We propose Descriptive Knowledge Graph (DKG) – an open and interpretable form of modeling relationships between entities. In DKGs, relationships between entities are represented by relation descriptions. For instance, the relationship between entities of machine learning and algorithm can be described as “Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.” To construct DKGs, we propose a self-supervised learning method to extract relation descriptions with the analysis of dependency patterns and a transformer-based relation description synthesizing model to generate relation descriptions. Experiments demonstrate that our system can extract and generate high-quality relation descriptions for explaining entity relationships.

1 Introduction

Relationships exist widely between entities. For example, a person may be related to another person or an institution, and a scientific concept can be connected to another concept. At the same time, relationships between entities can be subtle or complex, e.g., the relationship between machine learning and algorithm.

To model relationships between entities, researchers usually construct knowledge graphs (KGs) (Ji et al., 2021; Hogan et al., 2021), where nodes are entities, e.g., machine learning, and edges are relations, e.g., subclass of (Figure 2). However, KGs usually require a pre-specified set of relation types, and the covered relation types are usually coarse-grained and simple. This indicates existing KGs lack two desired features. The first is openness: for entities with a relationship not covered by the type set, KGs cannot handle their relationship directly. Besides, in many cases, the relationship between entities is complex or idiosyncratic that it cannot be simply categorized to a relation type (usually denoted by a short phrase). For instance, for related entities machine learning and algorithm, Wikidata (Vrandečić and Krötzsch, 2014) does not include a relation for them, and it is also not easy to come up with a relation type to describe their relationship.

The second feature is about interpretability. With the relational facts in KGs, humans may still have difficulty in understanding entity relationships. For instance, from fact “(data mining, facet of, database)” in Wikidata, humans may guess data mining and database are related fields, but they cannot understand how exactly they are related. Although techniques like knowledge graph reasoning (Lao et al., 2011; Xiong et al., 2017; Chen et al., 2018) or open relation extraction (Etzioni et al., 2008) can represent more complex relationships to some extent, they do not fundamentally solve the limitations (Huang et al., 2022). For instance, neither a multi-hop reasoning path in KGs nor a triple extracted by open relation extraction, e.g., (data mining methods, to be integrate within, the framework of traditional database systems), is easy to interpret.

Based on the above analysis, we propose a new form of modeling relationships between entities: Descriptive Knowledge Graph (DKG). We define a DKG as a graph, where nodes are entities and edges are descriptive statements of entity relationships (refer to Figure 1 for an example). DKGs have great potential to help users understand entity relationships more easily and intuitively by providing relation descriptions for any two related entities (including those not present in traditional KGs) and facilitate downstream tasks on entities and entity relationships such as definition modeling.

Asterisk indicates equal contribution.
Figure 1: Descriptive Knowledge Graph. Here we show machine learning and several of its related entities, with corresponding relation descriptions extracted by our system in the edges.

Figure 2: Prescriptive Knowledge Graph (Wikidata), where ` denotes the inverted direction and ` means the relation is not present in the graph.

(Noraset et al., 2017), relation extraction (Bach and Badaskar, 2007), and knowledge graph completion (Lin et al., 2015).

Compared to traditional KGs, what we refer to as Prescriptive Knowledge Graphs (PKGs) later, DKGs are 1) more open: a DKG does not require a pre-specified set of relation types like a PKG. In principle, all entity relationships, either explicit or implicit, can be represented by DKG, as long as they can be connected in a sentence – which is not possible for PKGs. For instance, for machine learning and algorithm, the relationship between them can be represented by “Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.”; 2) more interpretable: the relationships between entities are represented by interpretable sentence relation descriptions, instead of a simple short phrase like “facet of”.

It should be noted that we do not emphasize that DKGs are superior to PKGs. In fact, they can complement each other: DKGs provide a more open and interpretable (i.e., descriptive) way to understand entity relationships, while PKGs help understand relationships from a more normative and structural (i.e., prescriptive) perspective.

The key to building a DKG is to acquire high-quality relation descriptions. However, writing or collecting relation descriptions manually requires enormous human efforts and expertise (in our human evaluation in Section 6.1, it takes ~3 minutes to evaluate whether a sentence is a good relation description). Considering this, we propose a novel two-step approach to construct a DKG with Wikipedia, where no manual annotation is required. Specifically, we first extract relation descriptions from corpus in a self-supervised manner, where a scoring function is introduced to measure the explicitness, i.e., how explicit is the relationship represented by the sentence, and significance, i.e., how significant is the relationship represented, with the analysis of dependency patterns. Second, a transformer-based relation description synthesizing model is introduced to generate relation descriptions for interesting entity pairs whose relation descriptions are not extracted in the first step.

Both quantitative and qualitative experiments, along with ablation study, demonstrate the effectiveness of our proposed methods. We also conduct case study and error analysis and suggest several promising directions for future work.

