

Singapore Management University

## Institutional Knowledge at Singapore Management University

---

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

---

1-2023

### Contextual Path Retrieval: A contextual entity relation embedding-based approach

Pei-chi LO

Singapore Management University, pclo.2017@phdcs.smu.edu.sg

Ee-peng LIM

Singapore Management University, eplim@smu.edu.sg

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)



Part of the [Databases and Information Systems Commons](#)

---

#### Citation

LO, Pei-chi and LIM, Ee-peng. Contextual Path Retrieval: A contextual entity relation embedding-based approach. (2023). *ACM Transactions on Information Systems*. 41, (1), 1-38.

Available at: [https://ink.library.smu.edu.sg/sis\\_research/7780](https://ink.library.smu.edu.sg/sis_research/7780)

This Journal Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [cherylds@smu.edu.sg](mailto:cherylds@smu.edu.sg).



# Contextual Path Retrieval: A Contextual Entity Relation Embedding-based Approach

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

**Contextual path retrieval** (CPR) refers to the task of finding contextual path(s) between a pair of entities in a knowledge graph that explains the connection between them in a given context. For this novel retrieval task, we propose the Embedding-based Contextual Path Retrieval (ECPR) framework. ECPR is based on a three-component structure that includes a context encoder and path encoder that encode query context and path, respectively, and a path ranker that assigns a ranking score to each candidate path to determine the one that should be the contextual path. For context encoding, we propose two novel context encoding methods, i.e., context-fused entity embeddings and contextualized embeddings. For path encoding, we propose PathVAE, an inductive embedding approach to generate path representations. Finally, we explore two path-ranking approaches. In our evaluation, we construct a synthetic dataset from Wikipedia and two real datasets of Wikinews articles constructed through crowdsourcing. Our experiments show that methods based on ECPR framework outperform baseline methods, and that our two proposed context encoders yield significantly better performance than baselines. We also analyze a few case studies to show the distinct features of ECPR-based methods.

CCS Concepts: • **Computing methodologies** → **Reasoning about belief and knowledge**;

Additional Key Words and Phrases: Knowledge base, reasoning, information retrieval, embedding learning

## ACM Reference format:

Pei-Chi Lo and Ee-Peng Lim. 2023. Contextual Path Retrieval: A Contextual Entity Relation Embedding-based Approach. *ACM Trans. Inf. Syst.* 41, 1, Article 1 (January 2023), 38 pages.

<https://doi.org/10.1145/3502720>

## 1 INTRODUCTION

### 1.1 Motivation

With knowledge graphs covering many different topical domains, mentions of entities in documents can be linked to knowledge graphs. Knowledge graph-assisted question-answering and exploratory search are becoming very important. These applications typically involve different types of retrieval tasks, querying, or extracting pieces of knowledge from the knowledge graphs. Knowledge graph completion task, for example, is a kind of knowledge retrieval task that predicts entities connecting to a given query-entity via a specific relation. Knowledge-based

This research is supported by the National Research Foundation, Singapore under its Strategic Capabilities Research Centres Funding Initiative.

Authors' address: P.-C. Lo and E.-P. Lim, Singapore Management University; emails: pclo.2017@phdcs.smu.edu.sg, eplim@smu.edu.sg.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2023 Association for Computing Machinery.

1046-8188/2023/01-ART1 \$15.00

<https://doi.org/10.1145/3502720>

question-answering is another task that aims to answer factual questions by searching relations in a knowledge graph connecting some question entity to the answer entities [27, 49, 52, 57]. Both tasks address the “what” questions (i.e., what are the possible items that could be recommended to the user?) but not the “how” and “why” questions (i.e., how does this item appeal to the use? Why should we recommend this item to the user?). To cope with the latter, we need a different type of knowledge retrieval task that returns relevant parts of the knowledge graph as answers or part of answers.

In this article, we thus propose a novel knowledge retrieval task called *contextual path retrieval*. Unlike knowledge-based completion and question-answering, contextual path retrieval requires a good understanding of the query context before the correct answers can be returned. The query in this case is a pair of input entities mentioned in some document, and the answer is a path connecting the entities extracted from the knowledge graph. To the best of our knowledge, this task has not yet been studied, and it represents an early attempt to mimic the human wisdom in establishing contextual connections between two entities. We shall elaborate on the task definition below.

## 1.2 Research Objective

**Contextual path retrieval (CPR)** is defined as the task of finding *path(s)* between a pair of *query entities* in a knowledge graph to explain the connection between them when they appear together in a given *context*. In this definition, the two query entities form the input query and the result path is invariant to the query entity order. We define the context to be a document covering some common topic or event. The knowledge graph contains the entities and relations from which every result path is to be retrieved.

**Example:** Consider this query: “Why is *Roger Moore* related to *Daniel Craig* in the context of a movie entertainment news article?” *Roger Moore* and *Daniel Craig* are the two input query entities, and the movie entertainment news is the context. Here, the background knowledge graph is assumed to be movie related entities and relations extracted from DBpedia. For this pair of query entities, the correct result is a path with two relations, *Roger Moore*  $\xrightarrow{\text{Portrayer}}$  *James Bond*, and *James Bond*  $\xrightarrow{\text{Portrayer}}$  *Daniel Craig*. In the example, there could be several other paths connecting the input entities (e.g., *Roger Moore*  $\xrightarrow{\text{Nationality}}$  *British*, and *Daniel Craig*  $\xrightarrow{\text{Nationality}}$  *British*) but they are not relevant to the context and hence not included in the CPR result. In some cases, a query may have no paths as results. It could be that the knowledge graph does not have a path between the two query entities. Second, there may be path(s) but none is relevant to the query context. While the former can be easily handled, the latter requires relevant paths to be accurately determined.

