OSKGC: A benchmark for Ontology Schema-based Knowledge Graph Construction from Text

Dali Wang[1,*], Mizuho Iwaihara[1]

[1] *Waseda University*

Abstract

Knowledge graph construction is a critical task in natural language processing, providing structured information that supports downstream applications such as recommendation systems and question answering. To ensure the accuracy and completeness of the information stored in knowledge graphs, it is essential to continuously update them based on new textual data while maintaining consistency with the existing knowledge graph architecture. To this end, constructing knowledge graphs from text based on a predefined ontology schema is crucial. However, existing benchmark only considers simplistic schema, lacking subsumptions. A new task shall be constructed, that requires not only accurate extraction of factual triples, but also structural alignment between the constructed knowledge graph and the predefined hierarchical ontology schema. For benchmarking this task, we introduce OSKGC, a benchmark dataset specifically designed for ontology schema-based knowledge graph construction, along with a new evaluation metric that measures the structural similarity between the constructed knowledge graphs and the predefined ontology schema. Compared to existing datasets, OSKGC ensures alignment between textual data, triples, and the ontology schema, provides fine-grained ontology annotations, and incorporates hierarchy within the ontology schema. To validate the utility of OSKGC, we propose two baseline methods under joint extraction and pipeline settings and conduct experiments using several mainstream large language models.

Source Repo: https://github.com/HeraclesWang/OSKGC

DOI: https://zenodo.org/records/15297020

Keywords

Benchmark, Knowledge Graph, Knowledge Graph Construction, Large Language Models

1. Introduction

In recent years, the task of extracting triples from text to construct knowledge graphs has garnered significant attention in the field of information extraction [1, 2]. Despite containing rich semantic information, text is challenging to use directly for complex tasks, such as automated reasoning and question answering [3, 4]. Knowledge graphs, in contrast, provide structured means of storing information, typically representing facts in the form of triples, where each triple consists of a subject, predicate, and object. For instance, the triple (Alabama, country, United States) represents the fact that Alabama belongs to the country United States. Numerous studies have demonstrated that knowledge graphs effectively support downstream tasks, including question answering [5, 6], decision-making [7, 8], recommendation systems [9, 10, 11], and search engines [12, 13]. However, knowledge graphs often suffer from issues of incompleteness due to their collaborative or semi-automatic construction processes [14]. Researchers have long sought to use newly retrieved information to augment existing knowledge graphs [15, 16]. A critical step in bridging unstructured text with structured knowledge involves extracting triples that align with the schema of existing knowledge graphs. This task has become a research hotspot, especially with the increasing demand for precise and consistent knowledge representation [17].

Current tasks for constructing knowledge graphs from text can be categorized into two main settings: Open information extraction (OIE) setting and ontology-driven, as shown in Figure 1. The OIE setting takes text as input and produces triples as output. For example, given the input text "The capital

Joint proceedings of KBC-LM and LM-KBC @ ISWC 2025

*Corresponding author.

dlwang@fuji.waseda.jp (D. Wang); iwaihara@waseda.jp (M. Iwaihara)
0009-0006-7934-1626 (D. Wang); 0000-0001-6985-9671 (M. Iwaihara)

of France is Paris," the output would be the triple (France, capital, Paris). To evaluate this setting, benchmark datasets such as WebNLG [18] and NYT [19] have been widely used. However, they only provide text-triple pairs without offering an ontology-based schema for constraint, which we refer to as "ontology schema" in this paper, to constrain the extracted triples. While suitable for open triple extraction, these datasets often fail to ensure the consistency and standardization of constructed triples, making them less applicable for tasks requiring domain-specific knowledge graph updates. In contrast, the ontology-driven setting takes both text and a predefined ontology as input and output triples. TEXT2KGBENCH [20] represents one of the few works in this domain, introducing an ontology-driven dataset to evaluate the performance of large language models in constructing knowledge graphs. However, its ontology schema is critically simple, primarily exhibiting a single star-shaped structure that inadequately captures the hierarchical information of fine-grained entity types. This limitation reduces its effectiveness for handling complex tasks requiring precise distinctions between entity types. Additionally, mismatches between the ontology schema and the text-triple data undermine its utility, highlighting the need for a high-quality dataset with better alignment.

To address these gaps, we propose a novel task setting for constructing knowledge graphs from text based on an ontology schema. In our setting, the input consists of the text and a predefined ontology schema, and the output includes both triples and their corresponding schemas. We evaluate not only the factual extraction performance but also the alignment between the schema of the constructed knowledge graph and the predefined ontology schema. This aims to extract structured information from new texts to expand the knowledge graph without altering its existing architecture. To support this task, we propose a benchmark dataset, OSKGC, along with an evaluation metric tailored to our setting. Considering the need to design distinct ontology schemas for different thematic domains, we selected WebNLG, which characterized by its clear topic separation, as the foundation. We provide fine-grained annotations for entities, such as subdividing "politician" into categories like president, vice president, and mayor, to better reflect the types of entities in the text. Furthermore, we introduce a subsumption hierarchy derived from DBpedia [21] to support fine-grained entity type recognition. Through these enhancements, OSKGC spans multiple domains, features fine-grained annotations, and ensures well-aligned ontology schemas. Furthermore, we adopt several mainstream LLMs as baselines and conducted experiments using joint extraction and pipeline settings to confirm OSKGC as a challenging benchmark.

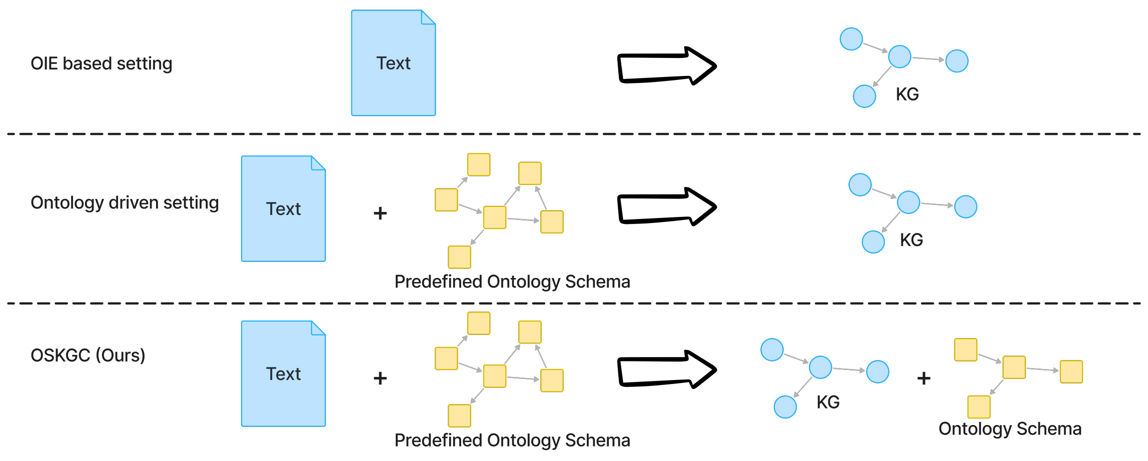


Figure 1: Task settings for constructing knowledge graph from text.

