3-2024

Non-monotonic generation of knowledge paths for context understanding

Pei-chi LO
Singapore Management University, pclo.2017@phdcs.smu.edu.sg

Ee-peng LIM
Singapore Management University, eplim@smu.edu.sg

Non-monotonic Generation of Knowledge Paths for Context Understanding

PEI-CHI LO and EE-PENG LIM, Singapore Management University, Singapore

Knowledge graphs can be used to enhance text search and access by augmenting textual content with relevant background knowledge. While many large knowledge graphs are available, using them to make semantic connections between entities mentioned in the textual content remains to be a difficult task. In this work, we therefore introduce contextual path generation (CPG), which refers to the task of generating knowledge paths, *contextual path*, to explain the semantic connections between entities mentioned in textual documents with given knowledge graph. To perform the CPG task well, one has to address its three challenges, namely, path relevance, incomplete knowledge graph, and path well-formedness. This article designs a *two-stage framework* comprised of the following: (1) a *knowledge-enabled embedding matching* and *learning-to-rank with multi-head self-attention* context extractor to determine a set of context entities relevant to both the query entities and context document, and (2) a *non-monotonic path generation method with pretrained transformer* to generate high-quality contextual paths. Our experiment results on two real-world datasets show that our best performing CPG model successfully recovers 84.13% of ground truth contextual paths, outperforming the context window baselines. Finally, we demonstrate that the non-monotonic model generates more well-formed paths compared to the monotonic counterpart.

CCS Concepts: • **Information systems** → **Retrieval tasks and goals;**

Additional Key Words and Phrases: Information retrieval, knowledge graph, contextual path generation, generation model

ACM Reference format:
Pei-Chi Lo and Ee-Peng Lim. 2024. Non-monotonic Generation of Knowledge Paths for Context Understanding. *ACM Trans. Manag. Inform. Syst.* 15, 1, Article 1 (March 2024), 28 pages.
https://doi.org/10.1145/3627994

1 INTRODUCTION

1.1 Background

A knowledge graph represents human knowledge in the form of semantic entities and relations among these entities. In recent years, knowledge graphs have been applied to many information system applications, including **decision support systems (DSS)** [24, 36, 41, 67], **information**

This research is partially supported by the Lee Kong Chian Professorship and National Research Foundation, Singapore under its Strategic Capabilities Research Centres Funding Initiative.
Authors' address: P.-C. Lo and E.-P. Lim, Singapore Management University, 80 Stamford Rd., Singapore, 178902; e-mails: pclo.2017@phdcs.smu.edu.sg, eplim@smu.edu.sg.

2158-656X/2024/03-ART1
https://doi.org/10.1145/3627994

retrieval (IR) [4, 17, 34, 62], **question answering (QA)** [49, 61], and recommendation [8, 12, 57, 71] to help with deriving or explaining the results of these applications. Meanwhile, industries and user communities are also developing large knowledge graphs for commercial and public use. For example, Google's knowledge graph, which contains hundreds of millions of entities, is being used for improving the search of people, locations, and other interesting objects.[1] DBpedia is another knowledge graph constructed by the Wikipedia user community and it contains at least 6 million entities [50].

To illustrate how one leverages a knowledge graph for information retrieval, we consider the task of identifying fake news among a collection of news articles [19, 22]. In a recent MIS work on fake news identification [19], the trustworthiness of each third party digital supply chain used by online news websites has been determined to be a useful feature to predict if a given online news article is fake. Another feature that aids fake news prediction is the checking of facts mentioned in the news article. As a knowledge graph provides entities and relations possibly mentioned by a news article, verifying the facts in a news article against knowledge graph entities and relations is thus a good strategy to improve the fake news prediction accuracy. By performing the same verification for a sampled collection of news articles from the same digital supply chain against the knowledge graph, one may also calibrate the supply chain's trustworthiness more accurately. Once a fake news article is detected, one can provide warnings or interventions to help users make appropriate judgments about the article content [7].

Despite the several useful knowledge graph applications, many existing knowledge graphs are incomplete as they may not be up-to-date with the latest entities or relations [10, 65]. Hence, many research projects on knowledge graph completion, i.e., identifying missing entities, entity attributes, and relations, have been carried out lately [45, 52]. The earlier works in this domain focus on predicting missing relations from observed entities and relations [5, 33]. However, such methods fail to handle unseen entities, or paths of relations that semantically connect two entities of a knowledge graph. As a result, some research works turn to knowledge graph completion with input unstructured data such as text [44, 60]. As part of information systems research, there are several research efforts focusing on knowledge extraction and construction [11, 51, 63]. Specifically, Gil et al. designed software systems to extract knowledge from different software artifacts (e.g., purpose, features, information, deployment strategy) [20]. These methods learn to infer new relations from input textual information, and use them to augment the knowledge graph. Still, most of these works focus on extracting one-hop relations from text instead of paths made up of entities and relations.

Example 1. Consider the news example in Figure 1(a) published at Wikinews on October 14, 2005. There are mentions of several entities including actors, novelists, films, companies, drama, countries, and studios, which can be found in some knowledge graphs. For the two entities *Daniel Craig* and *Martin Campbell*, the semantic connection underlying the co-occurrence of their mentions is that *Daniel Craig* starred in the movie *Casino Royale* directed by *Martin Campbell*, which exists as a path in the knowledge graph:

$$e_{\text{Daniel Craig}} \xrightarrow{\textbf{starring}} e_{\text{Casino Royale}} \xrightarrow{\textbf{director}} e_{\text{Martin Campbell}},$$

where $e_{\text{Daniel Craig}}$, $e_{\text{Casino Royale}}$, and $e_{\text{Martin Campbell}}$ are the relevant entities in the knowledge graph, and $\xrightarrow{\textbf{starring}}$ and $\xrightarrow{\textbf{director}}$ are the relevant relations.

[1] https://blog.google/products/search/introducing-knowledge-graph-things-not/

Daniel Craig to be new James Bond

Friday, October 14, 2005

British actor **Daniel Craig** has been confirmed as the man to follow Pierce Brosnan as the sixth James Bond.

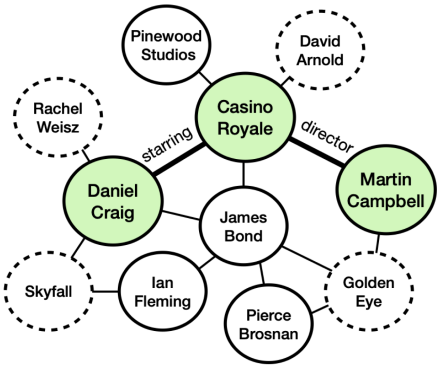
Producer Barbara Broccoli and director **Martin Campbell** called Craig 'a superb actor who has all the qualities needed to bring a contemporary edge to the role'.

"It is a huge iconic figure in movie history, and those things don't come along very often," Craig said. The role makes Craig the second Englishman to act as the British spy code-named 007, based on the famed literary character created by novelist Ian Fleming in 1953.

Craig's film credits include Road to Perdition, Enduring Love, Layer Cake and Sylvia. On British television, his major breakthrough was a starring role in the 1996 BBC drama Our Friends in the North.

The next Bond film Casino Royale is due to film in Italy, the Bahamas, the Czech Republic and Pinewood Studios. The film is due for release in 2006.

(a) Example Wikinews article. The query entities mentions are shown in **red**. Other entities mentions are shown in **blue**.



(b) Example knowledge graph associated with the example article. Entities in the contextual path are shown in **green** circles while other entities are shown in white circles. Entities mentioned in the example Wikinews article and those are not mentioned are shown in white circles with solid and dashed outlines, respectively.

Fig. 1. Contextual path from an example Wikinews article.

For the above example, the existing methods that infer one-hop relations would fail to recover the two relations $e_{\text{Daniel Craig}} \xrightarrow{\textbf{starring}} e_{\text{Casino Royale}}$ and $e_{\text{Casino Royale}} \xrightarrow{\textbf{director}} e_{\text{Martin Campbell}}$, as *Casino Royale* occurs very far from both *Daniel Craig* and *Martin Campbell* in the news text.

1.2 Problem Definition

In this article, we therefore aim to infer a path of relations from a piece of input text capturing the semantic connection between two given entities. This inference is performed based on an incomplete knowledge graph, and such an inferred path is called *contextual path*. For the example shown in Figure 1(a), the two given entities are *Daniel Craig* and *Martin Campbell* and the input text is the news article in Figure 1(a). The input knowledge graph is shown in Figure 1(a) and (b). The contextual path inferred is $e_{\text{Daniel Craig}} \xrightarrow{\textbf{starring}} e_{\text{Casino Royale}} \xrightarrow{\textbf{director}} e_{\text{Martin Campbell}}$. In this example, the two relations in the contextual path happen to exist in the knowledge graph. In general, we would also like to infer new relations for the knowledge graph when they are required to construct contextual paths.

Suppose the contextual paths connecting between many different pairs of entities mentioned in a collection of news articles are inferred or generated; the paths and their relations can support many new interesting ways to search and browse the articles. For instance, one can easily search news articles mentioning actors or actresses who star in Martin Campbell's movies.

In the above example, we assume that text data such as news articles have their entity mentions extracted and linked to the corresponding entities of the given knowledge graph. Among the methods that recognize such entity mentions, named entity recognition methods help to detect person names, organization names, place names, and other names in textual content followed by linking these entity mentions with entities of the knowledge graph (also known as entity linking) [42, 43]. Entity linking methods perform the matching of mentions in textual content with knowledge graph entities.

In this article, we therefore focus on addressing the following question: "how exactly do two entities connect to each other given a textual content as the context?" Context is crucial in this

question as the same entities can be semantically connected differently in different contexts. To answer the above question, we formulate the *contextual path generation problem*.

Definition 1 (Contextual Path Generation (CPG)). Given a textual document d and two entities e_h and e_t, which are mentioned in d, the CPG problem is to generate a path between e_h and e_t that provides the semantic connection between e_h and e_t relevant to d using the entities and relation edges from a given knowledge graph. We call the textual document and the resultant path the *context* and *contextual path*, respectively.

Example 2. In the earlier example, the Wikinews article serves as a context. The path $e_{\text{Daniel Craig}} \xrightarrow{\textbf{starring}} e_{\text{Casino Royale}} \xrightarrow{\textbf{director}} e_{\text{Martin Campbell}}$ depicts the contextual path as it accurately explains how Daniel Craig is connected with Martin Campbell in this Wikinews.

Finding the contextual path is challenging for a number of reasons. First, instead of returning all paths that connect e_h and e_t, CPG has to determine the path that is most relevant to the context d, including the involvement of entities not mentioned in the context. We call this the "path relevance" challenge. This challenge is largely caused by noisy and sparse data in the context, as well as the implicit patterns of constructing relevant contextual paths. The former is caused by other entities mentioned in the context but may not be relevant, and absence of entities not mentioned in the context but may be relevant, which could be due to very little content in the context. The latter refers to the expected pattern of contextual path for e_h and e_t of specific types. For example, a contextual path linking an actor entity with a director entity via a movie entity is certainly more plausible than via a concert entity. Second, the given knowledge graph may not cover all the required relation edges to form the contextual path. We call this the "incomplete knowledge graph" challenge. Given an incomplete knowledge graph, we have to generate contextual paths in CPG instead of constructing paths using observed relation edges. In the generation process, we face the third challenge of ensuring the resultant paths to be of good quality. This is also known as the "path well-formedness" challenge. Finally, as there are no prior works on CPG, we also need to address the "data challenge" of constructing datasets for training and evaluating any proposed CPG solution models.