For CPR to be interesting, the given knowledge graph should be rich in its coverage of entities and relations. This ensures that relevant paths, if exist, can be retrieved. Examples of rich knowledge graphs covering general domains include DBpedia and WordNet. There are also knowledge graphs covering specialized domains, e.g., **Global Research Identifier Database (GRID)** for educational and research entities.<sup>1</sup> In domains where knowledge graphs may not exist or may be incomplete, researchers have looked into automated construction of knowledge graphs from domain-relevant text [14, 29] or pre-trained embedding models [56].

CPR is related to knowledge graph-based explainable recommendation, which aims to discover user preference through investigating the underlying knowledge graphs constructed for items and

<sup>1</sup><https://www.grid.ac/>.

users [2]. For instance, KGAT learns the importance of paths based on their abilities to predict the target items [45]. Another similar IR task is explainable question-answering, which extracts answer to a question by traversing the knowledge graph. CPR differs from the two by returning paths as results instead of items or entities. CPR is also different due to the consideration of query context. This, therefore, rules out the possibility of directly applying the solutions of the two tasks.

There are several technical challenges to be addressed in CPR. The first challenge is (a) the incorporation of query context in the query entity representation. Query context representation is non-trivial, because we want to capture sufficient and accurate semantics of the query context, which contains limited amount of textual content and possibly other entity mentions. The second challenge is (b) the representation of paths in knowledge graph in an inductive manner. The encoding of path should be inductive, as we not only have to encode paths in training phase, but we also need to handle new paths at query phase. The final challenge is (c) matching paths against the query entities even when they are in heterogeneous forms. Paths are formed by possibly multiple entities and relations from a knowledge graph while the query context is textual. The comparison between the two is non-trivial unless (i) they can be represented in the same space where some similarity measure in this space can be proposed to rank them in an unsupervised manner; or (ii) they are represented in different spaces and a separate model is trained to determine the relevance of path with respect to query context. While option (i) provides easier comparison between paths and contexts, it is difficult to jointly embed them in the same vector space. Option (ii), however, is more flexible, as we can train the path and context representations separately. In this article, we want to mimic the way an intelligent human user would handle the CPR task. We formulate a CPR framework based on learning embeddings for context representation and path representation, which in turn can be matched for returning the correct contextual path(s) using a supervised learning approach. For example, in the context of a movie awards ceremony news article, it is more semantically appropriate to relate an actor with a director through a movie production, than an award given out by the director to the actor at a ceremony. The CPR task has to consider both the context underlying the query entities and the path's semantics to determine the contextual relevance of the latter for result generation.

As the number of annotated paths is expected to be small, we want our models to be able to acquire the underlying semantics of paths by introducing path representations based on the knowledge graph representations of entities and relations. We also need to capture the semantics of the given context using a good representation for matching with path representation.

Finally, this research requires datasets with ground truth paths and their associated context documents. This dataset construction requires annotators with good domain knowledge and text comprehension abilities. We therefore have to design smaller annotation steps and support them with user-friendly annotation UI.

### 1.3 Contribution

Our contribution can be summarized as follows:

- We formally define the **Contextual Path Retrieval Problem (CPR)** and propose a novel ECPR framework to determine contextual paths between two entities in a knowledge graph. The framework is generic and consists of three components: context encoder, path encoder, and path ranker.
- In ECPR, we propose various context encoders that effectively leverage on all semantics in context and to incorporate background knowledge using pre-trained model.
- We propose PathVAE, which utilizes LSTM-VAE as our path encoder to generate a representation inductively for any given path in knowledge graph in a self-supervised manner.

- Our experiments on two real datasets show that ECPR-based model with contextualized entity/word embedding as context encoder, pathVAE as path encoder, and learning-to-rank path ranker outperforms other baselines on both MRR and hit@k, as well as two similarity-based metrics NGeo and PED. Moreover, we conduct case studies to reveal the salient characteristics of ECPR.
- We conduct analysis on two synthetic datasets to compare how selected models perform on datasets of large-scale data and how they perform on queries of different degrees of complexity.

## 2 RELATED WORK

### 2.1 Knowledge Graph Prediction Tasks Related to CPR

CPR is related to prediction tasks in knowledge graphs that recover missing relations and relation labels. Knowledge graph completion and relation/link prediction are two such tasks.

*2.1.1 Knowledge Graph Completion.* Knowledge graph completion, or simply knowledge completion, is a prediction task in knowledge graph to find the tail entities associated with a given head entity by a given relation label [22, 27, 36, 39, 49]. Knowledge completion has often been used to evaluate the effectiveness of a knowledge graph embedding model to infer new (head,tail) entity relations. While knowledge completion also involves a knowledge graph and input head entity, it is very different from CPR in three aspects: (a) it does not involve context external to the knowledge graph, (b) it returns inferred relations only instead of paths, and (c) it assumes that the specific relation label for connecting head with tail entities is already known. Given the above major differences, knowledge completion methods cannot be used to solve CPR.