2. Related Work

Constructing knowledge graphs from unstructured text typically involves extracting entities and their relations and organizing them into structured triples. In this field, several benchmark datasets have been proposed for model training and evaluation, which are summarized in Table 1. CoNLL04 [22] as an early representative benchmark, employed manual annotation, clearly defining entity types and relation categories, providing a solid experimental platform for named entity recognition (NER) and relation extraction (RE). However, it lacks fine-grained entity type classification and does not incorporate a formal ontology schema, focusing primarily on general-domain relation extraction task.

With the integration of large-scale textual resources and knowledge bases, distant supervision has been adopted to automatically construct training data. This approach assumes that if two entities have a certain relation in a knowledge base, they also express the same relation when they co-occur in the same sentence. Typical datasets such as NYT [19] and REBEL [23] were constructed using distant supervision, automatically annotating large volumes of triple data, significantly improving the efficiency of construction. However, this method also introduces considerable noise, such as incorrect entity pairings or missing relation labels. Moreover, such datasets typically do not define entity types or build extensible ontology schemas, which limits their support for canonicalization of extracted triples and knowledge graph expansion.

Table 1
Overview of related benchmark datasets

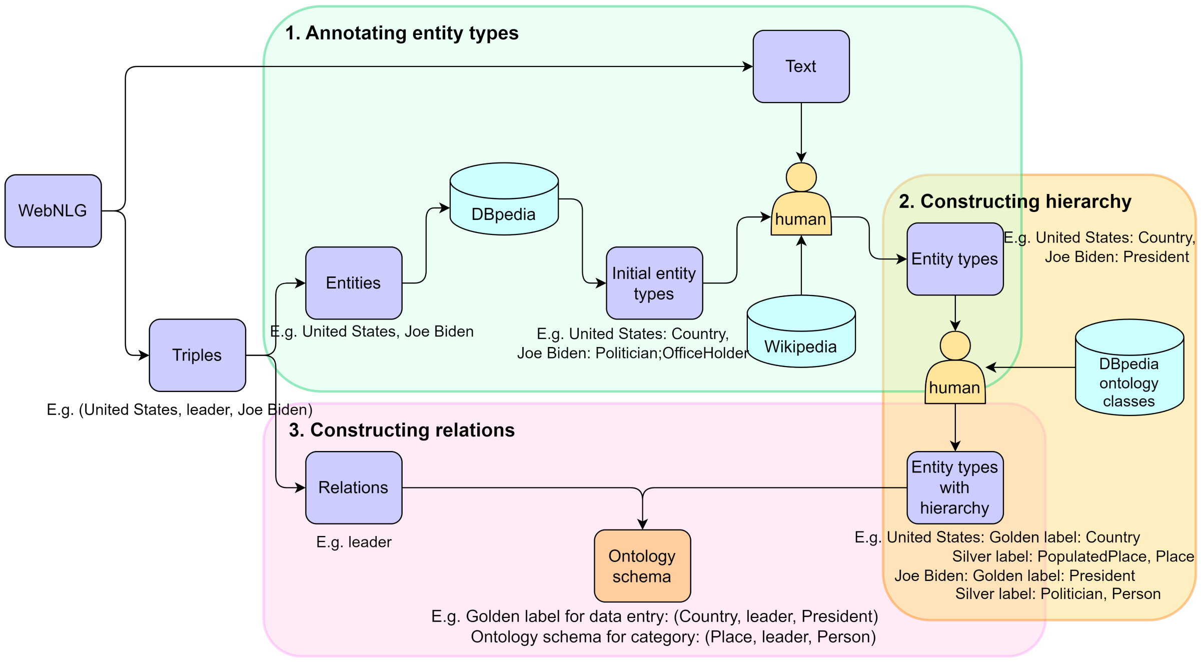
Dataset	Defined Entity Types	Fine-Grained Typing	Ontology Schema	Ontology Hierarchy	Build Method	Thematic Distinction
CoNLL04 [22]	✓	✗	✗	✗	human annotation	✗
WebNLG [18]	✗	✗	✗	✗	human annotation	✓
TekGen [24]	✗	✗	✗	✗	distant supervision	✗
NYT [19]	✗	✗	✗	✗	distant supervision	✗
REBEL [23]	✗	✗	✗	✗	distant supervision	✗
TEXT2KGBENCH [20]	✓	✗	✓	✗	human annotation	✓
Ours	✓	✓	✓	✓	human annotation	✓

Other works attempt to generate corresponding text directly from structured data to construct training data, such as WebNLG [18] and TekGen [24]. These datasets pair triples with natural language descriptions, aiming to train models to generate factual text or recover triples from text. The WebNLG dataset uses manual annotation and covers multiple thematic domains, providing good topic differentiation. In contrast, TekGen was built using distant supervision, lacking entity type definitions and the design of ontology schema. Furthermore, the recently proposed TEXT2KGBENCH [20] introduced a more systematic ontology design for ontology-driven triple extraction. However, the constructed ontology schema is overly simplistic, with each category forming a star-shaped structure centered around a core ontology, which makes it incapable of handling associations between entities with deep hierarchical structures. In addition, the ontology schema does not align well with the text and triples, because the context of the text was not fully considered during construction, resulting in low data quality.

Overall, most mainstream datasets today focus primarily on triple extraction itself, with limited attention to ontology schema construction and fine-grained entity typing. These limitations hinder the ability of current datasets to support incremental updates and structural alignment within existing KG resources, thereby restricting the scalability of KGs in practical applications. In contrast, OSKGC provides fine-grained entity annotations along with an explicitly ontology schema with hierarchy, filling the gap left by existing datasets.