2 Related Work

There are several previous attempts on acquiring entity relation descriptions. For instance, Voskarides
et al. (2015) study a learning to rank problem of ranking relation descriptions by training a Random Forest classifier with manually annotated data. Subsequently, Huang et al. (2017) build a pairwise ranking model based on convolutional neural networks by leveraging query-title pairs derived from clickthrough data of a Web search engine, and Voskarides et al. (2017) attempt to generate descriptions for relationship instances in KGs by filling created sentence templates with appropriate entities. However, all these methods are not “open”. First, they rely and demand heavily on features of entities and relations. Second, these models only deal with entities with several pre-specified relation types, e.g., 9 in (Voskarides et al., 2015) and 10 in (Voskarides et al., 2017), and only explicit relation types, e.g., isMemberOfMusicGroup, are covered. Besides, these works do not systematically define what constitutes a good relational description.

The work most relevant to ours is Open Relation Modeling (Huang et al., 2022), which aims to generate relation descriptions for entity pairs. To achieve this, the authors propose to fine-tune BART (Lewis et al., 2020) to reproduce definitions of entities. Compared to their problem, i.e., text generation, the focus of this paper is on graph construction. Besides, their relation descriptions are limited to definitional sentences, which assumes that one entity appears in the other’s definition; however, the assumption is not true for many related entities. In addition, their methodology does not incorporate sufficient knowledge about entities and relations for generation.

There are also some less relevant works that can be related. For example, Lin et al. (2020); Liu et al. (2021) study CommonGen, which aims to generate coherent sentences containing the given common concepts. Dognin et al. (2020); Agarwal et al. (2021) study the data-to-text generation (Kukich, 1983), which aims to convert facts in KGs into natural language. None of these works meet the requirements on openness and interpretability.

3 Descriptive Knowledge Graph

A Descriptive Knowledge Graph (DKG) is a graph that represents relationships between entities with sentence descriptions. Formally, we define a DKG as \( G = (E, R) \), where \( E \) is the set of entities and \( R \) is the set of relation description facts. A relation description fact is denoted as a triple \( (x, s, y) \), where \( x \in E \) and \( y \in E \). \( s \) is a sentence describing the relationship between \( x \) and \( y \) (Figure 1).

To build a DKG, the first step is to collect entities and identify related entity pairs, which can be simply achieved by utilizing existing resources, e.g., Wikipedia, and entity relevance analysis, e.g., cosine similarity of entity embeddings in Wikipedia2vec (Yamada et al., 2020). And then, we need to acquire high-quality relation descriptions for entity pairs. Taking entity pair (machine learning, algorithm) as an example, a relation description of them can be \( s_1 \) in Table 1. From the perspective of human understanding, we identify three requirements for a good relation description:

• **Explicitness**: The relationship of the target entities is described explicitly. E.g., in \( s_1 \), “machine learning explores the study and construction of algorithms” describes the relationship explicitly; while in \( s_2 \), the relationship between machine learning and algorithm is expressed implicitly so that the relationship is difficult to reason.

• **Significance**: The relationship of the target entities is the point of the sentence. In \( s_1 \), all the tokens in the sentence are associated with the relationship between machine learning and algorithm; while in \( s_3 \), although the description is explicit, “many ... far” only characterizes algorithm, but not the target entity relationship.

• **Correctness**: The relationship between target entities is described correctly.

There are other requirements to ensure a good relation description, e.g., the sentence is coherent, grammatical, of reasonable length. Compared to the above ones, these requirements are general requirements for any sentence, but not specific to our problem; therefore, we put less emphasis on them.

To acquire relation descriptions that satisfy the above requirements, we propose a novel two-step approach: first extracting relation descriptions from a corpus in a self-supervised manner (Section 4), and then generating relation descriptions for interesting entity pairs whose relation descriptions are not extracted in the previous step (Section 5).

| # | Sentence |
|---|---------|
| 1 | Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. |
| 2 | Machine learning is employed in a range of computing tasks where designing and programming explicit, rule-based algorithms is infeasible. |
| 3 | In machine learning and optimization, many algorithms are adaptive or have adaptive variants, which usually means that the algorithm parameters are automatically adjusted according to statistics about the optimisation thus far. |

Table 1: Example sentences containing both machine learning and algorithm.
4 Relation Description Extraction

In this section, we introduce a self-supervised learning method for extracting entity relation descriptions from Wikipedia according to the requirements discussed in Section 3.

4.1 Preprocessing and Filtering

The goal of preprocessing and filtering is to collect entities and map entity pairs to candidate relation descriptions. To ensure correctness, we use Wikipedia as the source corpus, which is a high-quality corpus covering a wide range of domains. Because this process mainly relies on heuristic rules and existing tools, to save space, we refer the readers to Appendix A for the details.

4.2 Scoring

In this section, we design a scoring function to measure the quality of relation descriptions. Since we use Wikipedia as the source corpus, the correctness of the extracted sentences can be largely guaranteed; thus, we focus on measuring explicitness and significance of candidate relation descriptions.

