DEER: Descriptive Knowledge Graph for Explaining Entity Relationships

Jie Huang[*,1] **Kerui Zhu**[*,1] **Kevin Chen-Chuan Chang**[1]
Jinjun Xiong[2] **Wen-mei Hwu**[1,3]
[1]University of Illinois at Urbana-Champaign, USA
[2]University at Buffalo, USA
[3]NVIDIA, USA
{jeffhj, keruiz2, kcchang, w-hwu}@illinois.edu
jinjun@buffalo.edu

Abstract

We propose DEER (**D**escriptive Knowledge Graph for **E**xplaining **E**ntity **R**elationships) – an open and informative form of modeling entity relationships. In DEER, relationships between entities are represented by free-text relation descriptions. For instance, the relationship between entities of *machine learning* and *algorithm* can be represented as "*Machine learning* explores the study and construction of *algorithms* that can learn from and make predictions on data." To construct DEER, we propose a self-supervised learning method to extract relation descriptions with the analysis of dependency patterns and generate relation descriptions with a transformer-based relation description synthesizing model, where no human labeling is required. Experiments demonstrate that our system can extract and generate high-quality relation descriptions for explaining entity relationships. The results suggest that we can build an open and informative knowledge graph without human annotation.[1]

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. 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 ***informativeness***. 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, e.g, *why is it a facet?* and *what is the facet?*

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 as discussed in Huang et al. (2022a). 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: DEER (**D**escriptive Knowledge Graph for **E**xplaining **E**ntity **R**elationships). We define DEER as a graph, where nodes are entities and edges are descriptive statements of entity relationships (refer to Figure 1 for an example). DEER is ***open*** since it does not require a pre-specified set of relation types. In principle, all entity relationships, either explicit or implicit, can be represented by DEER, as long as they can be connected in a sentence – which is not possible for KGs. It is ***in-***

[1]Code and data are available at https://github.com/jeffhj/DEER. *Asterisk indicates equal contribution.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6686–6698
December 7-11, 2022

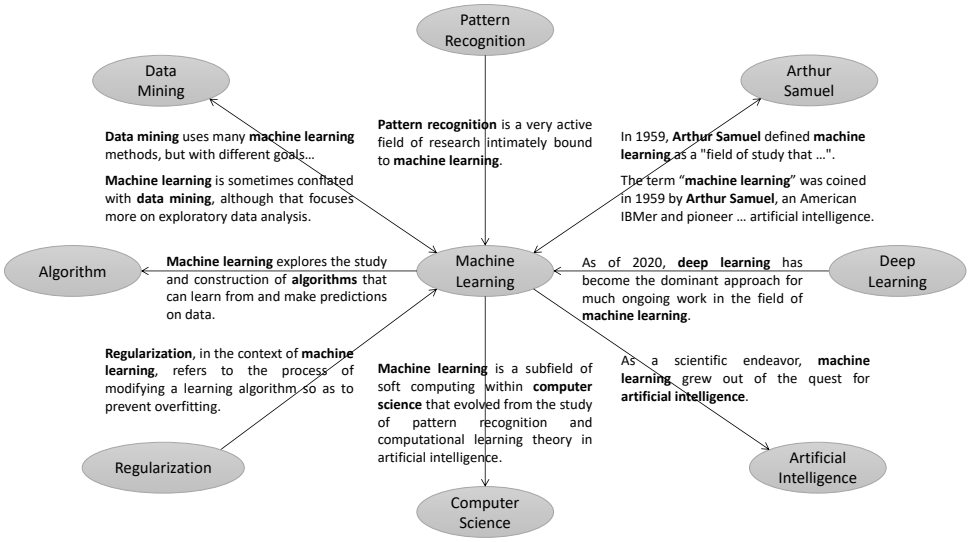

Figure 1: Relations in DEER. Here we show *machine learning* and several of its related entities, with corresponding relation descriptions produced by our model (only extraction) in the edges.