3. Benchmark Construction

In this section, we present the construction method of the OSKGC dataset, consisting of 57 categories, where each category has two main components: The ontology schema and the corresponding text-triple pairs. To ensure alignment among the text, triples, and ontology schema in the dataset, we construct the ontology schema from scratch based on WebNLG [18] and cleaned the text-triple pairs. Additionally, we introduce an evaluation metric designed to assess the alignment between the knowledge graph constructed by the model and the predefined ontology schema.



1. Annotating entity types
Text
WebNLG
DBpedia
human
Entities
Initial entity types
Triples
E.g. United States, Joe Biden
E.g. United States: Country, Joe Biden: Politician;OfficeHolder
Wikipedia
E.g. (United States, leader, Joe Biden)
2. Constructing hierarchy
Entity types
E.g. United States: Country, Joe Biden: President
human
DBpedia ontology classes
3. Constructing relations
Relations
E.g. leader
Entity types with hierarchy
E.g. United States: Golden label: Country
Silver label: PopulatedPlace, Place
Joe Biden: Golden label: President
Silver label: Politician, Person
Ontology schema
E.g. Golden label for data entry: (Country, leader, President)
Ontology schema for category: (Place, leader, Person)

Figure 2: Overview of the ontology schema construction process for OSKGC.

3.1. Construction of Ontology Schema

The construction of the ontology schema serves two primary objectives: (1) to generate a schema label for each text-triple pair, which is used for training and evaluation; and (2) to build a predefined ontology schema for each category, which serves as a structural guide for knowledge graph construction. In WebNLG, the data are presented as pairs of text and corresponding triples, without explicit entity type information. We selected data entries that contain one, two, or three triples as the basis for constructing our ontology schema. To ensure alignment between the ontology schema and the text-triple data, we first annotate the entities in each triple with their corresponding entity types, thereby transforming each triple into a type-level schema that serves as the label for the associated data entry. Subsequently, based on the subsumption hierarchy of the DBpedia ontology, we enrich each annotated entity type by recursively adding more general types, thus constructing a subsumption hierarchy. Finally, for each category, we aggregate all schema labels associated with the data entries. For every entity type in the head and tail positions of these schema labels, we identify its corresponding root-level general type within the subsumption hierarchy. Using the relations from the schema labels, we link these root-level types to form a core structural schema in the form of triples. By summarizing the core structural schemas and their associated hierarchical structures, we construct a predefined ontology schema for each category. The complete ontology schema construction process is illustrated in Figure 2, consisting of three main steps: annotating entity types, constructing the subsumption hierarchy, and constructing relations.

3.1.1. Annotating Entity Types

In this step, we annotate the entity types in the triples of each data entry. For each entity in the triples, we query DBpedia to find the deepest level type of the entity and use it as the initial type for the entity. We manually reviewed the queried entity types to ensure the accuracy of the annotations. On this basis, we performed fine-grained annotation manually for each entity. Specifically, we used external knowledge sources including Wikipedia and Wikidata [25] as references to assign more precise entity types to each entity. For example, for the entity "Rome," the initial type obtained from DBpedia was

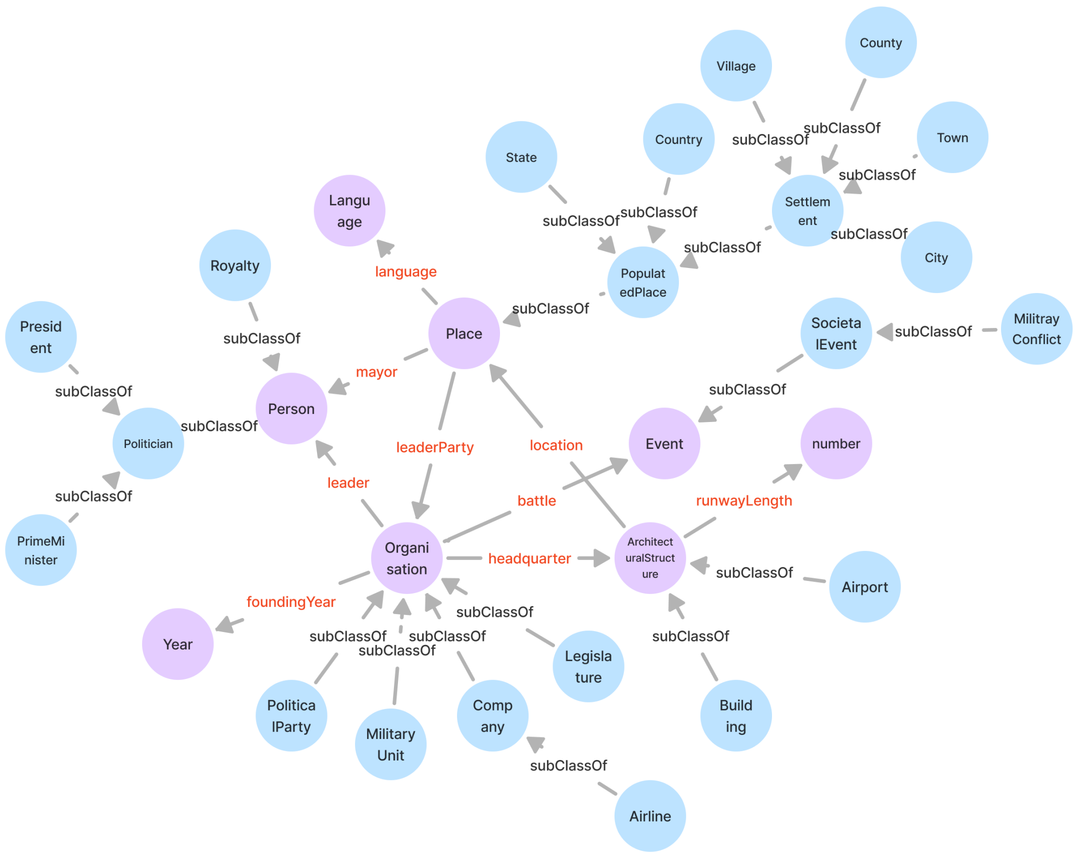


Figure 3: An example of a portion of the predefined ontology schema. The purple nodes represent general entity types located at the root level, which are connected via category-specific relations to form the core structure. The blue nodes denote fine-grained entity types in the subsumption hierarchy and are linked to the core structure through the root nodes.