*2.1.2 Relation/Link Prediction.* Relation prediction, or link prediction, is similar to knowledge graph completion in the sense that it tries to augment the missing information in the knowledge graph [7]. However, instead of focusing in finding tail entities given the head entity and a relation label, relation prediction task aims to predict whether a new relation exist between two given entities. Some of the important works in this line of research include translation-based methods that focus on representing relation as translation operation [7, 40, 54], and others using CNN on the knowledge graph structure to determine whether a relation exists [15, 32]. Although relation/link prediction is seemingly similar to our CPR problem, there are still two major disparities: (1) same as KG completion, it does not involve context external to the knowledge graph; and (2) it infers new relations by observing other existing relations in the KG, while CPR tries to find a multi-hop knowledge path that best describes the semantic relationship between two entities.

### 2.2 Retrieval Tasks Related to CPR

One related IR task to CPR is entity relation explanation. As opposed to CPR, which retrieves contextual paths given a textual input, entity relation explanation finds passages that explain a relation between two entities in a knowledge graph [1, 4, 34, 42, 43]. Entity support passage retrieval, however, aims to find a support passage that explains why an entity is connected to a given query [5, 12, 23]. In the following, we discuss in detail about two tasks that also try to retrieve knowledge graph paths to explain the relation between two entities, namely, recommendation with knowledge graph and question-answering with knowledge graph.

*2.2.1 Recommendation with Knowledge Graph.* Given a knowledge graph and past user-item interaction, recommendation system predicts items users may like and may also provide explanation in the form of reasoning structure extracted from the knowledge graph. Path-based recommendation methods provide intuitive reasoning that is easy to comprehend by human using

connectivity patterns or meta-paths [13, 18, 21, 58, 59]. Some works also integrate both user-item interaction and item knowledge graph to generate explainable recommended items. For instance, KPRN appends the user-item interaction to an item knowledge graph [46] and utilizes the paths between a user and an item in the knowledge graph as features to recommend items. It also returns an importance weight for each path (from the user-item graph) to tell the user which path is likely to explain the choice of a recommended item. KTUP utilizes the information in knowledge graph as an auxiliary data to enhance the user-item interaction modeling [10]. Finally, there are also some works focusing on explicit reasoning in recommendations. With the help of reinforcement learning, PGPR performs a causal inference procedure to provide explanations [50]. Each recommendation is based on a set of knowledge graphs. In spite of all the advancement in recommendation with knowledge graph, recommendation models cannot be directly used to solve CPR, as the prediction targets of the two problems differ. CPR seeks to identify the semantic path in knowledge graph most relevant to the query context covering the two input entities, while the recommendation works use the knowledge graph paths as features to predict a user's preference toward an item. Furthermore, most of the recommendation works do not involve context.

**2.2.2 Question-answering with Knowledge Graphs.** QA is an information retrieval task that determines for a given question the important entities ( $e_h$ ) and traverses the knowledge graph from these entities to find possible answers ( $e_t$ ). One popular QA task is multi-relation QA over knowledge graph, which reasons over multiple facts in the knowledge graph to determine the answers [52, 57, 61]. Some works base on reinforcement learning and use policy-based agents to generate the most probable paths on a knowledge graph [26, 51]. Text-based QA reasons over multiple documents instead of knowledge graph to generate answers [35, 38, 48, 55]. For instance, given a set of input paragraphs and a question, DFGN constructs an entity graph from the paragraphs and extracts the answer from the graph. The reasoning process first identifies the important entity from the question, then iteratively excludes irrelevant subgraphs from the entity graph. The answer is later determined according the entity graphs that are relevant. The above QA tasks are different from CPR due to their focus on answering questions rather than finding the correct path for explanation. Path extraction/construction in QA with knowledge graph essentially aims to weigh the different possible answer options, which is similar to that of recommendation with knowledge graphs.

### 2.3 Text Encoding and Entity Embeddings

An important part in the definition of CPR is context. Context is an input to the framework to identify which path in knowledge graph best represents the relationship between two query entities. In our proposed framework, we look for ways to encode the unstructured context to be a vector representation. Specifically, we prefer unsupervised learning methods, as the encoding of context can be separated from other components in the framework. Classic unsupervised ways to represent a document include one-hot encoding or TF-IDF vectorization. However, such methods often suffer from sparsity. Dense word embeddings (word embeddings) research is therefore proposed to tackle the sparsity problem. These works learn the word/document representation by looking into statistics of a word in its context such as GloVe, Word2vec, and doc2vec [25, 30, 33].

State-of-the-art unsupervised text embedding works incorporate contextual information by introducing a modified transformer structure [41]. The most representative work in this field is BERT [16]. BERT utilizes multiple layers of transformer encoders that are trained to predict masked words with large text corpora so a different representation of a word is contextualized, i.e., the same word has different representations when occurring in different textual context.