4.2.1 Shortest Dependency Path as Relation

Inspired by Wu and Weld (2010), we use the shortest dependency path to represent the relation pattern between the target entities in a sentence. For instance, Figure 3 shows the dependency tree of sentence $s_1$ processed by spaCy. The shortest path between $\text{machine learning}$ and $\text{algorithm}$ is: “learning $\text{nsubj}$ explores $\text{dobj}$ study $\text{prep}$ of $\text{pobj}$ algorithms”. Following their notation, we call such a path a corePath. To represent the relation pattern, we collect dependencies in the path and append “i_” to the dependencies with an inverted direction. E.g., the relation pattern for the above path is $[i_\text{nsubj}, \text{dobj}, \text{prep}, \text{pobj}]$. We remove dependencies that do not affect human understanding. Specifically, we drop the $\text{conj}$ and $\text{appos}$ dependencies and replace two consecutive $\text{prep}$ with one.

Besides corePath, we also collect the shortest paths between the corePath and the tokens outside the corePath to represent the relationships between entity relationships and tokens. For instance, in Figure 3, construction is a token outside the corePath between $\text{machine learning}$ and $\text{algorithm}$. The shortest path between it and the corePath is: “study $\text{conj}$ construction”. We call this kind of path as subPath. Similar to corePath, we generate the relation pattern from subPath and drop the $\text{conj}$, $\text{appos}$ and $\text{compound}$ dependencies.

4.2.2 Explicitness

Given two entities and a candidate relation description $s$, we measure the explicitness by calculating the normalized logarithmic frequency of the relation pattern of the corePath:

$$\text{ExpScore}(s) = \frac{\log(f_p + 1)}{\log(f_{\text{max}} + 1)},$$

where $f_{\text{max}}$ is the frequency of the most frequent corePath relation pattern and $f_p$ is the frequency of the relation pattern in the present corePath. The intuition here is that humans tend to use explicit structure to explain relations. Thus, we assume that a relation description is more explicit if its relation pattern is more frequent. Intuitively, if a relation pattern is unpopular, it is likely that this pattern is either too complicated or contains some rarely used dependencies. Both of these cases may increase the difficulty in reasoning.

Similar to (Wu and Weld, 2010), we only consider patterns that start with $\text{nsubj}$ or $\text{nsubjpass}$, indicating that one of the target entities is the subject of the sentence. This restriction helps increase the explicitness of the selected relation description sentences because if one entity is the subject, the sentence is likely to contain a “argument-predicate-argument” structure connecting the target entities.

4.2.3 Significance

We measure the significance as the proportion of information that is relevant to the entity relationship in a sentence. To measure the relevance of each token in the sentence to the entity relationship, we divide tokens into three categories: 1) core token if the token is in the corePath; 2) modifying token if the token is in a subPath that is connected to the corePath through a modifying dependency; and 3) irrelevant token for the rest tokens. The intuition here is that a sub-dependency tree connected to the corePath with a modifying dependency is supposed to modify the relationship. We predefined a set of modifying dependencies in Table 7.

We calculate a score for each token in the sentence based on its category and dependency analysis. Then, the significance score is the average of all the token’s scores. Formally, for a candidate relation description $s$, the significance score is calculated as

$$\text{SigScore}(s) = \frac{\sum_{t \in s} w(t)}{|s|},$$
where

\[ w(t) = \begin{cases} 
1 & \text{if } t \in ct \\
\frac{\log(f_{pt}^{t} + 1)}{\log(f_{pt}^{t_{max}} + 1)} & \text{if } t \in mt \\
0 & \text{otherwise}
\end{cases} \quad (3) \]

where \( ct \) is the set of core tokens and \( mt \) is the set of modifying tokens. \( f_{pt}^{t} \) is the frequency of the subPath relation pattern from the corePath to the present token \( t \) and \( f_{pt}^{t_{max}} \) is the frequency of the most frequent subPath relation pattern. The intuition is: with higher relation pattern frequency, the modifying token is more explicitly related to the entity relationship, and thus, should have a higher score. This also comes with another useful characteristic: the score will decrease token by token as we move along the subPath because the frequency of a subPath relation pattern cannot be greater than the frequency of its parent. With this characteristic, we can penalize the long modifying subPath as it will distract the focus from the entity relationship and is less explicitly related to the relationship.

4.2.4 Relation Descriptive Score

To calculate the explicitness and significance, we need to build a database of relation patterns for both corePath and subPath. We construct both databases with the candidate relation descriptions and corresponding entity pairs collected from Section 4.1 with spaCy. We also require the two target entities in the sentence are highly related. Intuitively, if two entities are more related, the sentences containing them are more likely to be relation descriptions; therefore, the extracted corePath relation patterns are more likely to indicate entity relationships. We measure the relevance of two entities by calculating the cosine similarity of the entity embeddings in Wikipedia2Vec. We filter out entity pairs (and the associated sentences) with a relevance score < 0.5.

With the databases of relation patterns, we can calculate the explicitness and significance scores for a candidate relation description. The final score, named Relation Descriptive Score (RDScore), is computed as the harmonic mean:

\[
\text{RDScore}(s) = \frac{\text{ExpScore}(s) \cdot \text{SigScore}(s)}{\text{ExpScore}(s) + \text{SigScore}(s)},
\]

For each entity pair, we calculate RDScore for all the candidate relation descriptions and select the candidate with the highest score as the final relation description. To build an initial DKG, we keep edges with an entity relevance score \( \geq 0.5^{4} \) and with a relation description whose RDScore \( \geq 0.6^{5} \). We refer to this graph as \( \text{Wiki-DKG}_0 \).