"City." After manual annotation, it was refined to "CapitalCity." In this process, we also referred to the context of each entity in the text to ensure that the annotated entity type aligns with the content in the text. This is crucial due to the polysemy of entities. For example, the entity "Christian Panucci" was a soccer player during his early career and later became a soccer manager. For such polysemous entities, we annotate them with the type that aligns with the context of the text, ensuring that the constructed ontology schema remains consistent with the textual content. After this step, we obtain the corresponding entity type for each entity in different text contexts.

3.1.2. Constructing Hierarchy

In this step, we add a subsumption hierarchy to all entity types. The goal is to effectively categorize the annotated entity types into a hierarchy. Specifically, we use the hierarchy of DBpedia ontology classes as a reference. For each entity type, we query its hierarchical path to the root node. For example, for the entity type "President," the obtained path is "President $\rightarrow$ Politician $\rightarrow$ Person $\rightarrow$ Animal $\rightarrow$ Eukaryote $\rightarrow$ Species $\rightarrow$ Thing". For these paths, we removed the nodes that do not contribute to effective classification. Specifically, we remove overly general nodes that are close to the root and retain the nodes starting from the most general node in the path that can identify the annotated entity type. For example, for the entity type "President", the most general entity type that can identify it is "Person". Therefore, the retained path is "President $\rightarrow$ Politician $\rightarrow$ Person", with overly general nodes above the "Person" level removed. In addition, some nodes in the hierarchy of DBpedia have sibling nodes that do not have instantiated entities in OSKGC. These nodes not only fail to contribute to effective classification but also add unnecessary complexity to our hierarchy. Therefore, we consider them redundant and remove them from the paths where they appear. For example, for the entity type "FormulaOneRacer", its path is "FormulaOneRacer → RacingDriver → MotorsportRacer → Athlete → Person". Since the sibling nodes of "MotorsportRacer" and "RacingDriver" do not have instantiated entities in OSKGC, these siblings are removed. Since we have performed fine-grained annotation of entity types, not all entity types are included in DBpedia. For these entity types, we add them to appropriate positions within the hierarchy.

In OSKGC, not all entity types have a hierarchy. Since the purpose of introducing the hierarchy is to classify the finely annotated entity types, we did not build additional hierarchies for entity types that can be distinct on their own. For instance, entity types like "EthnicGroup" and "Language", which are inherently easy to distinguish, do not need additional hierarchy for classification. For all entity types with a hierarchy, we define the manually annotated entity type in the schema label as the golden label, and all the higher-level nodes in the hierarchy as silver labels. For example, for the type "President," the golden label is "President," while the silver labels include "Politician" and "Person". The golden label represents the fine-grained and precise entity type, whereas the silver labels correspond to more general entity types. These silver labels are used to evaluate the structural similarity of the constructed knowledge graph.

3.1.3. Constructing Relations

In this step, we construct relations between entity types based on the instantiated triples from the WebNLG corpus. Specifically, we connected annotated entity types and their root-level types within the hierarchy using relations derived from the triples, which allowed us to generate the schema label for the text-triple data and the predefined ontology schema. For example, given the triple (United States, leader, Joe Biden), where the type of "United States" is "Country" and the type of "Joe Biden" is "President," the schema label for the data entry is (Country, leader, President). Subsequently, based on the hierarchy, we determined that the root nodes corresponding to the entity types "Country" and "President" are "Place" and "Person", respectively. Thus, we derived (Place, leader, Person) and included it in the predefined ontology schema. Figure 3 illustrates a portion of the predefined ontology schema we constructed as an example. Since the entity types connected by relations in the ontology schema are root nodes in the hierarchy, which represents the most general classification of the entity types, the schema labels specific to each data instance are not directly reflected here. We followed the classification standards of the WebNLG and divided the data into three groups based on the number of triples they contain. These were further subdivided into 19 thematic categories, resulting in a total of 57 categories. For each category, we independently summarized a predefined ontology schema.

Table 2
Issues on the text-triple pairs in WebNLG and corresponding examples. The problematic parts are highlighted in bold.

Issue	Text	Label
Entity inconsistency	In Indiana, the language spoken is American English.	(Indiana, language, **English Americans**)
Semantic inconsistency	Bacon and sausage are the main ingredients in a Bacon Explosion, which comes from the United States.	(Bacon Explosion, **ingredient**, Bacon); (Bacon Explosion, mainIngredient, Sausage)
Irrelevant triple	The comic character Asterix, was created by René Goscinny and Albert Uderzo.	(Asterix, creator, René Goscinny); **(Asterix, alternativeName, Astérix)**; (Asterix, creator, Albert Uderzo)

3.2. Cleaning Text-Triple Pairs

Apart from the ontology schema, another crucial component of OSKGC is the text-triple pairs. This portion of the data is sourced from WebNLG. To ensure consistency and alignment between the text and the triples, we conducted a thorough review and validation of the text-triple pairs. As shown in Table 2, the issues we found in WebNLG primarily include inconsistencies between entities in the text and those in the triples, mismatches between the semantics of the text and the triples, and triples in the labels containing information not present in the text. For entity inconsistency, an example is such that the text mentions "American English" as a language, while the tail entity in the triple is "English

Algorithm 1: Compute Structural Similarity (SS)

```
**Input:** Ontology hierarchy $\mathcal{H}$, Golden schemas $G$, Predicted schemas $P$
**Output:** Structural similarity score $SS \in [0, 1]$
Initialize $total\_score \leftarrow 0$
**foreach** *triple* $(h_p, r_p, t_p)$ *in* $P$ **do**
    Find $(h_g, r_g, t_g)$ in $G$ such that $r_p = r_g$;
    $score_h \leftarrow$ ComputeSS$(\mathcal{H}, h_g, h_p)$, $score_t \leftarrow$ ComputeSS$(\mathcal{H}, t_g, t_p)$;
    $total\_score \leftarrow total\_score + (score_h \times score_t)$;
$SS \leftarrow \begin{cases} total\_score/|G|, & \text{if } |G| \geq |P| \\ \left(total\_score \times \left(\frac{|G|}{|P|}\right)^2\right)/|G|, & \text{otherwise} \end{cases}$
**return** $SS$;
```

Americans," which is an ethnic group. In such cases, we correct the entity in the triple to ensure it aligns with the text. In cases of semantic inconsistency, the triples may include the correct entities, but the semantic information they convey does not align with the text. For example, the text states that both "Bacon" and "Sausage" are the main ingredients of "Bacon Explosion." However, the triples in the label indicate only that "Sausage" is the main ingredient, while "Bacon" is merely an ingredient. Such inconsistencies can mislead the model, thereby affecting the evaluation of the results. To resolve this issue, we modified the triples to ensure they accurately reflect the content of the text. In cases where the triples in the label contain information not present in the text, for example, the label includes a triple (Asterix, alternativeName, Astérix), but the text does not mention this information. In such cases, we removed the triples that were not related to the text. Through the above cleaning process, we corrected the errors in WebNLG and ensured mutual alignment between the text and the triples, thus ensuring the quality of OSKGC.