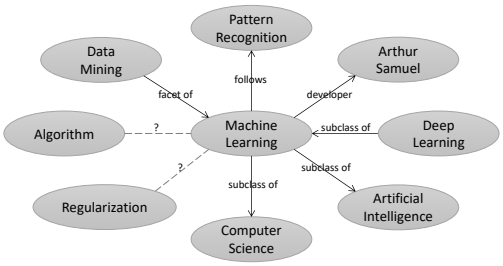

Figure 2: Relations in Wikidata (Knowledge Graph), where ? means the relation is not present in the graph.

formative since the relationships between entities are represented by informative free-text relation descriptions, instead of simple short phrases like "facet of".

DEER has great potential to help users understand entity relationships more easily and intuitively by providing relation descriptions for any two related entities and facilitate downstream tasks on entities and entity relationships such as entity profiling (Noraset et al., 2017; Cheng et al., 2020; Huang et al., 2022b), relation extraction (Bach and Badaskar, 2007), and knowledge graph completion (Lin et al., 2015). For example, in Figure 1, we can understand the semantic meaning of the terms by connecting them with familiar ones. In e-commerce, the system (e.g., Amazon online shopping website) may recommend *tripods* to a photography novice who is browsing *cameras*. An explanation in DEER, e.g., "*tripods* are used for both motion and still photography to prevent *camera* movement and provide stability", could not only help users make a better purchase decision but also justify the recommendation. In KG construction and completion, the relation descriptions can serve as knowledge to improve the performance or as explanations to justify the relations in KGs.

The key to building DEER 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 DEER 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, based on the extracted graph, 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. This allows DEER to handle a large number of entity pairs, including those that do not co-occur in the corpus.

Both quantitative and qualitative experiments demonstrate the effectiveness of our proposed methods. We also conduct case study and error analysis and suggest several promising directions for future work – DEER not only serves as a valuable application in itself to help understand entity relationships, but also has the potential to serve as a knowledge source to facilitate various tasks on entities and entity relationships.

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. Notably, Handler and O'Connor (2018) propose to extract relation statements, i.e., natural language expressions that begin with one entity and end with the other entity, from a corpus to describe entity relationships. However, the "*acceptability*" used in their work cannot ensure a good relation description. Moreover, these works do not systematically analyze and define what constitutes a good relational description.

The work most relevant to ours is *Open Relation Modeling* (Huang et al., 2022a), 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 other 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. Gunaratna et al. (2021) propose to construct an entity context graph with contexts as random paragraphs containing the target entities to help entity embedding. None of them meets the requirements for high-quality relation descriptions.

3 Descriptive Knowledge Graph for Explaining Entity Relationships

𝔻𝔼𝔼ℝ🦌 is a graph representing entity relationships with sentence descriptions. Formally, we define 𝔻𝔼𝔼ℝ as a directed graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relation description facts. A relation description fact is a triple (x, s, y), where $x, y \in \mathcal{E}$ are the *subject* and *object* of s, respectively. s is a sentence describing the relationship between x and y (Figure 1).

To build 𝔻𝔼𝔼ℝ, 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, "*which ... far*" mainly 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

#	Sentence
s_1	*Machine learning* explores the study and construction of *algorithms* that can learn from and make predictions on data.
s_2	*Machine learning* is employed in a range of computing tasks where designing and programming explicit, rule-based *algorithms* is infeasible.
s_3	*Machine learning* includes *algorithms* that 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*.

approach: first extracting relation descriptions from a corpus with the analysis of dependency patterns (Section 4), and then generating relation descriptions for interesting entity pairs whose relation descriptions are not extracted in the previous step (Section 5).