5 Relation Description Generation

In the previous section, we extract relation descriptions for entity pairs in a self-supervised manner and build an initial DKG with Wikipedia automatically. However, for some related entity pairs, there may not exist a sentence that contains both entities; and although such a sentence exists, it may not be extracted by the system. To solve this problem, in this section, we introduce Relation Description Generation – generating relation descriptions for interesting entity pairs.

We form relation description generation as a conditional text generation task: given two entities, generating a sentence describing the relationship between them with the initial DKG. Formally, we apply the knowledge-enhanced sequence-to-sequence formulation (Yu et al., 2020): given an entity pair \((x, y)\) and an initial DKG \(G_0\), the probability of the output relation description \(s\) is computed auto-regressively:

\[
P(s \mid x, y, G_0) = \prod_{i=1}^{m} P(s_i \mid s_{0:i-1}, x, y, G_0), \quad (5)
\]

where \( m \) is the length of \( s \), \( s_i \) is the \( i \)-th token of \( s \), and \( s_0 \) is a special start token.

To incorporate the DKG for generation, we propose Relation Description Synthesizing (RelationSyn). RelationSyn consists of two processes: first retrieving relevant relation descriptions (reasoning paths) from the graph and then synthesizing them into a final relation description (Figure 4).

5.1 Retrieval

To generate a relation description, the model needs knowledge about the target entities and their relations.

---

4Since there is no boundary that delineates whether two entities are related, we consider the relevance threshold as a hyperparameter.

5This threshold is also a hyperparameter to balance the density of the graph and the quality of relation descriptions.
In DKG, we define a reasoning path \( q \) as a path connecting the target entities, which is called \( k \)-hop if it is connected by \( k \) edges. For instance, in Figure 4, there are two 2-hop reasoning paths between \( x \) and \( y \): \((x, s_{11}, e_{11}, s_{12}, y)\) and \((x, s_{21}, e_{21}, s_{22}, y)\), and two 3-hop reasoning paths: \((x, s_{21}, e_{21}, s_{23}, e_{32}, s_{33}, y)\) and \((x, s_{31}, e_{31}, s_{32}, e_{32}, s_{33}, y)\) in the DKG. To measure the quality of reasoning paths, we define PathScore as the harmonic mean of RDScore of relation descriptions in the path:

\[
\text{PathScore}(q) = \frac{|S_q|}{\sum_{s \in S_q} \text{RDScore}(s)}, \quad (6)
\]

where \( S_q \) is the set of relation descriptions in \( q \), and \(|S_q| = k\).

Reasoning paths are helpful for relation description generation. For instance, from reasoning path (deep learning, \( s'_{1} \), machine learning, \( s'_{2} \), computer science) (refer to Figure 1 for \( s'_{1} \) and \( s'_{2} \)), we can infer the relationship between deep learning and computer science: deep learning is a subset of ML, while ML is a subset of CS; therefore, deep learning is a subset of CS. Besides, from the sentence description, we can acquire rich entity information about deep learning and computer science, e.g., deep learning focuses heavily on the use of artificial intelligence.

However, not all reasoning paths are equally useful. Longer reasoning paths are usually more difficult to reason, while paths with higher PathScore usually contain more explicit and significant relation descriptions. Therefore, when retrieving reasoning paths for an entity pair, we first sort the paths by their length (shorter first) and then by their PathScore (higher first).

5.2 Synthesizing

According to Section 5.1, we may retrieve multiple reasoning paths for an entity pair whose relation description is missed in the initial DKG. In this section, we focus on synthesizing relation descriptions in the retrieved reasoning paths into a final relation description of the target entities based on T5 (Rafael et al., 2020) and Fusion-in-Decoder (Izacard and Grave, 2021).

We first convert each reasoning path to a sequence using the following encoding scheme: \((x, s_{31}, e_{31}, s_{32}, e_{32}, s_{33}, y) \rightarrow \text{“entity1: } x \text{ entity2: } y \text{ path: } x; e_{31}; e_{32}; y \text{ sentence1: } s_{31} \text{ sentence2: } s_{32} \text{ sentence3: } s_{33} \text{”}. And then, we encode the sequence with the encoder of T5. In this way, the relation descriptions in each reasoning path are synthesized into a latent vector, named \text{“local synthesizing”}.

After local synthesizing, we concatenate the latent vectors of all the retrieved reasoning paths to form a global latent vector. The decoder of T5 performs attention over the global latent vector and produces the final relation description. We name this process as \text{“global synthesizing”}.

Combining retrieval and synthesizing, given two entities, we first retrieve \( m \) reasoning paths connecting the target entities according to their length and PathScore, and then synthesize them to produce the target relation description. We refer to this model as RelationSyn-\( m \).

6 Evaluation

In this section, we verify the proposed methods for building DKGs by conducting experiments on relation description extraction and generation.