3.3. Evaluation Metric

In our task setting, we aim to extract correct triples from the text while ensuring that the constructed knowledge graph aligns with the predefined ontology schema. To this end, we propose an evaluation metric called Structural Similarity (SS) to measure the degree of alignment between the schema of the constructed knowledge graph and the predefined ontology schema, as shown in Algorithm 1.

The proposed metric begins by evaluating the structural similarity between individual pairs of entity types, specifically between a gold entity type g and a predicted entity type p. Let D denote the path length from the gold type g to the root node of the ontology. Based on the structural relation between g and p within the ontology hierarchy, we define two types of matching cases: If the predicted type p lies on the ancestral path from the gold type g to the root, the match is considered an ancestor match. In this case, the structural similarity score is computed using Equation 1,

$$\text{SS} = \exp\left(-\alpha \cdot \frac{d}{D}\right) \tag{1}$$

where d is the path length from g to p, and α is a parameter that controls the decay rate of the score, which we set to 2. This design reflects the intuitive principle that the closer the predicted type is to the gold type in the hierarchy, the higher the SS score it receives. If the predicted type p is not on the ancestral path of g but they share a lowest common ancestor (LCA), the match is regarded as a lowest common ancestor (LCA) match. In this case, the similarity score simultaneously considers the distances from both g and p to their LCA. Additionally, we introduce local structural entropy as a penalty factor to model the complexity of the local topological structure around the predicted node. The scoring is defined by Equation 2,

$$\text{SS} = \exp\left(-\alpha \cdot \frac{d}{D}\right) \times \exp\left(-\beta \cdot \frac{d'}{D + \log_2(S_p + 1)}\right) \tag{2}$$

where d and d' denote the path lengths from g and p to the LCA, respectively; S_p is the number of sibling nodes of the predicted type p; and β is a parameter that controls the strength of the entropy penalty, which we set to 1.5. In graph representation learning, structural entropy is commonly used to quantify the complexity and uncertainty of graph structures [26]. The local structural entropy term $\log_2(S_p + 1)$ is introduced to capture the branching complexity of the subtree where the predicted node resides. A larger number of sibling nodes indicates higher local uncertainty and an increased likelihood of misclassification, thus resulting in a greater penalty. This design is consistent with the principle of entropy in information theory, which is used to quantify uncertainty and complexity in systems [27]. If no common ancestor exists between g and p, they belong to different connected components in the ontology, and the SS score is 0.

For each predicted schema, the final SS score is calculated by multiplying the SS scores of its head and tail entity types. At the data entry level, where a single entry may contain multiple schemas, the overall SS is computed by summing the scores of all schemas and dividing by the number of corresponding golden labels. Notably, when the number of predicted triples exceeds that of the golden triples, we introduce an additional penalty factor to suppress redundancy in predictions, thereby ensuring that the metric maintains strong discriminative power and robustness across results of varying scales.

4. Dataset Statistics

In this section, we present the statistical information of OSKGC. OSKGC consists of two components: text-triple-schema pairs and predefined ontology schema. Each data entry consists of a textual instance labeled with its corresponding subgraph of the knowledge graph in triple form and the schema label associated with the subgraph. For each category, we provide its predefined ontology schema. As shown in Table 3, we divide the data into three groups based on the number of triples contained in the text: those with one triple, two triples, and three triples. Within each group, the text data is further categorized into 19 categories based on the topic of the text, resulting in a total of 57 categories. For each category, although there are overlapping parts in the ontology schema, each also has unique components. Therefore, we provide a dedicated predefined ontology schema for each category. The data in each category are divided into a training set, a validation set, and a test set in a 7:1:2 ratio. We ensured that the test set contains entities that do not appear in the training or validation sets. OSKGC contains a total of 207 entity types, 382 relations, and 10,183 text entries. The knowledge graph includes 3,446 entities, and the predefined ontology schema consists of 691 edges, 207 of which represent the subsumption hierarchy. The maximum depth of the hierarchy is 4, with an average of 1.6. Among the entity types that have descendant nodes, the maximum in-degree is 21, with an average of 4.6.

5. Experiment

In this section, we introduce the baseline experiments conducted on OSKGC. To evaluate the performance of OSKGC, we employed two experimental setups: joint extraction and pipeline, as shown in Figure 4. We conducted experiments using several current mainstream LLMs, including open-source models Llama3-8b [28], Phi-3-small [29], Qwen2.5-7b [30], Mistral-7b [31] and proprietary models GPT-4o [32], Gemini 1.5 pro [33] and Claude 3.5 sonnet [34].

5.1. Joint Extraction Setup

In this part of the experiment, we designed a single prompt to jointly extract entities and relations, enabling the construction of the knowledge graph in one step, as shown in Prompt 1. The goal is to evaluate whether the LLM is capable of handling a large amount of complex ontology schema information. In this experimental setup, the prompt consist of the entity types, relations, and hierarchy from the ontology schema along with the test text. Additionally, we provide a one-shot example to assist the model in understanding and specify the format for the output. The input ontology schema information is raw data without any additional processing, and the ontology schema for data belonging

Table 3
Statistics of OSKGC at the category level, including the number of entity types, the number of relations, and the count of textual data.

