DESCRIPTIVE KNOWLEDGE GRAPH FOR EXPLAINING ENTITY RELATIONSHIPS

BY

KERUI ZHU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Adviser:

Professor Kevin Chen-Chuan Chang

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.

We also present a novel system that automates the extraction or generation of informative and descriptive sentences from biomedical corpus and builds a descriptive knowledge graph to facilitate efficient search for relational knowledge. In contrast to previous search engines or exploration systems that retrieve unconnected passages, our system organizes descriptive sentences into a graph, enabling researchers to explore relationships between entities. Our system also includes a relation synthesis model that generates concise descriptive sentences from retrieved sentences, reducing the need for human reading effort. With our system, researchers can quickly obtain a high-level overview of directly related entities to a query entity (e.g., diseases treated by a chemical) or indirect connections between two entities (e.g., candidate drugs for treating a disease). This information can guide literature surveys and facilitate the discovery of potential research topics. Our system also speeds up the literature curation and drug repurposing process. We demonstrate the effectiveness of our system on the CORD-19 dataset, but it can be deployed on any biomedical corpus without manual adaptation.

TABLE OF CONTENTS

CHAPTER 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) [1, 2], where nodes are entities, e.g., *machine learning*, and edges are relations, e.g., *subclass of* (Figure 1.1). 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 [3] 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 [4, 5, 6] or open relation extraction [7] can represent more complex relationships to some extent, they do not fundamentally solve the limitations as discussed in [8]. 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*

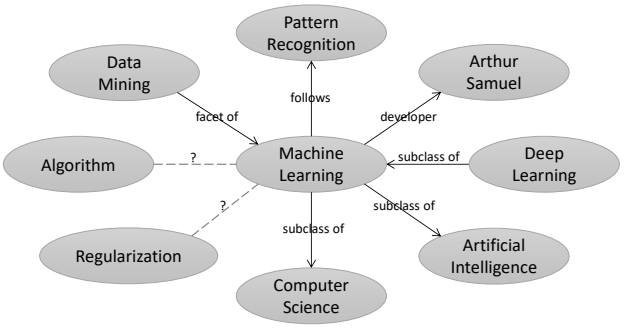

Figure 1.1: Relations in Wikidata (Knowledge Graph), where ? means the relation is not present in the graph.

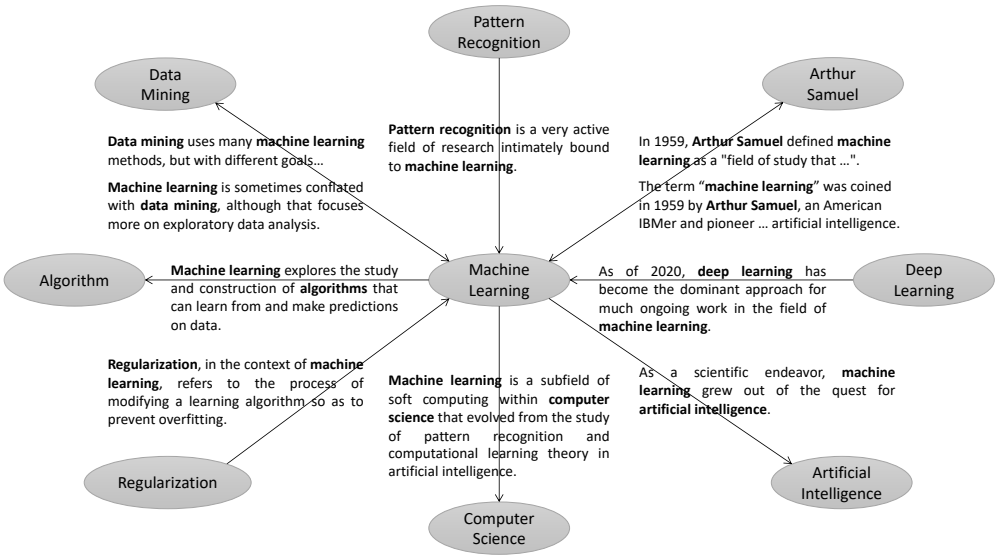

Figure 1.2: Relations in DEER. Here we show *machine learning* and several of its related entities, with corresponding relation descriptions produced by our model (only extraction) in the edges.

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 1.2 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 ***informative*** 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 facilitating downstream tasks on entities and entity relationships such as entity profiling [9, 10, 11], relation extraction [12], and knowledge graph completion [13]. For example, in Figure 1.2, 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 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 3.3.1, 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.

CHAPTER 2: BACKGROUND

There are several previous attempts on acquiring entity relation descriptions. For instance, [14] study a learning-to-rank problem of ranking relation descriptions by training a Random Forest classifier with manually annotated data. Subsequently, [15] 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 [16] 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 [14] and 10 in [16], and only explicit relation types, e.g., *isMemberOfMusicGroup*, are covered. Notably, [17] 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* [8], which aims to generate relation descriptions for entity pairs. To achieve this, the authors propose to fine-tune BART [18] 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 generations.

There are also some other works that can be related. For example, [19, 20] study *CommonGen*, which aims to generate coherent sentences containing the given common concepts. [21, 22] study the data-to-text generation [23], which aims to convert facts in KGs into natural language. [24] proposes 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.

CHAPTER 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 1.2).

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 [25]. 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 3.1. 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 approach: first extracting relation descriptions from a corpus with the analysis of dependency patterns (Section 3.1), and then generating relation descriptions for interesting entity pairs whose relation descriptions are not extracted in the previous step (Section 3.2).

#	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 3.1: Example sentences containing both *machine learning* and *algorithm*.

3.1 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.

3.1.1 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.

We introduce our preprocessing to the raw Wikipedia dump[1]. For each article, we extract the plain text by WikiExtractor[2]. We split the Wikipedia articles into sentences with the NLTK library[3] 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 [25] 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")

[1] `https://dumps.wikimedia.org` (enwiki/20210320)
[2] `https://github.com/attardi/wikiextractor`
[3] `https://www.nltk.org`

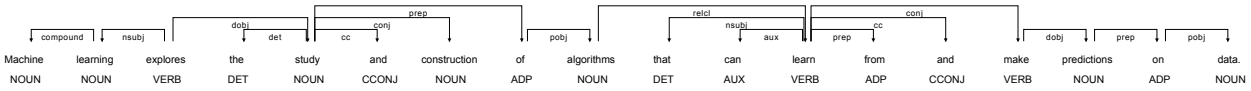

Figure 3.1: Dependency tree of s_1.