While the aforementioned methods have been already proven useful in tasks such as document classification, they can only cover words. As the context may consists of both mentions of entities and words, we need to explore entity embeddings. Given an observed relation triplet  $(e_h, r, e_t)$  in the KG, Translation-based entity embeddings such as TransE, TransR, and TransH learn entity and relation embeddings  $z_*$  in an alignment that satisfies the translation operation  $z_{e_h} + z_r \rightarrow z_{e_t}$  [7, 27, 47]. However, multiplication-based methods such as DISTMult and ComplEx represent a relation as a matrix  $W_r$  that satisfy  $z_{e_h}^T W_r z_{e_t} = 1$  [40, 54].

Finally, we review hybrid embedding methods that jointly learn the word and entity representations in a common vector space. Wikipedia2vec jointly learns the representations of entities and words in a common vector space using skip-gram model [53]. There are also works that try to learn contextualized hybrid entity embedding based on BERT. KG-BERT augments BERT with knowledge graph by tuning a pre-trained BERT with triplet identification or relation prediction loss function [56]. Other works such as K-BERT and KEPLER further learn the entity to word interaction in a knowledge graph and text documents mentioning its entities [28, 44]. K-BERT injects relation triplets extracted from the knowledge graph into for a BERT-transformer to encode, for a BERT-transformer to encode words and entities simultaneously. Last but not least, KEPLER jointly optimizes knowledge graph loss (e.g., TransE) and masked language model loss to implicitly incorporate knowledge into contextualized word embeddings. The entity is represented by encoding the description from its Wikipedia page in KEPLER.

### 3 CONTEXTUAL PATH RETRIEVAL

In this section, we formally define the CPR task. We will also describe our proposed ECPR framework in both training and query phases. To ease reading, we use italic font for entities (e.g., *Apple Inc.*) and boldface font for relations (e.g., **director**). Table 1 shows the list of notations used in Sections 3–4.

#### 3.1 Definitions

We formally define a **knowledge graph**  $G$  to be a tuple  $(E, L, R, F)$  where  $E$  denotes a set of entities,  $L \subseteq E \times E$  denotes a set of edges,  $R$  denotes a set of relation labels, and for each  $l = (e_h, e_t) \in L$ ,  $F(l) \rightarrow R$ , or simply  $e_h \xrightarrow{r} e_t$  where  $r = F(l)$ . For the purpose of establishing connections between entities, we assume that a knowledge graph has both a set of relations (e.g., *participateIn*) and their corresponding reverse relations (e.g., *participateIn<sup>-1</sup>*). For example, in DBpedia, *John Cena* and *Heath Slater* are two person entities connected to each other via a wrestling match denoted

by  $x$ , i.e.,  $e_{JohnCena} \xrightarrow{\textit{participateIn}} e_x$ , and  $e_x \xrightarrow{\textit{participateIn}^{-1}} e_{HeathSlater}$ .

Among the paths between two entities in the knowledge graph, only a few carry useful contextual semantics that can best explain the connection between the entities. We call these the contextual paths.

*Definition 1. Contextual Path  $p$ :* A path  $p = \langle e_h, r_1, e_2, \dots, r_{L-1}, e_t \rangle$  is contextual to a pair of entities  $e_h$  and  $e_t$  and a piece of textual content  $d$ , when it is composed of entities  $e_i$ 's and relations  $r_k$ 's of a knowledge graph that describe the semantic connection between  $e_h$  and  $e_t$  for the given context  $d$ .

In the above definition, we assume that the entities  $e_h$  and  $e_t$  as well as other entities/relations in the path are not only mentioned in  $d$ , but also found in the knowledge graph  $G$ . This assumption is reasonable given the existence of several large text corpora of documents with entity mentions linked to knowledge graphs, e.g., Wikipedia, Freebase [6], and AIDA CoNLL-YAGO Dataset [20],

Table 1. Table of Notations

Symbol	Definition
General Notations	
$G$	Knowledge graph
$E(e)$	Set of entities
$L(l)$	Set of edges
$R(r)$	Set of relation labels
$F$	Set of relation triplets
$D(d)$	Set of context documents
$Q(q)$	Set of query triplets $\langle e_h, e_t, d \rangle$
$P(p)$	Set of knowledge graph paths
$e_h(e_t)$	Head (tail) entity from a edge $l$
$P^+(P^-)$	Set of positive paths (negative paths)
$z^{cx}(z^{pt})$	Context (path) representation
Context-fused Entity Embeddings	
$z_e$	Entity embedding of $e$
$z'_e$	Retrofitted entity embedding of $e$
$E_{d,e}^{CW}$	Set of entities that appear in the context window of $e$ in $d$
$E_{d,e}^{SD}$	Set of entities that appear in $d$
$E_{d,e}^{NC}$	Set of entities that have relation with $e$ but not present in $d$
$k^*$	Hyper-parameters in retrofit cost function
Path Encoder	
$w$	An element in a path
$q_\phi(p_\theta)$	VAE encoder (decoder)
Path Ranker	
$x_{m,k}$	Input to the path ranker $[z^{cx,m}, z_k^{pt,m}]$
$y$	Label for $x$ , 1: relevant, 0: irrelevant
$P_m$	set of candidate paths for query $q_m$
$Y_m$	Ground truth ranking of candidate paths
$s_{m,k}$	Cosine similarity score between representation of path $p_k$ and ground truth path for $q_m$
$S_m$	Cosine similarity score vector of all candidate paths for $q_m$

and the increasingly more accurate NER and entity link methods capable of linking entity mentioned in documents to knowledge graphs. We use  $|p|$  to denote the length of a path  $p$ .