Category	1 triple			2 triples			3 triples		
	Ont	Rel	Text	Ont	Rel	Text	Ont	Rel	Text
Airport	44	54	350	42	50	240	39	35	228
Artist	36	44	324	36	34	276	31	30	298
Astronaut	28	37	86	23	30	67	28	36	84
Athlete	46	45	326	39	31	219	32	32	258
Building	47	39	256	47	39	195	47	37	219
CelestialBody	11	27	196	10	25	157	11	25	153
City	23	26	279	19	21	244	22	21	256
ComicsCharacter	20	20	111	18	16	94	17	19	78
Company	33	35	104	21	21	101	21	24	99
Film	25	33	74	18	17	41	19	22	51
Food	48	28	271	45	25	280	43	22	310
MeanOfTransportation	41	72	333	36	66	240	36	63	249
Monument	23	24	38	21	20	39	23	24	49
MusicalWork	16	28	88	12	15	49	17	22	53
Politician	53	46	334	46	38	281	48	39	295
Scientist	29	37	73	20	23	52	25	27	50
SportsTeam	31	29	283	28	26	199	24	23	197
University	29	40	71	25	31	59	29	41	72
WrittenWork	35	49	250	32	42	230	30	40	274

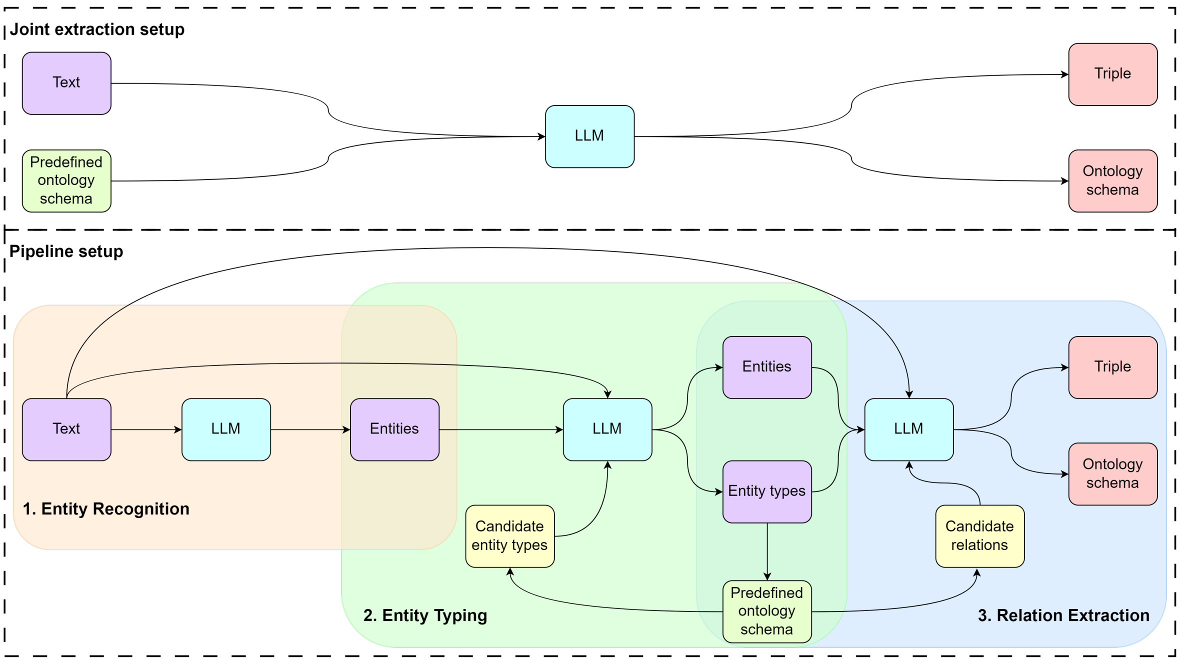


Figure 4: Overview of the baseline experiment setups.

to the same category is constant. We used two methods to select the one-shot example: random selection and similarity-based selection. For random selection, we randomly choose a fixed data sample from the training set of each category to serve as the example. All test data within the category share this example. For similarity-based selection, we used SBERT [35] to find the most similar text in the training set of the corresponding category for each test data based on text similarity. Each test data entry has its own unique example. The output includes the triples and their corresponding ontology schema.

Prompt 1: Joint Extraction

Your task is to construct a knowledge graph from the input text based on the given ontology schema. The goal is to extract triples based on the given ontology schema's entity types, relations, and hierarchy, and provide the most accurate corresponding ontology schema possible.
Ontology schema:
entity type: {entity type}
relation: {relation}
hierarchy: {hierarchy}
Example text: {example text}
Example output: {example output}
The output format must strictly follow the example, with no additional text or explanations.
Input text: {text}
Output:

Prompt 2: Entity Recognition

Your task is to find all the named entities in the given text, including numbers, codes, and dates. You need to find at least two entities. Only focus on named entities and never extract adjectives or numerical units. Please strictly follow the format of the following example and only provide the named entities you find, without any additional words.
Example text: {example text}
Output: {example output}
Text: {input text}
Output:

Prompt 3: Entity Typing

Please select the appropriate entity type for each given entity from the candidate entity types based on the text and your knowledge.
Text: {text}
Entities: {entities}
Candidate entity types: {candidate types}
Each square bracket in the candidate entity types contains a candidate entity type. If a candidate entity type is followed by a colon, then each entity type enclosed in square brackets within the subsequent curly brace is a sub-type of that entity type.
Example:
Text: {example text}
Entities: {example entities}
Output: {example output}
Please select only one most appropriate entity type from the brackets for each entity, strictly following the format shown in the example, without any additional words or explanation.
Output:

Prompt 4: Relation Extraction

Please select relations from the candidates to connect given entities in the text, if they exist.
Example:
Text: {example text}
Given entities: {example entities}
Candidate relations: {example relations}
Output: {example output}
Please strictly follow the output format in the example without any additional words.
Text: {input text}
Given entities: {entities}
Candidate relations: {relations}
Output:

5.2. Pipeline Setup

In addition to the joint extraction method, we also adopted a pipeline experimental setup to construct the knowledge graph step by step. Specifically, the process is divided into three steps: Entity recognition, entity typing, and relation extraction, with separate prompts designed for each step. Similar to the joint extraction experimental setup, we provided a one-shot example in the prompt, using both random selection and similarity-based selection methods. Entity recognition aims to identify entity names from the text. As shown in Prompt 2, we input the text into the prompt. To guide the LLM in generating the expected output, we provided a one-shot example to specify the output format. The entity typing step involves assigning the corresponding entity types to the extracted entity names. The prompt template is shown in Prompt 3. The inputs for this step include the entity names output from entity recognition step, the candidate entity types for the category of the input data, the input text, and a one-shot example. The output is the corresponding entity type for each entity. The candidate entity types are derived from the ontology schema of the category to which the input data belongs. Finally, relation extraction involves identifying the relations between entities to form triples. The prompt template for this step is shown in Prompt 4. The input for this step includes the entity names, candidate relations, the input text, and a one-shot example. For the candidate relations, we did not use all the relations from the category of the input data. Instead, we traced back the entity types obtained in the entity typing step to their root node types based on the hierarchy. Then, we compiled all the relations existing between these root nodes as the candidate relations. Through this approach, we eliminated the relations that were irrelevant to the input data, thereby reducing the number of candidate relations and streamlining the prompt.