6.1 Relation Description Extraction

We first present the statistics of the initial DKG built with Wikipedia in Table 2.
Table 2: The statistics of Wiki-DKG

| # nodes | # edges | average sentence length |
|---------|---------|-------------------------|
| 1,294,502 | 2,503,695 | 19.6 |

Table 3: Qualitative results of extraction.

| Rating (1-5) |  |
|-------------|---|
| Random      | 2.84 |
| ExpScore    | 3.80 |
| SigScore    | 3.92 |
| RDScore     | 4.20 |

Table 4: The statistics of data for generation.

|                          | train | valid | test |
|--------------------------|-------|-------|------|
| size                     | 672,930 | 14,019 | 14,020 |

6.2.1 Experimental Setup

Data construction. We build the dataset for relation description generation as follows: for an entity pair with a relation description in Wiki-DKG, we hide the relation description and consider it as the target for generation. The goal is to recover/generate the target relation description with the rest of the graph\(^7\). For instance, in Figure 4, we hide the edge (relation description \(s\)) between \(x\) and \(y\) and use the remaining reasoning paths to recover \(s\). We train and test on entity pairs with \(\geq 5\) reasoning paths connecting them. The statistics of the data are reported in Table 4.

Models. The task of relation description generation is relevant to Open Relation Modeling (Huang et al., 2022) – a recent work aimed at generating sentences capturing general relations between entities conditioned on entity pairs. To the best of our knowledge, no other existing work can generate relation descriptions for any two related entities (since open relation modeling has only just been introduced). Therefore, we mainly compare the models proposed in (Huang et al., 2022) with several variants of our model:

- **RelationBART (Vanilla)**: The vanilla model proposed in (Huang et al., 2022) for generating entity relation descriptions, where BART (Lewis et al., 2020) is fine-tuned on a training data whose inputs are entity pairs and outputs are corresponding relation descriptions.
- **RelationBART-MP + PS**: The best model proposed in (Huang et al., 2022), which incorporates Wikidata by selecting the most interpretable and informative reasoning path in the KG automatically for helping generate relation descriptions.
- **RelationSyn-0**: A reduced variant of our model, where the encoding scheme of the input is only \(\text{entity1}: x \text{ entity2}: y\), i.e., no reasoning path and relation description is input to the encoder.
- **RelationSyn-\(m\)**: The proposed relation description synthesizing model (Section 5), where \(m\) is the maximum number of retrieved reasoning paths for an entity pair.

6.2 Relation Description Generation

In this section, we perform experiments on relation description generation.

\(^6\)More specifically, for better comparison with generation later, we sample 100 entity pairs from the test set in Table 4.

\(^7\)To increase the difficulty of the task, we assume these two entities do not co-occur in the corpus, i.e., we do not utilize any sentence containing both the target entities for generation.
### Metrics

We perform both quantitative and qualitative evaluation. Following (Huang et al., 2022), we apply several automatic metrics for, including BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang et al., 2019). Among them, BLEU, ROUGE, and METEOR focus on measuring surface similarities between the generated relation descriptions and the target relation descriptions, and BERTScore is based on the similarities of contextual token embeddings. We also ask three human annotators to evaluate the output relation descriptions with the same rating scale in Table 8.8

### 6.2.2 Quantitative Evaluation

Table 5 reports the results of relation description generation with the automatic metrics. We observe that our best model RelationSyn-5 outperforms the state-of-the-art model for open relation modeling significantly. We also observe that RelationSyn-1 performs better than RelationSyn-0, which means that reasoning paths in DKG are helpful for relation description generation. In addition, as the number of reasoning paths, i.e., $m$, increases, the performance of RelationSyn-$m$ improves. This demonstrates that the proposed model can synthesize multiple relation descriptions in different reasoning paths into a final relation description.

### 6.2.3 Qualitative Evaluation

We also conduct qualitative experiments to measure the quality of generated relation descriptions. For a better comparison with extraction, we sample the same 100 entity pairs from the test set as in Section 6.1. From the results in Table 6, we observe that the quality of generated relation descriptions is higher than that of random sentences containing the target entities. The best model, RelationSyn-5, achieves a rating of 3.48, which means the model can generate reasonable relation descriptions. However, the performance is still much worse than Oracle, i.e., relation descriptions extracted by our best extraction model (RDScore). This indicates that generating high-quality relation descriptions is still a challenging task.

### 6.3 Ablation Study, Case Study and Error Analysis

We provide ablation study in Appendix C and case study and error analysis in Appendix D.

### 7 Conclusion and Discussion

In this paper, we propose DKG – an open and interpretable form of modeling relationships between entities. To avoid tremendous human efforts, we design a novel self-supervised learning approach to extract relation descriptions from Wikipedia. To provide relation descriptions for related entity pairs whose relation descriptions are not extracted in the previous step, we study relation description generation by synthesizing relation descriptions in the retrieved reasoning paths. Experiments on both extraction and generation demonstrate that our system can produce high-quality relation descriptions for building DKGs.