4 Relation Description Extraction

In this section, we introduce our approach 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 s_1 processed by spaCy[2]. The shortest path between *machine learning* and *algorithm* is: "learning $\overleftarrow{nsubj}$ explores $\overrightarrow{dobj}$ study $\overrightarrow{prep}$ of $\overrightarrow{pobj}$ 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 inversed direction. E.g., the relation pattern for the above path is $[i\_nsubj, dobj, prep, pobj]$. We remove dependencies that do not affect human understanding. Specifically, we drop the *conj* and *appos* dependencies and replace two consecutive *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 *machine learning* and *algorithm*. The shortest path between it and the *corePath* is: "study $\overrightarrow{conj}$ construction". We call this kind of path as *subPath*. Similar to *corePath*, we generate the relation pattern from *subPath* and drop the *conj*, *appos* and *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*:

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

where f_{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 *nsubj* or *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

[2] https://spacy.io

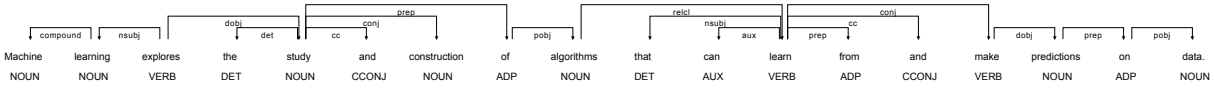

Figure 3: Dependency tree of s_1.

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

$$SigScore(s) = \frac{\sum_{t \in s} w(t)}{|s|}, \tag{2}$$

where

$$w(t) = \begin{cases} 1 & \text{if } t \in ct \\ \frac{\log(f'_{p_t}+1)}{\log(f'_{max}+1)} & \text{if } t \in mt \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where ct is the set of *core tokens* and mt is the set of *modifying tokens*. f'_{p_t} is the frequency of the *subPath* relation pattern from the *corePath* to the present token t and f'_{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 related to a certain threshold. 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. This leads to a collection of 7,186,996 *corePaths* and 83,265,285 *subPaths*.

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:

$$RDScore(s) = 2 \cdot \frac{ExpScore(s) \cdot SigScore(s)}{ExpScore(s) + SigScore(s)}. \tag{4}$$

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 $\mathbb{DEER}$, we keep edges with an entity relevance score ≥ 0.5[3] and with a relation description whose *RDScore* ≥ 0.75[4]. We refer to this graph as **Wiki-**$\mathbb{DEER}_0$.

5 Relation Description Generation

In the previous section, we extract relation descriptions for entity pairs with the analysis of dependency patterns and build an initial $\mathbb{DEER}$ 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 $\mathbb{DEER}$. For-

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

[4]This threshold is also a hyperparameter to balance the density of the graph and the quality of relation descriptions.

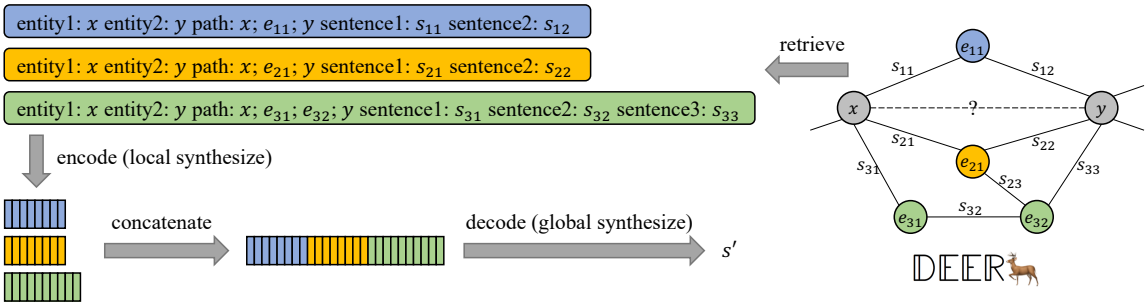

Figure 4: The framework of *RelationSyn*. Given entity pair (x, y) whose relation description is not present in the initial DEER, we first retrieve several reasoning paths from the graph. And then, we encode (local synthesize) each reasoning path into a latent vector and concatenate all the latent vectors. Finally, we decode (global synthesize) the vector to produce relation description s' for (x, y).

mally, we apply the knowledge-enhanced sequence-to-sequence formulation (Yu et al., 2020): given an entity pair (x, y) and an initial DEER $\mathcal{G}_0$, the probability of the output relation description s is computed auto-regressively:

$$P(s|x, y, \mathcal{G}_0) = \prod_{i=1}^{m} P(s_i|s_{0:i-1}, x, y, \mathcal{G}_0), \quad (5)$$

where m is the length of s, s_i is the ith token of s, and s_0 is a special start token.