5.3. Experimental Results

Table 4 present the experimental results of open-source LLMs and proprietary LLMs. Precision, recall, and F1 are metrics used to evaluate the triples extracted by the model. SS is the metric we proposed to assess the similarity between the ontology schema constructed by the model and the predefined ontology schema. Regarding the selection of one-shot example, the results from both the pipeline and joint extraction setups indicate that using examples similar to the input text yields better results than random selection. Comparing the pipeline and joint extraction experimental setups, we can observe that the results of the joint extraction setup generally outperform those of the pipeline setup in terms of the evaluation of the extracted triples. However, when evaluating the similarity between the constructed knowledge graph's ontology schema and our predefined ontology schema, the pipeline approach generally performs better. These baseline results leave room for improvement in future work.

Table 4
Experimental results of open-source and proprietary LLMs on OSKGC. Rand refers to the results obtained by randomly selecting a one-shot example, while SBERT refers to the results obtained by selecting the most similar one-shot example to the test data using SBERT.

Method	Metrics	Llama3-8b		Phi-3-small		Qwen2.5-7b		Mistral-7b		GPT-4o		Gemini 1.5 Pro		Claude 3.5 Sonnet	
		Rand	SBERT	Rand	SBERT	Rand	SBERT	Rand	SBERT	Rand	SBERT	Rand	SBERT	Rand	SBERT
Pipeline	Precision	0.328	0.464	0.241	0.349	0.417	0.544	0.299	0.464	0.550	0.673	0.465	0.600	0.494	0.602
	Recall	0.483	0.683	0.513	0.685	0.660	0.801	0.538	0.758	0.750	0.851	0.617	0.806	0.694	0.835
	Micro F1	0.391	0.553	0.328	0.463	0.511	0.648	0.385	0.576	0.635	0.751	0.531	0.688	0.577	0.700
	Macro F1	0.347	0.493	0.346	0.460	0.529	0.649	0.403	0.571	0.634	0.745	0.486	0.661	0.576	0.702
	SS	0.161	0.264	0.161	0.299	0.351	0.485	0.212	0.381	0.519	0.658	0.433	0.597	0.466	0.584
Joint extraction	Precision	0.274	0.392	0.388	0.485	0.423	0.572	0.353	0.479	0.596	0.694	0.580	0.679	0.601	0.682
	Recall	0.693	0.815	0.700	0.812	0.742	0.861	0.677	0.821	0.803	0.893	0.805	0.871	0.838	0.900
	Micro F1	0.393	0.530	0.499	0.607	0.539	0.687	0.464	0.605	0.684	0.781	0.674	0.763	0.700	0.776
	Macro F1	0.481	0.601	0.551	0.627	0.582	0.701	0.518	0.646	0.693	0.766	0.682	0.763	0.705	0.786
	SS	0.107	0.228	0.190	0.423	0.158	0.440	0.259	0.471	0.404	0.627	0.580	0.691	0.482	0.619

5.4. Error Analysis

We examined the responses generated by the baseline LLMs and found that the main issues include incorrect entity extraction, relation extraction errors, mismatches between the constructed triples and the semantic meaning of the text, hallucinations, and inconsistencies between the structure of the triples and the predefined ontology schema. Table 5 presents examples of such errors from Claude. For relation errors, the model generated "locatedIn," which was not among the candidate relations; the correct relation should have been "location." Regarding entity errors, the model produced the entity "Bininigt," which differs from the "Binignit" mentioned in the text. In the case of factual errors, the output triple (Faversham, IsA, Town) does not align with any semantic evidence in the text. The model also produced hallucinated entities not mentioned in the input text, such as "A Wizard of Mars." Lastly, the structure of some output triples violated the predefined ontology schema, for example, the subject and object were reversed compared to the expected format. In particular, these examples come from Claude, a proprietary model that generally performed better among the baselines. These types of error are even more frequent in open-source models with fewer parameters.

Table 5
Errors identified in the response from the baseline LLMs.

Error Type	Text	Error Triple	Comment
Relation Error	Lady Anne Monson was born in Darlington which is located in Kingdom of England.	(Darlington, locatedIn, Kingdom of England)	`locatedIn` is not among the candidate relations.
Entity Error	Sago is the main ingredient of binignit recipes, which also contain sweet potatoes, and come from the Philippines.	(Bininigt, country, Philippines)	The entity in the text is `Binignit`, not `Bininigt`.
Factual Error	Faversham was the birthplace of Adam Holloway, who resided in Gravesend.	(Faversham, IsA, Town)	The information in the triple is not expressed in the text.
Hallucination	Written in English, the book Alcatraz Versus the Evil Librarians is from The United States where one of the ethnic groups is Asian Americans.	(A Wizard of Mars, language, English)	The entity `A Wizard of Mars` does not appear in the text.
Ontology Violation	Robert A. M. Stern is the architect of Alan B. Miller Hall of which The Mason School of Business is the current tenant.	(Robert A. M. Stern, architect, Alan B. Miller Hall)	The schema (Person, architect, ArchitecturalStructure) conflicts with the predefined ontology (ArchitecturalStructure, architect, Person).

6. Discussion

The primary task setting described in this paper for OSKGC involves constructing triples and their corresponding schemas from text based on a predefined ontology schema. However, this is not the only task applicable to the dataset. OSKGC provides aligned data for text, triples, and schemas, as well as a predefined ontology schema at the category level. Researchers can define other potential task settings by using the fine-grained annotations provided in OSKGC. Inverse tasks are also possible, for example, given an entity type label such as "airport" and a text as context, one could formulate a question-answering task to predict the specific entity name.

