Link Prediction Based on Large Language Model and Knowledge Graph Retrieval under Open-World and Resource-Restricted Environment

Ryu Takeda
rtakeda@sanken.osaka-u.ac.jp
SANKEN, Osaka University
Ibaraki, Osaka, Japan

Hokuto Munakata
h_munakata@ei.sanken.osaka-u.ac.jp
SANKEN, Osaka University
Ibaraki, Osaka, Japan

Kazunori Komatani
komatani@sanken.osaka-u.ac.jp
SANKEN, Osaka University
Ibaraki, Osaka, Japan

ABSTRACT

A pre-trained large language model (LLM) has the potential to solve a knowledge graph completion (KGC) problem, i.e., link prediction, under an open-world environment where unknown entities of knowledge graph (KG) need to be predicted. However, many studies on LLM-based KGC either require other resources than the given KG for constructing LLM prompts or cannot predict the unknown entities consisting of several words, e.g., noun phrases such as seafood pizza, minced pork, etc. We therefore propose a probabilistic model based on a latent space model that incorporates LLM for such environments. Monte Carlo implementation of the model enables us to obtain KG-entities and their posterior probability even when they consist of several words. Prompts for LLM are constructed by retrieving related facts from the given KG, which does not require additional resources. We utilized both domain-specific KG extracted from Wikidata and human prediction results by crowdworkers in our closed- and open-world evaluations. The results showed that real unknown entities could be accurately predicted and that the number of retrieved facts affected the KGC performance.

CCS CONCEPTS

• **Computing methodologies** → *Information extraction.*

KEYWORDS

Link Prediction, Open-world Environment, Knowledge Graph Retrieval, Large Language Model

ACM Reference Format:
Ryu Takeda, Hokuto Munakata, and Kazunori Komatani. 2023. Link Prediction Based on Large Language Model and Knowledge Graph Retrieval under Open-World and Resource-Restricted Environment. In *Proceedings of 12th International Joint Conference On Knowledge Graphs (IJCKG2023).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

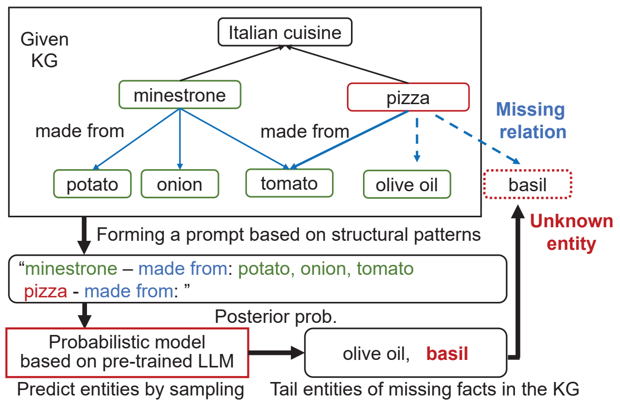
Given KG
Italian cuisine
minestrone
pizza
made from
made from
potato
onion
tomato
olive oil
basil
Missing relation
Unknown entity
Forming a prompt based on structural patterns
"minestrone – made from: potato, onion, tomato
pizza - made from: "
Posterior prob.
Probabilistic model based on pre-trained LLM
olive oil, basil
Predict entities by sampling
Tail entities of missing facts in the KG

Figure 1: Concept of our method to estimate a target triple: *("pizza", "made from", "?")*. **Dashed lines represent a missing entity.**

1 INTRODUCTION

1.1 Motivation

The construction of a *specialized* knowledge base (KB) is important when developing dialogue systems in a specific domain. A graph-structured KB, known as a knowledge graph (KG) [2, 30, 34], is widely utilized in many applications [38] due to its flexible representation of knowledge: for example, some studies have focused on a food-domain KG for food recommendation and dialogue systems [9, 13, 16]. KG is defined by sets of entities, relations, and facts represented as a triple (*head entity, relation, tail entity*), abbreviated hereafter as (h, r, t), e.g., (*"pizza", "made from", "olive oil"*).

Since KGs usually have missing entities, we need to fill them in (especially well-known ones) either manually or by machine learning methods to prevent poor system responses. For example, although "basil" can be an ingredient of "pizza", there is no link between basil and pizza in Wikidata. Because some people may not acknowledge the relationship, it is useful to represent such possibility by, e.g., posterior probability or confidence score. Additionally, the *well-known* entities themselves may not exist in the seed KG. Hereafter, we focus on the KG completion (KGC) problem: specifically, a *link prediction* problem where the tail entity t is predicted given a query of a head entity h and a relation r in a food-domain KG, as shown in the upper part of Fig. 1. The predicted entities will be utilized to make the system responses more reliable and accurate in some tasks, such as question answering and explanation of food recipe.

Missing entities of KG need to be completed even under an *open-world* and *resource-restricted* environment. An *open-world* environment [25] means that some entities may not be present in the KG, while a *closed-world* environment means that all entities are already known. For example, "basil" is a missing entity under open-world environment while "olive oil" is not in Fig. 1. It is necessary to predict any relationships between known entities and entities not in the KG, such as new words and orthographical variants. It is not realistic to assume that all such entities appearing in an application (e.g., dialogue) are included in KGs under an open-world environment. Additionally, domain-specific KGs may have no external resources such as definitions or descriptive texts about entities. In such cases, KGC should be performed utilizing only information of the KG itself, which means a resource-restricted environment.

Recent KGC methods utilizing pre-trained large language models (LLM) [1] have the potential to handle KGC under an *open-world* environment. Since LLMs are trained with a large amount of text data, missing triples of KG can be predicted by extracting fact information from the LLM. The textual representation of unknown entities is acceptable as an input sentence of LLM in both triple classification [20] and link prediction [22, 26] techniques. Here, the triple classification problem predicts whether a given triple (h, r, t) exists or not [23]. The design of the conditional sentence of LLM (input/*prompt*) affects the prediction performance, so template-based hard prompts are typically used in KGC [1]. Some methods require additional descriptions or definitions of an entity, or may require fine-tuning [4, 20, 26, 29].