1.3 Research Objectives and Contributions

This article therefore aims to solve the novel CPG problem while addressing all the above challenges. We propose a general *two-stage framework* for solving CPG. We then use that to develop solution models. In the following, we outline our research contributions:

- Our proposed two-stage framework is general for the purpose of creating different CPG solution models. The framework consists of context extraction and CPG stages, each of which can be implemented with different methods. In this article, we introduce two context extraction methods, one based on a classification approach and another based on a learning-to-rank approach. Both methods are designed to extract relevant entities from the context and knowledge graph so as to address the path relevance challenge.
- We also propose a non-monotonic CPG method for the CPG stage. This method is unique in its ability to generate well-formed paths and to create new but relevant relation edges to address the path well-formedness, path relevance, and incomplete knowledge graph challenges. Unlike the usual monotonic approach where a path is constructed from one end, say e_h, to another end, say e_t, in a left-to-right manner, our non-monotonic based method is trained to generate path elements in a flexible order that increases the accuracy of the generated path. For instance, the monotonic approach to generate the contextual path in Example 1

will generate $e_{\text{Daniel Craig}} \xrightarrow{\textbf{starring}} e_{\text{Casino Royale}}$ before $e_{\text{Casino Royale}} \xrightarrow{\textbf{director}} e_{\text{Martin Campbell}}$. The non-monotonic approach will be able to generate the two relation edges in any order.

— To address the data challenge, we construct two real-world datasets with ground truth contextual paths. We use these datasets to show that our best performing CPG solution model recovers nearly 85% of ground truth contextual paths, which is 14.8% higher than the baseline model. Our experiment also shows that the non-monotonic approach generates better-formed and more accurate contextual paths than the monotonic approach. We also design a synthetic dataset for evaluating the abilities of our best performing model in coping with large data and relation edge inference as covered in Appendix A.

We organize the rest of this article as follows. We present our literature review in Section 2. Section 3 covers our proposed Two-Stage framework and our proposed Non-Monotonic CPG model. We construct the two real-world datasets and introduce the evaluation metrics specially designed for CPG in Section 4. In Section 5, we cover the experiment setup and results/findings. Finally, Section 7 concludes the article and outlines the future research directions.

2 RELATED WORKS

In this section, we first survey the use of knowledge graphs in various types of information systems and how they may benefit from contextual paths generated for a given context. As the CPG problem is also closely related to knowledge reasoning and conditional sequential generation problems, we also examine works in these two areas.

2.1 Knowledge-enhanced Information Systems

Using knowledge graphs for enhancing information systems has been an active research area in information systems [9, 16, 69]. The common goal of these works is to capture and employ existing human knowledge to improve the functional features of information systems. For instance, Liu et al. derived inter-company connections from relations of an enterprise knowledge graph and utilized these connections to derive for a target company news sentiment features of connected companies together with news sentiment features of the target company in order to predict the target company's stock movement [37]. Lim et al. proposed to improve the accuracy of adverse drug reaction prediction by learning better vector representation of clinical concepts and relations in a knowledge graph using the confidence weights assigned to them based on co-occurrence and NLP patterns found in medical literature [30]. Xu et al. proposed to measure social proximity between two company entities using the similarity between the embeddings of the two entities in the knowledge graph [64]. CPG can be applied to the above company and medical domains to return contextual paths between entities (e.g., companies or medical concepts) mentioned in text collection (e.g., news articles and medical literature). These contextual paths can be used to establish new inter-entity connections or to derive confidence weights between entities so as to improve prediction accuracy or social proximity measurement.

2.2 Knowledge Reasoning

Finding a contextual path is somewhat similar to reasoning with knowledge graphs. The previous knowledge reasoning works adopt a wide range of reasoning techniques, including logical rules [2, 26], Bayesian models [59], distributed representations [47, 55], neural networks [14, 25, 29, 32], and reinforcement learning [13, 21, 28, 35].

Knowledge reasoning has been widely applied to knowledge graph–based **question answering (QA)**. In QA, Asai et. al proposed a retriever-reader framework on the HotpotQA dataset that reasons over Wikipedia using its graph structure [1]. While the HotpotQA dataset itself, however,

does not include a knowledge graph, it can be linked to the Wikipedia knowledge graph for explaining the questions' answers. MetaQA traverses through a knowledge graph in order to retrieve the answer to a given question [68]. Although the traversed path could explain the answer, the QA task itself is not designed to evaluate the traversed path against a human judged explanation path [31, 61]. By definition, traversed path can only be constructed with observed relations of the knowledge graph. MHQA-GRN addresses reading comprehension QA by constructing a directed acyclic graph from a knowledge graph for each passage of the query article and using its representation as features to predict the answer [54]. Nevertheless, MHQA-GRN does not return a path similar to contextual path. LEGO is a framework designed to find the answer entities for a given question by alternating between generation of query tree and reasoning with latent space [49]. Starting from the question entities as the root, LEGO expands the query tree with extracted relations from the question. In the representation space, the projection of the relations form several candidate answer spaces near the question entities. LEGO then finds the answer entities lying within the intersection of the candidate answer spaces. In [73], researchers combined a convolutional neural network for feature extraction with a recurrent neural network to predict the answer entity given a query entity and a knowledge graph that covers different paths from the query entity to candidate answer entities. CogKR conducts multi-hop reasoning over a knowledge graph by iterating between an expansion module and a reasoning module to predict the answer entity for a given query entity using a neighborhood knowledge subgraph [18].

In summary, we found that none of the above models considers context documents when selecting the knowledge paths leading to the answer entities. They also do not perform any evaluation on the knowledge paths derived as a side result. Hence, our CPG work can contribute to existing knowledge reasoning works by bridging these two research gaps.

2.3 Conditional Sequential Generation

Both the path relevance and the well-formedness challenges require us to generate well-formed contextual paths relevant to the input context document and query entity pair. We can thus treat CPG as a conditional sequence generation task with the context document and query entity pair as the input condition.

Our survey found that most conditional sequential generation works focus on generating text conditioned on a simple label such as tense, sentiment polarity, or formality of the text to be generated [15, 23, 53, 66]. They could not cope with the complex input condition of CPG that involves entity and text data. Among the few recent works that consider text as input condition, BERT-fused translation uses a pretrained encoder to first obtain the representation of text condition before performing text generation [72]. KG-BART takes a set of concepts from a commonsense knowledge graph as input and generates a piece of text covering these input concepts [39]. Other related works include generating sentences conditioned on a text document following a given syntax of an exemplar [27, 46]. So far, all these works focus on generating natural language sentences instead of knowledge paths. To our knowledge, there is so far no research focusing on generating knowledge paths with complex input conditions.

Unlike the above works, we propose to learn a pretrained model to generate knowledge paths before fine-tuning it to generate conditional knowledge paths. Our research also focuses on making the paths well-formed as well as semantically relevant to the query and context document.

3 PROPOSED FRAMEWORK AND CPG MODELS

In this section, we first describe our proposed two-stage framework. We then present the proposed methods for the context extractor and contextual path generator modules of the framework that form the different CPG solution models.

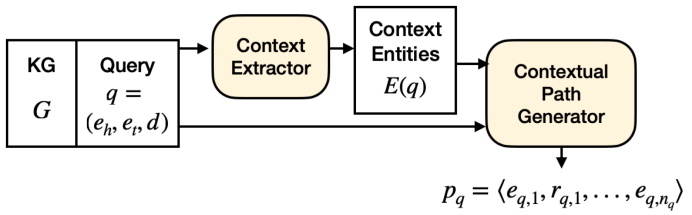


Fig. 2. The two-stage framework for contextual path generation.

3.1 Two-Stage Framework

Before we present the framework, we give the definition of knowledge graph. We define the *knowledge graph* G to be a tuple (E, L, R) where E denotes a set of entities, L denotes a set of relation edges, and R denotes a set of relation labels or simply relations. Each relation edge (e_i, r_k, e_j) of L directly links entity e_i to entity e_j via a relation edge with label r_k that can be found in R. We can also denote the same edge by $e_i \xrightarrow{r_k} e_j$. Notation wise, we use italic for entities (e.g., *Biden*) and boldface (e.g., **presidentof**) for relations. For the purpose of establishing connections between entities, we assume that each relation and its reverse can always be found in R. For example, for the knowledge graph to capture *Biden* is the president of *USA*, R should contain both $e_{\text{Biden}} \xrightarrow{\textbf{presidentof}} e_{\text{USA}}$ and $e_{\text{USA}} \xrightarrow{\textbf{presidentof}^{-1}} e_{\text{Biden}}$.

The input of CPG can be captured as a query q represented by (e_h, e_t, d) where e_h and e_t are the input entities and d is the context document. The output of CPG is a path p represented as a sequence of path elements denoted by $\langle e_{q,1}, r_{q,1}, e_{q,2}, r_{q,2}, \ldots, r_{q,(n_q-1)}, e_{q,n_q}\rangle$ where $e_{q,1} = e_h$ and $e_{q,n_q} = e_t$. The path elements include the entities and relation labels involved in the path. We assume that all entities and relation labels that are used in any path should be found in E and R, respectively, i.e., $\forall i, e_{q,i} \in E$ and $\forall k, r_{q,k} \in L$. Nevertheless, it is possible that some relation edges of a path may not exist in the knowledge graph G as the latter is incomplete.

As shown in Figure 2, our proposed framework consists of a *context extractor* and a *contextual path generator*. The former derives *context entities* $E(q) = \{e_1, e_2, \ldots, e_{|E(q)|}\}$ from the query q and input knowledge graph G. Context entities are entities relevant to the query and they may be mentioned in d. To enrich the sparse content of the context document, one should also select relevant entities from the knowledge graph to be context entities thereby addressing part of the path relevance challenge (as the remaining part is to be addressed by the contextual path generator).

The context entities together with the query are then provided to the contextual path generator to construct the resultant contextual path(s) leveraging the interaction between context entities to finally determine the relevant entities and relation edges and sequence them to form the resultant path. In this process, new relation edges may be inferred to address the incomplete knowledge graph challenge. We further propose a non-monotonic path generation process to ensure that the resultant path is both semantically relevant and well-formed addressing the path relevance and path well-formedness challenges, respectively.

The above non-monotonic path generation approach is novel and is different from the existing *single object-type monotonic* sequence generation approach, considering that a knowledge path is a type of sequence. Single object-type sequence generation is one that generates elements of the same type, instead of entities and relation edges as alternating path elements. The monotonic sequence generation is one that always generates elements from left to right only. Such generation may fail to reach the tail entity e_t, and repeats the same path element(s). In this case, the generated path is not well-formed. As non-well-formed paths are considered incorrect, our goal is thus to design a generation model that always generates well-formed contextual paths.

3.2 Context Extractor

The context extractor takes a query $q = (e_h, e_t, d)$ and a knowledge graph G as input, and decides the set of context entities $E(q)$. Ideally, $E(q)$ contains the set of entities for constructing the correct contextual path. In this section, we propose two context extraction methods: **Knowledge-enabled Embedding Matching Model (KEMatch)** and **Learning-To-Rank with Multi-Head Self-Attention Model (LTRMHSA)**. The former is based on binary classification, and the latter is based on learning-to-rank on multi-head self-attention.

The two methods share a common **Query-induced Entity Selector** that derives a subset of knowledge graph's entities $E^r(q) = \{e_1, e_2, \ldots, e_{m_q}\}$ as candidate entities using the query entities e_h and e_t where $m_q = |E^r(q)|$.

Let $E^r(h)$ and $E^r(t)$ be the sets of entities in the knowledge graph G that are reachable from e_h and e_t, respectively, in k-hops. The set of candidate entities is then obtained by $E^r(q) = E^r(h) \cup E^r(t)$. The remaining modules of KEMatch and LTRMHSA then select a subset of entities from $E^r(q)$ to form the context entity set $E(q)$.

3.2.1 KEMatch: Knowledge-enabled Embedding Matching. In KEMatch, we train a classifier to decide whether a candidate entity is related to the context or not. Thus, we introduce a simple binary classifier that takes an entity and the context as input, and outputs a probability between 0 and 1 that suggests the relatedness between the two inputs. Given $E^r(q)$ already derived by the query-induced entity extractor, the classifier performs matching of every candidate entity e from $E^r(q)$ with the vector representation of the query z_q. We define this matching model by $f_{kem}(z_e, z_q) \rightarrow \{0, 1\}$ where z_e and z_q denote the candidate entity representation and query representation, respectively. The candidate entities are predicted with label 1 (or matching) if they are deemed as semantically relevant, and label 0 (or non-matching) otherwise. Only the predicted matching entities will be returned as the set of context entities $E(q)$. We denote the size of $E(q)$ by m.