In terms of baseline experiments, we primarily focus on evaluating whether current LLMs, known for their powerful information processing capabilities, can directly handle our complex knowledge graph construction task. According to the baseline results, while LLMs demonstrate better performance than previous similar work [20] under various evaluation metrics, they still suffer from issues such as incorrect entity extraction, failure to conform to the predefined ontology schema, and hallucinations, even with high-performing proprietary LLMs. In this work, we did not incorporate multi-model frameworks, external knowledge graph lookups, or retrieval-augmented generation techniques, which we consider promising directions for future research.

7. Conclusion

In this paper, we proposed a novel task setting for constructing knowledge graphs from text based on ontology schema. Additionally, we introduced a corresponding benchmark dataset, OSKGC, and an evaluation metric. Compared to existing datasets, OSKGC features fine-grained annotations and incorporates hierarchy. While ensuring quality, OSKGC introduces more complex ontology schemas, thus increasing the level of challenge. We conducted baseline experiments with joint extraction and pipeline settings on OSKGC using the current mainstream LLMs. The baseline results indicate that there is room for improvement on our fine-grained dataset.

8. Acknowledgments

Weixi Wang contributed to the construction of OSKGC. This work was in part supported by JSPS KAKENHI Grant Number 22K12044.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Z. Wei, J. Su, Y. Wang, Y. Tian, Y. Chang, A novel cascade binary tagging framework for relational triple extraction, arXiv preprint arXiv:1909.03227 (2019).

[2] H. Ye, N. Zhang, S. Deng, M. Chen, C. Tan, F. Huang, H. Chen, Contrastive triple extraction with generative transformer, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 14257–14265.

[3] W. Zhong, S. Wang, D. Tang, Z. Xu, D. Guo, J. Wang, J. Yin, M. Zhou, N. Duan, Ar-lsat: Investigating analytical reasoning of text, arXiv preprint arXiv:2104.06598 (2021).

[4] A. Asai, E. Choi, Challenges in information-seeking qa: Unanswerable questions and paragraph retrieval, arXiv preprint arXiv:2010.11915 (2020).

[5] A. Bordes, S. Chopra, J. Weston, Question answering with subgraph embeddings, arXiv preprint arXiv:1406.3676 (2014).

[6] L. Dong, F. Wei, M. Zhou, K. Xu, Question answering over freebase with multi-column convolutional neural networks, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, pp. 260–269.

[7] L. Guo, F. Yan, Y. Lu, M. Zhou, T. Yang, An automatic machining process decision-making system based on knowledge graph, International journal of computer integrated manufacturing 34 (2021) 1348–1369.

[8] L. T. H. Lan, T. M. Tuan, T. T. Ngan, N. L. Giang, V. T. N. Ngoc, P. Van Hai, et al., A new complex fuzzy inference system with fuzzy knowledge graph and extensions in decision making, Ieee Access 8 (2020) 164899–164921.

[9] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, Kgat: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 950–958.

[10] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering 34 (2020) 3549–3568.

[11] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The world wide web conference, 2019, pp. 3307–3313.

[12] J. Liu, J. Ren, W. Zheng, L. Chi, I. Lee, F. Xia, Web of scholars: A scholar knowledge graph, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 2153–2156.

[13] C. Xiong, R. Power, J. Callan, Explicit semantic ranking for academic search via knowledge graph embedding, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 1271–1279.

[14] J. Xu, K. Chen, X. Qiu, X. Huang, Knowledge graph representation with jointly structural and textual encoding, arXiv preprint arXiv:1611.08661 (2016).

[15] X. Lv, Y. Lin, Z. Yao, K. Zeng, J. Zhang, L. Hou, J. Li, Step out of kg: Knowledge graph completion via knowledgeable retrieval and reading comprehension, arXiv preprint arXiv:2210.05921 (2022).

[16] K. Zeng, H. Jin, X. Lv, F. Zhu, L. Hou, Y. Zhang, F. Pang, Y. Qi, D. Liu, J. Li, et al., Xlore 3: A large-scale multilingual knowledge graph from heterogeneous wiki knowledge resources, ACM Transactions on Information Systems (2024).

[17] L. Zhong, J. Wu, Q. Li, H. Peng, X. Wu, A comprehensive survey on automatic knowledge graph construction, ACM Computing Surveys 56 (2023) 1–62.

[18] C. Gardent, A. Shimorina, S. Narayan, L. Perez-Beltrachini, Creating training corpora for nlg

micro-planning, in: 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Association for Computational Linguistics (ACL), 2017, pp. 179–188.

[19] S. Riedel, L. Yao, A. McCallum, Modeling relations and their mentions without labeled text, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III 21, Springer, 2010, pp. 148–163.

[20] N. Mihindukulasooriya, S. Tiwari, C. F. Enguix, K. Lata, Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text, in: International Semantic Web Conference, Springer, 2023, pp. 247–265.

[21] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: international semantic web conference, Springer, 2007, pp. 722–735.

[22] D. Roth, W.-t. Yih, 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, 2004, pp. 1–8.

[23] P.-L. H. Cabot, R. Navigli, Rebel: Relation extraction by end-to-end language generation, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 2370–2381.

[24] O. Agarwal, H. Ge, S. Shakeri, R. Al-Rfou, Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training, arXiv preprint arXiv:2010.12688 (2020).

[25] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[26] D. Zou, H. Peng, X. Huang, R. Yang, J. Li, J. Wu, C. Liu, P. S. Yu, Se-gsl: A general and effective graph structure learning framework through structural entropy optimization, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 499–510.

[27] C. E. Shannon, A mathematical theory of communication, The Bell system technical journal 27 (1948) 379–423.

[28] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al., The llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024).

[29] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, et al., Phi-3 technical report: A highly capable language model locally on your phone, arXiv preprint arXiv:2404.14219 (2024).

[30] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al., Qwen2. 5 technical report, arXiv preprint arXiv:2412.15115 (2024).

[31] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7b, 2023. URL: https://arxiv.org/abs/2310.06825. arXiv:2310.06825.

[32] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al., Gpt-4o system card, arXiv preprint arXiv:2410.21276 (2024).

[33] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, et al., Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, arXiv preprint arXiv:2403.05530 (2024).

[34] A. Anthropic, The claude 3 model family: Opus, sonnet, haiku, Claude-3 Model Card 1 (2024) 4.

[35] N. Reimers, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).