The above LLM-based KGCs and prompt designs cannot be applied to our link prediction problem under an open-world and resource-restricted environment because unknown entities themselves are not directly estimated in the previous works, which means their candidates are required in advance due to their triple classification formulation. For example, if "basil" is given as a candidate for an unknown entity, the previous methods can only be applied by making a hypothetical triple ("pizza", "made from", "basil"). We need to prepare such entity candidates in some way. Although a single word entity is directly estimated via masked-LLM [22], entities consisting of several words, e.g., noun phrases such as seafood pizza and minced pork, should also be considered. Existing prompt strategies are also unusable because they require additional resources [26] or assume a triple classification setting [20]. While Cohen et al. [6] examined an automatic construction of KG using LLM from a few seed entities, their assumption is different from ours.

In light of the above, we propose a probabilistic model based on LLM and KG retrieval for the open-world link prediction problem. Generated text (character sequence) from LLM is treated as a latent variable in terms of a generative probabilistic model [5], which enables us to obtain the posterior probability of estimation. A tail entity is estimated and extracted from the text, and its posterior probability is then calculated through Monte Carlo approximation. Both single word and noun phrase are treated as candidates of entities in this paper. Prompts for LLM are constructed by retrieving facts related to a given query from the given KG itself, which means no additional resources are required. Both domain-specific KG extracted from Wikidata and real predicted facts by crowd-workers were utilized in our evaluation. We investigated 1) the link prediction performance of our retrieval-based prompts and 2) the prediction ability of real unknown entities using a crowdsourcing dataset by experiments.

1.2 Related Work

Our work is related to KGC methods based on embedding, pre-trained LLM with fine-tuning, and LLM with prompting (without fine-tuning). With the LLM-based methods, we focus on the "Access" operation mentioned in [1], such as the "Fine-tuning" and "Prompting" techniques.

Conventional embedding-based KGC approaches [3, 21, 28, 32, 35, 36] are well-known for their *closed-world* assumption. In particular, link prediction methods usually estimate a missing entity *from their entity candidates* by capturing the structural patterns of KG. A fact's confidence score, such as $p(z|h, r, t)$, which represents whether it is true $(z = 1)$ or false $(z = 0)$ is usually evaluated. Such score itself corresponds to the score for the triple classification problem. In the case of the link prediction setting, the score of each entity candidate $\hat{t}$, e.g. $p(z = 1|h, r, \hat{t})$, is first calculated one by one, and then the entity with a maximum score among candidates is estimated as the tail entity of $(h, r, ?)$.

Although KGC methods using a pre-trained LLM with fine-tuning (e.g., KG-BERT [37]) have performed well for certain kinds of KGs, we cannot straightforwardly apply them in an open-world environment because they require the embedding of missing entities. COMET (commonsense transformers) [4] presented an automatic knowledge base construction from a seed KG by utilizing LLM. Since COMET is trained to generate tail object tokens from a given head entity and relation tokens, this method is categorized as a "Fine-tuning" technique whereas our approach is categorized as a "Prompting" technique that does not assume fine-tuning. Note that our main research question is to clarify the impact of *KG-retrieval-based prompts* for link prediction (the way of knowledge utilization).

Some researchers have tackled the open-world environment by using graph neural networks [8] or word embeddings [25]. However, such methods also require entity candidates with auxiliary information to compute the posterior score of a fact. Hence, if we cannot prepare the candidates in advance, or if they are too numerous to calculate each score by enumerating facts, such methods are not feasible.

As stated earlier, other approaches based on LLM for the open-world environment are problematic in terms of unknown entity prediction and prompt design. As for the prompt design, although discrete or soft prompting is now widely utilized in the natural language processing field, relatively few researchers have applied it to the KGC problem. For example, there are manually designed prompts based on a query for missing entities [22], prompts that exploit external text data [26, 29], and soft prompts with fine-tuning [20]. In addition, prompts based on multiple sub-tasks have been utilized for the automatic construction of KG from seed entities [6].

2 PRELIMINARIES

2.1 KGC in Open-world Environment

Let $\mathcal{E}_c$ and $\mathcal{R}_c$ denote a set of closed entities and relations, i.e., entities and relations that are present in the existing KG. Let also $\mathcal{E}_o \supseteq \mathcal{E}_c$ denote a set of open entities, which includes missing unknown entities. We define an *unknown* entity as an entity that does not appear in the triples or textual information of the KG. Note that no list of unknown entities is available because literally no information is given about them. The existing (given) KGs consist of a set of facts $\mathcal{G} \subset (\mathcal{E}_c \times \mathcal{R} \times \mathcal{E}_c)$. Our objective is to find a set of gold facts holding all the true facts, including the missing facts denoted by $\mathcal{G}_{gold} \subset (\mathcal{E}_o \times \mathcal{R} \times \mathcal{E}_o)$. Note that $\mathcal{G} \subset \mathcal{G}_{gold}$. Hence, KGC in an open environment is defined as finding facts $x \in \mathcal{G}_{gold} \backslash \mathcal{G}$, and similarly, KGC in a closed-world environment is defined as finding facts $x \in (\mathcal{E}_c \times \mathcal{R} \times \mathcal{E}_c) \backslash \mathcal{G}$. We assume that all relations are known because the definition of relations is well maintained thanks to the research area of ontology.

We formulate KGC as a link prediction task, i.e., predicting the tail entity t for a given missing fact $(h, r, ?)$ in $\mathcal{G}_{gold} \backslash \mathcal{G}$. Here, all possible notions can be candidates of the tail entity in the open-world environment. To clarify the problem setting, we assume that the candidates can be represented by flexible text (character sequences), not by a fixed label set or an embedding set.