There are several promising directions for future work. First, in this paper, DKGs are modeled as undirected graphs, it is interesting to explore the direction of the edge and the diversity of relation descriptions. Second, we build a DKG with Wikipedia, future work may explore scaling up the graph and constructing domain-specific DKGs with other corpora. Third, DKGs can be utilized as knowledge sources to facilitate downstream applications and related tasks such as relation extraction (Bach and Badaskar, 2007) and knowledge graph completion (Lin et al., 2015).

### Table 5: Quantitative results of relation description generation.

| Method                  | BLEU  | ROUGE | METEOR | BERTScore |
|-------------------------|-------|-------|--------|-----------|
| RelationBART-Vanilla     | 20.36 | 40.95 | 20.97  | 82.27     |
| RelationBART-MP + PS    | 20.97 | 41.22 | 21.25  | 82.25     |
| RelationSyn-0           | 21.50 | 40.35 | 21.08  | 82.68     |
| RelationSyn-1           | 23.61 | 42.16 | 22.28  | 83.45     |
| RelationSyn-3           | 24.12 | 43.00 | 22.67  | 83.77     |
| RelationSyn-5           | 24.66 | 43.25 | 22.93  | 83.79     |

Table 6: Qualitative results of generation.

| Method                  | Rating (1-5) |
|-------------------------|--------------|
| Random                  | 2.84         |
| RDScore (Oracle)        | 4.20         |
| RelationBART-MP + PS    | 3.12         |
| RelationSyn-0           | 3.10         |
| RelationSyn-1           | 3.36         |
| RelationSyn-5           | 3.48         |

8Details about implementation are in Appendix B.
References

Oshin Agarwal, Heming Ge, Siyamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3554–3565.

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. Literature review for Language and Statistics II, 2:1–15.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pages 65–72.

Wenhua Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1823–1832.

Pierre Dognin, Igor Melnyk, Inkit Padhi, Cicero dos Santos, and Payel Das. 2020. Dualtkb: A dual learning bridge between text and knowledge base. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 8605–8616.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. Communications of the ACM, 51(12):68–74.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. Synthesis Lectures on Data, Semantics, and Knowledge, 12(2):1–257.

Jie Huang, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022. Open relation modeling: Learning to define relations between entities. In Findings of the Association for Computational Linguistics: ACL 2022.

Jizhou Huang, Wei Zhang, Shiqi Zhao, Shiqiang Ding, and Haifeng Wang. 2017. Learning to explain entity relationships by pairwise ranking with convolutional neural networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 4018–4025.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics, pages 874–880. Association for Computational Linguistics.

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.

Karen Kukich. 1983. Design of a knowledge-based report generator. In 21st Annual Meeting of the Association for Computational Linguistics, pages 145–150.

Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In Proceedings of the 2011 conference on empirical methods in natural language processing, pages 529–539.

Mike Lewis, Yinlin Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. Commongen: A constrained text generation challenge for generative commonsense reasoning. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 1823–1840.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pages 74–81.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In Twenty-ninth AAAI conference on artificial intelligence.

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

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pages 311–318.

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*, 21:1–67.

Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 2017. Generating descriptions of entity relationships. In *European Conference on Information Retrieval*, pages 317–330. Springer.

Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. 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)*, pages 564–574.

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

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*. 
### Table 7: Manually collected modifying dependencies in spaCy.

| Dependency label | Description |
|------------------|-------------|
| acl              | clausal modifier of noun (adjectival clause) |
| advcl            | adverbial clause modifier |
| advmod           | adverbial modifier |
| amod             | adjectival modifier |
| det              | determiner |
| mark             | marker |
| meta             | meta modifier |
| neg              | negation modifier |
| nmod             | modifier of nominal |
| npmod            | noun phrase as adverbial modifier |
| nummod           | numeric modifier |
| poss             | possession modifier |
| prep             | prepositional modifier |
| quantmod         | modifier of quantifier |
| relcl            | relative clause modifier |
| appos            | appositional modifier |
| aux              | auxiliary |
| auxpass          | auxiliary (passive) |
| compound         | compound |
| coop             | copula |
| ccomp            | clausal complement |
| xcomp            | open clausal complement |
| expl             | expletive |
| punct            | punctuation |
| nsubj            | nominal subject |
| csbj             | clausal subject |
| csubjpass        | clausal subject (passive) |
| dobj             | direct object |
| iobj             | indirect object |
| obj              | object |
| poly             | object of preposition |

### Table 8: Annotation guidelines excerpt.

| Rating | Criterion |
|--------|-----------|
| 5      | The relation description is explicit, significant, and correct, with which users can understand the relationship correctly and easily. |
| 4      | The relation description is a bit less explicit (reasoning is a bit indirect or description is a bit unclear), less significant (containing a little irrelevant content), and less correct (containing minor errors that do not affect the understanding). |
| 3      | The relation description is fairly explicit, significant, and correct, while users can still understand the relationship. |
| 2      | The relation description is not explicit (reasoning is difficult or description is unclear), significant (containing much irrelevant content), or correct (containing major errors that affect the understanding), while users can still infer the relationship to some extent. |
| 1      | The relation description is completely wrong or does not show any relationship between the two entities. |

### A Preprocessing and Filtering

We introduce our preprocessing to the raw Wikipedia dump\(^9\). For each article, we extract the plain text by WikiExtractor\(^10\). We split the Wikipedia articles into sentences with the NLTK library\(^11\) and map entity pairs to candidate relation descriptions with the following steps:

**Entity collection.** We collect Wikipedia page titles (surface form) as our entities. To acquire knowledge and utilize the pre-trained entity embeddings in Wikipedia2Vec (Yamada et al., 2020) in the later steps, we only keep entities that can be recognized by Wikipedia2Vec.