KEMatch uses k-bert [38], a knowledge-enabled contextualized word embeddings, to derive z_q and z_e. We introduce four different query representation schemes:

- **Average Entity Representation:** $z_q = \frac{\sum_{e' \in E(d)} z_{e'}}{|E(d)|}$, where $E(d)$ is the set of entities mentioned in context document d (which includes e_h and e_t), and the representation of an entity e is defined by $z_e = \frac{1}{|W(e)|} \sum_{w \in W(e)} z_w$ where $W(e)$ denotes the words in the entity name of e, and z_w denotes the k-bert embedding of word w. Note that $E(d)$ is not necessarily identical to $E(q)$ as not all entities in $E(d)$ are from $G(q)$ and the latter may contain entities not in d.
- **Mention Paragraph Representation:** $z_q = ENC(\mathrm{p}_q)$, where $\mathrm{p}_q = \mathrm{p}_h + \mathrm{p}_t$ where p_h and p_t are the paragraphs in d mentioning e_h and e_t, respectively. ENC is an encoder that derives a paragraph representation by averaging the k-bert embeddings of words in the paragraph.
- **Title and Mention Paragraph Representations:** This representation concatenates the representations of the title and mention paragraphs denoted by p_1 and p_q, respectively. Title paragraph p_1 refers to the first paragraph in d. Hence, $z_q = [ENC(\mathrm{p}_1), ENC(\mathrm{p}_q)]$.
- **Contextual Query Entity Representation.** This scheme combines e_h, e_t, and the context document d as a sequence of word tokens for input to ENC. z_q is then defined as the average k-bert embedding of word tokens in e_h, e_t, and d.

The matching model of KEMatch f_{kem} is trained with a set of queries with their corresponding positive and negative entity sets denoted by $E(q)^+$ and $E(q)^-$, respectively. For a query q, we use p_q^* to denote the ground truth contextual path. $E(q)^+$ is then assigned the (positive) entities in p_q^*. To avoid false-negative entities, $E(q)^-$ is assigned with a set of hard negative entities sampled

from entities with few or no common neighbor with entities of $E(q)^+$ in the knowledge graph G. That is, $E(q)^- = \{e | e \in E(q) - E(q)^+$ and $\sum_{e' \in E(q)^+} \text{co-occur}(e, e') \leq \delta\}$, where co-occur$(\cdot)$ is a function that returns the number of common neighbors of the two input entities and δ is a threshold parameter. In this article, we use the averaged co-occur$(\cdot)$ between positive pairs in the training set as δ, and choose $E(q)^-$ to be 10 times the size of $E(q)^+$.

We learn the matching function f_{kem} as a logistic regression classifier that takes z_q and z_e as input, and outputs the prediction probability $\hat{y}$. In this article, we consider three input feature combinations, namely, (i) *concatenation* $[z_q, z_e]$, (ii) *Hadamard product* $z_q \odot z_e$, and (iii) *subtraction* $z_q - z_e$. Note that when $z_q = [ENC(\mathrm{p}_1), ENC(\mathrm{p}_q)]$, the Hadamard product and subtraction methods are not applicable due to dimension mismatch between z_q and z_e. f_{kem} is learned with cross entropy loss. Once f_{kem} is learned, given a query q, we select top n^{CXT} entities in $G(q)$ with highest prediction probability as context entities $\hat{E}(q)$. In our experiment, we empirically set $n^{CXT} = 10$.

3.2.2 LTRMHSA. While KEMatch is easy to train with low computation overheads, it suffers from a major shortcoming of not considering how an entity determines the relevance of other entities when it co-occurs with different entities. For instance, when the entity *Ben Affleck* (an actor) is mentioned together with *Rosamund Piker* (an actress) in the context, the candidate movie entity *Gone Girl* will become relevant and thus should be included in $E(q)$, as they both starred in this movie. On the other hand, when the entity *Ben Affleck* appears with *Matt Damon* (another actor) in the context, another movie entity *Good Will Hunting* should be included in $E(q)$ instead. As a result, we propose LTRMHSA, an attention-based learning-to-rank model that considers the interaction among candidate entities when determining their relevance. We adapt the MHSA from Tu et al. [56], to model interaction among entities of $E^r(q)$, and propose a MHSA layer and Pairwise Bi-Linear layer to select context entities.

As depicted in Figure 3, LTRMHSA also uses the query-induced entity selector to obtain $E^r(q)$. It then concatenates all its entities with the context document d and encodes these pairs using k-bert [38]. Each document-candidate entity pair (d, e_i) is first represented as a sequence of word tokens started with a "[cls]" tag token followed by d's word tokens, "[sep]" tag token, the word tokens of e_i's entity name, and "[sep]" tag token. The output representation of "[cls]" from k-bert for each (d, e_i) pair, denoted by $z_{cls,i}$, then summarizes the semantics of the context document d and the corresponding entity e_i. The MHSA layer takes the sequence of m_q output representations z_1^{cls} to $z_{m_q}^{cls}$ and allows them to interact using multi-head self-attention:

$$\begin{aligned} Multihead(Q, K, V) &= Concat(head_1, \ldots, head_n)W^0, \\ head_h &= Attention(QW_h^Q, KW_h^k, VW_h^v), \quad h \in \{1, \ldots, n_h\}, \\ Attention(Q_h, K_h, V_h) &= softmax\left(\frac{Q_h K_h^T}{\sqrt{\mathrm{d}_k}}\right)V_h, \end{aligned} \tag{1}$$

where Q, K, and V are $m_q \times \mathrm{d}_z$ matrices where d_z is the dimensionality of $z_{cls,i}$. Q_h, K_h, and V_h are $m_q \times \mathrm{d}_k$ matrices where d_k is the inner dimension of self-attention. W_h^Q, W_h^K, and W_h^V are learnable $\mathrm{d}_z \times \mathrm{d}_k$ projection matrices of the sequence of $z_{cls,i}$'s for different heads. For n_h heads, d_k is determined by d_z divided by n_h. In this work, we set $n_h = 8$,

A pairwise bi-linear learning-to-rank classifier f_{ltr} then ranks all candidate entities by their relevance to the query. We assume that a context entity should have more inter-entity interaction with entities in $E(q)^+$ (or entities in the ground truth contextual path p_q^*) than other entities. The classifier $f_{ltr}(e_i, e_j) \rightarrow \{0, 1\}$ is therefore expected to return 1 if the entity e_i is more relevant to the context document d than entity e_j, 0 otherwise.

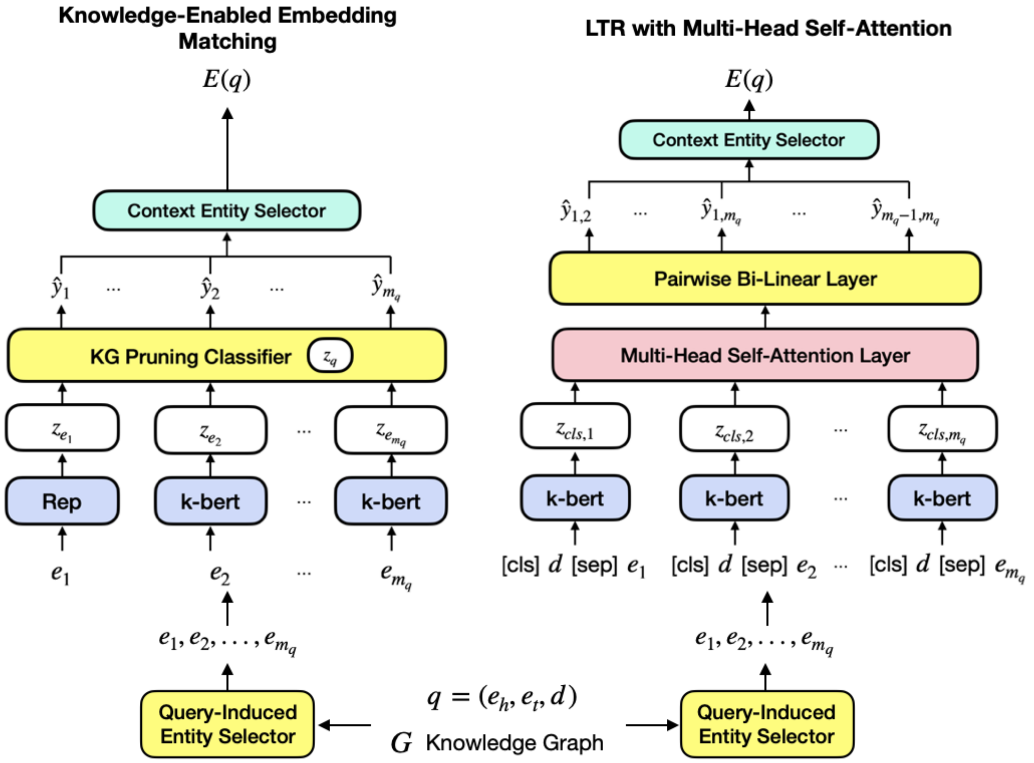


Fig. 3. Context extractors.

We also introduce a score function $\mathbb{S}(\cdot)$ to measure the relevance of the entity.

$$\mathbb{S}(e_i) = \begin{cases} 2 & if\ e_i \text{ can be found in } p_q^* \\ 1 & if\ e_i \in E(d) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For a given query q, the ground truth label $y_{i,j}$ assigned to $f_{ltr}(e_i, e_j)$ is

$$y_{i,j} = \begin{cases} 1 & if\ \mathbb{S}(e_i) > \mathbb{S}(e_j) \\ 0 & if\ \mathbb{S}(e_i) \leq \mathbb{S}(e_j). \end{cases} \tag{3}$$

The Pairwise Bi-Linear layer of LTRMHSA f_{ltr} is then optimized with binary cross entropy on the ground truth $y_{i,j}$ and predicted label $\hat{y}_{i,j}$ of (e_i, e_j).

During the inference phase, we derive a relevance score $\hat{\mathbb{S}}(e_i)$ to determine the relevance of e_i to q by gathering the pairwise prediction results of $f_{ltr}(e_i, e_j)$ for $e_i, e_j \in E^r(q)$. $\hat{\mathbb{S}}(e_i) = \sum_{j=1}^{m_q} \mathbb{1}(\hat{y}_{i,j} > 0.5)$ where $\mathbb{1}(\cdot)$ is an indicator function. Finally, we use a context entity selector to select the n^{CXT} entities with the highest $\hat{\mathbb{S}}$ scores to construct $E(q)$, similar to that in KEMatch. In this article, we empirically use $n^{CXT} = 10$.

3.3 CPG

CPG is responsible for generating a path p_q given $E(q)$ and for inferring missing relations when needed. We propose a **Non-Monotonic Contextual Path Generation with Pretrained Transformer (NMCPGT)** method as shown in Figure 4. NMCPGT addresses the two limitations of

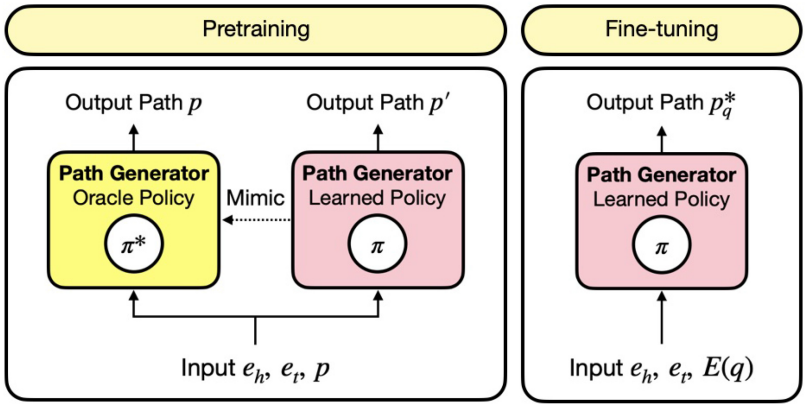


Fig. 4. Pretraining and fine-tuning steps of non-monotonic contextual path generation with pretrained transformer (NMCPGT).

traditional monotonic generation models such as n-gram [3] or neural language models [48] where the order of element generation is always from left to right. The first limitation is the generation of *unfinished paths* when the model generates a path that starts with e_h but could not end with e_t. The second limitation is the difficulty to leverage on both e_h and e_t to determine the next element to generate in the monotonic manner. Thus, in this article, we propose the non-monotonic path generation model that substantially increases the odds of finished paths and is trained to generate the more likely path elements using e_h, e_t, and the other already generated path elements in each generation step.