Since entities indicating food names often consist of several noun words, both of a single word and noun phrase including several words can be candidates of a tail entity in our work. For example, "*seafood pizza*" includes two words. This is the key advantage of our link prediction method as compared to Petroni et al. [22], which only predicted a single word/token entity.

2.2 Language Model for Text Generation

An autoregressive language model is usually formulated as conditional probability $p(w_l|w_1, ..., w_{l-1})$ with token symbols $w_i (i = 1, ..., l)$. This model calculates the probability that token w_l follows input token sequence $w_1, ..., w_{l-1}$. Here, the token can be a word or subword such as a sentence piece [14].

The output text (character sequence) W is generated from the pre-trained LM by sampling in an autoregressive manner. For example, w_l^* is drawn from $p(w_l|w_1^*, ..., w_{l-1}^*)$, and $p(w_{l+1}|w_1^*, .., .w_l^*)$ is evaluated using w_l^* to draw w_{l+1} until the number of drawn tokens reaches a certain threshold or the "end of sentence" token, such as [EOS], is sampled. Here, the symbol * means an actual drawn token, and the initial token w_1^* is usually the "start of sentence" token [SOS]. If the token is a word, the output text W is a concatenation of the generated tokens $[w_1, ..., w_L]$ with length L. If the token is a sentence-piece, the output text is recovered from the drawn tokens by a appropriate processing. In contrast to a bi-directional LM, such as BERT [10] or RoBERTa [18], the autoregressive model can generate tokens with various lengths. This generation scheme is suitable for KGC in our usage scenario because entity candidates may consist of several tokens.

Transformer-based neural LMs [33] (e.g., GPT [31]) are trained with a massive amount of data: for example, the training data for GPT-3 collected by web crawling exceeded 570 GB after preprocessing. We therefore assume that LM has rich information about missing unknown entity $e \in \mathcal{E}_o \backslash \mathcal{E}_c$ through training because massive training datasets include almost all the noun phrases, especially *well-known* ones that can be prediction candidates.

GPT-2, 3, and 4 no longer require fine-tuning, and only prompts (conditional sentences of LM) are required. For example, a prompt might be "*pizza - made from:*" and the output is "*cheese*" [31]. To improve the performance, the prompt is combined with a few examples such as "*minestrone - made from: potatoes*." The input text template or format utilized for the prompt usually affects the prediction performance or the structure of the output text. As such, both the text format and text content (information) used for the prompt are important.

3 PROPOSED METHOD

In this section, we introduce our probabilistic generative model based on LLM for link prediction and implementation based on Monte Carlo sampling. We then explain our strategy for prompts based on retrieving facts related with a given query in the given KG to improve the prediction performance. Figure 2 presents an overview of our proposed method consisting of KG-retrieval and entity- sampling steps.

3.1 Probabilistic Formulation and Entity Sampling via LLM

Our formulation of link prediction is based on the posterior probability $p(\tilde{t}|h_T, r, \mathcal{G})$ that represents the conditional probability of $\tilde{t}$ being the tail of a triple, given the head entity h_T, the relation r, and the KG of $\mathcal{G}$. This probability is utilized as the confidence score. We assume the posterior probability is represented as the following marginalized probability:

$$p(\tilde{t}|h_T, r, \mathcal{G}) = \sum_W p_{\mathrm{E}}(\tilde{t}|W, \mathcal{G}) p_{\mathrm{LM}}(W|h_T, r, \mathcal{G}), \quad (1)$$

where W represents a latent text in terms of a probabilistic model [5]. $p_{\mathrm{LM}}(W|h_T, r, \mathcal{G})$ is a conditional probability of W being the latent text, given the head entity h_T, the relation r and the KG $\mathcal{G}$, and $p_{\mathrm{E}}(\tilde{t}|W, \mathcal{G})$ represents a conditional probability of a tail entity $\tilde{t}$ given the text W and the KG $\mathcal{G}$. Here, $p_{\mathrm{LM}}(W|h_T, r, \mathcal{G})$ is implemented by LLM, and W corresponds to (recovered) tokens $w_l (l = 1, ..., L)$ generated from LLM. h_T, r, and $\mathcal{G}$ in p_{LM} correspond to information for the prompt of LLM. Since $\mathcal{G}$ itself is too large for prompting, it is replaced by a specific example E in $\mathcal{G}$, i.e., $p_{\mathrm{LM}}(W|h_T, r, E)$. $p_{\mathrm{E}}(\tilde{t}|W, \mathcal{G})$ generally measures the similarity between entity candidates $\tilde{t}$ and text W. It is also interpreted as an entity generator using the given text W in the case of Monte Carlo sampling. We can utilize the information of $\mathcal{G}$ in p_{E} if necessary (e.g., for entity identification or named entity recognition).

We evaluate the posterior probability using the Monte Carlo method because a noun phrase (i.e., $\tilde{t}$) generally consists of two or more words, which complicates defining it as a probability. The latent text W is drawn from p_{LM}, and $\tilde{t}$ is then drawn from p_{E} given the W. The posterior probability is approximated by normalizing the frequency of the drawn samples. Note that this formulation can be applied to any LLM and to approximate the posterior probability of predicted entities.