To incorporate $\mathcal{G}_0$ 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 relationship. To provide knowledge, we retrieve reasoning paths of the target entities from the graph.

In DEER, 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 graph[5]. To measure the quality of reasoning paths, we define *PathScore* as the harmonic mean of *RDScore* of relation descriptions in the path:

$$PathScore(q) = \frac{|\mathcal{S}_q|}{\sum_{s \in \mathcal{S}_q} \frac{1}{RDScore(s)}}, \quad (6)$$

where $\mathcal{S}_q$ is the set of relation descriptions in q, and $|\mathcal{S}_q| = k$.

Reasoning paths are helpful for relation description generation. For instance, from reasoning path (deep learning, s'_1, machine learning, s'_2, artificial intelligence) (refer to Figure 1 for s'_1 and s'_2), we can infer the relationship between *deep learning* and *AI*: *deep learning* is the dominant approach for *ML*, while *ML* grew out of the quest for *AI*; therefore, *deep learning* is an important technology for the development 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 DEER. 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 (Raffel 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: e.g., $(x, s_{31}, e_{31}, s_{32}, e_{32}, s_{33}, y) \rightarrow$ "entity1: x entity2: y path: x; e_{31}; e_{32}; y sentence1: s_{31} sentence2: s_{32} sentence3: s_{33}". 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 "**local synthesizing**".

After local synthesizing, we concatenate the la-

[5]In order to collect more reasoning paths as knowledge for generation, we ignore the directions of edges.

# nodes	# edges	average sentence length
1,378,471	2,890,718	19.9

Table 2: The statistics of Wiki-$\mathbb{DEER}_0$.

tent 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 "**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 $\mathbb{DEER}$ by conducting experiments on relation description extraction and generation.

6.1 Relation Description Extraction

We first present the statistics of the initial $\mathbb{DEER}$ built with Wikipedia in Table 2.

To evaluate the quality of relation descriptions in the graph, we randomly sample 100 entity pairs from the graph[6] and ask three human annotators (graduate students doing research on computational linguistics) to assign a graded value (1-5) for each relation description according to Table 8.

Since previous works on relation description extraction are supervised and only limited to several explicit relation types, e.g., 9 in Voskarides et al. (2015), it is impractical and meaningless to compare with them. For instance, the relationship of (*Arthur Samuel*, *Machine Learning*) is not available or even not considered by the previous methods. Therefore, we verify the effectiveness of our model by comparing different variants of the model:

- **Random**: A sentence containing the target entities is randomly selected as the relation description.
- **ExpScore**: The sentence with the highest *explicitness* is selected according to Eq. (1).
- **SigScore**: The sentence with the highest *significance* is selected according to Eq. (2).
- **RDScore**: The sentence with the highest *RDScore* is selected according to Eq. (4).

[6]More specifically, for better comparison with generation later, we sample 100 entity pairs from the test set in Table 4.

	Rating (1-5)
Random	2.75
ExpScore	3.77
SigScore	3.84
RDScore	**4.18**

Table 3: Qualitative results of extraction.

	train	valid	test
size	847,792	17,662	17,663

Table 4: The statistics of data for generation.

Table 3 shows the human evaluation results for relation description extraction, with an average pairwise Cohen's κ of 0.66 (good agreement). From the results, we observe that both our explicitness and significance measurements are important to ensure a good relation description. In addition, *RDScore* achieves an average rating of 4.18, which means that most of the selected sentences are high-quality relation descriptions, further indicating that the quality of Wiki-$\mathbb{DEER}_0$ is high.

6.2 Relation Description Generation

6.2.1 Experimental Setup

Data construction. We build a dataset for relation description generation as follows: for an entity pair with a relation description in Wiki-$\mathbb{DEER}_0$, 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 ≥ 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., 2022a) – 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. (2022a) with several variants of our model:

[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.