NMCPGT is obtained in two steps, namely, the pretraining and fine-tuning steps. In pretraining, a pretrained path generation model is trained to be familiar with the knowledge graph's entities and relations and the ways entities of different types are connected with one another via relation edges, and it is able to generate knowledge paths given an (e_h, e_t) pair. In fine-tuning, we further train the model to include candidate entities $E(q)$ as additional input and to return the contextual path.

3.3.1 Pretraining of Non-monotonic Path Generation Model. Our non-monotonic path generation model learns to generate path elements in any order. Borrowing the idea from an earlier work [6], we represent the generation order of all the path elements (i.e., entities and relations of the contextual path) using a binary tree in which each tree node is either an entity/relation or an "end" (or "E") item. The model generates these elements in a top-down and left-to-right manner, until the binary tree has "E" items generated for all the leaf nodes. Once the generation process is completed, the generated contextual path is the sequence of path elements (excluding the "E" items) determined by a **Deep-first Search (DFS)** traversal of generated path elements in the binary tree.

Consider the binary tree output example of our non-monotonic path generation module in Figure 5(a). The generation order starts with the root node, followed by its left child node, and then its right child node. The generation order numbers of the three nodes are thus assigned 1, 2, and 3 as shown in **blue** in the figure. Following that, the nodes to be generated next are those at the third and fourth layers with generation order numbers assigned from 4 to 11. For each node, we generate a path element or "E" item. As shown in Figure 5(a), we finally obtain a contextual path from the binary tree by constructing a sequence of the path elements (excluding the "E" items) following the DFS order numbers shown in **green**. The generated path is

$$e_{\text{MattDamon}} \xrightarrow{\textbf{almaMater}} e_{\text{HavardUniversity}} \xrightarrow{\textbf{state}} e_{\text{Massachusetts}}.$$

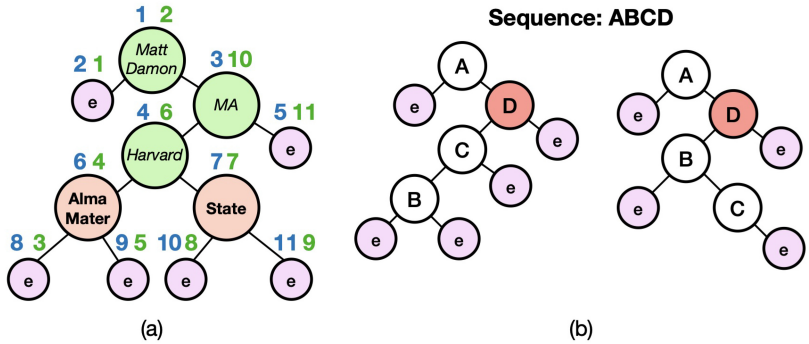


Fig. 5. Binary trees of non-monotonic generation: (a) A terminal binary tree that shows the generation steps for a path (with generation order numbers and DFS order numbers shown in **blue** and **green**, respectively). (b) Two binary trees with the same generated sequence.

Our non-monotonic path generation model is based on reinforcement learning. We let V be the set of all entities and relations, i.e., $V = E \cup R$. Let $Y = (w_1, \ldots, w_N)$ denote a sequence of path elements where $w_i \in V$ and $\tilde{V}$ be $\{V \cup \langle \text{e} \rangle\}$ where $\langle \text{e} \rangle$ denotes the terminal or "E" item. Let D denote the collection of Y's. The generation process is regarded as deterministically navigating a state space $\mathcal{S}$. A state $s \in \mathcal{S}$ corresponds to a binary tree of nodes from $\tilde{V}$. For instance, the example we show in Figure 5(a) has an initial state s_1=(*Matt Damon*), and a final state s_{11}=(*Matt Damon*,$\langle$e$\rangle$,*MA*, . . ., **State**, . . ., $\langle$e$\rangle$). The subscript of state s_i is the generation order number. An action a is an element of $\tilde{V}$ that is chosen to be added to the tree for the next available generation order index. As mentioned previously, when every leaf node in the binary tree is the terminal node $\langle$e$\rangle$, the generation reaches the terminal state s_T. $T = 2N + 1$ denotes the number of nodes (or states) that exist in the terminal binary tree. We use $\tau(i)$ to represent the generation order index of the generated element with DFS order number = i, e.g., $\tau(10) = 3$ in Figure 5(a).

The goal of the non-monotonic path generation model is to learn a policy π that imitates an oracle policy π^* that always generates a knowledge graph path. A policy is a stochastic mapping from states to actions. It decides for each generation order index which element or terminal symbol to generate. When the binary tree is terminal, the entire path is determined by the sequence of elements following the DFS order numbers. The probability of an action $a \in \tilde{V}$ given a state s under policy π is denoted as $\pi(a|s)$.

To ensure that we generate a path from e_h to e_t, the policy is trained to generate e_h, $\langle$e$\rangle$, e_t, and $\langle$e$\rangle$ for the states s_1, s_2, s_3, and s_5, respectively. That is, $\pi(e_h|s_1) = \pi(\langle \text{e} \rangle|s_2) = \pi(e_t|s_3) = \pi(\langle \text{e} \rangle|s_5) = 1.0$.

Let $U[T]$ be the uniform distribution over all the states of a binary tree $\{1, \ldots, T\}$ and d_π^t be the state distribution obtained from running π for t-many steps. When generating an element or terminal symbol, the model updates the current learned policy π by comparing its predicted cost to an observed cost-to-go estimated with states drawn from π^{in} and actions from π^{out}. In other words, we train π to pick actions that minimize $C(\pi; \pi^{\text{out}}, s_t)$. $C(\pi; \pi^{\text{out}}, s_t)$ measures the loss incurred by π against the cost-to-go estimates under π^{out} for a given state s_t. The model then learns π^{out} that minimizes the following cost function:

$$\mathbb{E}_{Y \sim D} \mathbb{E}_{t \sim U[T]} \mathbb{E}_{s_t \sim d_{\pi^{\text{in}}}^t} [C(\pi; \pi^{\text{out}}, s_t)], \tag{4}$$

where s_t is the state corresponding to the top-down traversal of the generated binary tree at step t. This process finds a policy that performs on-par or better than the oracle policy π^* with access to only states s_t.

The non-monotonic generation model can be implemented with a LSTM units model, or with transformer structure [58]. In this article, we build the model with the latter.

Oracle Policies. Given a partially generated target path $p = \langle w_1, \ldots, w_n \rangle$ at state s_t corresponding to the generation step t, we define Y_t as the set of elements in p that can be generated for state s_t for the non-monotonic generation model to be able to generate p eventually. An oracle policy is defined as

$$\pi^*(a|s_t) = \begin{cases} 1 & if\ a = \langle \text{e} \rangle \text{ and } Y_t = \langle\rangle \\ P(a) & if\ a \in Y_t \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $P(a)$ is defined such that $\sum_{a \in Y_t} P(a) = 1$. For every generation step t, the oracle policy always generates valid action a (i.e., if $a \in Y_t$) with positive probabilities, invalid actions with zero probabilities, and $\langle \text{e} \rangle$ when no other elements are required to be generated. We can design different oracle policies by defining $P(a)$ differently. Let g and h be the nearest left parent and nearest right parent, respectively, of the tree node corresponding to step t based on the DFS order. Let the actions generated at steps g and h be $a_g = w_g$ and $a_h = w_h$, respectively. We now derive Y_t as follows:

$$Y_t = \begin{cases} \{w_{g+1}, \ldots, w_{h-1}\} \cup \{\langle \text{e} \rangle\} & \text{if } a_g \text{ and } a_h \text{ are found} \\ \{w_1, \ldots, w_{h-1}\} \cup \{\langle \text{e} \rangle\} & \text{if only } a_h \text{ is found} \\ \{w_{g+1}, \ldots, w_n\} \cup \{\langle \text{e} \rangle\} & \text{if only } a_g \text{ is found.} \end{cases} \tag{6}$$

In this article, we use an **annealed coaching oracle** that combines a uniform oracle and a coaching oracle to address the problem of not exploring diverse sets of generation orders. The **uniform oracle** treats all possible generation orders that lead to the target sequence as equally likely, without preferring any specific set of orders. It gives uniform probabilities $P(a) = 1/|V|$ for all elements in the sequence. On the other hand, **coaching oracle** ensures no invalid action is assigned by any probability. It prefers actions that are preferred by the current parametrized policy and reinforces the selection by the current policy π if it is valid. In other words, $\pi^*_{\text{coaching}}(a|s) \propto \pi^*_{\text{uniform}}(a|s)\pi(a|s)$. The annealed coaching oracle can therefore be represented as $\pi^*_{\text{annealed}}(a|s) = \beta\pi^*_{\text{uniform}}(a|s) + (1-\beta)\pi^*_{\text{coaching}}(a|s)$.

Cost Functions. Since the generation is non-monotonic, the generation order does not necessarily match the DFS order, and hence there can be multiple terminal binary trees that share the same generated sequence of path elements. For a desired sequence to be finally generated based on a partially generated binary tree, we need to determine if the element to be generated could violate the desired sequence. Therefore, we consider all entities and relations that can be generated as correct generation. We show two valid binary trees of a sequence *ABCD* generated by the non-monotonic generation model in Figure 5(b). When deciding the left child of *D*, any token that locates before it in the sequence can lead to generation of a valid binary tree. For instance, *C* is chosen in the left tree and the right tree chooses *B*. As a result, we consider *B* or *C* as correct generation when computing the cross entropy. Any other tokens are considered incorrect, including the termination node $\langle \text{e} \rangle$. We define the *Cross Entropy Loss* as follows:

$$\begin{aligned} &C(\pi; \pi^{\text{out}}, s_t) = \\ &-\left(\sum_{w \in V^+} y_{\pi, s_t} \log(p_{\pi, s_t}) + (1 - y_{\pi, s_t}) \log(1 - p_{\pi, s_t}) \right), \end{aligned} \tag{7}$$

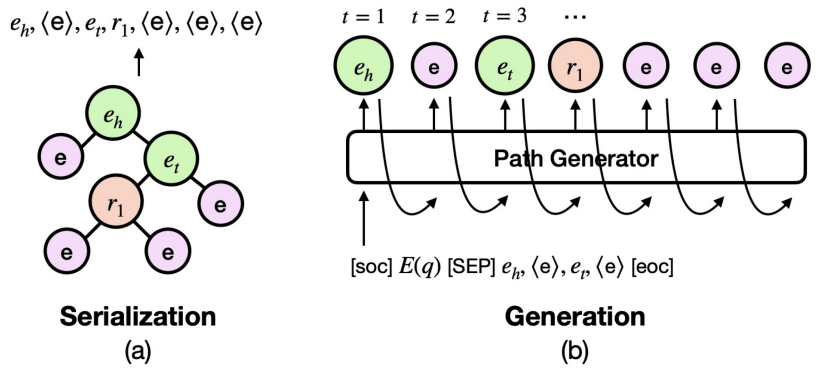


Fig. 6. Illustration of non-monotonic path generation: (a) Serialize an input tree with DFS traversal; (b) Generation of a Contextual Path.

where V^+ denotes the set of entities/relations that are deemed correct at this timestep t. $y_{\pi,s_t} \in \{0,1\}$ such that $y_{\pi,s_t} = 1$ if the generation predicted by π in state s_t leads to a valid DFS tree, and p_{π,s_t} is the generation probability.

In our experiment, we generated 100,000 paths from the knowledge graph for the pretraining. The paths are generated by first sampling e_1 from the graph as the head entity of the path, followed by sampling the next entity e_2 connected to e_1 with an edge $e_1 \xrightarrow{r1} e_2$ and adding that to the path. This process is repeated until we have sampled L edges, or stops with a chance of 20%. We will use the first and last entities of the path as e_h and e_t, respectively.