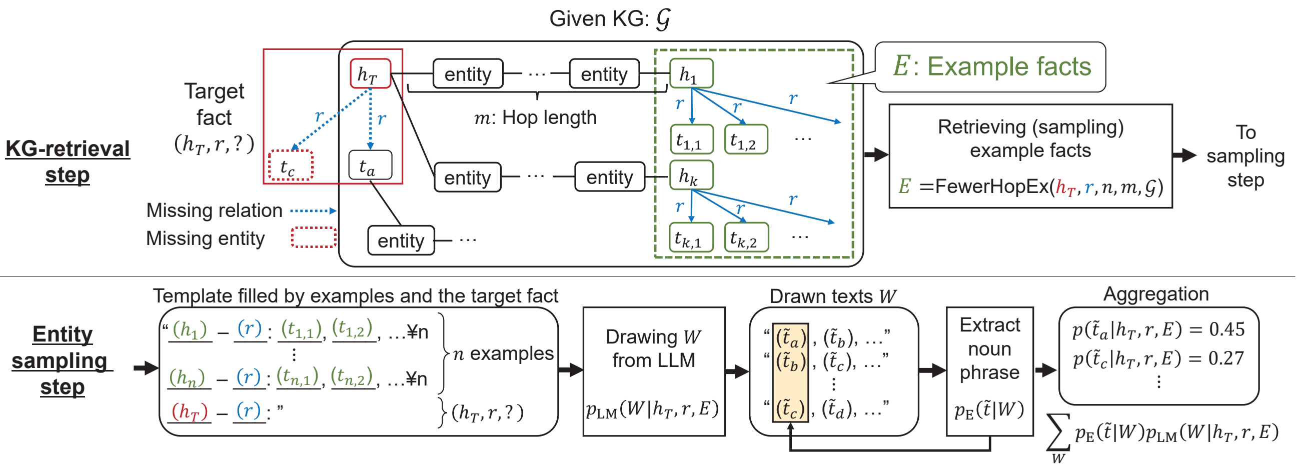
Given KG: $\mathcal{G}$
KG-retrieval step
Target fact $(h_T, r, ?)$
Missing relation
Missing entity
m: Hop length
E: Example facts
Retrieving (sampling) example facts
E =FewerHopEx($h_T, r, n, m, \mathcal{G}$)
To sampling step
Entity sampling step
Template filled by examples and the target fact
n examples
Drawing W from LLM
$p_{\mathrm{LLM}}(W|h_T, r, E)$
Drawn texts W
Extract noun phrase
$p_{\mathrm{E}}(\tilde{t}|W)$
Aggregation
$p(\tilde{t}_a|h_T, r, E) = 0.45$
$p(\tilde{t}_c|h_T, r, E) = 0.27$
$\sum_W p_{\mathrm{E}}(\tilde{t}|W) p_{\mathrm{LLM}}(W|h_T, r, E)$

Figure 2: Overview of proposed method. Target fact (h_T, r, ?) and n examples $\{\{(h_1, r, t_{1,1}), (h_1, r, t_{1,2}, ...)\}$, ..., $\{(h_n, r, t_{n,1}), (h_n, r, t_{n,2}, ...)\}\}$, related to h_T, are retrieved from KG by our algorithm. Obtained facts are then incorporated into a template (lower left) from which output texts are repeatedly generated by LLM. Finally, the first noun phrase is extracted as predicted fact $(\tilde{t}_a, \tilde{t}_b, ...)$ and aggregated as $p(\tilde{t}|h_T, r, E)$.

The details of each process are as follows. First, we repeatedly generate text W by LLM with an input character sentence (prompts) using a template text filled with h, r, and example E in the graph $\mathcal{G}$. (The details of the template and prompt strategy are described in the next section.) Second, we estimate and extract the tail entity $\tilde{t}$ from the W. Here, we utilize the tendency that such a delimiter as "," appears with high probability after the noun phrase under our heuristic prompt. This tendency is helpful for estimating entities in the case of Japanese text because word segmentation method or morphological analyzer is usually required to parse text. In this case, $p_{\mathrm{E}}(\tilde{t}|W, \mathcal{G})$ works like a *delta* function $\delta(\tilde{t} - t')$ with the first noun phrase t' in W. For example, if the output text is "*mozzarella cheese, tomato, basil, ...*", "*mozzarella cheese*" is the extracted noun phrase, i.e., $\hat{t}$. Note that since a delimiter does not always appear, this case remains a failure (no entity drawn).

3.2 Prompt based on KG-retrieval

Previous research has shown that using the appropriate prompts improves the performance in many tasks [11]. For KGC, a straightforward approach to creating prompts is to give a total KG $\mathcal{G}$ as a text, as mentioned earlier. This strategy is obviously infeasible in terms of computational cost, and it will degrade the prediction performance without LLM fine-tuning due to the inclusion of irrelevant facts in prompts. Therefore, we retrieve a small number of examples relevant to the given query from KG and utilize them to fill our template to construct prompts. The influence of irrelevant facts on the link prediction performance will be discussed later.

Our prompt is based on the following two restrictions on examples considering the head entity h_T and the relation r in a given query.

(1) The relations for the examples are identical to the relation r.
(2) The head entities for the examples are reached from the head entity h_T in fewer hops in the KG.

The first restriction ensures relational consistency between examples and the query. We assume the relation set used in Wikidata. If we use relations different from the target relation r for the examples, prompts become ambiguous and the performance degrades. For example, the relation r of the query is "*said to be the same as*" and one of the examples is "*different from*". This is because the relation "different from" can be applied to almost all entities in the KG. The second restriction ensures entity's relevance to the given head entity h_T. Here, we assume that the relevance between the entity h_T and the head entities of an example fact can be measured by the hop length (distance), e.g., the number of relations (edges in graph) in the shortest path connecting them. Therefore, "fewer hops" means strong relevance with h_T.

Algorithm 1 shows **FewerHopsEx**, which samples examples with fewer hops from KG, and Algorithm 2 shows **RandEx**, a sampling algorithm without our second restriction, which we also utilized for ablation study. Tails$(x, r, \mathcal{G})$ is a function that gives a set of tail entities of the fact $(x, r, ?)$ in $\mathcal{G}$, and kHopEntities$(h_T, k, \mathcal{G})$ is a function that gives entities reached from entity h_T in just k undirected hops in the shortest path in $\mathcal{G}$. $|\cdot|$ represents the number of elements in the set. Y, which is the return value of Tails, is a set of tail entities, i.e., $\{y_1, y_2, ...\}$ and (x, r, Y) form a set of facts $\{(x, r, y_1), (x, r, y_2), ...\}$. Hence, E is nested by example head entities. Each algorithm obtains n examples, including relation r from $\mathcal{G}$. Specifically, **FewerHopsEx** gives examples with head entities that were reached from a given h_T within m hops. If there are fewer entities than n in the m hops, we substitute examples sampled from **RandEx** because they reduce the calculation cost, and entities with many hops are no longer related to h_T. Sampling$(\mathcal{G})$ in **RandEx** is a function that samples randomly from $\mathcal{G}$. In line 4 of **FewerHopsEx**, x is taken randomly from X.