Next, we formulate the **Contextual Retrieval Problem (CPR)** as follows:

**Definition 2. Contextual Path Retrieval Problem (CPR):** Given a knowledge graph  $G$ , a query  $\langle e_h, e_t, d \rangle$  consists of a context document  $d \in D$  and two entities  $e_h$  and  $e_t$  mentioned in  $d$ , we want to retrieve from  $G$  the contextual paths between  $e_h$  and  $e_t$ .

For example, in Figure 1, *Alfonso Cuarón* and *Children of Men* are two entities in a context document containing several sentences. The CPR problem is to retrieve the contextual path(s) from knowledge graph that explains the connection between *Alfonso Cuarón* and *Children of Men*. In the figure, the path  $\langle \textit{Alfonso Cuarón}, \textbf{director}, \textit{Children of Men} \rangle$  is a candidate contextual path.

We divide the CPR task into two subtasks: (a) construction of candidate paths connecting  $e_h$  and  $e_t$  and (b) determination of contextual paths among the candidates. For subtask (a), we can focus



### "Children of Men" wins Scriptor Award for writing

- 1 **"Children of Men"**, a movie based on a P.D. James book, has won the USC Scriptor Award 2006 for its writing.
- 2 Both the original author, James, and the screenwriting team will be honored by the University of South California for their work.
- 3 The winning screenwriters are **Alfonso Cuarón**, Timothy J. Sexton, David Arata, Mark Fergus, and Hawk Ostby. "
- 4 The Children of Men" was James' 12th book, written in 1992.
- 5 USC School of Cinematic Arts Writing Division Chair Howard A. Rodman commented For nineteen years, the USC Libraries Scriptor Award has honored "writers for the best achievement in adaptation among English-language films released during the previous year and based on a book, novella or short story."
- 6 While there are many awards for either screenwriting in general, or adapted screenwriting, the Scriptor is the only award to recognize both the screenwriters and the original authors.

Does the following relation capture the relation between **Children of Men** and **Alfonso Cuarón** in the article?

Alfonso Cuarón  $\xrightarrow{\text{director}}$  Children of Men


Yes
 No

WIKIPEDIA

### Children of Men

This article is about the film adaptation. For the original novel, see *The Children of Men*.

**Children of Men** is a 2006 science fiction action-thriller film<sup>[4][9][8][7]</sup> co-written and directed by Alfonso Cuarón. The screenplay, based on P. D. James' 1992 novel *The Children of Men*, was credited to five writers, with Clive Owen making uncredited contributions. The film takes place in 2027, when two decades of human infertility have left society on the brink of collapse. *Asylum seekers* seek sanctuary in the United Kingdom, where they are subjected to detention and *refoulement* by the government. Owen plays civil servant Theo Faron, who must help refugee Kee (*Clare-Hope Ashitey*) escape the chaos. *Children of Men* also stars Julianne Moore, Michael Caine, Chiwetel Ejiofor, Pam Ferris, and Charlie Hunnam.



Theatrical release poster

Directed by Alfonso Cuarón

Fig. 1. Annotation interface.

on a set of simple yet effective heuristics to constrain the set of candidate paths. In this article, we focus on subtask (b), which requires a new approach to context and path representations as well as incorporating them into path ranking. We shall elaborate on our solution framework below.

### 3.2 Proposed ECPR Framework

Our proposed the Embedding-based Contextual Path Retrieval (**ECPR**) framework, as shown in Figure 2, involves a knowledge graph  $G = (E, L, R, F)$  covering entities and relations of some domain. We divide the framework into two phases: training phase and query phase. During the training phase, ECPR assumes that the training data includes a set of documents  $D$  of the same domain, a set of query entity pairs  $Q$ , and their corresponding contextual paths (positive paths)  $P^+$  and other non-contextual paths (negative paths)  $P^-$ .  $Q$  is a set of head-tail entity pairs and the corresponding context documents, i.e.,  $Q = \{q_m | 1 \leq m \leq |Q|\}$ . Each query  $q_m$  is a triplet  $\langle e_h, e_t, d \rangle$ ,  $e_h, e_t \in E$ , and  $d \in D$ . Both head and tail entities  $e_h$  and  $e_t$  are mentioned in the document  $d$ . Every query  $q_m$  is associated with a set of candidate paths  $P_m$  (size of the set is denoted by  $|P_m|$ ). The contextual path and the set of non-contextual paths between  $e_h$  and  $e_t$  in  $d$  are denoted by  $P_m^+$  and  $P_m^-$ , respectively.

In the training phase, three major components are trained, namely, context encoder, path encoder, and path ranker. First, the **context encoder** is trained to return *context representations*  $z_h^{cx,m}$  and  $z_t^{cx,m}$  of entities  $e_h$  and  $e_t$ , respectively, for each query  $q_m = \langle e_h, e_t, d \rangle \in Q$ . The context encoder may combine the embeddings of other words and/or entities in the relevant section of the context document as it generates the two context representations.