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 occurrences on the page.

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.

3.1.2 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.

Shortest dependency path as relation Inspired by [26], we use the shortest dependency path to represent the relation pattern between the target entities in a sentence. For instance, Figure 3.1 shows the dependency tree of s_1 processed by spaCy[4]. 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

[4] `https://spacy.io`

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 3.1, *construction* is a token outside the *corePath* between *machine learning* and *algorithm*. The shortest path between it and the *corePath* is: "study $\overleftarrow{conj}$ 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.

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*:

$$ExpScore(s) = \frac{\log(f_p + 1)}{\log(f_{max} + 1)}, \tag{3.1}$$

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 [26], 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.

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 3.2

We calculate a score for each token in the sentence based on its category and dependency

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 3.2: Manually collected modifying dependencies in spaCy.

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

$$SigScore(s) = \frac{\sum_{t \in s} w(t)}{|s|}, \tag{3.2}$$

where

$$w(t) = \begin{cases} 1 & \text{if } t \in ct \\ \frac{\log(f'_{p_t}+1)}{\log(f'_{max}+1)} & \text{if } t \in mt \\ 0 & \text{otherwise} \end{cases}, \tag{3.3}$$

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.

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 3.1.1 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:

$$RDScore(s) = 2 \cdot \frac{ExpScore(s) \cdot SigScore(s)}{ExpScore(s) + SigScore(s)}. \tag{3.4}$$

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 DEER, we keep edges with an entity relevance score ≥ 0.5[5] and with a relation description

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

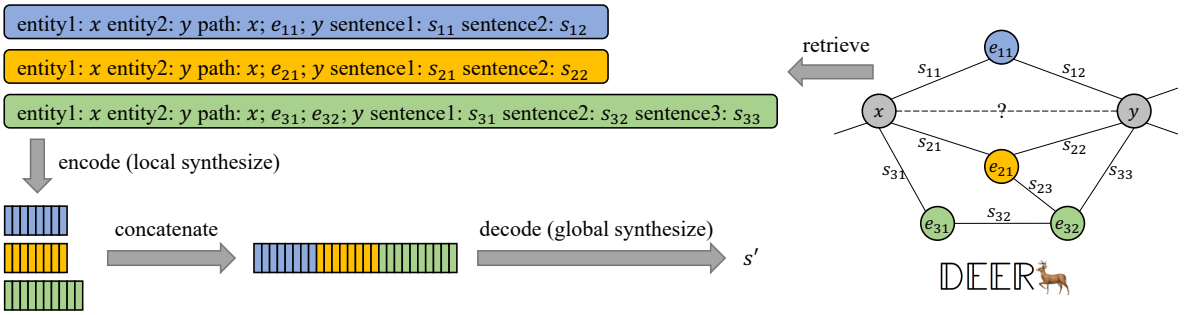

Figure 3.2: The framework of *RelationSyn*. Given entity pair (x, y) whose relation description is not present in the initial DEER, we first retrieve several reasoning paths from the graph. And then, we encode (local synthesize) each reasoning path into a latent vector and concatenate all the latent vectors. Finally, we decode (global synthesize) the vector to produce relation description s' for (x, y).

whose $RDScore \geq 0.75$[6]. We refer to this graph as **Wiki-DEER**$_0$.

3.2 RELATION DESCRIPTION GENERATION

In the previous section, we extract relation descriptions for entity pairs with the analysis of dependency patterns and build an initial 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 DEER. Formally, we apply the knowledge-enhanced sequence-to-sequence formulation [27]: 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:

$$P(s|x, y, \mathcal{G}_0) = \prod_{i=1}^{m} P(s_i|s_{0:i-1}, x, y, \mathcal{G}_0), \tag{3.5}$$

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

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

descriptions (reasoning paths) from the graph and then synthesizing them into a final relation description (Figure 3.2).

3.2.1 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 3.2, 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[7]. To measure the quality of reasoning paths, we define *PathScore* as the harmonic mean of *RDScore* of relation descriptions in the path:

$$PathScore(q) = \frac{|\mathcal{S}_q|}{\sum_{s \in \mathcal{S}_q} \frac{1}{RDScore(s)}}, \tag{3.6}$$

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 1.2 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).

3.2.2 Synthesizing

According to Section 3.2.1, 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 [28] and Fusion-in-Decoder [29].

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

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 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 "**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.

3.3 EVALUATION

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

3.3.1 Relation Description Extraction

We first present the statistics of the initial DEER built with Wikipedia in Table 3.3. To evaluate the quality of relation descriptions in the graph, we randomly sample 100 entity pairs from the graph[8] 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 3.7.

Since previous works on relation description extraction are supervised and only limited to several explicit relation types, e.g., 9 in [14], 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.

[8]More specifically, for better comparison with generation later, we sample 100 entity pairs from the test set in Table 3.5.

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

Table 3.3: The statistics of Wiki-DEER$_0$.

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

Table 3.4: Qualitative results of extraction.

- **ExpScore**: The sentence with the highest *explicitness* is selected according to Eq. (3.1).
- **SigScore**: The sentence with the highest *significance* is selected according to Eq. (3.2).
- **RDScore**: The sentence with the highest *RDScore* is selected according to Eq. (3.4).

Table 3.4 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-DEER$_0$ is high.

3.3.2 Relation Description Generation

Data construction We build a dataset for relation description generation as follows: for an entity pair with a relation description in Wiki-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[9]. For instance, in Figure 3.2, we hide the edge (relation description s) between x and y and use the remaining reasoning paths to recover s.

[9]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.

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

Table 3.5: The statistics of data for generation.

	BLEU	ROUGE	METEOR	BERTScore
RealtionBART-Vanilla [8]	19.61	41.52	20.48	82.99
RealtionBART-MP + PS [8]	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 3.6: Quantitative results of relation description generation.

We train and test on entity pairs with ≥ 5 reasoning paths connecting them. The statistics of the data are reported in Table 3.5.

Models The task of relation description generation is relevant to *Open Relation Modeling* [8] – 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 [8] with several variants of our model:

- **RelationBART (Vanilla)**: The vanilla model proposed in [8] for generating entity relation descriptions, where BART [18] 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 [8], 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.

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 3.7: Annotation guidelines excerpt.

- **RelationSyn-**m: The proposed relation description synthesizing model (Section 3.2), where m is the maximum number of retrieved reasoning paths for an entity pair.

Metrics We perform both quantitative and qualitative evaluation. Following [8], we apply several automatic metrics, including BLEU [30], ROUGE-L [31], METEOR [32], and BERTScore [33]. 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 3.7.