**Local mention-entity mapping.** Wikipedia2Vec uses hyperlinks to collect a global mention-entity dictionary to map the entity mention to the referent entities, like mapping “algorithm” and “graph” wrapped by parenthesis and the content after the first comma.

\(^9\)https://dumps.wikimedia.org (version: enwiki/20210320)

\(^10\)https://github.com/attardi/wikiextractor

\(^11\)https://www.nltk.org

Hyperlink mapping correction. Using hyperlinks to collect entities will lead to errors under some conditions: 1) The original link is redirected to a new page, where the title does not match with the entity in the link; 2) The entity in the link is lower-cased and thus, does not match with any title. Under the first condition, we just skip this entity because we require that the entity mention must appear in the sentence to prove its occurrence. Under the second situation, if there is only one page title matching with the entity under the case-insensitive setting, we correct the entity to this page title. Otherwise, if there is more than one match, we use the entity embeddings in Wikipedia2Vec to measure the cosine similarity between each matched title and the title of the current page and correct the entity with the most relevant one.

Filtering. Sometimes the entity mention extracted from the sentence may be part of a bigger noun phrase, which is not an entity mention. For example, suppose we recognize “algorithm” and “graph”
as entity mentions in the sentence “The breadth-first-search algorithm is a way to explore the vertices of a graph layer by layer.” However, this is not a good relation description between “algorithm” and “graph” because the subject is “breadth-first-search algorithm” rather than “algorithm”. Therefore, it is necessary to determine the completed noun phrase for each entity mention. With the dependency tree of the sentence, we recursively find all the child tokens and the ancestor tokens that are connected to the entity mention with a compound dependency. To avoid any confusion, we simply reject the entity occurrence if its completed noun phrase and entity mention are different.

Besides, to ensure that the length of relation descriptions is reasonable, we only remain the sentences with the number of tokens $e \in [5, 50]$. We also only remain sentences whose shortest dependency path pattern between two target entities starts with $nsubj$ or $nsubjpass$ (more details are in Section 4.2.2).

## B Implementation Details

We train and evaluate all the baselines and variants on the same train/valid/test split. For RelationBART (Vanilla) and RelationBART-MP + PS, we apply the official implementation\(^\text{12}\) and adopt the default hyperparameters. The training converges in 50 epochs. For our models, we modify the implementation of Fusion-in-Decoder\(^\text{13}\) and initialize the model with the T5-base configuration. All the baseline models for RelationSyn are trained under the same batch size of 8 with a learning rate of 0.0001 and evaluated on the validation set every 5000 steps. The training is considered converged and terminated with no better performance on the validation set in 20 evaluations. The training of all models converges in 20 epochs. The training time is about one week on a single NVIDIA A40 GPU.

## C Ablation Study

To further validate the proposed methods, we conduct an ablation study with the following variants:
- **RelationSyn-1 (w/o sentence)**: a variant of RelationSyn-1, where the encoding scheme of the input is only “entity1: x entity2: y path: x; $e_{31}$; $e_{32}$; $y'$”, i.e., no relation description is input to the encoder.
- **RelationSyn-1 (w/ Random)**: The input relation descriptions to RelationSyn are extracted by Random (i.e., a random sentence containing the target entities), instead of RDScore.
- **RelationSyn-1 (w/ ExpScore)**: The input relation descriptions to RelationSyn are extracted by ExpScore.
- **RelationSyn-1 (w/ SigScore)**: The input relation descriptions to RelationSyn are extracted by SigScore.

From the results in Table 9, we observe that RelationSyn-1 achieves better results than all the reduced variants. However, the difference is not quite significant. We think this is because, although relation descriptions extracted by Random, ExpScore and SigScore are worse than those extracted by RDScore, they can still provide reasonable knowledge for generating relation descriptions for the target entities. In addition, the results are also consistent with the results in Table 6. This indirectly verifies that the combination of explicitness and significance, i.e., RDScore, can extract better relation descriptions, since better relation descriptions of related entities are expected to be synthesized into better relation descriptions of the target entities.

## D Case Study and Error Analysis

In Table 10, we show some sample outputs in the test set of relation description generation of three extraction models: ExpScore, SigScore, RDScore, and three generation models: RelationSyn-0, RelationSyn-1, RelationSyn-5.