The second component **path encoder** returns a *path representation* for each path found in  $G$ . In particular, it generates the representations  $\{z_i^{pt,m}\}$  for the candidate paths  $\{p_i^m\}$  of the query  $q_m$ . While paths are sequences of entities and relations, path encoder turns them into vector representation forms that can be matched against representation of the query context. During the training phase, both context encoder and path encoder are learned from the queries with input entity pairs and their labelled contextual and non-contextual paths.

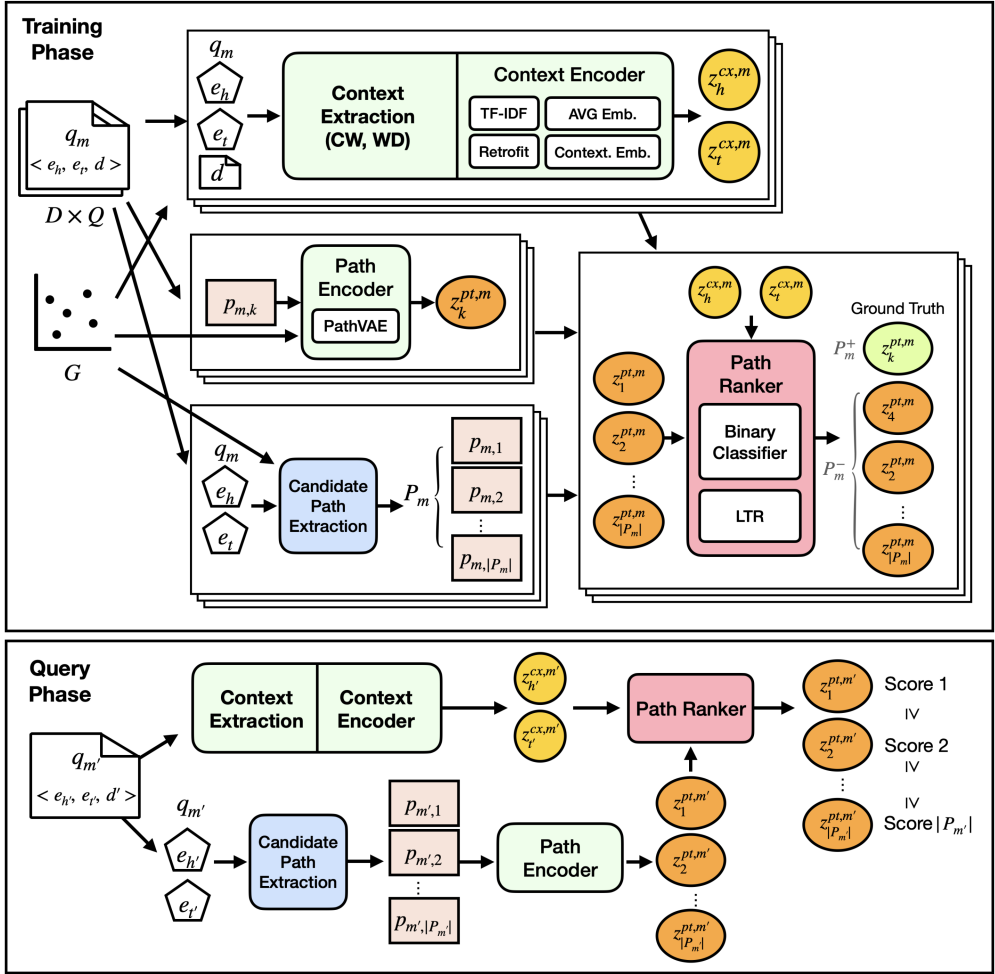


Fig. 2. ECPR framework.

The final component is **path ranker**, which ranks a set of candidate paths against the query context during the query time using their representations (i.e., context representation and path representation), respectively. As each query  $q_m$  is associated with a positive candidate path  $P_m^+$  and negative candidate paths  $P_m^-$  in Figure 2, we assume  $p_m^k$  is the ground truth contextual path. Path ranker is thus trained to differentiate the positive candidate path from the negative ones through a supervised learning model. In this article, we train path ranker using both a binary classification and a learning to rank method.

In the query phase, when given a query  $q_{m'} = \langle e_{h'}, e_{t'}, d' \rangle$  consisting of entity pair  $(e_{h'}, e_{t'})$  from the context document  $d'$ , we use the trained context encoder and path encoder (context encoding and path encoding) to obtain the context representation  $z_{h'}^{cx,m'}/z_{t'}^{cx,m'}$  as well as candidate path representations  $z_k^{pt,m'}$ . The path ranker will then compare each entity-path pair and rank the candidate paths by the likelihood of being the contextual path. In this work, we propose different options for context encoders and path rankers. We elaborate on the details of the three components in the framework in Sections 4, 5, and 6.

## 4 CONTEXT ENCODER

The context encoder in our ECPR framework learns the latent representation of entities  $e_h$  and  $e_t$  with respect to the context. The context representation is the concatenation of the representations of  $e_h$  and  $e_t$ , denoted by  $z_h^{cx,m}$  and  $z_t^{cx,m}$ , respectively. The context representation is then provided to the path ranker for identifying the correct contextual path. When encoding the context, we need to consider the context range as the context involving the query entity pairs can be as loose as the whole context document  $d$  (denoted by **whole-context-document (WD)**) or as tight as within some context window covering the query entities in  $d$  (denoted by **CW**). In this section, we describe four different context encoders in detail. Among them, we propose the context-fused embedding and contextualized embedding that incorporate context constraints and background knowledge, respectively.