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[10] and adopt the default hyperparameters. The training converges in 50 epochs. For our models, we modify the implementation of Fusion-in-Decoder[11] 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

[10] `https://github.com/jeffhj/open-relation-modeling`
[11] `https://github.com/facebookresearch/FiD`

	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 3.8: Qualitative results of generation.

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).

Quantitative evaluation Table 3.6 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.

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 3.3.1. From the results in Table 3.8, 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.

3.3.3 Case Study and Error Analysis

In Table 3.9, 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 3.9 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.

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

CHAPTER 4: DESCRIPTIVE KNOWLEDGE GRAPH FOR COVID LITERATURE

4.1 INTRODUCTION

Efficiently extracting knowledge from the vast and ever-growing corpus of literature is crucial for researchers to remain up-to-date with the latest discoveries and trends in their field. The COVID-19 pandemic has highlighted this need, with thousands of related studies being published in a short period when a new disease emerges. However, surveying the latest findings requires significant effort, and researchers may struggle to see the big picture, leading to duplicated work and delaying the development of treatments[34]. To address this challenge, many tools for information retrieval in scientific literature have been proposed in recent years.

Previous works have focused on document retrieval question answering[35], key phrase queries, and relation extraction. Document retrieval question-answering systems retrieve original documents or passages as independent answers to user queries. Key phrase queries [36]retrieve sentences containing one or more specified key phrases, with modifiers provided by the user. Relation extraction[37, 38, 39] extracts possible relationships between entities from corpus sentences to collect relational knowledge. However, these systems do not serve as comprehensive knowledge discovery tools for researchers. With document retrieval QA systems, researchers must still read retrieved documents to find relevant information, which is time-consuming[40]. Key phrase queries limit searching flexibility, relying on the user's choice of query words and hindering the retrieval of unknown knowledge. Relation extraction systems require supervised training or hand-crafted rules to build models that can only detect relations from a predefined set, leading to limited coverage and reliance on training datasets or rules. Additionally, none of these systems display retrieved documents or sentences in a connected manner, forcing users to organize and conclude information on their own, which can be both exhausting and incomplete. This sets a barrier for researchers to discover related knowledge across different literature.

In this paper, we present a novel system called *CovidDEER*, which overcomes the limitations of the three types of systems mentioned earlier by leveraging a COVID-related corpus. Our system enables users with little prior knowledge to interactively retrieve up-to-date, comprehensive, and easily understandable relation description sentences, and explore relational knowledge between entities in one-hop or multi-hop connections. Additionally, we introduce an innovative approach to generate succinct relation descriptions for entity pairs from the retrieved relation descriptions to aid users in acquiring information. We demonstrate that

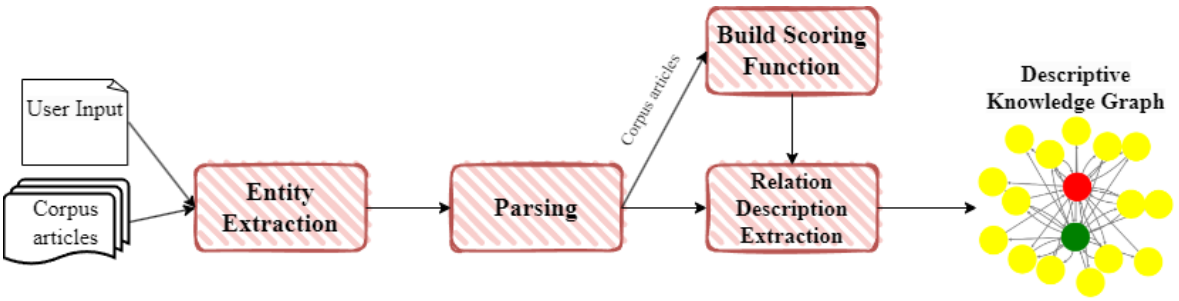

Figure 4.1: Descriptive knowledge graph construction pipeline.

CovidDEER can benefit literature surveys, literature curation[41], and drug repurposing task. Furthermore, our system is automatically built without any supervised training or hand-crafted rules, making it seamlessly adaptable to any biomedical corpus with ease, and it can serve as a frontrunner for collecting knowledge in any future emergency.

In essence, our system builds and manages a descriptive knowledge graph (DKG)[42] in which nodes represent biomedical entities, and edges represent relation description sentences collected from the corpus. Users can explore knowledge or retrieve sentences by querying this graph using entities, modifiers, or entity types. To maximize the potential of this resource, our system provides three useful tools:

- A neighbor query module that can retrieve entities of specific types (such as Disease or Chemical) within one or two-hop neighbors of the query entity.
- An article parser that enables users to build a DKG over a biomedical article and identify the related entities and relation descriptions in the article.
- A relation synthesis model that can automatically generate relation descriptions for entity pairs connected through a one or two-hop path using the retrieved relation descriptions. This model can aggregate and summarize the information for the user, providing a brief overview of the entity pair before the user begins reading the retrieved sentences.

We will provide further details about the implementation of our system in the subsequent sections.

4.2 GRAPH CONSTRUCTION

To enhance users' understanding of the retrieved information related to their queries, our system goes beyond traditional text search engines that only return independent passages. Instead, our system extracts descriptive sentences and constructs a descriptive knowledge graph (DKG), as introduced in DEER[42]. Unlike traditional text search engines, which leave users to extract and organize knowledge from each text piece on their own, our system

organizes entities and sentences in the DKG. In this graph, nodes represent entities, and edges represent sentences that describe the relationship between the two nodes, called *relation descriptions*, pointing from the subject to the object in the sentences. With this graph, our system enables users to retrieve sentences with efficient graph queries and view the result from a connected perspective, allowing them to gain a more holistic understanding of the information related to their queries.

In this section, we will introduce some adjustments to the original DEER work for building this DKG in the biomedical domain.

4.2.1 Corpus

Our system is built on a DKG constructed from the COVID-19 Open Research Dataset (CORD-19) [43]. CORD-19 is a corpus comprising scientific papers related to COVID-19 and other coronaviruses, but it has not been updated since 2022. For the demonstration purposes, we used the August 8th, 2020 version of CORD-19 to simulate a collection of papers in the context of a new disease outbreak and some clinical experimental results have been published. Through this simulation, we demonstrate how our system can retrieve valuable information for disease research and lead to meaningful findings in drug repurposing.

4.2.2 Pipeline

To construct the DKG, our system employs a pipeline that processes the corpus as follows. A visualization of the pipeline is shown in Figure 4.1.