These sampled examples are utilized for the template filling shown in the lower left of Fig. 2. In the template, "–" separates the head entity and the relation, ":" separates the relation and the tail

Algorithm 1 FewerHopsEx: Sampling facts for examples with fewer hops

```
Require: $h_T, r, n, m, \mathcal{G}$
 1: $E = \{\phi\}$
 2: for $k = 1, ...m$ do
 3:     $X = \text{kHopEntities}(h_T, k, \mathcal{G})$
 4:     for $x \in X$ do
 5:         $Y = \text{Tails}(x, r, \mathcal{G})$
 6:         if $|Y| \neq 0$ then
 7:             $E = E \cup \{(x, r, Y)\}$
 8:         end if
 9:         if $|E| = n$ then
10:             return $E$
11:         end if
12:     end for
13: end for
14: $E = E \cup \text{RandEx}(r, n - |E|, \mathcal{G})$
15: return $E$
```

Algorithm 2 RandEx:Sampling facts for examples by random sampling

```
Require: $r, n, \mathcal{G}$
1: $E = \{\phi\}$
2: while $|E| \leq n$ do
3:     $x \sim \text{Sampling}(\mathcal{G})$
4:     $Y = \text{Tails}(x, r, \mathcal{G})$
5:     if $|Y| \neq 0$ then
6:         $E = E \cup (x, r, Y)$
7:     end if
8: end while
9: return $E$
```

entity, and each tail entity is separated by a comma. The new line marker separates each fact. By using these specific symbols, generated text tends to be the same format of the template: the output text W is also comma-separated, so parsing it to extract an entity becomes easy. Note that there is a possibility that LLM may answer the same words in the prompts. Therefore, to avoid trivial answers that may degrade the link prediction accuracy, some restriction sentences may be required in the prompt, for example, "don't reuse these words" to avoid trivial answers that may degrade link prediction accuracy. In the evaluation of link prediction, the "known" triples that may include entities in the prompts are excluded in the "filtered setting".

4 EXPERIMENT

We conducted experiments for both open- and closed-world environments with a food subgraph of Japanese Wikidata and real facts predicted by crowdworkers.

4.1 Dataset

We used a subgraph of Japanese Wikidata [34] related to food entities. In addition, we utilized crowdsourcing to collect ground-truth data for real missing facts. The items in the food subgraph were used for example generation in the LLM prompts and as training data for the baseline KGC methods, while the crowdsourcing data were utilized as a test set. We focused on the six relations that frequently appeared in the KG: "*subclass of*", "*has part*", "*made from material*", "*has use*", "*different from*", and "*said to be the same as*". Note that the relations between two food-related entities were almost always oen of these six. In the actual prompt, we used Japanse translations of these names, e.g., "材料" for the relation "made from material".

For the food subgraph, we extracted seed entities that were reached from the entity "*food*" within ten hops of "*subclass of*" as a tail entity. In addition, we extracted the tail entities connected with the extracted seed entities. Finally, the facts consisting of the extracted head and tail entities were obtained. Almost all extracted entities were food-related ones. For baselines that require training or fine-tuning, we divided the food subgraph into training and validation datasets at a ratio of 9:1.

We hired 201 crowdworkers and obtained 55,807 facts. The crowdworkers were presented with the head entity h, selected the relation r out of six, and filled the corresponding tail entity t of (h, r, t) in Japanese. We had selected the h from the entities of the food subgraph with the top 1,000 hits in an Internet search to narrow down popular (well-known) entities. This is because one of our purposes is to provide the missing well-known facts in KG. Crowdworkders could search the Internet if they did not know a head or tail entity. Since our objective is to determine the distribution of tail entities and the well-known entities, we feel that the influence of noise responses and the agreement rate among workers are not important for our experiments.

Table 1: Summary of dataset statistics. $|\mathcal{E}|$, $|\mathcal{R}|$, and $|\mathcal{G}|$ represent the number of entities, relations, and facts. FS, WD, and CS represent the food subgraph, Wikidata, and crowdsourcing data. In the test dataset, "Closed" means that the subset of the crowdsourcing data consists of entities included in food subgraphs, and "Open" means that the whole set of crowdsourcing data is included. Facts included in Wikidata were removed from the crowdsourcing data.

Dataset		$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{G}\rvert$
-	**WD**	2,414,044	969	13,584,645
Training	**FS**	8,699	110	15,029
Test	**CS** (Closed)	1,538	6	12,939
	CS (Open)	7,187	6	27,177

The statistics of our dataset are presented in Table 1. We extracted 8,699 entities, 110 relations, and 15,029 facts for the food subgraph from Wikidata. After duplicated facts of crowdsourcing data were removed, 7,187 entities and 27,177 facts remained. For the evaluation described in the next section, we generated a closed-world subset from the total crowdsourcing data, i.e., the subgraph consisting of entities that appeared in the food subgraph. The subset included 1,538 entities and 12,939 facts. Figure 3 shows the relationships between the entity sets of each dataset.

The tail entities of the crowdsourcing data utilized for the test dataset and the predicted results of our method were susceptible to notation inconsistency. In Japanese, some words have the same

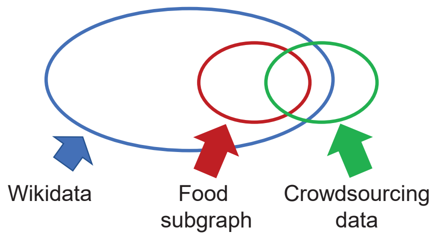

Figure 3: Venn diagram of entity set of each dataset.

meaning and the same reading but can have different notations, i.e., both “ジャガイモ” and “じゃがいも” mean “*potato*” and have the same pronunciation. We therefore reduced the effect of notation inconsistency by pronunciation matching, where we gave each entity a pronunciation by MeCab [15] (a Japanese tokenizer) and identified entities with an identical pronunciation.