### 4.1 TF-IDF

TF-IDF determines the relevance of a word or an entity to a context in document  $d$  by combining term frequency and inverse document frequency together. The TF-IDF context representation of  $d$  is thus a TF-IDF vector where each element is the TF-IDF score of a word in a vocabulary of words and entities. We learn a TF-IDF vectorizer using our dataset. The vocabulary consists of all words in entity surfaces plus the top 300 words ranked by TF-IDF found in the context document set  $D$ . The size of vocabulary is chosen based on grid search on settings that yield the best MRR. Further, stop-words are removed from all context documents. We subsequently use  $z_h^{cx,m}$  ( $z_t^{cx,m}$ ) to represent the TF-IDF vector of  $e_h$  ( $e_t$ )'s context.

### 4.2 Averaged Embeddings

TF-IDF as a simple method suffers from sparsity of words in the context. Thus, we propose to use dense embedding methods to address the sparsity issue. Specifically, we encode the context using the averaged embeddings of tokens in the context. In this work, we choose three different average embeddings methods that cover only entities, cover only words, or cover both entities and words.

**4.2.1 Entity-only Embeddings.** There are many knowledge graph embeddings models that can be used for entity-only embeddings. In this article, we use TransE, which has been widely used in past research. TransE encodes entities and relations in a common embedding space such that the translation operation of a pair of entities corresponds to the relation's embedding [7]. The context of  $e_h$  ( $e_t$ ) in  $d$ , denoted by  $z_h^{cx,m}$  (or  $z_t^{cx,m}$ ), is defined by the averaged TransE embeddings of all entities appeared in  $e_h$  ( $e_t$ )'s context window. We train a TransE model with our knowledge graph  $G$  (see Section 7.2.1 for details about how our knowledge graph is constructed in our experiments).

**4.2.2 Word-only Embeddings.** In this averaged embeddings method, the context of  $e_h$  ( $e_t$ ) is the averaged Word2vec embeddings [30] of all words appeared in  $e_h$  ( $e_t$ )'s context window as denoted by  $z_h^{cx,m}$  ( $z_t^{cx,m}$ ).

**4.2.3 Word-entity Embeddings.** The context of  $e_h$  ( $e_t$ ), in this method, is the averaged Wikipedia2vec embeddings of all words and entities appeared in  $e_h$  ( $e_t$ )'s context window as denoted by  $z_h^{cx,m}$  ( $z_t^{cx,m}$ ). Wikipedia2vec jointly embeds words and entities in a common vector space [53]. It models word-to-word, word-to-entity, and entity-to-entity interaction using skip-gram model.

### 4.3 Context-fused Entity Embeddings

By averaging the embeddings of words and/or entities in the context, the averaged embeddings methods treat everything in the context equally instead of differentiating the entities or words

by their relevance or importance to the context. For instance, for the query (*Daniel Craig, Casino Royale*) and the context “British actor Daniel Craig has been confirmed... The next Bond film Casino Royale is due to film in Italy, the Bahamas, the Czech Republic and Pinewood Studios,” all the location names are irrelevant to the retrieval of contextual path *Daniel Craig*  $\xleftarrow{\text{starring}}$  *Casino Royale*. These irrelevant information should be treated as noise that should be excluded from the context representation.

Therefore, we propose the context-fused entity embedding method using a retrofitting approach and its variant(s) to incorporate background knowledge as constraints into context representations [17]. Retrofitting is originally designed to refine pre-trained word representations with synonym information from semantic lexicons and assign synonymous words to have similar representations. Counter-fitting, another form of retrofitting, repels representations of antonym words from each other [31]. At the beginning of retrofitting (counter-fitting), synonymous word pairs (e.g., **good** and **nice**) and antonymous ones (e.g., **good** and **bad**) are extracted from a thesaurus and used as constraints. The model then updates a pre-trained word embedding model to satisfy all these constraints while preserving the structure of the original embeddings. The retrofitted (counter-fitted) word embeddings not only carry all its own original attributes and semantic alignment, but also the semantics in synonymous and antonym constraints.

In context-fused embeddings, we use retrofitting and counterfitting to augment the original word and entity embeddings with context constraints. In other words, given the head(tail) entity embedding  $z_h^{KG}(z_t^{KG})$  from a knowledge graph embedding such as TransE, we will retrofit it with constraints made up of an entity set  $E^*$  extracted from the context. The resultant entity embedding  $z_h^{cx,m}(z_t^{cx,m})$  will then be used as context representation. For the rest of this section, we overload the notation for simplicity and represent the entity embedding of an entity  $e$  as  $z_e$ . The retrofitted representation of  $z_e$  is denoted by  $z'_e$ .

**4.3.1 Constraints.** Constraints tell us how entity embeddings should be updated. In this section, we introduce all constraints used in the retrofitting cost function:

$$C(z_e, z'_e) = CNA + SDA + NCR + VSP + RP.$$

To obtain a context-fused entity embedding method that leverages knowledge from its context, we propose two in-context constraints (*CNA* and *SDA*), one out-context constraint (*NCR*), and two regularization terms (*VSP* and *RP*). This is the only context encoder that does not differentiate the setting of context range of WD and CW, as the constraints are retrieved from both in and out-context.