Entity extraction and linking . Initially, we extract biomedical entities from each sentence in the articles and link them to biomedical ontologies using the NCBI Pubtator API and the SciSpacy library[44]. Specifically, we link the extracted entities to Cellosaurus, OMIM, MeSH, Gene, Taxonomy, and UMLS metathesaurus.

Parsing . Next, we parse the sentences and gather the parameters for the scoring function. We use the relation description score (RDS) introduced in DEER to rank the relation descriptions. The RDS's parameters are the dependency path frequencies, and we parse all the sentences in CORD-19 using the SciSpacy library. We then collect the dependency paths between the extracted entities to ensure that the scoring function is customized to the biomedical domain. These parameters can be fixed and reused for new papers in the biomedical domain.

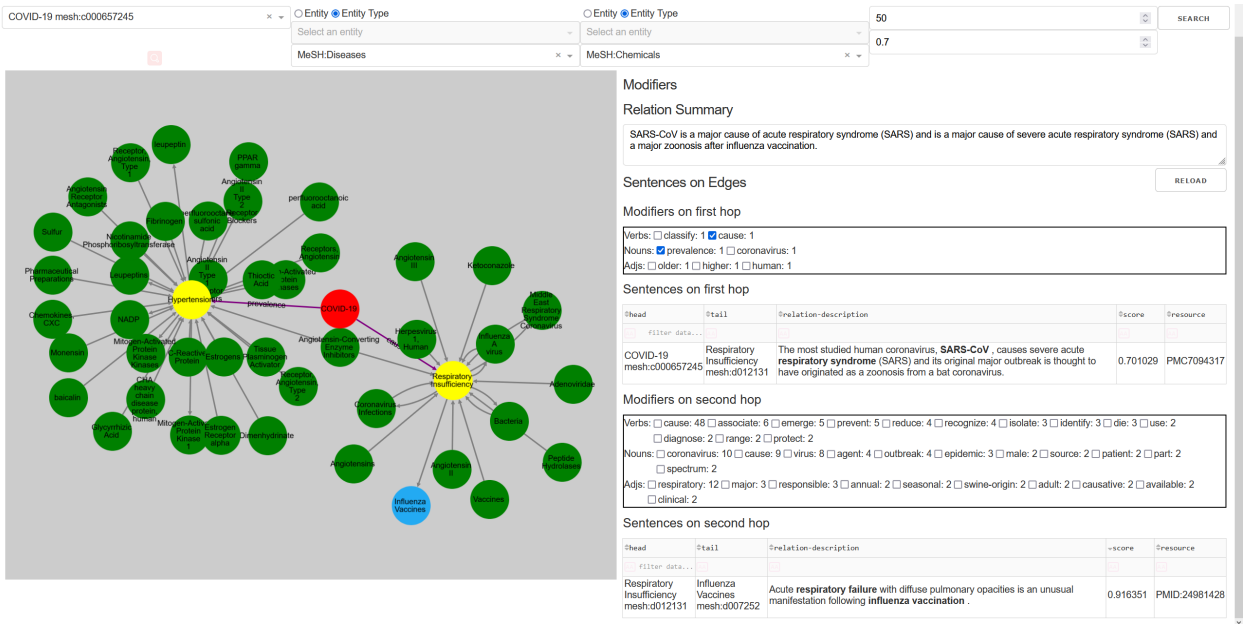

Figure 4.2: The web interface of *CovidDEER*.

Relation description extraction . For each sentence that contains a pair of entities, we apply the scoring function to evaluate the sentence's syntax in describing the relation between the entities. A higher score indicates that the sentence has a more explicit expression, and the relation information comprises a more significant portion of the sentence's meaning. We collect all the sentences that score higher than a certain threshold and use them to form the edges in the DKG.

4.3 SYSTEM DESCRIPTION

In this section, we will introduce the three main modules of our system that enable users to effectively query the DKG for relevant sentences and extract the maximum value from this resource.

4.3.1 Graph Query

The Graph Query module provides easy access to the DKG and performs several kinds of queries. Like most search engines, our system allows boolean queries, which allow users to retrieve sentences where the queried entities co-occur. In terms of querying the DKG, this is similarly achieved by retrieving the sentences on the edges between the queried entities. However, our system's approach is more effective since it retrieves relation descriptions,

Type	Frequent Modifiers
Nouns	treatment (34), patient (19), therapy (12), chloroquine (7), efficacy (6), drug (5), risk, (4), use, (4), hydroxychloroquine (4), option (3), effect (3), diagnosis (3), vaccine (3), trial (3), case (3), candidate (3), choice (3), level (3), increase (3), agent (3), disease (3), activation (3)
Verbs	use (19), show (14), treat (10), test (7), approve (7), reduce (7), inhibit (7), induce (7), play (6), increase (6), investigate (5), propose (5), prevent (5), cause (4), consider (4), block (4), appear (4), prove (3), become (3), associate (3), provide (3), protect (3), lead (3), find (3)
Adjs	potential (8), antiviral (7), therapeutic (7), effective (7), beneficial (5), clinical (5), apparent (5), severe (4), good (3)

Table 4.1: Frequent Modifiers between Chemicals and COVID-19.

which not only contain the queried entities but also provide clear relational information between them, which is cleaner and more useful for the user's understanding.

Moreover, users can query entity types that co-occur with an entity to obtain a comprehensive overview. Our system allows querying of several specific types of neighbors of an entity. For instance, users can retrieve all the *Chemicals* connected to *COVID-19* to gain insights into Chemical-Disease interactions related to COVID-19.

The retrieval result is displayed as a graph, as shown in Figure 4.2, allowing users to easily locate sentences containing entities of interest. To assist users in navigating the retrieved sentences, we have augmented the graph query module with two additional features: **modifier filtering** and **multi-hop neighbor query**.

Modifier filtering We define the words and phrases that convey the relation between two entities as *modifiers* of the relation, and our system provides a feature that allows users to filter the retrieved relation descriptions based on these modifiers. When querying the co-occurrence of a popular entity, retrieval systems may return an overwhelming number of independent sentences, which can distract users from extracting the general relations between the entities. Instead of asking the user to form a more specific query by adding co-occurred words, our system collects a set of frequent modifiers from the retrieved sentences. These modifiers are the noun phrases, verbs, or adjectives on the dependency path between the two entities and provide insights into the relational information in the sentences. For example, Table 4.1 presents the most common modifiers collected from the query of searching the *Chemical* neighbors of *COVID-19*. Users can read the list of modifiers first and then check out the sentences that might interest them. With our system, users do not require significant

background knowledge to form a more specific query, and they can explore knowledge with the utmost flexibility. Our system also labels the edges with the modifiers collected from the sentences on each edge to provide users with a general idea of the relational information between the entities before they start reading the sentences.