	BLEU	ROUGE	METEOR	BERTScore
RealtionBART-Vanilla (Huang et al., 2022a)	19.61	41.52	20.48	82.99
RealtionBART-MP + PS (Huang et al., 2022a)	21.64	42.62	21.40	83.29
RelationSyn-0	20.83	41.46	20.66	82.84
RelationSyn-1	22.43	42.74	21.65	83.41
RelationSyn-3	23.26	43.33	22.12	83.63
RelationSyn-5	**23.88**	**43.56**	**22.40**	**83.70**

Table 5: Quantitative results of relation description generation.

- **RelationBART (Vanilla)**: The vanilla model proposed in Huang et al. (2022a) 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. (2022a), 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 "entity1: x entity2: y", i.e., no reasoning path and relation description is fed 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.

Metrics. We perform both quantitative and qualitative evaluation. Following Huang et al. (2022a), we apply several automatic metrics, 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.

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[8] and adopt the default hyperparameters. The training converges in 50 epochs. For our models, we modify the implementation of Fusion-in-Decoder[9] 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. For evaluation, the signature of BERTScore is: roberta-large-mnli L19 no-idf version=0.3.11(hug trans=4.15.0).

	Rating (1-5)
Random	2.75
RDScore (Oracle)	4.18
RealtionBART-MP + PS	3.12
RelationSyn-0	3.08
RelationSyn-1	3.34
RelationSyn-5	**3.47**

Table 6: Qualitative results of generation.

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 DEER 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 Sec-

[8] https://github.com/jeffhj/open-relation-modeling

[9] https://github.com/facebookresearch/FiD

tion 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 3.47, 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 Case Study and Error Analysis

In Table 9 of Appendix B, 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.

For extraction, we observe that if we only consider the explicitness of the sentence, the selected sentence may contain a lot of stuff that is irrelevant to the entity relationship, e.g., (*Mucus*, *Stomach*). 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., (*Surfers Paradise*, *Queensland*) and (*Knowledge*, *Epistemology*). And the combination of them, i.e., *RDScore*, yields better relation descriptions.

For generation, we notice that *RelationSyn*-0 suffers severely from *hallucinations*, i.e., generating irrelevant or contradicted facts. E.g., the relation descriptions generated for (*Dayan Khan*, *Oirats*) is incorrect. 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 0.886, compared to 0.853 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 attempted to find the eight entity pairs in Table 9 in Wikidata. Among them, only (*Surfers Paradise*, *Queensland*) is present in Wikidata. This further confirms that DEER can model a wider range of entity relationships.

7 Conclusion and Discussion

In this paper, we propose DEER – an open and informative 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. We believe that DEER can not only serve as a direct application to help understand entity relationships but also be utilized as a knowledge source to facilitate related tasks such as relation extraction (Bach and Badaskar, 2007) and knowledge graph completion (Lin et al., 2015).

Limitations

In this paper, we focus on designing methods to construct DEER and evaluating DEER on serving as a system for entity relationship understanding, which has direct applications in, e.g., encyclopedias and concept maps. Due to limited space, we do not fully investigate its use as a knowledge source to facilitate other tasks, e.g., relation extraction and knowledge graph completion, which we leave as future work for the whole research community.

Acknowledgements

We thank the reviewers for their constructive feedback. This material is based upon work supported by the National Science Foundation IIS 16-19302 and IIS 16-33755, Zhejiang University ZJU Research 083650, IBM-Illinois Center for Cognitive Computing Systems Research (C3SR) – a research collaboration as part of the IBM Cognitive Horizon Network, grants from eBay and Microsoft Azure, UIUC OVCR CCIL Planning Grant 434S34, UIUC CSBS Small Grant 434C8U, and UIUC New Frontiers Initiative. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

Oshin Agarwal, Heming Ge, Siamak 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.

Wenhu 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.

Liying Cheng, Dekun Wu, Lidong Bing, Yan Zhang, Zhanming Jie, Wei Lu, and Luo Si. 2020. Ent-desc: Entity description generation by exploring knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1187–1197.

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.

Kalpa Gunaratna, Yu Wang, and Hongxia Jin. 2021. Entity context graph: Learning entity representations fromsemi-structured textual sources on the web. *arXiv preprint arXiv:2103.15950*.