3.3.2 Fine-tuning Non-monotonic CPG. To direct the NMCPGT model to generate a path according to a context, we fine-tune our pretrained path generation model with additional context entities $E(q)$.

The context entities are arbitrarily ordered and separated by spaces. They are then concatenated with the query entities before being given to the pretrained transformer. We then fine-tune the transformer to return the correct contextual path. The fine-tuning step is very similar to that of pretraining except that the pretraining step involves sampled paths instead of ground truth contextual paths. The fine-tuning step is also different due to additional context entities. Its input sequence is in the format of "[soc] e_1 e_2 ... $e_{|E(q)|}$ [sep] e_h [e] e_t [e] [eoc]" where [sep] is a special token separating the context entities and the initial tree. We illustrate the contextual path generation process in Figure 6(b).

In the prediction phase, we construct an initial tree with e_h and e_t as the first and last nodes in the generation order leaving the left child of e_t missing. The serialized sequence of the initial tree is then fed to the decoder as prefix in the format of "[soc] e_h [e] e_t [e] [eoc]" where [soc] and [eoc] are special tokens signaling the start and the end of the prefix, and [e] is the token for the termination signal $\langle e \rangle$. The model then completes the binary tree and recovers the predicted contextual path by traversing the tree in DFS order.

3.4 Tree Serialization

During both the pretraining and fine-tuning phases, we need to feed the initial generation state represented as a binary tree to the transformer for training. Since the transformer can only generate a sequence of objects, it is trained to decode or generate the sequence of actions that represents the serialized binary tree of the training path. We serialize a path by traversing it in a top-down, left-to-right manner. The serialization process is shown in Figure 6(a). For each timestep t, the

Table 1. Dataset Statistics

	Wikinews Dataset	
	Wiki-film	**Wiki-music**
# context documents	40	40
# entities mentioned	563	471
# (e_h, e_t) entity pairs	1,396	1,237
Knowledge Graph		
# entities	59,173	44,886
# relations	651	513
Ground Truth Contextual Paths		
Avg./Max. path length	3.87/6	3.62/6
# unique path entities	563	471
# unique path relations	139	108

transformer decoder generates (or predicts) the next action of binary tree, which is in turn used as input to the transformer decoder for generating the next action. This repetitive process ends when the binary tree is terminal.

4 DATASET AND EVALUATION METRICS

4.1 Collection of Wikinews Datasets

In the absence of suitable datasets, we constructed two datasets, Wiki-film and Wiki-music, each consisting of 40 Wikinews articles as context documents. We then identify 563 and 471 entities mentioned in these two sets of context documents, respectively, which can be found in a knowledge graph extracted from DBpedia.[2] In the DBpedia knowledge graph, each entity corresponds to an article in Wikipedia. Every attribute within the infobox of the article that refers to another article is extracted as a relation edge linking the entity of the first article to the entity corresponding to the second article. The attribute label in the infobox is used as the relation label. To derive the set of (e_h, e_t) entity pairs, we identify 1,396 and 1,237 entity pairs from the context documents of the Wiki-film and Wiki-music datasets such that both entities of each pair can be found in a context document with paths connecting them in the knowledge graph. The detailed statistics of the datasets are shown in the second and third columns of Table 1.

We crowdsourced the ground truth contextual path for each (e_h, e_t) entity pair using AMT workers annotating the path using a custom-developed web-based annotation interface. To make the annotation reasonable to most annotators, we limit the contextual paths to have a maximum length of six. At the end of crowdsourcing, we derive one ground truth contextual path for each (e_h, e_t) pair with majority agreement.[3]

4.2 Evaluation Metrics

To measure the CPG performance, we divide the (e_h, e_t) entity pairs of each dataset into five folds. Each fold takes a turn to be used for testing while the remaining four folds are used for model training. We repeat this process five times and report the averaged performance across five folds in the experiment. We measure the model performance using three types of metrics, namely, (a) percentage of recovered ground truth paths; (b) averaged pairwise similarity; and (c) normalized graph edit distance.

[2] https://wiki.dbpedia.org

[3] Due to space constraints, we will elaborate the annotation process in an addendum upon publication.

Percentage of recovered ground truth paths (%Path Recovered). This metric measures the proportion of generated paths that are identical to the ground truth path. In this metric, the similarity between the generated and ground truth paths is not considered.

Average pairwise similarity (AVG PW Sim). This metric shows how much a generated path overlaps with the ground truth contextual path. Given a path p generated by the model, we compute its pairwise similarity with the ground truth path p^* by

$$\mathbf{s}(p, p^*) = \frac{\{E(p) \cup L(p)\} \cap \{E(p^*) \cup L(p^*)\}}{\{E(p) \cup L(p)\} \cup \{E(p^*) \cup L(p^*)\}}, \tag{8}$$

where $E(p)$ and $L(p)$ represent the set of entities and relation labels existing in p, respectively. We then compute the average pairwise similarity for all the generated paths. While this metric can effectively determine how close the generated path is to ground truth, it does not consider the ordering of the entities and relations. Moreover, it does not take into consideration the semantic relatedness between the two paths. The next normalized graph edit distance metric thus addresses these limitations.

Normalized Graph Edit Distance (NGEO). The **Graph Edit Distance (GEO)** is a metric originally designed to measure the similarity of two graphs by counting the number of operations needed to transform one graph to another [70]. We adapt it to measure the similarity between a generated element sequence q and the ground truth element sequence q^*. In this work, we report the normalized GEO:

$$NGEO(q, q^*) = min\left(\frac{GEO(q, q^*)}{|q^*|}, 1\right), \tag{9}$$

where $|q^*|$ is the length of the ground truth sequence. From the full path $(e_1, r_1, \ldots, r_{L-1}, e_L)$, we derive an *entity sequence* where only entities are included in a path $(e_1, .., e_L)$, and a *relation sequence* $(r_1, \ldots, r_{L-1})$ defined in a similar manner. We denote the NGEO for relation and entity sequences as **NGEO(E)** and **NGEO(R)**, respectively.

Let $OP(q, q^*)$ be the sequence of edit operations for converting the former to the latter. The GEO metric is defined by

$$GEO(q, q^*) = \sum_{(op_j, t_j, t'_j) \in OP(q, q^*)} c_{op_j}(t_j, t'_j). \tag{10}$$

There are six operations defined: entity insertion, entity deletion, entity substitution, relation insertion, relation deletion, and relation substitution. The semantic cost of an operation is defined by entity type or relation label distance determined by the ontology structure underlying the entities and relation labels. GEO makes use of the ontology structure of the knowledge graph to determine the semantic similarity between entities and relations. Let t (or r) and t' (or r') be the inserted entity's type (or inserted relation label) and the deleted entity's type (or deleted relation label), respectively. t_0 and r_0 are the root entity and relation, respectively, in the knowledge ontology. For each operation *op* performed on a path, the semantic cost $c_{op}(\cdot)$ incurred is defined below:

$$\begin{aligned}
\text{Entity Insertion: } c_{vi}(t) &= dist(t, t_0),\\
\text{Entity Deletion: } c_{vd}(t') &= dist(t', t_0),\\
\text{Entity Substitution: } c_{vs}(t, t') &= dist(t, t'),\\
\text{Relation Insertion: } c_{ri}(r) &= dist(r, r_0),\\
\text{Relation Deletion: } c_{rd}(r') &= dist(r', r_0),\\
\text{Relation Substitution: } c_{rs}(r, r') &= dist(r, r').
\end{aligned} \tag{11}$$

When a semantic operation is performed, the semantic distance is defined as the co-topic distance $dist(t_1.t_2)$ (or $dist(r_1, r_2)$) of the two ontological types t_1 and t_2 (or relation labels r_1 and r_2) in a knowledge graph ontology:

$$dist(t_1, t_2) = 1 - \frac{|S(t_1) \cap S(t_2)|}{|S(t_1) \cup S(t_2)|,} \tag{12}$$

where $S(t_i)$ is the set of supertypes of t_i in the ontology. For example, given an ontology consisting of supertype-subtype relations, {Agent → Person, Person → Artist, Artist → Actor, and Person → MovieDirector}, the distance between Actor and MovieDirector is $1 - \frac{2}{5} = \frac{3}{5}$.

5 EXPERIMENT RESULTS

To compare the effectiveness of our proposed models and baselines, we design a series of experiments to measure the correctness of their generated paths. As our framework consists of context extractor and non-monotonic contextual path generator, we first evaluate the former before analyzing the overall contextual path generation performance on our two Wikinews datasets. Additionally, we complete a series of experiments on a synthetic dataset to show the scalability of our dataset, and the ability to infer missing relations of the knowledge graph. We cover the detail of how we generate the synthetic dataset and the experiment result in Appendix A.

5.1 Effectiveness of Context Extractor Methods

Our first set of experiments evaluates the effectiveness of context extraction models returning the ground truth contextual path entities as context entities for a set of queries. Other than our proposed context extractor methods KEMatch and LTRMHSA, we also include a context window baseline method described below.

- **Context window baseline.** We introduce a context extractor baseline method to compare against our proposed KEMatch and LTRMHSA methods. In this baseline method, we extract the entities mentioned in the text surrounding e_h and that in the text surrounding e_t within a distance of cw word tokens in context document d. In this article, we empirically use $cw = 20$.
- **KEMatch.** KEMatch prunes the query-induced entities graph with a classifier, and keeps only the top n^{CXT} entities with highest prediction probability as context entities. As described in Section 3, we experiment with three different methods of query representation z_q, namely, *(1) average entity representation*, *(2) mention paragraph representation*, *(3) title and mention paragraph representation, and (4) contextual query entity representation*. We also propose three feature combination schemes to combine z_q and z_e, the representation of entity e from $G(q)$: *(a) concatenation* +, *(b) Hadamard product* $\odot$, and *(c) subtraction* −, and an additional *(d) all*, which concatenates (a), (b), and (c) together as features. The parameters $k = 2$, $\delta = 4$, and $n^{CXT} = 10$ are chosen based on grid search.
- **LTRMHSA.** We employ a learning-to-rank with multi-head self-attention model to extract the most context-relevant entities as context entities. We set $n^{CXT} = 10$ to stay consistent with KEMatch.

We use a pretrained DBpedia k-bert model as the context encoder [38]. The model uses the following configuration: $L = 12$, $A = 12$, $H = 768$.

We measure the performance of context entity extraction by precision and recall defined by **Precision**= $\frac{|E(q) \cap E^*(q)|}{|E(q)|}$ and **Recall**= $\frac{|E(q) \cap E^*(q)|}{|E^*(q)|}$, respectively. $E(q)$ denotes the set of context entities extracted by the extractor and $E^*(q)$ is the set of ground truth context entities. We utilize the negative sampling process described under Section 3 and conduct 10-fold cross validation. As we have consistent observations on Wiki-film and Wiki-music and due to limitations in space,

Table 2. Performance in Context Entity Extraction (Wiki-Film)

Context Extractor		Feature Combinations	Precall	Recall
Context Window			68.4	62.3
KEMatch	AVG Entity Rep.	+	73.5	70.9
		⊙	75.6	73.8
		−	72.9	70.1
		all	72.1	69.3
	Mention Paragraph Rep.	+	75.4	73.2
		⊙	80.1	77.6
		−	72.7	71.6
		all	71.5	69.8
	Title + Mention Paragraph Rep.	+	69.20	65.4
	Cxt. Query Entity Rep.	+	74.7	72.3
		⊙	81.3	78.9
		−	72.5	71.7
		all	71.9	70.4
LTRMHSA			**85.9**	**83.1**

we only show the experiment result of the Wiki-film dataset in Table 2. In this comparison, we additionally include the context window baseline.

LTRMHSA, which considers inter-entity interaction, yields the best precision (85.9%) and recall (83.1%). KEMatch using contextual query entity representation with the Hadamard product feature combination option, returns the next best precision (81.3%) and recall (78.9%). This is followed by KEMatch using mention paragraph representation also with the Hadamard product feature combination option. These results suggest that the contextualized query entity representation is the best option for KEMatch. Finally, the context window baseline is the worst-performing method as it does not involve the knowledge graph entities.