Multi-hop query Since our system manages the sentences in a graph structure, users can perform multi-hop queries to find connections between entities across different documents. By specifying the entity or entity type at each hop, users can create queries tailored to their purposes. For example, a user may begin with *COVID-19*, with *Symptom* as the first-hop entity type and *Chemical* as the second-hop entity type to explore candidate drugs for COVID-19 treatment. Compared to traditional knowledge graphs, our system can provide the most up-to-date information as soon as new articles are published, which is valuable for researchers keeping track of disease trends. In contrast to text-based search engines, our system retrieves sentences on the multi-hop path in a single query, while traditional search engines require multiple queries. This makes our system more efficient for retrieving knowledge and facilitates multi-hop reasoning for users.

4.3.2 Relation Synthesis Model

Multi-hop reasoning is a common method to explore new knowledge from knowledge graphs. However, this is challenging in a DKG because each edge is a set of relation descriptions instead of a single relation label. Users need to read and comprehend the knowledge on each edge before they can perform inference, which can be time-consuming and laborious. To help users gain a quick understanding of the relationship between two entities, we employ a relation synthesis model based on DEER's work. The model automatically generates concise relation descriptions from the sentences retrieved on the paths. Besides training the model on two-hop and three-hop paths, we also trained it on one-hop paths, allowing it to learn to summarize relation descriptions for individual entity pairs. By reading the generated relation description first, users can obtain a general understanding of the relation before delving into the details in the retrieved sentences.

As the CORD-19 dataset we use for demonstration is not large enough to train a relation synthesis model, we collected a training, validation, and test dataset from a subset of articles randomly selected from PubMed. The resulting dataset is comparable in size to the one used in DEER, and we trained the model for 20 epochs. In Section 4.4, we discuss our manual evaluation of the quality of generation.

4.3.3 Article Parser

Our system offers a module called the "Article Parser" in addition to the search engine. The Article Parser takes any user-provided passage or article and runs the pipeline described in Figure 4.1 to generate a local DKG for the user's purpose. However, since the extracted local DKG may contain various extracted entities, users can select specific entities or entity types. Moreover, we compute the normalized pointwise mutual information (NPMI) score for connected entity pairs in the local DKG using corpus records. We then highlight highly correlated pairs to indicate possibly informative edges. The graph generated by the Article Parser visually represents the entity connections in the article. It helps users locate interesting relation descriptions or conclusions. In the following section, we present a case study demonstrating how the Article Parser can be used for literature curation.

4.4 EVALUATION

In this work, we assess the performance of CovidDEER in two distinct parts: the ability of the extracted descriptive knowledge graph and the effectiveness of the relation synthesis model. To evaluate the extracted DKG, we conducted case studies and developed two workflows to showcase how CovidDEER can assist with biomedical tasks in practical scenarios. In addition, we assessed the reliability of the relation synthesis model by manually examining the generated text's faithfulness in relation to the input relation descriptions.

4.4.1 Case Study 1: Drug Repurposing

Drug repurposing involves identifying new uses for drugs that were originally developed to treat other diseases. CovidDEER can aid researchers in identifying candidate drugs through the following steps:

- Begin with the target disease as the starting node.
- Search the first-hop neighborhood for diseases and symptoms related to the target disease.
- Search the second-hop neighborhood for drugs used to treat those related diseases and symptoms.

Suppose a researcher wants to discover the candidate drugs for COVID-19. By searching the *Diseases* and *Symptoms* neighbors of *COVID-19*, the system retrieved several frequent verb modifiers as shown in Figure 4.3. We select several modifiers that might indicate a correlation between the *Diseases* and *COVID-19*. Then, we pick 10 out of these "related" Disease entities as the first-hop neighbors and search for the Chemicals or Drugs in the

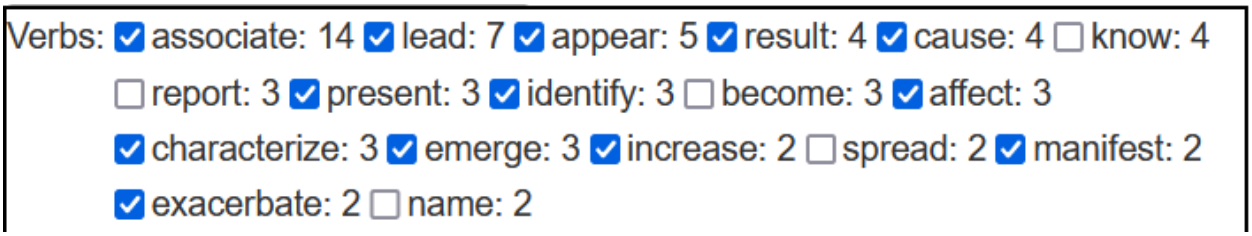

Figure 4.3: Verb Modifiers between *COVID-19* and *Disease* or *Symptom*.