4.2 Evaluation

We adopted Hits@k as an evaluation metric, which represents the ratio of the true tail entity t of $(h_T, r, ?)$ that is scored within the top k as the predicted entity $\hat{t}$. To more accurately evaluate facts with many true tail entities, we excluded true facts in the training, validation, and test datasets (except the target fact) from the predicted entities (i.e., filtered setting), as in a previous work [3].

First, we evaluated the performance of the closed-world environment to compare our proposed method with the baselines by using closed-world test data constructed from the crowdsourcing data. The closed-world test data included only the entities that appeared in the food subgraph. For our proposed method, we removed the predicted tail entities that do not exist in the food subgraph from the prediction result.

We utilized the following methods as baselines for the closed-world environment: KG-BERT [37], TransE [3], and ComplEx [32]. Here, TransE and KG-BERT were used as baselines of embedding-based and pre-trained LLM methods in [20]. We also set our method without the examples as a baseline because it closely resembles another method [22] except for the number of acceptable words (single or multiple).

As for the open-world environment, we used all of the crowdsourcing data as test data. With this setting, we evaluated the overall performance, the number of entities in the food subgraph the method can find, and the number of errors (i.e., prediction results without any food). Note that the open-world test data included all of the closed-world data.

4.3 Model Configuration

We used “japanese-gpt-1b[1]”, a GPT-2 based model with 1.3 billion parameters trained with Japanese text data as an LM. The training data were made from Japanese Wikipedia data, C4 [24], and CC-100 [7]. The number of hops m was set to 5, and the number of examples n was examined from $\{1, 3, 5\}$. The text W was sampled 3,000 times from LLM to approximate the posterior probability. As delimiters, we adopted commas, periods, dots, and brackets in the Japanese and English notations.

[1]https://huggingface.co/rinna/japanese-gpt-1b

Table 2: Average prediction performance in closed-world environment: n is the number of examples used in prompt.

	Method	Hits@10	Hits@1
Baseline	TransE	0.176	0.011
	ComplEx	0.048	0.005
	KG-BERT	0.198	0.030
Baseline (for ablation study)	No Examples	0.321	0.091
	RandEx ($n = 1$)	0.330	0.078
	RandEx ($n = 3$)	0.383	0.118
	RandEx ($n = 5$)	0.398	0.116
Proposed	FewerHopEx ($n = 1$)	0.399	0.112
	FewerHopEx ($n = 3$)	0.426	0.127
	FewerHopEx ($n = 5$)	**0.446**	**0.136**

The hyper parameters of the baselines were optimized with Hits@10 scores of the validation dataset. The negative sampling ratio of the embedding-based methods was from 1 to 5 for TransE and ComplEx, and fixed at 5 for KG-BERT. The learning ratio for all the models was 1.0×10^{-5}. The embedding dimensions of TransE were from 50 to 500 and of ComplEx were from 5 to 300. We used Japanese BERT for KG-BERT(available on GitHub[2]). As an optimizer, we adopted Adam [12] for TransE and ComplEx, and AdamW [19] for KG-BERT.

4.4 Results

Table 2 shows the average Hits@k scores in the closed-world environment for our proposed method and the baselines. As we can see, our method outperformed the best of the baselines (KG-BERT) by at least 0.10 for Hits@10, and **FewerHopsEx** outperformed **RandEx** by 0.125 for Hits@10 and by 0.045 for Hits@1. In particular, Hits@k improved with more examples ($n = 3, 5$). Considering that the hop length of **RandEx** was higher than that of **FewerHopsEx**, this result implies that entities with fewer hops to the target head entity contained more information about the target head entity and thus contributed to effective knowledge extraction from the pre-trained LM.

Figure 4 shows the detailed relationship between Hit@10 and the number of facts of the food subgraph that are available for the training and predictions. We found relation-dependent performance differences between our proposed method and the baselines, especially for “*made from material*”. In particular, **FewerHopsEx** with five examples outperformed KG-BERT for “*made from material*” by 0.326. On the other hand, there was only a small difference for “*said to be the same as*”. This result indicates that the performance of our method was affected by the meaning of the relation. The knowledge tended to be extracted from the LLM correctly for relations in which the answer for $(h_T, r, ?)$ was clearly defined as true or false, such as with “*made from material*”. In contrast, the embedding-based method was less affected by the meaning of the relations. Most of the tail entities of the examples with “*has use*” sampled by **RandEx** were “*food additives*” and “*ph adjuster*”. This

[2]https://huggingface.co/cl-tohoku/bert-large-japanese

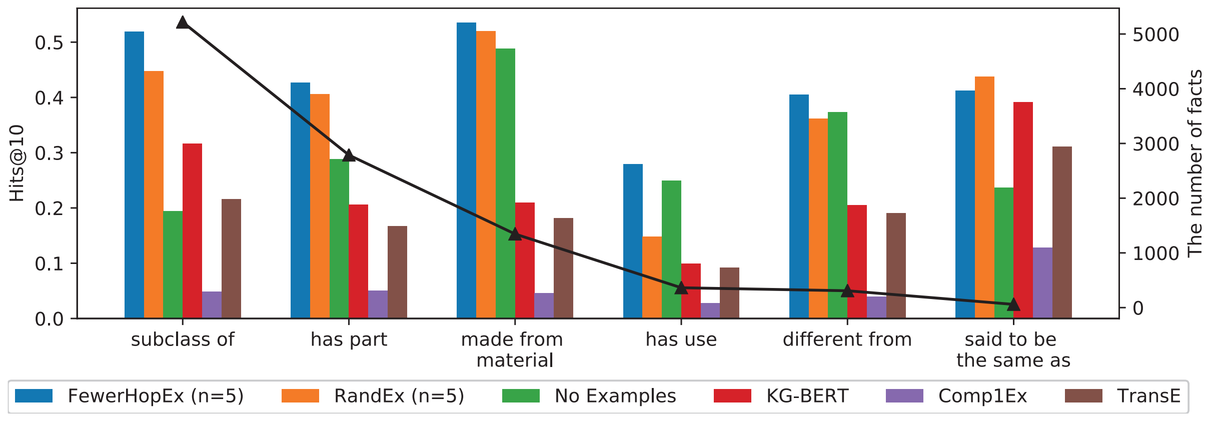

Figure 4: Hits@10 and the number of facts of the food subgraph available for prediction by six relations. Left axis and bar graph represent Hits@10, and right axis and black line graph represent the number of facts.