Among the other KEMatch variants, average entity representation performs better than title and mention paragraph representation, possibly due to the latter's high feature dimensionality. We observe features with a high coefficient have been assigned to both p_1 and p_q. This observation supports our hypothesis that the title paragraph contains useful information about the contextual relationship between the query entities. Finally, among the different feature representations, the Hadamard product achieves the best performance, followed by concatenation and subtraction. Although all three of these feature representations show decent predictive results, combining them together does not result in better performance due to high feature dimension.

While Table 2 shows that LTRMHSA enjoys higher accuracy than all of the KEMatch variants, the former involves a computational complexity of $O(m_q^2)$ compared with KEMatch's $O(m_q)$. When m_q is large, the computational overhead of LTRMHSA could be significantly higher, which adversely affects the time needed for generating a contextual path. As a result, one should take into consideration the tradeoff between performance and execution time when deciding which context extractor to use together with NMCPGT in practice.

5.2 Performance in CPG

Next, we conduct experiments to evaluate the models' performance in generating the contextual paths. We compare our proposed models combining NMCPGT with different context extractor options, namely, KEMatch and its variants, LTRMHSA, context window entities, and random context

Table 3. Path Generation Performance on Wiki-Film and Wiki-Music Datasets

	Dataset	Feat. Comb.	Wiki-Film %Recov	Wiki-Film AVG PW Sim	Wiki-Film NGEO(E), NGEO(R)	Wiki-Music %Recov	Wiki-Music AVG PW Sim	Wiki-Music NGEO(E), NGEO(R)
NCNMPG			19.7	0.44	0.29, 0.24	20.32	0.46	0.29, 0.23
NMCPGT	Random Context		62.33	0.59	0.26, 0.22	60.15	0.58	0.27, 0.22
	Context Window		73.27	0.75	0.2, 0.19	75.22	0.74	0.21, 0.2
	KEMatch AVG Ent Rep.	+	76.76	0.83	0.17, 0.16	78.13	0.81	0.17, 0.15
		⊙	78.14	0.84	0.17, 0.16	78.92	0.81	0.17, 0.15
		−	71.11	0.78	0.19, 0.18	70.37	0.80	0.2, 0.18
		all	73.2	0.79	0.18, 0.17	74.52	0.78	0.18, 0.16
	KEMatch Mention Para. Rep.	+	77.41	0.86	0.17, 0.15	76.49	0.85	0.17, 0.16
		⊙	80.13	0.87	0.16, 0.15	80.41	0.87	0.16, 0.16
		−	73.29	0.81	0.18, 0.18	72.48	0.82	0.19, 0.18
		all	74.19	0.81	0.18, 0.17	74.07	0.83	0.19, 0.18
	KEMatch Title + Mention Para. Rep.	+	72.28	0.78	0.18, 0.16	71.78	0.77	0.19, 0.16
	KEMatch Cxt. Query Ent Rep.	⊙	81.2	0.87	0.15, 0.15	82.89	0.88	0.16, 0.15
	LTRMHSA		**84.13**	**0.89**	**0.14, 0.14**	**85.37**	**0.91**	**0.14, 0.14**

entities. The *random context entities* method randomly selects n^{CXT} entities from $G(q)$ as the context entities. We set $n^{CXT} = 10$, which is consistent with the n^{CXT} for KEMatch and LTRMHSA. For simplicity, we name our models by "NMCPGT+⟨context extractor method⟩" where the ⟨context extractor method⟩ options are Random Context, Context Window, KEMatch, and LTRMHSA.

To evaluate the importance of the context document, we additionally include a **non-contextual non-monotonic path generation (NCNMPG)** baseline that generates the contextual path from path generator that only takes e_h and e_t as input, without any knowledge of the context document. In other words, this model does not use any context entities as condition, and only relies on the pretrained transformer when generating paths.

For the path generation, we only include the annealed coaching oracle $\pi^*_{annealed}$ as it has been shown to outperform the uniform and coaching oracles in [6]. The model is optimized with cross entropy loss. The non-monotonic path generator uses a four-layer transformer structure with four attention heads, hidden dimension of 256, and feed-forward dimension of 1,024. Adam optimizer is used with the initial learning rate of 10^{-5}. The model is trained with 100 epochs. After the 50 burn-in epochs, β is linearly annealed from 1.0 by 0.01 in each epoch. The implementation of all neural structures is based on Pytorch.

We report the % ground truth recovered (**%Recov**), averaged pairwise similarity (**AVG PW Sim**), **NGEO(E)**, and **NGEO(R)** of the generated contextual paths by different models in Table 3. We have similar observations based on the experiments on the two real-world datasets. The CPG performance is also consistent with the results in Table 2. The model NMCPGT+LTRMHSA outperforms other models for all metrics, followed by NMCPGT+KEMatch with context query entity representation and mention paragraph representation. Among the NMCPGT+KEMatch model variants, the Hadamard product is the best feature combination method while subtraction is the worst. The simple baselines that do not extract context entities, including Non-Contextualized Generation and Random Context baseline, perform the worst. This indicates the importance of a good context extractor—the better we extract context entities, the more accurate the generated contextual paths will be. In the following, we use two case examples to illustrate the difference between the different models.

Table 4. Non-monotonic (Non-M) and Monotonic (M) Generation

	%Unf	%Recov	AVG PW Sim	NGEO(E),NGEO(R)
$\pi^*_{annealed}$ (Non-Monotonic)	0	84.13	0.89	0.14, 0.14
$\pi^*_{L\text{-}R}$ (Monotonic)	29.73	42.85	0.73	0.19, 0.16

Case example 1. Consider the query entities *Zacharias Kunuk* and *Inuit* in the following context from Wiki-film:

> *"Produced by an Igloolik, Nunavut company, the film is titled The Journals of Knud Rassmusen, and co-directed by* ***Zacharias Kunuk*** *of Igloolik and Norman Cohn of Montreal. The company received critical acclaim for their first film, Atanarjuat, ... The film portrays the pressures on traditional* ***Inuit*** *culture...."*[4]

The ground truth contextual path is $e_{\text{Kunuk}} \xrightarrow{\textbf{director}} e_{\text{TheJournalOfKnudRasmussen}} \xrightarrow{\textbf{pageLink}} e_{\text{Inuit}}$, which is successfully generated by both NMCPGT+LTRMHSA and NMCPGT+KEMatch with context query entity representation. NMCPGT+KEMatch with mention paragraph representation, on the other hand, generates $e_{\text{Kunuk}} \xrightarrow{\textbf{producer}} e_{\text{TheJournalOfKnudRasmussen}} \xrightarrow{\textbf{pageLink}} e_{\text{Inuit}}$. Although this is not a ground truth path, it is semantically correct as *Kunuk* is the co-founder of the production company *Igloolik, Nunavut company*. Both NMCPGT+context window and NMCPGT+KEMatch with AVG entity representation extractors generate $e_{\text{Kunuk}} \xrightarrow{\textbf{director}} e_{\text{Atanarjuat}} \xrightarrow{\textbf{pageLink}} e_{\text{Inuit}}$, which suggests the generation of path is affected by entities within the context document. Finally, the non-contextualized generation and NMCPGT+random context baseline generates the shortest path between the query entities in knowledge graph: $e_{\text{Kunuk}} \xrightarrow{\textbf{pageLink}} e_{\text{Inuit}}$.

Case example 2. Consider query entities *James Bond* and *Pinewood Studio* in the following context from Wiki-film:

> *"Firefighters have confirmed that the large* ***James Bond*** *sound stage at* ***Pinewood Studios*** *has been destroyed by fire... where filming for Casino Royale, the latest Bond movie, has been completed... Pinewood, which was created in 1935, was the filming ground for Dr No, the first ever James Bond movie in 1962."*[5]

Both NMCPGT+LTRMHSA and NMCPGT+KEMatch with context query entity representation generate $e_{\text{JamesBond}} \xrightarrow{\textbf{isSeriesOf}} e_{\text{Dr.No}} \xrightarrow{\textbf{pageLink}} e_{\text{PinewoodStudio}}$, which is different from the ground truth path $e_{\text{JamesBond}} \xrightarrow{\textbf{isSeriesOf}} e_{\text{CasinoRoyale}} \xrightarrow{\textbf{pageLink}} e_{\text{PinewoodStudio}}$. Nevertheless, the path is considered semantically correct as the event carried by both paths exists in the context document. NMCPGT+KEMatch with mention paragraph representation successfully generates the ground truth path as it only focuses on the paragraph in which *Casino Royale* is mentioned. It is not affected by the mention of *Dr. No*.

5.3 Comparison between Non-monotonic and Monotonic Path Generation

In this section, we compare our non-monotonic path generator NMCPGT+LTRMHSA with a left-to-right counterpart by evaluating them on the Wiki-film dataset. As discussed in Section 3.3, we

[4] https://en.wikinews.org/wiki/Film_from_Nunavut_in_Canada%27s_north_to_open_TIFF
[5] https://en.wikinews.org/wiki/James_Bond_set_at_Pinewood_Studios_destroyed_by_fire

choose to use a non-monotonic generation model because left-to-right generation models are likely to generate unfinished paths.

To conduct this experiment, we replace $\pi^*_{annealed}$ of NMCPGT by a Left-to-Right Oracle $\pi^*_{L\text{-}R}$, which always assigns probability of 1 to the leftmost to-be-generated path element of the sequence. As suggested in [6], $\pi^*_{L\text{-}R}$ results in maximum likelihood learning of an autoregressive sequence model, which makes the generation process identical to neural sequence models such as GPT-2 [48]. We use the same input sequence, *"[soc] e_1 e_2 ... $e_{|E(q)|}$ [sep] e_h [e] e_t [e] [eoc]"*, for fine-tuning the non-monotonic and monotonic models. In this experiment, we include a new metric representing the percentage of **%unfinished path** in the result (%Unf) to evaluate the model's ability to complete the path. We define an unfinished path as one that starts with e_h but could not end with e_t within $L_{MAX} = 6$ relation edges. Note that the generated paths, finished or not, may involve inferred relation edges. We report the experiment result on the Wiki-film dataset for NMCPGT+LTRMHSA in Table 4. Since NMCPGT+LTRMHSA is required to generate path elements between e_h and e_t, it has %Unf = 0. The monotonic model, however, sees 29.73% of the generated paths unfinished. With fewer finished paths, the %path recovered of the monotonic model is also substantially less than NMCPGT+LTRMHSA.

While the monotonic model performs badly in well-formed path generation, it achieves decent average pairwise similarity and NGEO results. This suggests that the monotonic model generates entities and relation labels that are still relevant to the context with the help of LTRMHSA context extractor. For instance, for the query with entities ($e_{\text{Brokeback Mountain}}$, $e_{\text{Mel Gibson}}$) and the following context document:

> *"...Ledger starred in the 2005 movie **Brokeback Mountain** where he was nominated for the Academy Award and the Golden Globe Award for Best Actor. He also starred in the 2000 movie The Patriot with **Mel Gibson**..."*[6]

The ground truth contextual path of this query should be $e_{\text{BrokebackMountain}} \xrightarrow{\textbf{starring}} e_{\text{Ledger}} \xrightarrow{\textbf{starring}} e_{\text{ThePatriot}} \xrightarrow{\textbf{starring}} e_{\text{Gibson}}$. The non-monotonic model using $\pi^*_{annealed}$ generates the ground truth path perfectly. On the other hand, the monotonic model using $\pi^*_{L\text{-}R}$ generates a path that is unfinished: $e_{\text{BrokebackMountain}} \xrightarrow{\textbf{subject}} e_{\text{AcademyAward}} \xrightarrow{\textbf{subject}} e_{\text{AcademyAwardForBestActor}} \xrightarrow{\textbf{subject}} e_{\text{Ledger}} \xrightarrow{\textbf{starring}} e_{\text{ThePatriot}} \xrightarrow{\textbf{pageLink}} e_{\text{Ledger}}$. Nevertheless, the generated path is still very relevant to the context. In fact, if we do not force the model to stop at the length of 6, it will eventually reach $e_{\text{Mel Gibson}}$ at the 8th hop. This example explains why we obtain relatively high pairwise similarity and NGEO for the monotonic model.