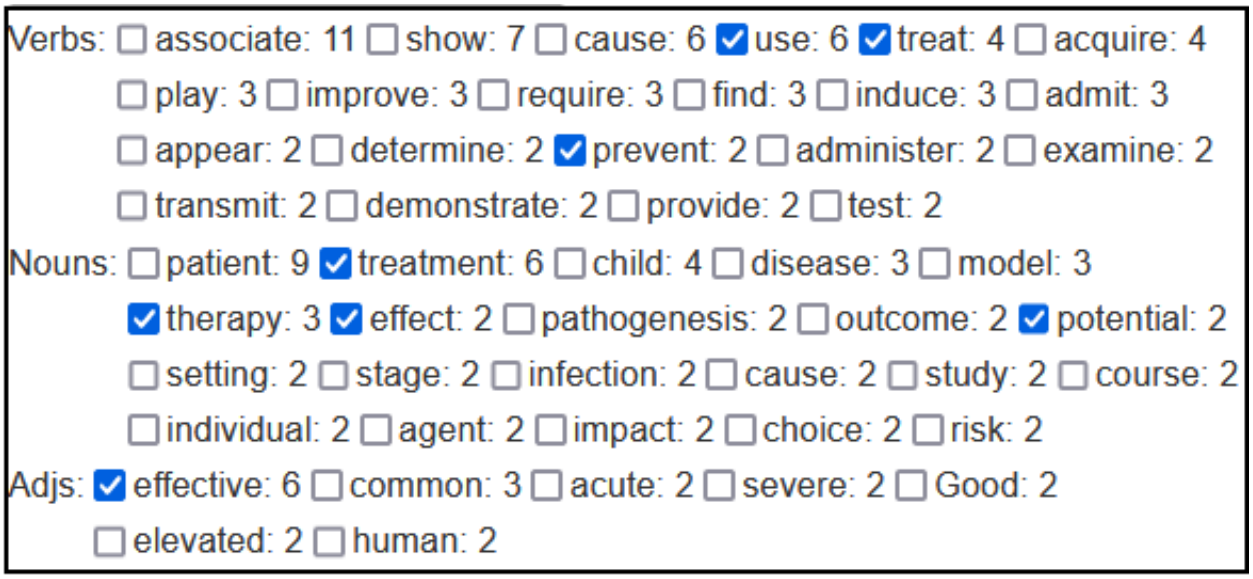

Figure 4.4: Modifiers between COVID-19 related and *Disease* and *Chemicals*.

Candidate drugs
nitric oxide, lamb preparation, beta-Lactams, Leukotriene B4, sphingosine 1-phosphate, amoxicillin, Macrolide Antibiotics, Macrolides, beta-Lactams, rifampin, Hydroxymethylglutaryl-CoA Reductase Inhibitors, methylprednisolone, trivalent influenza vaccine, Fibrates, lipid modifying drugs, plain, Corticosteroid ophthalmologic and otologic preparations, metformin, inhibitors, Corticosteroid otologicals, Bilirubin, Fibrates, nitazoxanide, atorvastatin, Artemisinins, antagonists

Table 4.2: Collected candidate drugs for COVID-19 treatment.

second-hop neighbors. The retrieved two-hop graph can be seen in Figure 4.2. Similarly, we select several modifiers that might indicate a "treatment" relation as shown in Figure 4.4 and candidate drugs collected are shown in Table 4.2.

To identify potential candidate drugs for COVID-19, a researcher using CovidDEER would begin by searching for diseases and symptoms related to COVID-19. Figure 4.3 displays several frequent verb modifiers retrieved by the system, from which the researcher could select several modifiers that may indicate a correlation between Diseases and COVID-19.

	(COVID-19, Pneumonia)	(Pneumonia, Vaccines)
Extracted relation descriptions	**Coronavirus disease 2019** (COVID-19) is a novel type of highly contagious **pneumonia** caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Conversely, SARS-CoV, MERS-CoV, and **COVID-19** may initially present asymptomatically, but can progress to **pneumonia**, shortness of breath, renal insufficiency and, in some cases, death.	Despite the availability of safe and effective antibiotics and **vaccines** for treatment and prevention, **pneumonia** is a leading cause of death worldwide and the leading infectious disease killer. Despite advances in managerial practices, **vaccines**, and clinical therapies, **pneumonia** remains a widespread problem and methods to enhance host resistance to pathogen colonization and pneumonia are needed.
1-hop relation summary	**COVID-19** is a highly contagious **pneumonia** caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2).	Despite the availability of safe and effective antibiotics and **vaccines** for treatment and prevention, **pneumonia** remains a major cause of death worldwide.
2-hop relation synthesis (COVID-19, Vaccines)	COVID-19 is a major cause of death worldwide, despite the availability of safe and effective antibiotics and vaccines for treatment and prevention of pneumonia.	

Table 4.3: Example of relation description extracted or generated by the relation synthesis model.

Using these "related" Disease entities as first-hop neighbors, the researcher would search for Chemicals or Drugs in the second-hop neighbors, resulting in the retrieval of a two-hop graph, as depicted in Figure 4.2.

To identify potential treatments for COVID-19, the researcher would identify modifiers that suggest a "treatment" relation, as shown in Figure 4.4. The candidate drugs collected from this search are presented in Table 4.2.

4.4.2 Case Study 2: Literature Curation

Literature curation involves collecting knowledge facts from articles and adding them to a knowledge graph. Curators are typically assigned an entity and a list of articles, and they need to read each article in its entirety to find related facts to the entity[41, 45]. This process

In cultured human Upcyte hepatocytes and HepG2 cells, CLAV was more cytotoxic than AMOX, and, at subcytotoxic concentrations, it altered the expression of more than 1,300 genes. CLAV, but not AMOX, downregulated the expression of key genes for BA transport (BSEP, NTCP, OSTalpha and MDR2) and synthesis (CYP7A1 and CYP8B1). CLAV also caused early oxidative stress, with reduced GSH/GSSG ratio, along with induction of antioxidant nuclear factor erythroid 2-related factor 2 (NRF2) target genes. Activation of NRF2 by sulforaphane also resulted in downregulation of NTCP, OSTalpha, ABCG5, CYP7A1 and CYP8B1. CLAV also inhibited the BA-sensor farnesoid X receptor (FXR), in agreement with the downregulation of FXR targets BSEP, OSTalpha and ABCG5.

Figure 4.5: Example of specific entity type extraction with modifiers.

can be time-consuming. With CovidDEER, curators can save time by using the article parser to extract relation descriptions from articles, allowing them to focus on sentences that are likely to contain relevant facts.

To use the article parser, curators can provide a list of entities and entity types of interest and submit the article to build a local DKG. The resulting graph shows the connection between the entity of interest and other entities, and the extracted modifiers suggest possible relations between the entities. Curators can then read the sentences on the edges to curate the knowledge. Figure 4.5 shows the extraction of entities and modifiers from a passage where the entity of interest is *Clavulanic Acid* and the entity types are *Antibiotic*, *Gene*, *Organic Chemical*, and *Cell or Molecular Dysfunction*. The resulting graph is shown in Figure 4.6, with each edge labeled using the extracted modifiers.

4.4.3 Relation Synthesis Model Evaluation

The DEER work has demonstrated the capability of the relation synthesis model to generate easily understandable relation descriptions. However, in the biomedical domain, it is crucial for the model to generate truthful sentences and not mislead the reader with erroneous information. Table 4.3 provides an example of the model's generation, where the extracted relation descriptions for (*COVID-19*, *Pneumonia*) and (*Pneumonia*, *Vaccines*) are the inputs to the model. The 1-hop relation summary is the summarized relation description over the sentences of one pair of entities, and the 2-hop relation synthesis is the synthesized relation description for (*COVID-19*, *Vaccines*) through aggregating the 2-hop path (*COVID-19*, *Pneumonia*, *Vaccines*).