Table 3: Average prediction performance under open-world environment. *n* is the number of examples used in prompt. The right side of the table shows the ratio of the top 10 predicted entities included in the entity set of each dataset. FS, WD, and CS represent the food subgraph, Wikidata, and crowdsourcing data. The relationships between entity sets follows that in Fig. 3. "None" represents the ratio of the top 10 predicted entities not included in the entity set of any dataset.

	Method	Evaluation metrics		Ratio of top 10 predicted entities included in dataset				
		Hits@10	Hits@1	**FS**	**WD**	**CS**	**CS \ FS**	None
Baseline (for ablation study)	No Examples	0.155	0.032	0.434	0.685	0.593	0.169	0.234
	RandEx ($n = 1$)	0.182	0.038	0.537	0.785	0.794	0.186	0.123
	RandEx ($n = 3$)	0.218	0.059	0.671	0.807	0.842	0.194	0.095
	RandEx ($n = 5$)	0.230	0.058	0.670	0.803	0.853	0.201	0.093
Proposed	FewerHopEx ($n = 1$)	0.219	0.054	0.647	0.785	0.812	0.192	0.114
	FewerHopEx ($n = 3$)	0.240	0.062	0.689	0.814	0.838	0.181	0.093
	FewerHopEx ($n = 5$)	**0.251**	**0.066**	0.715	0.823	0.857	0.174	0.080

implies that the randomly sampled examples might be irrelevant to the target facts, especially for facts with relations that do not impose a strong constraint on the predicted entities to be foods. For example, when we predict the tail entity of "(pizza, made from, ?)", the examples based on food additives will be noise, and the generation probability of entities related with the food additives will be enhanced due to the attention mechanism of LLM. As a result, these examples obscured the KGC task and degraded the performance.

Table 3 shows the average Hits@k in the open-world environment and the ratio of the top 10 predicted entities included in the entity set of each dataset. Note that the relationships between the entity sets follow those in Fig. 3. **FewerHopsEx** was the best for both Hits@1 and Hits@10. "None" in Table 3 represents the ratio of the top 10 predicted entities not included in any dataset, i.e., the food subgraph, Wikidata, or crowdsourcing data. Most entities of "None" were not foods or were foods with notational inconsistency. We found that the predicted result with a higher number of examples tended to decrease the ratio of the predicted entities not included in any dataset. This result implies that the prediction with a higher number of examples about foods clarified KGC's task for the pre-trained LM and gave a constraint to output entities related to foods to the LM. On the other hand, considering the entity subset of the crowdsourcing data minus the food subgraph (i.e., **CS \ FS**) as newly discovered food entities, **RandHopsEx** with five examples was the best. In contrast, **RandHopsEx** with a large number of examples degraded the ratio of **CS \ FS**.

Table 4 shows examples of a prompt, predicted tail entities, and the true tail entity. We found that the prediction performance suffered from notational inconsistency. For example, although キャベツ (*cabbage*) and キャベツの葉 (*cabbage leaves*) are very close in meaning, they were treated as different entities. As another example, ピッツァ (*pizza*) means exactly the same things as ピザ (*pizza*), but ピッツァ (*pizza*) did not appear in any dataset. These cases occurred frequently and degraded the performance of the open-world environment, which is a problem of the design of the entity extractor $p_{\mathrm{E}}(\tilde{t}|W, \mathcal{G})$. On the other hand, entities that were obviously not foods were sometimes predicted, such as "ミケーレ (*Michele*)", but these entities were rarely among the top 10 predicted entities included in the crowdsourcing data (Fig. 3). Therefore, efficient named entity extraction from the output text and disambiguation of extracted entities are both necessary functions to achieve an effective p_{E}.

Table 4: Examples of a prompt, predicted tail entities and their score with FewerHopEx, and true tail entities. Italic font indicates English translations.

Prompt	Predicted tail entity: Score	True tail entity
チキンライス - 材料: ケチャップ，米，鶏肉 \n ロールキャベツ - 材料: *chicken rice - made from material:* *ketchup, rice, chicken \n* *cabbage roll - made from material:*	キャベツ (*cabbage*): 0.679, 玉ねぎ (*onion*): 0.056, 鶏肉 (*chicken*): 0.051, 野菜 (*vegetable*): 0.040, じゃがいも (*potato*): 0.027, 米 (*rice*): 0.023, ⋮	キャベツの葉 (*cabbage leaves*), 玉ねぎ (*onion*), 豚ひき肉 (*minced pork*) ひき肉 (*minced meat*) 肉 (*meat*), コンソメ (*consommé*)
乾パン - 同一とされる事物: ツヴィーバック \n マルゲリータ - 同一とされる事物: *hardtack - said to be the same as:* *zwieback \n* *Margherita - said to be the same as:*	ピザ (*pizza*): 0.380, ピッツァ (*pizza*): 0.072, パン (*bread*): 0.044, ツヴィーバック (*zwieback*): 0.037, ミケーレ (*Michele*): 0.027, ミケランジェロ (*Michelangelo*): 0.019, ⋮	ナポリピッツァ (*Neapolitan pizza*)

4.5 Discussion and Limitations

Although we demonstrated the effectiveness of our model and achieved retrieval-based prompts for LLM in link prediction, there are some limitations in terms of the generalization of our method. Note that our probabilistic formulation itself is general and the concept can be applied to other settings.