Abram Handler and Brendan O'Connor. 2018. Relational summarization for corpus analysis. 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 1760–1769.

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 Chang, Jinjun Xiong, and Wen-Mei Hwu. 2022a. Open relation modeling: Learning to define relations between entities. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 297–308.

Jie Huang, Hanyin Shao, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022b. Understanding jargon: Combining extraction and generation for definition modeling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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, Yinhan 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*.

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
nn	noun compound 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
cop	copula
ccomp	clausal complement
xcomp	open clausal complement
expl	expletive
punct	punctuation
nsubj	nominal subject
csubj	clausal subject
csubjpass	clausal subject (passive)
dobj	direct object
iobj	indirect object
obj	object
pobj	object of preposition

Table 7: Manually collected modifying dependencies in spaCy.

A Preprocessing and Filtering

We introduce our preprocessing to the raw Wikipedia dump[10]. For each article, we extract the plain text by WikiExtractor[11]. We split the Wikipedia articles into sentences with the NLTK library[12] 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 "apple" to "Apple Inc" or "Apple (food)". In this work, we follow a similar approach to build the mapping. To maintain high accuracy and low ambiguity, we craft the entity mention from the entity by removing the content wrapped by parenthesis and the content after the first comma. For example, a mention-entity pair could be ("Champaign", "Champaign, Illinois") or ("Python", "Python (programming language)"). Unlike Wikipedia2Vec, we create a local dictionary for each Wikipedia page. When processing a page, we dynamically update the dictionary with mention-entity pairs collected from the hyperlinks, and extract the entity occurrence with the updating dictionary in one pass. This can reduce the ambiguity when two entities with the same entity mention co-occur on one page and also avoid collecting trivial entity occurrence on the page.

[10] https://dumps.wikimedia.org (version: enwiki/20210320)

[11] https://github.com/attardi/wikiextractor

[12] https://www.nltk.org

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.

Table 8: Annotation guidelines excerpt.