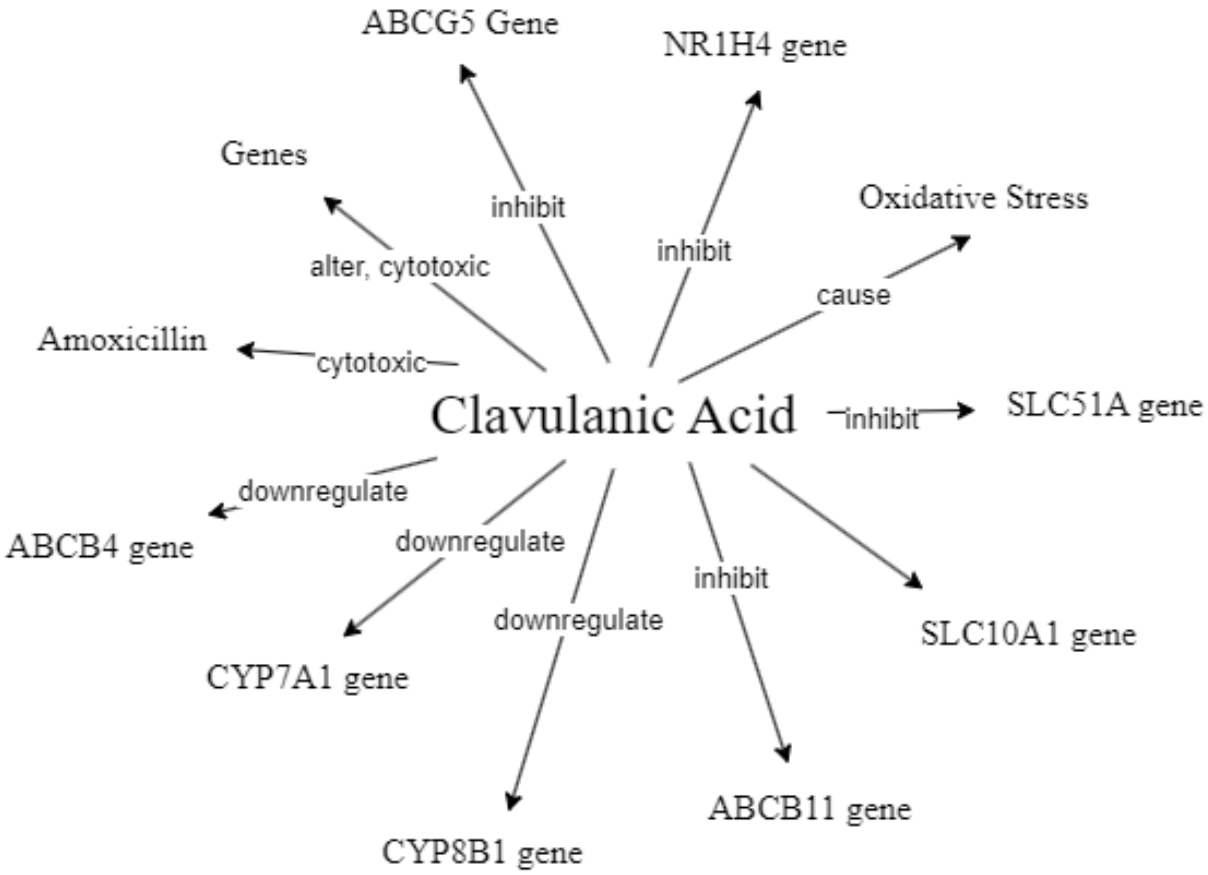

Figure 4.6: The local graph built from the passage in Figure 4.5

To evaluate the model's faithfulness, we randomly selected 20 samples from the test dataset and provided an evaluator with the input relation descriptions, the expected relation description output, and the generation of the model for each sample. The evaluator was instructed to find supporting evidence from the input for the generation and give a score from 1 to 5 for each generation to indicate its faithfulness to the input. The final average score for the 20 samples is 4.10, indicating that the generation is generally supported by the input. However, there is still a gap before we can fully trust it and we suggest users read the retrieved sentences to acquire reliable knowledge and only use the generated relation description as a reference. We observed that the model tends to copy or make minor modifications to the input relation description if the input already contains the target entity pair. During training, some input relation descriptions were already in a good form, and the model could get a high score by finding these input sentences and copying them to the output. This feature helps the human evaluator find supporting evidence and enhances the model's faithfulness.

CHAPTER 5: CONCLUSION AND DISCUSSION

5.1 DEER

In this work, 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 [12] and knowledge graph completion [13].

Limitations 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.

5.2 COVIDDEER

In the CovidDEER work, we developed a retrieval system in the biomedical domain that operates on a COVID-related corpus, facilitating efficient retrieval of relational knowledge and enabling tasks such as drug repurposing and literature curation. We demonstrate the advantages of managing a raw text corpus in a descriptive knowledge graph, including streamlined management, support for multi-hop reasoning across sentences from various articles, and comprehensive visualization of entity connections in the domain. Additionally, we equipped users with a modifier filtering module and a relation synthesis model that offer an overview of the relations on the edge before reading, and an article parser tool to aid in user tasks. In future work, we aim to enhance the accuracy and reliability of the relation descriptions generated for user reference.

REFERENCES

[1] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[2] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier et al., "Knowledge graphs," *Synthesis Lectures on Data, Semantics, and Knowledge*, vol. 12, no. 2, pp. 1–257, 2021.

[3] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[4] N. Lao, T. Mitchell, and W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 529–539.

[5] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 564–573.

[6] W. Chen, W. Xiong, X. Yan, and W. Y. Wang, "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)*, 2018, pp. 1823–1832.

[7] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[8] J. Huang, K. Chang, J. Xiong, and W.-M. Hwu, "Open relation modeling: Learning to define relations between entities," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 297–308.

[9] T. Noraset, C. Liang, L. Birnbaum, and D. Downey, "Definition modeling: Learning to define word embeddings in natural language," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[10] L. Cheng, D. Wu, L. Bing, Y. Zhang, Z. Jie, W. Lu, and L. Si, "Ent-desc: Entity description generation by exploring knowledge graph," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1187–1197.

[11] J. Huang, H. Shao, K. C.-C. Chang, J. Xiong, and W.-m. Hwu, "Understanding jargon: Combining extraction and generation for definition modeling," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[12] N. Bach and S. Badaskar, "A review of relation extraction," *Literature review for Language and Statistics II*, vol. 2, pp. 1–15, 2007.