The generalization ability is limited to the language (Japanese), KG (food-domain), and LLM (GPT-2) used in our method. In particular, the typical graph structure of KG usually depends on its domain, so it is important to clarify the relationship between the property of KG and the performance of entity prediction. Evaluation using different languages and LLMs is also important to characterize our method because the construction of prompts is obviously affected by the performance of the LLM and the design of $p_{\mathrm{E}}(\tilde{t}|W,\mathcal{G})$. Although the format of our prompt is language-independent, the notation of relations and entities are language-dependent, which may affect the structure of text drawn from LLM. The modification of noun phrase extraction process may be required in such case. However, we can expect the performance of our link prediction method to improve when the latest LLM is applied.

5 CONCLUSION

We proposed a probabilistic model using a pre-trained LLM with prompts based on KG-retrieval for an open-world environment. Our formulation can provide the posterior probability of a predicted entity by assuming a latent text generated by LLM. We introduced a prompt with example facts consisting of similar entities reached from the target head entity within fewer hops. Both domain-specific KG extracted from Wikidata and real facts predicted by crowdworkers were utilized in our evaluation. Experimental results showed that our retrieval-based prompts improved the KGC performance and that real unknown entities were also accurately predicted by our model.

Our future work will take two directions. First, we will create more effective prompts by incorporating graph information in a prompt-tuning method [17, 27]. Second, we will adopt an embedding-based technique for the post-processing of our method. By calculating the similarity of the predicted entity in the embedding space, we should be able to reduce the notational inconsistency caused in $p_{\mathrm{E}}(\tilde{t}|W,\mathcal{G})$ and simplify the combinations with other applications using KG embeddings [35].

ACKNOWLEDGMENTS

This work was partly supported by JSPS KAKENHI Grant Numbers JP23H03457 and JP22H00536, and JST Moonshot R&D Grant Number JPMJPR1857, Japan.

REFERENCES

[1] Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A Review on Language Models as Knowledge Bases. *arCiv:2204.06031* (2022). arXiv:2204.06031 [cs.CL]

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proc. Int. Conf. Management of Data.* 1247–1250.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proc. NIPS.* 2787–2795.

[4] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proc. ACL.* 4762–4779. https://doi.org/10.18653/v1/P19-1470

[5] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning.* Springer-Verlag New York.

[6] Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling The Internal Knowledge-Base of Language Models. In *Proc. Findings of EACL.* 1856–1869. https://aclanthology.org/2023.findings-eacl.139

[7] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, et al. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proc. ACL.* 8440–8451.

[8] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge Transfer for Out-of-Knowledge-Base Entities: A Graph Neural Network Approach. In *Proc. IJCAI.* 1082–1088.

[9] Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L. McGuinness, and Mohammed J. Zaki. 2019. FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation. In *Proc. ISWC.* 146–162.

[10] Devlin Jacob, Chang Ming-Wei, Lee Kenton, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. NAACL.* 4171–4186.

[11] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *TACL* 8 (2020), 423–438.

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*. 1–15.

[13] Kazunori Komatani, Yuma Fujioka, Keisuke Nakashima, Katsuhiko Hayashi, and Mikio Nakano. 2021. Knowledge Graph Completion-based Question Selection for Acquiring Domain Knowledge through Dialogues. In *Proc. IUI*. 531–541.

[14] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proc. EMNLP*. 66–71. https://doi.org/10.18653/v1/D18-2012

[15] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proc. EMNLP*. 230–237.

[16] Diya Li and Mohammed J Zaki. 2022. Food Knowledge Representation Learning with Adversarial Substitution. In *Proc. AACL-IJCNLP*. 653–664. https://aclanthology.org/2022.aacl-main.50

[17] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *arXiv:2103.10385* (2021). arXiv:2103.10385

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692* (2019).

[19] Ilya Loshchilov and Frank Hutter. 2018. Decoupled Weight Decay Regularization. In *Proc. ICLR*.

[20] Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do Pre-trained Models Benefit Knowledge Graph Completion? A Reliable Evaluation and a Reasonable Approach. In *Proc. Findings of ACL*. 3570–3581. https://doi.org/10.18653/v1/2022.findings-acl.282

[21] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proc. ICML*. 809–816.

[22] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *Proc. EMNLP-IJCNLP*. 2463–2473.

[23] Wang Quan, Mao Zhendong, Wang Bin, and Guo Li. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, et al. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[25] Haseeb Shah, Johannes Villmow, Adrian Ulges, Ulrich Schwanecke, and Faisal Shafait. 2019. An open-world extension to knowledge graph completion models. In *Proc. AAAI*, Vol. 33. 3044–3051.

[26] Baoxu Shi and Tim Weninger. 2018. Open-World Knowledge Graph Completion. In *Proc. AAAI*. 1957–1964.

[27] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proc. EMNLP*. 4222–4235.

[28] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proc. NIPS*, Vol. 26. 926–934.

[29] Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proc. EMNLP*. 1784–1789.

[30] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. WWW*. 697–706.

[31] Brown Tom, Mann Benjamin, Ryder Nick, Subbiah Melanie, Jared D Kapla, Prafulla Dhariwal, et al. 2020. Language Models are Few-Shot Learners. In *Proc. NIPS*, Vol. 33. 1877–1901.

[32] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proc. ICLR*, Vol. 48. 2071–2080.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NIPS*, Vol. 30. 6000–6010.

[34] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledge Base. *Commun. ACM* 57, 10 (2014), 78–85.

[35] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[36] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proc. ICLR*. 1–12.

[37] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *arXiv:1909.03193* (2019). arXiv:1909.03193

[38] Xiaohan Zou. 2020. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series* 1487 (2020).