6 DISCUSSION

Our proposed two-stage CPG framework consisting of context extractor and contextual path generator is unique in knowledge graph reasoning research. It allows context entities relevant to a query to be used as input to contextual path generator to control the path generated by the latter. Moreover, the path generator outputs a relevant and well-formed contextual path using a non-monotonic path generation approach. To our best knowledge, such an approach has not been attempted in previous works. In the following, we discuss the potential impact this article may have on knowledge reasoning research and applications.

In the research aspect, this article represents an early work to address the emerging problem of inferring new relation edges or facts for a knowledge graph from input textual data. When

[6] https://en.wikinews.org/wiki/Australian_actor_Heath_Ledger_found_dead_in_New_York_City

combined with information retrieval research, CPG can offer useful explanations to the content found in search results. This paves the foundational work for future research in information retrieval with explainable AI. From the technical standpoint, the NMCPGT is also a research breakthrough as it combines the strengths of efficient relation edge inference, well-formed CPG, and efficient path generation together in a model. Most of the previous works that use a knowledge graph to reason (e.g., MHQA-GRN [54] and LEGO [49]) generate knowledge paths by traversing the knowledge graph. Such methods suffer from missing knowledge graph relation edges, long relation edge inference times, and poorly formed paths. In contrast, our pretrained transformer method for knowledge path generation is more versatile, and can be easily fine-tuned for many different tasks. It may subsequently evolve into a large pretrained generative model for complex reasoning and as a problem solving tool.

In the practical aspect, CPG overlays a context document with contextual paths. This enhances content understanding as both the known and inferred relation edges of contextual paths between entities can be highlighted to the readers. For instance, a content analysis system could process documents with CPG and obtain contextual paths of entity pairs mentioned in the documents. The users can therefore access these contextual paths and entity descriptions provided by the knowledge graph as additional background information about the entity pairs, and to acquire a more complete contextual knowledge about the document content for knowledge-enhanced information retrieval tasks.

Finally, it is possible to automatically enrich a knowledge graph with CPG. As discussed in Section 2, knowledge graphs can be incomplete and outdated, which is a common limitation faced by many knowledge graph–based applications. CPG can help to address this by inferring new relation edges from a large enough set of documents mentioning the knowledge graph entities. This can be especially helpful for information systems that require up-to-date information to deliver the best outcomes, e.g., decision-making systems, competitive intelligence analytics systems, and so forth.

7 CONCLUSION

In this article, we aim to enhance the understanding of textual documents by deriving knowledge paths among the mentions of entities. To address the challenges in path generation, we proposed a two-stage CPG framework that can handle (i) noisy context information, (ii) missing relation edges in knowledge graphs, and (iii) generate well-formed paths. We propose novel context entity extraction methods and develop non-monotonic generation with pretraining to overcome the above challenges. For context entity extraction, we introduce knowledge-enabled embedding matching and learning-to-rank with multi-head self-attention methods. The latter yields higher accuracy than the former. For CPG, we propose NMCPGT, which is capable of generating contextual paths that resemble an oracle policy. Our experiments demonstrate that our proposed framework yielded the best accuracy, surpassing previous state-of-the-art methods.

Due to the high cost of annotation efforts, we only conducted experiments on small real datasets built with Wikinews articles. However, with more datasets annotated in the future, we will be able to conduct experiments on larger real datasets to produce more comprehensive results and findings. Moreover, the current work assumes that only one ground truth contextual path exists between a query entity pair, which may not hold in a real-world scenario. Therefore, we plan to further refine both the datasets and experiment design to address this limitation in our follow-up works.

There are several other interesting future directions for CPG research. First, CPG can be adapted to other research tasks in different types of knowledge graph–enhanced information systems. For example, in a recommendation system, we can extend CPG to recommend items by constructing a path out of the user's product purchase and browsing history. In search engine research, one can

also conduct design novel methods to summarize search results using contextual paths. Secondly, CPG can be viewed as an interesting means to derive new relation edges not found in the existing knowledge graphs. For this to work at scale, we require many context documents of the same knowledge domain to be provided. It should also be compared and evaluated against other knowledge completion methods. Thirdly, CPG research can be extended to find contextual subgraphs (instead of contextual paths) connecting a set of input entities in a context document. The generated subgraphs will be more easily used in downstream applications. Finally, it is worthwhile to explore how to create a large pretrained knowledge path model using the proposed NMCPGT. With recent advancements in large pretrained language models, we hope that a large pretrained knowledge path model can be developed to generate high-quality and accurate paths across different knowledge domains. We hypothesize that such pretrained large knowledge path models will have better generation and reasoning abilities, which can make them useful in many downstream applications.

APPENDIX A

A EXPERIMENTS ON SYNTHETIC DATASET

In this section, we explore the scalability of our proposed framework using a large-scale synthetic dataset. We use the synthetic dataset from [40], and show the statistics of the dataset in Table 5. This large-scale dataset overcomes the limitation of real-world datasets that are usually small due to the high data construction cost. We show the experiment results of our CPG models on the synthetic dataset, which focus on evaluating how well our proposed CPG models cope with a large dataset. Finally, we inspect our models' ability to infer new relation edges given an incomplete knowledge graph.

Table 5. Synthetic Dataset Statistics

	Synthetic Dataset
# context documents	2,000
# entities mentioned	19,173
# entity pairs	5,000
AVG/MAX ground truth path length	4/6
Knowledge Graph	
# entities	59,173
# relations	651
Ground Truth Paths	
AVG path length	4
# unique path entities	19,173
# unique path relations	648

A.1 Coping with Large Dataset

We conduct the evaluation of selected models and baselines with the synthetic dataset, as shown in Table 6. The results are generally consistent with those in the Wiki-film and Wiki-music datasets. However, we observe that the performance results of all models are poorer in the synthetic dataset. For example, the % Path Recovered of NMCPGT+LTRMHSA is about 10% less than that of the same model on Wiki-music. This may be due to the additional noises introduced to the contextual documents and paths when constructing the synthetic dataset.

As the synthetic dataset is significantly larger, we also examine the efficiency of our proposed model. We conduct the experiments on a Tesla V100 GPU server with 32 GB memory. With 4,000 training instances, the training process costs 1 h 20 min 34 sec. The generation of contextual path

Table 6. Path Generation Performance on Synthetic Dataset

Dataset		Feature Combination	%Recov	AVG PW Sim	NGEO(E), NGEO(R)
NCNMPG			16.2	0.32	0.32, 0.25
NMCPGT	Random Context		55.43	0.47	0.29, 0.23
	Context Window		61.83	0.56	0.25, 0.22
	KEMatch AVG Ent Rep.	+	63.46	0.68	0.25, 0.22
		$\odot$	64.27	0.68	0.24, 0.2
		−	57.2	0.55	0.27, 0.23
		all	62.35	0.64	0.25, 0.22
	KEMatch Mention Para. Rep.	+	63.59	0.65	0.25, 0.21
		$\odot$	73.66	0.71	0.21, 0.18
		−	58.74	0.53	0.26, 0.21
		all	61.57	0.58	0.26, 0.21
	KEMatch Title + Mention Para. Rep.	+	60.91	0.58	0.27, 0.21
	KEMatch Cxt. Query Ent Rep.	$\odot$	<u>75.37</u>	<u>0.74</u>	<u>0.18, 0.17</u>
	LTRMHSA		**75.92**	**0.76**	**0.17, 0.16**

Table 7. Inferring Missing Relations using NMCPGT+LTRMHSA

k	%Recov	AVG PW Sim	NGEO(E),NGEO(R)	$\%E^{-}_{CP}$	$\%E^{-}_{CP}$ Recov
0 (Full KG)	84.13	0.89	0.14, 0.14	-	-
10	80.22	0.86	0.15, 0.14	5.37	65.92
30	65.23	0.71	0.19, 0,17	19.31	57.23
50	47.37	0.59	0.24, 0.23	38.45	40.85

E^{-}_{CP}: Removed edges that exist in the ground truth contextual path.

for a test query requires around 0.86 second. This result shows the ability of our two-stage model in handling large datasets.

A.2 Inferring New Relation Edges

To evaluate the models' abilities to infer edges that are not observed in the current knowledge graph, we randomly drop k% of the edges from the knowledge graph of the synthetic dataset. We report the performance of MNCPGT+LTRMHSA only, our best performing model, in Table 7. In addition to % path recovered (**%Recov**), AVG pairwise similarity (**AVG PW Sim**), and normalized graph edit distance (**NGEO**), we also report $\%E^{-}_{CP}$ **Recov**, which represents the percentage of ground truth path edges that are recovered.

The results show that as k increases, our model performs poorer by all metrics. When k is small (e.g., $k = 10$), the model successfully recovers 65.92% of the ground truth contextual path edges that are removed from the knowledge graph. When $k = 50$, the %Recov performance drops significantly to 47.37% but the model still recovers 40.85% of removed ground truth edges. Empirically, we also observe that NMCPGT+LTRMHSA is more likely to recover edges between entities that are logically or semantically related. For instance, we find that a movie m is inferred to be made in country c (i.e., $e_m \xrightarrow{\textbf{country}} e_{\text{France}}$) if it mostly stars actors and actresses from c. As a

result, the model successfully recovers $e_{\text{Top Gun}} \xrightarrow{\textbf{country}} e_{\text{United States}}$, and $e_{\text{Amour(film)}} \xrightarrow{\textbf{country}} e_{\text{France}}$. On the other hand, the model fails to recover edges like $e_{\text{Ridley Scott}} \xrightarrow{\textbf{producer}} e_{\text{American Gangster(film)}}$, $e_{\text{Crimson Tide}} \xrightarrow{\textbf{starring}} e_{\text{Viggo Mortensen}}$, and $e_{\text{Clint Eastwood}} \xrightarrow{\textbf{artist}} e_{\text{Bar Room Buddies}}$. These are edges with specific semantics and thus more difficult to infer.

ACKNOWLEDGMENTS

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over Wikipedia graph for question answering. In *International Conference on Learning Representations (ICLR '20)*. https://doi.org/10.48550/arXiv.1911.10470

[2] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss Markov random fields and probabilistic soft logic. *The Journal of Machine Learning Research* 18, 1 (Jan. 2017), 3846–3912. https://doi.org/10.48550/arXiv.1505.04406

[3] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5, 2 (1983), 179–190. https://doi.org/10.1109/TPAMI.1983.4767370

[4] Laura Biester, Katie Matton, Janarthanan Rajendran, Emily Mower Provost, and Rada Mihalcea. 2021. Understanding the impact of COVID-19 on online mental health forums. *ACM Transactions on Management Information Systems* 12, 4 (Sept. 2021), 31:1–31:28. https://doi.org/10.1145/3458770

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, Vol. 26.

[6] Kiante Brantley, Kyunghyun Cho, Hal Daumé, and Sean Welleck. 2019. Non-monotonic sequential text generation. In *Workshop on Widening NLP*.

[7] Kirill Bryanov and Victoria Vziatysheva. 2021. Determinants of individuals' belief in fake news: A scoping review determinants of belief in fake news. *PLoS One* 16, 6 (Jun. 2021), e0253717. https://doi.org/10.1371/journal.pone.0253717

[8] Chi Chen, Xin Peng, Zhenchang Xing, Jun Sun, Xin Wang, Yifan Zhao, and Wenyun Zhao. 2022. Holistic combination of structural and textual code information for context based API recommendation. *IEEE Transactions on Software Engineering* 48, 8 (Aug. 2022), 2987–3009. https://doi.org/10.1109/TSE.2021.3074309

[9] Qiangui Chen and Bing Li. 2018. Retrieval method of electronic medical records based on rules and knowledge graph. In *ICEB 2018 Proceedings*.