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-

	ExpScore	**SigScore**	**RDScore**	**RelationSyn-0**	**RelationSyn-1**	**RelationSyn-5**
(Mucus, Stomach)	As the first two chemicals may damage the stomach wall, *mucus* is secreted by the *stomach*, providing a slimy layer that acts as a shield against the damaging effects of the chemicals.	The *mucus* produced by these cells is extremely important, as it prevents the *stomach* from digesting itself.	The *mucus* produced by these cells is extremely important, as it prevents the *stomach* from digesting itself.	*Mucus* is a fluid that is produced by the *stomach*.	*Mucus* is the main barrier to mucus from the *stomach*.	*Mucus* is a thick, protective fluid that is secreted by the *stomach*.
(Surfers Paradise, Queensland)	*Surfers Paradise* is a coastal town and suburb in the City of Gold Coast, *Queensland*, Australia.	In 2009 as part of the Q150 celebrations, *Surfers Paradise* was announced as one of the Q150 Icons of *Queensland* for its role as a "location".	*Surfers Paradise* is a coastal town and suburb in the City of Gold Coast, *Queensland*, Australia.	*Surfers Paradise* is a coastal suburb in the City of Brisbane, *Queensland*, Australia.	*Surfers Paradise* is a coastal town and locality in the Shire of Mareeba, *Queensland*, Australia.	*Surfers Paradise* is a coastal suburb in the City of Redland, *Queensland*, Australia.
(Parkinson's disease, Dopamine)	Thus for the first time the reserpine-induced Parkinsonism in laboratory animals and, by implication, *Parkinson's disease* in humans was related to depletion of striatal *dopamine*.	*Parkinson's disease* is characterized by the death of cells that produce *dopamine*, a neurotransmitter.	*Parkinson's disease* is associated with the degeneration of *dopamine* and other neurodegenerative events.	*Parkinson's disease* is a neurodegenerative disease involving the loss of *dopamine* in the brain.	*Parkinson's disease* is a neurodegenerative disease characterized by the loss of *dopamine*.	*Parkinson's disease* is a neurodegenerative disorder characterized by a slow and steady loss of *dopamine* in the substantia nigra.
(Dayan Khan, Oirats)	Mandukhai and *Dayan Khan* defeated the *Oirats* and the taishis who ruled the Yellow River Mongols.	By 1510 *Dayan Khan* had unified the entire Mongol nation including *Oirats*.	By 1510 *Dayan Khan* had unified the entire Mongol nation including *Oirats*.	*Dayan Khan* was a khan of the *Oirats*.	*Dayan Khan* was a khan of the *Oirats*.	*Dayan Khan* defeated the *Oirats* in 1510 with the assistance of the Four Oirats.
(Knowledge, Epistemology)	In *epistemology*, descriptive *knowledge* is knowledge that can be expressed in a declarative sentence or an indicative proposition.	These questions, but particularly the problem of how experience and *knowledge* interrelate, have broad theoretical and practical implications for such academic disciplines as *epistemology*, linguistics, and psychology.	*Knowledge* is the primary subject of the field of *epistemology*, which studies what we know, how we come to know it, and what it means to know something.	In *epistemology*, *knowledge* is a description of the possible meaning of knowledge.	In philosophy, aristocratic *knowledge* is a form of knowledge that can be gained through experience, through the use of a method of *epistemology*.	In the philosophy of *epistemology*, *knowledge* is often referred to as "a priori" or "synthetic".
(Atlantic Coast Line Railroad, Seaboard Air Line Railroad)	The *Atlantic Coast Line Railroad* later merged with the *Seaboard Air Line Railroad* to form the Seaboard Coast Line Railroad.	In 1967, the *Atlantic Coast Line Railroad* merged with the *Seaboard Air Line Railroad*, forming the Seaboard Coast Line Railroad.	In 1967, the *Atlantic Coast Line Railroad* merged with the *Seaboard Air Line Railroad*, forming the Seaboard Coast Line Railroad.	The *Atlantic Coast Line Railroad* was merged into the *Seaboard Air Line Railroad* in 1887.	The *Atlantic Coast Line Railroad* merged with the *Seaboard Air Line Railroad* in 1986 to form CSX Transportation.	The *Atlantic Coast Line Railroad* merged with the *Seaboard Air Line Railroad* on July 1, 1967, to form the Seaboard Coast Line Railroad.
(Twilight, Sunset)	*Twilight* is the period of night after *sunset* or before sunrise when the Sun still illuminates the sky when it is below the horizon.	Near the summer solstice, there are less than 8 hours between *sunset* and sunrise, with *twilight* lasting past 10 pm.	*Twilight* is the period of night after *sunset* or before sunrise when the Sun still illuminates the sky when it is below the horizon.	*Twilight* is the period of daylight between sunrise and *sunset* when the Sun is below the horizon.	*Twilight* is the period of darkness when the Sun is below the horizon.	*Twilight* is the period of darkness from *sunset* to sunrise when the Sun is below the horizon.
(Rock shelter, Cliff)	*Rock shelters* form because a rock stratum such as sandstone that is resistant to erosion and weathering has formed a *cliff* or bluff, ..., and thus undercuts the cliff.	A *rock shelter* is a shallow cave-like opening at the base of a bluff or *cliff*.	A *rock shelter* is a shallow cave-like opening at the base of a bluff or *cliff*.	A *rock shelter* is a structure built on the top of a *cliff*.	A *rock shelter* is a *cliff* or clifftop that is surrounded by a rock.	A *rock shelter* is a small, relatively flat, cave or cave-like structure on a *cliff*.

Table 9: Sample of relation descriptions produced by *ExpScore*, *SigScore*, *RDScore*, and *RelationSyn-*m.

first-search algorithm is a way to explore the vertexes 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 keep the sentences with the number of tokens $\in [5, 50]$. We also only keep sentences whose shortest dependency path pattern between two target entities starts with *nsubj* or *nsubjpass* (more details are in Section 4.2.2).

B Generation Examples

We present sample outputs of the models in Table 9, with analysis of the results in Section 6.3.