From the results, we observe that if we only consider the explicitness of the sentence, the se-

| | BLEU | ROUGE | METEOR | BERTScore |
|---|---|---|---|---|
| RelationSyn-0 | 21.50 | 40.35 | 21.08 | 82.68 |
| RelationSyn-1 (w/o sentence) | 22.69 | 41.26 | 21.69 | 83.03 |
| RelationSyn-1 (w/ Random) | 23.04 | 41.93 | 22.05 | 83.30 |
| RelationSyn-1 (w/ ExpScore) | 23.32 | 41.95 | 22.13 | 83.29 |
| RelationSyn-1 (w/ SigScore) | 23.31 | 42.14 | 22.18 | 83.39 |

Table 9: Ablation study for relation description generation.
|ExpScore| SigScore| RDScore| RelationSyn-0| RelationSyn-1| RelationSyn-5|
|--------|---------|--------|---------------|---------------|---------------|
| (Schizophrenia, Hallucination) | During the early 20th century, auditory hallucinations were second to visual hallucinations in frequency, but they are now the most common manifestation of schizophrenia, although rates vary between cultures and regions. | Auditory verbal hallucinations are often external, rather than internal, defining factor for the diagnosis of schizophrenia. | The hallucinations of the nocturnal syndrome are associated with schizophrenia. | Unlike most other forms of hallucination, schizophrenia does not cause a symptom change. | Schizophrenia is characterized by delusions, hallucinations, and other negative symptoms. |
| (Moth, Momphidae) | The Momphidae, or mompha moths, is a family of moths with some 115 described species. | Mompha terminula is a moth in the family Momphidae from Europe and North America. | The Momphidae are a family of moths with some 115 described species. | The Momphidae are a family of moths in the superfamily Gelechioidea. | The Momphidae are a family of moths in the superfamily Gelechioidea. |
| (Singapore, Indonesia) | China, Singapore, Malaysia, Australia, and Japan are the top five sources of visitors to Indonesia. | In 2007, Indonesia enacted a ban against exporting sand specifically to Singapore. | Indonesia is among the top source of foreign visitors to Singapore. | Singapore participated in the 2018 Asian Games in Jakarta and Palembang, Indonesia from 18 August to 2 September 2018. | Singapore is the second largest trading partner of Indonesia. |
| (Chota Nagpur Plateau, Giridih district) | Giridih district is a part of the Chota Nagpur plateau, with rocky soil and extensive forests. | Giridih district is a part of the Chota Nagpur plateau, with rocky soil and extensive forests. | Giridih district is located on the Chota Nagpur Plateau. | Giridih district is located on the Chota Nagpur Plateau. | Giridih district is spread over a part of the Chota Nagpur Plateau. |
| (Sun, Hipparchus) | Theon of Smyrna wrote that according to Hipparchus, the Sun is 1,880 times the size of the Earth, and the Earth twenty-seven times the size of the Moon; apparently this refers to volumes, not diameters. | Hipparchus already had developed a way to calculate the longitude of the Sun at any moment. | Both Aristarchus and Hipparchus drastically underestimated the distance of the Sun from the Earth. | Hipparchus, in his "History of the Sun", identifies the Sun with the Sun, a planet with a radius of. | Hipparchus, in his "History of the Sun", referred to the Sun as "the Sun of the sand". |

Table 10: Sample of relation descriptions produced by ExpScore, SigScore, RDScore, RelationSyn-0, RelationSyn-1, and RelationSyn-5.

Selected relation description may contain a lot of stuff that is irrelevant to the entity relationship, e.g., (Schizophrenia, Hallucination). And if we only consider the significance, the relationship between entities may be described implicitly; thus the relationship is difficult to reason out, e.g., (Singapore, Indonesia). And the combination of them, i.e., RDScore, yields better relation descriptions.

For generation, sometimes, the models generate similar relation descriptions, e.g., (Moth, Momphidae) and (Chota Nagpur Plateau, Giridih district); while in other cases the relation descriptions generated by the models are quite different. From the results, we notice that RelationSyn-0 suffers severely from hallucinations, i.e., generating irrelevant or contradicted facts. For instance, the relation descriptions generated for (Singapore, Indonesia) and (Sun, Hipparchus) are not correct or relevant. By incorporating relation descriptions in the reasoning paths as knowledge, hallucination is alleviated to some extent, leading to better performance of RelationSyn-1 and RelationSyn-5.

From the human evaluation results, we also find that the correctness of relation descriptions extracted by RDScore is largely guaranteed. However, sometimes, the extracted sentences are still a bit implicit or not significant. In contrast to this, the relation descriptions generated by RelationSyn are usually explicit and significant (the average RDScore of the relation descriptions generated by RelationSyn-5 is 0.83, compared to 0.80 of Oracle), but contain major or minor errors. We think this is because most of the relation descriptions extracted by RDScore are explicit and significant, and the generation model can mimic the dominant style of relation descriptions in the training set. However, it is still challenging to generate fully correct relation descriptions by synthesizing existing relation descriptions.

We also examine the five entity pairs in Table 10 in Wikidata. Among them, only the relation between Singapore and Indonesia is present in Wikidata. This further confirms that DKGs can model a wider range of entity relationships.