[10] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *IEEE Access* 8 (2020), 192435–192456. https://doi.org/10.1109/ACCESS.2020.3030076

[11] Billy Chiu, Eric SEE-TO Wing Kuen, and E. Ngai. 2021. Knowledge graph construction and applications in e-retailing: A review of literature. In *The annual Pacific Asia Conference on Information Systems*.

[12] Zhihong Cui, Hongxu Chen, Lizhen Cui, Shijun Liu, Xueyan Liu, Guandong Xu, and Hongzhi Yin. 2022. Reinforced KGs reasoning for explainable sequential recommendation. *World Wide Web* 25, 2 (Mar. 2022), 631–654. https://doi.org/10.1007/s11280-021-00902-6

[13] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations (ICLR '18)*.

[14] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL '17)*.

[15] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations (ICLR '20)*.

[16] Weiwei Deng, Peihu Zhu, and Jian Ma. 2018. Enhancing group recommendation by knowledge graph. *The Annual Pacific Asia Conference on Information Systems*.

[17] Laura Dietz, Alexander Kotov, and Edgar Meij. 2018. Utilizing knowledge graphs for text-centric information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. New York, NY, 1387–1390. https://doi.org/10.1145/3209978.3210187

[18] Zhengxiao Du, Chang Zhou, Jiangchao Yao, Teng Tu, Letian Cheng, Hongxia Yang, Jingren Zhou, and Jie Tang. 2021. CogKR: Cognitive graph for multi-hop knowledge reasoning. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. https://doi.org/10.1109/TKDE.2021.3104310

[19] Jordana George, Natalie Gerhart, and Russell Torres. 2021. Uncovering the truth about fake news: A research model grounded in multi-disciplinary literature. *Journal of Management Information Systems* 38, 4 (Oct. 2021), 1067–1094. https://doi.org/10.1080/07421222.2021.1990608

[20] Miriam Gil, Victoria Torres, Manoli Albert, and Vicente Pelechano. 2021. Extracting knowledge from software artefacts to assist software project stakeholders. *International Conference on Information Systems Development*.

[21] Fréderic Godin, Anjishnu Kumar, and Arpit Mittal. 2019. Learning when not to answer: A ternary reward structure for reinforcement learning based question answering. In *Conference of the North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.48550/arXiv.1902.10236

[22] Ram D. Gopal, Hooman Hidaji, Sule Nur Kutlu, Raymond A. Patterson, Erik Rolland, and Dmitry Zhdanov. 2020. Real or not? Identifying untrustworthy news websites using third-party partnerships. *ACM Transactions on Management Information Systems* 11, 3 (Jul. 2020), 10:1–10:20. https://doi.org/10.1145/3382188

[23] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *The 34th International Conference on Machine Learning*. https://doi.org/10.48550/arXiv.1703.00955

[24] Shi Ming Huang, David C. Yen, Ting Jyun Yan, and Yi Ting Yang. 2021. An intelligent mechanism to automatically discover emerging technology trends: Exploring regulatory technology. *ACM Transactions on Management Information Systems* 13, 2 (Dec. 2021), 17:1–17:29. https://doi.org/10.1145/3485187

[25] Annervaz K M, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. 2018. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing. In *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/N18-1029

[26] Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Veronika Thost. 2018. Attributed description logics: Reasoning on knowledge graphs. In *The 27th International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2018/743

[27] Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics* 8 (2020), 329–345. https://doi.org/10.1162/tacl_a_00318

[28] Zixuan Li, Xiaolong Jin, Saiping Guan, Yuanzhuo Wang, and Xueqi Cheng. 2018. Path reasoning over knowledge graph: A multi-agent and reinforcement learning based method. In *2018 IEEE International Conference on Data Mining Workshops*.

[29] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://doi.org/10.1145/3404835.3462963

[30] Abel Lim Jun Hong, Ragunathan Mariappan, and Vaibhav Rajan. 2020. Adverse drug event prediction using noisy literature-derived knowledge graphs. In *The International Conference on Information Systems* .

[31] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *The 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

[32] Qika Lin, Jun Liu, Yudai Pan, Lingling Zhang, Xin Hu, and Jie Ma. 2021. Rule-enhanced iterative complementation for knowledge graph reasoning. *Information Sciences* 575 (Oct. 2021), 66–79. https://doi.org/0.1016/j.ins.2021.06.040

[33] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *The 29th AAAI Conference on Artificial Intelligence*.

[34] Sanne van der Linden, Rita Sevastjanova, Mathias Funk, and Mennatallah El-Assady. 2022. MediCoSpace: Visual decision-support for doctor-patient consultations using medical concept spaces from EHRs. *ACM Transactions on Management Information Systems* (Sep. 2022). https://doi.org/10.1145/3564275

[35] Hao Liu, Shuwang Zhou, Changfang Chen, Tianlei Gao, Jiyong Xu, and Minglei Shu. 2022. Dynamic knowledge graph reasoning based on deep reinforcement learning. *Knowledge-Based Systems* 241 (Apr. 2022), 108235. https://doi.org/10.1016/j.knosys.2022.108235

[36] Jue Liu, Zhuocheng Lu, and Wei Du. 2019. Combining enterprise knowledge graph and news sentiment analysis for stock price prediction. In *The 52nd Hawaii International Conference on System Sciences*. 1247–1255.

[37] Jue Liu, Zhuocheng Lu, and Wei Du. 2019. Combining enterprise knowledge graph and news sentiment analysis for stock price prediction. In *Hawaii International Conference on System Sciences*.

[38] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 0303 (Apr. 2020), 2901–2908. https://doi.org/10.1609/aaai.v34i03.5681

[39] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021. KG-BART: Knowledge graph-augmented BART for generative commonsense reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 7 (May 2021), 6418–6425.

[40] Pei-Chi Lo and Ee-Peng Lim. 2023. Contextual path retrieval: A contextual entity relation embedding-based approach. *ACM Transactions on Information Systems* 41, 1 (Jan. 2023), Article 1. https://doi.org/10.1145/3502720

[41] Wanlun Ma, Xiangyu Hu, Chao Chen, Sheng Wen, Kkwang Raymond Choo, and Yang Xiang. 2022. Social media event prediction using DNN with feedback mechanism. *ACM Transactions on Management Information Systems* 13, 3 (May 2022), 33:1–33:24. https://doi.org/10.1145/3522759

[42] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticæ Investigationes* 30, 1 (Jan. 2007), 3–26. https://doi.org/10.1075/li.30.1.03nad

[43] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. 2021. Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys* 54, 1 (Feb. 2021), 20:1–20:39. https://doi.org/10.1145/3445965

[44] Byungkook Oh, Seungmin Seo, Jimin Hwang, Dongho Lee, and Kyong-Ho Lee. 2022. Open-world knowledge graph completion for unseen entities and relations via attentive feature aggregation. *Information Sciences* 586 (Mar. 2022), 468–484. https://doi.org/10.1016/j.ins.2021.11.085

[45] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio Rodriguez Mendez, and Armin Haller. 2022. Active knowledge graph completion. *Information Sciences* 604 (Aug. 2022), 267–279. https://doi.org/10.1016/j.ins.2022.05.027

[46] Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *The 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/N19-1263

[47] Meng Qu and Jian Tang. 2019. Probabilistic logic neural networks for reasoning. In *Advances in Neural Information Processing Systems*.

[48] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

[49] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. LEGO: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *The 38th International Conference on Machine Learning*.

[50] Daniel Ringler and Heiko Paulheim. 2017. One knowledge graph to rule them all? Analyzing the differences between DBpedia, YAGO, Wikidata & Co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, 366–372.

[51] Jonas Scharfenberger, Burkhardt Funk, and Benjamin Mueller. 2021. The augmented theorist—toward automated knowledge extraction from conceptual models. In *The International Conference on Information Systems*.

[52] Pengpeng Shao, Dawei Zhang, Guohua Yang, Jianhua Tao, Feihu Che, and Tong Liu. 2022. Tucker decomposition-based temporal knowledge graph completion. *Knowledge-Based Systems* 238 (Feb. 2022), 107841. https://doi.org/10.1016/j.knosys.2021.107841

[53] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *The 31st International Conference on Neural Information Processing Systems*.

[54] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2022. Evidence integration for multi-hop reading comprehension with graph neural networks. *IEEE Transactions on Knowledge and Data Engineering* 34, 2 (Feb. 2022), 631–639. https://doi.org/10.1109/TKDE.2020.2982894

[55] Xing Tang, Ling Chen, Jun Cui, and Baogang Wei. 2019. Knowledge representation learning with entity descriptions, hierarchical types, and textual relations. *Information Processing & Management* 56, 3 (May 2019), 809–822. https://doi.org/10.1016/j.ipm.2019.01.005

[56] Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (Apr. 2020), 9073–9080.

[57] Moshe Unger, Alexander Tuzhilin, and Amit Livne. 2020. Context-aware recommendations based on deep learning frameworks. *ACM Transactions on Management Information Systems* 11, 2 (May 2020), Article 8, 15 pages. https://doi.org/10.1145/3386243

[58] 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 *Advances in Neural Information Processing Systems*.

[59] Guojia Wan and Bo Du. 2021. GaussianPath: A Bayesian multi-hop reasoning framework for knowledge graph reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (May 2021), 4393–4401.

[60] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In *The Web Conference 2021*. 1737–1748. https://doi.org/10.1145/3442381.3450043

[61] Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. https://doi.org/10.18653/v1/2020.findings-emnlp.369

[62] Colby Wise, Miguel Romero Calvo, Pariminder Bhatia, Vassilis Ioannidis, George Karypus, George Price, Xiang Song, Ryan Brand, and Ninad Kulkani. 2020. COVID-19 knowledge graph: Accelerating information retrieval and discovery for scientific literature. In *Knowledgeable NLP: The First Workshop on Integrating Structured Knowledge and Neural Networks for NLP*. 1–10.

[63] Jiayi Xu, Shuang Zhang, Zhen Zhu, Lincan Zou, and Mengting Yang. 2022. Exploratory research on knowledge graph construction and attribute value extraction for large-scale textual data of tourism products. *The Annual Wuhan International Conference on E-Business*.

[64] Ruiyun Xu, Hailiang Chen, and J. Leon Zhao. 2020. Measuring social proximity via knowledge graph embedding. *The International Conference on Information Systems*.

[65] Bingcong Xue and Lei Zou. 2022. Knowledge graph quality management: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–1. https://doi.org/10.1109/TKDE.2022.3150080

[66] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*.

[67] Shuang (Sophie) Zhai and Zhu (Drew) Zhang. 2022. Read the news, not the books: Forecasting firms' long-term financial performance via deep text mining. *ACM Transactions on Management Information Systems* 14, 1 (May 2022), 1–37. https://doi.org/10.1145/3533018

[68] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *The 32nd AAAI Conference on Artificial Intelligence, The 30th Innovative Applications of Artificial Intelligence Conference and, The 8th AAAI Symposium on Educational Advances in Artificial Intelligence*.

[69] Yuxin Zhang, Kunlin Yang, Wei Du, and Wei Xu. 2018. Predicting stock price movement direction with enterprise knowledge graph. *The Annual Pacific Asia Conference on Information Systems*.

[70] Weiguo Zheng, Lei Zou, Wei Peng, Xifeng Yan, Shaoxu Song, and Dongyan Zhao. 2016. Semantic SPARQL similarity search over RDF knowledge graphs. *Proceedings of the VLDB Endowment* 9, 11 (Jul. 2016), 840–851. https://doi.org/10.14778/2983200.2983201

[71] J. Zhu, Y. He, G. Zhao, X. Bo, and X. Qian. 2022. Joint reason generation and rating prediction for explainable recommendation. *IEEE Transactions on Knowledge & Data Engineering* 01 (Jan. 2022), 1–1. https://doi.org/10.1109/TKDE.2022.3146178

[72] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating BERT into neural machine translation. In *International Conference on Learning Representations*.

[73] Qiannan Zhu, Xiaofei Zhou, Jianlong Tan, and Li Guo. 2021. Knowledge base reasoning with convolutional-based recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering* 33, 5 (May 2021), 2015–2028. https://doi.org/10.1109/TKDE.2019.2951103

Received 4 November 2022; revised 30 March 2023; accepted 18 April 2023