[13] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[14] N. Voskarides, E. Meij, M. Tsagkias, M. De Rijke, and W. Weerkamp, "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)*, 2015, pp. 564–574.

[15] J. Huang, W. Zhang, S. Zhao, S. Ding, and H. Wang, "Learning to explain entity relationships by pairwise ranking with convolutional neural networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 4018–4025.

[16] N. Voskarides, E. Meij, and M. d. Rijke, "Generating descriptions of entity relationships," in *European Conference on Information Retrieval*. Springer, 2017, pp. 317–330.

[17] A. Handler and B. O'Connor, "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)*, 2018, pp. 1760–1769.

[18] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "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*, 2020, pp. 7871–7880.

[19] B. Y. Lin, W. Zhou, M. Shen, P. Zhou, C. Bhagavatula, Y. Choi, and X. Ren, "Commongen: A constrained text generation challenge for generative commonsense reasoning," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1823–1840.

[20] Y. Liu, Y. Wan, L. He, H. Peng, and S. Y. Philip, "Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 7, 2021, pp. 6418–6425.

[21] P. Dognin, I. Melnyk, I. Padhi, C. dos Santos, and P. Das, "Dualtkb: A dual learning bridge between text and knowledge base," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 8605–8616.

[22] O. Agarwal, H. Ge, S. Shakeri, and R. Al-Rfou, "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*, 2021, pp. 3554–3565.

[23] K. Kukich, "Design of a knowledge-based report generator," in *21st Annual Meeting of the Association for Computational Linguistics*, 1983, pp. 145–150.

[24] K. Gunaratna, Y. Wang, and H. Jin, "Entity context graph: Learning entity representations fromsemi-structured textual sources on the web," *arXiv preprint arXiv:2103.15950*, 2021.

[25] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto, "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*, 2020, pp. 23–30.

[26] F. Wu and D. S. Weld, "Open information extraction using wikipedia," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010, pp. 118–127.

[27] W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang, "A survey of knowledge-enhanced text generation," *arXiv preprint arXiv:2010.04389*, 2020.

[28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020.

[29] G. Izacard and E. Grave, "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*. Association for Computational Linguistics, 2021, pp. 874–880.

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[31] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[32] S. Banerjee and A. Lavie, "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*, 2005, pp. 65–72.

[33] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2019.

[34] Q. Wang, M. Li, X. Wang, N. Parulian, G. Han, J. Ma, J. Tu, Y. Lin, R. H. Zhang, W. Liu, A. Chauhan, Y. Guan, B. Li, R. Li, X. Song, Y. Fung, H. Ji, J. Han, S.-F. Chang, J. Pustejovsky, J. Rah, D. Liem, A. ELsayed, M. Palmer, C. Voss, C. Schneider, and B. Onyshkevych, "COVID-19 Literature Knowledge Graph Construction and Drug Repurposing Report Generation," pp. 66–77, 7 2020. [Online]. Available: https://arxiv.org/abs/2007.00576v6

[35] E. Voorhees, T. Alam, S. Bedrick, D. Demner-Fushman, W. R. Hersh, K. Lo, K. Roberts, I. Soboroff, and L. L. Wang, "TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection." [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/

[36] H. Taub-Tabib, M. Shlain, S. Sadde, D. Lahav, M. Eyal, Y. Cohen, and Y. Goldberg, "Interactive Extractive Search over Biomedical Corpora," *BioNLP*, pp. 28–37, 6 2020. [Online]. Available: https://arxiv.org/abs/2006.04148v1

[37] A. Köksal, H. Dönmez, E. Ozkirimli, A. Arzucan¨, and A. A. Arzucan¨ozgür, "Vapur: A Search Engine to Find Related Protein-Compound Pairs in COVID-19 Literature," *EMNLP NLP-COVID*, 9 2020. [Online]. Available: https://arxiv.org/abs/2009.02526v3

[38] H. Kilicoglu, G. Rosemblat, M. Fiszman, and D. Shin, "Broad-coverage biomedical relation extraction with SemRep," *BMC Bioinformatics*, vol. 21, no. 1, pp. 1–28, 5 2020. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-3517-7http://creativecommons.org/publicdomain/zero/1.0/

[39] C. Deng, J. Zou, J. Deng, and M. Bai, "Extraction of gene-disease association from literature using BioBERT," *ACM International Conference Proceeding Series*, vol. PartF168982, 1 2021. [Online]. Available: https://doi.org/10.1145/3448734.3450772

[40] X. Wang, Y. Guan, W. Liu, A. Chauhan, E. Jiang, Q. Li, D. Liem, D. Sigdel, J. H. Caufield, P. Ping, and J. Han, "EVIDENCEMINER: Textual Evidence Discovery for Life Sciences," pp. 56–62, 7 2020. [Online]. Available: https://aclanthology.org/2020.acl-demos.8

[41] T. C. Wiegers, A. P. Davis, K. B. Cohen, L. Hirschman, and C. J. Mattingly, "Text mining and manual curation of chemical-gene-disease networks for the Comparative Toxicogenomics Database (CTD)," *BMC Bioinformatics*, vol. 10, no. 1, p. 326, 10 2009. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-10-326

[42] J. Huang, K. Zhu, K. C.-C. Chang, J. Xiong, and W.-m. Hwu, "DEER: Descriptive Knowledge Graph for Explaining Entity Relationships," 5 2022. [Online]. Available: https://arxiv.org/abs/2205.10479v2

[43] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. Kinney, Z. Liu, W. Merrill, P. Mooney, D. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. Raymond, D. S. Weld, O. Etzioni, and S. Kohlmeier, "CORD-19: The Covid-19 Open Research Dataset," *ArXiv*, 4 2020. [Online]. Available: /pmc/articles/PMC7251955//pmc/articles/PMC7251955/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC7251955/

[44] M. Neumann, D. King, I. Beltagy, and W. Ammar, "ScispaCy: Fast and robust models for biomedical natural language processing," in *BioNLP 2019 - SIGBioMed Workshop on Biomedical Natural Language Processing, Proceedings of the 18th BioNLP Workshop and Shared Task*. Association for Computational Linguistics (ACL), 2 2019, pp. 319–327.

[45] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, J. Wiegers, T. C. Wiegers, and C. J. Mattingly, "Comparative Toxicogenomics Database (CTD): update 2021," *Nucleic Acids Research*, vol. 49, no. D1, pp. D1138–D1143, 1 2021. [Online]. Available: https://academic.oup.com/nar/article/49/D1/D1138/5929242