First, the **in-context constraints** capture the co-occurrence(s) of an entity  $e'$  with a query entity  $e$  within some context range. With retrofitting, we adjust the embeddings of  $e$  and  $e'$ , i.e.,  $z_e$  and  $z_{e'}$ , to incorporate their context similarity. Through the retrofitting process, the entity will gradually move its representation towards the in-context constraints in the vector space to obtain more semantic similarity with them. We define two kinds of in-context constraints based on how close the entities are to  $e$  in the context document.

- **Close Neighbor Attract (CNA):**  $z_e$  is retrofitted with entity  $e' \in E_{d,e}^{CW}$ . For a query entity  $e$  and context document  $d$ , we defined *context window entity set*  $E_{d,e}^{CW}$  to be the set of entities that appear in the context window of  $e$  in  $d$ . For example, in the following context window of the query entity **Daniel Craig**: “British actor Daniel Craig has been confirmed as the man to follow Pierce Brosnan as the sixth James Bond.” CNA includes  $E_{\text{Daniel Craig}}^{CW} = \{\text{Pierce Brosnan, James Bond}\}$ . Let  $d(\cdot)$  be any kind of distance measurement and  $\tau(x) \triangleq \max(0, x)$ , CNA thus derives the new embeddings of  $e$  and  $e'$ , i.e.,  $z'_e$  and  $z'_{e'}$ , respectively, by minimizing

the cost function:

$$CNA(z'_e) = \sum_{e' \in E_e^{CW}} \tau(d(z'_e, z'_{e'}) - \gamma),$$

where  $\gamma$  is the ideal maximum distance between  $e$  and  $e' \in E_e^{CW}$ . Here, we empirically set  $\gamma = 0$ .

- **Same Document Attract (SDA):**  $z_e$  is retrofitted with entity  $e' \in E_{d,e}^{SD}$ . We defined *same document entity set*  $E_{d,e}^{SD}$  to be the set of entities that appear in the context document  $d$  of the query entity  $e$  and they do not exist in  $E_{d,e}^{CW}$ . Using the same example in CNA, towards  $E_{\text{Daniel Craig}}^{SD} = E^d - E_{\text{Daniel Craig}}^{CW}$ . Here,  $E^d$  is the set of entities included in  $d$ . Similar to CNA, SDA is designed to adjust the embeddings of  $e$  and  $e' \in E_{d,e}^{SD}$  by minimizing the cost function

$$SDA(z'_e) = \sum_{e' \in E_{d,e}^{SD}} \tau(d(z'_e, z'_{e'}) - \gamma), \gamma = 0.$$

The **out-context constraints**, however, are entities that we do not want the target entity  $e$  to be close to in the vector space. In the case of context encoding in CPR, such out-context constraints are negative context, that is, entities that do not appear in the context of  $e$ .  $z_e$  will learn to repel themselves from the these entities during the retrofitting(counter-fitting) process.

- **Negative Context Repel (NCR):**  $z_e$  is counter-fitted with entity  $e^\dagger \in E_e^{NC}$ , where  $E_e^{NC}$  is the set of entities that have relation with  $e$  but do not appear in the context document  $d$  containing  $e$ . From the previous example, entity **Daniel Craig**'s NCR includes  $E_{\text{Daniel Craig}}^{NC} = \{\text{Knives Out, Logan Lucky, ...}\}$ . Intending to push  $e$  away from  $e^\dagger$ s, AR seeks to minimize the following cost function:

$$NCR(z'_e) = \sum_{e^\dagger \in E_e^{NC}} \tau(\delta - d(z'_e, z'_{e^\dagger})),$$

where  $\delta$  is the ideal minimum distance between  $e$  and  $e^\dagger \in E_e^{NC}$ . Here, we empirically set  $\delta = 1$ .

Finally, the **regularization terms** make sure the adjusted embeddings after retrofitting (counterfitting) should not differ too much from their original embeddings. For instance, in a TransE entity embedding model, the closer two entities are in the vector space, the more semantic similarity they share with each other. However, when trying to minimize the cost function from CNA, SDA, and NCR, such property might be sacrificed. Thus, we introduce the following two preservation terms:

- **Vector Space Preservation (VSP):** Given the  $n$  neighboring vectors  $N(e)$  within a certain radius  $\rho$  around  $e$  in the original embedding model, the difference between distance from  $z_e$  to  $z_{\bar{e}}$  ( $\bar{e} \in N(e)$ ) and that from  $z'_e$  to  $z_{\bar{e}}$  should be minimized. This is to maintain the semantic alignment of the original embedding model. VSP minimizes the cost function

$$VSP(z_e, z'_e) = \sum_{\bar{e} \in N(e)} \tau(d(z_e, z_{\bar{e}}) - d(z'_e, z_{\bar{e}})).$$

- **Relation Preservation (RP):** Randomly sample  $m$  edges  $(e, \hat{r}, \hat{e}) \in KG$  to be  $L(e)$ , the distance between  $e + \hat{r}$  and  $\hat{e}$  should be minimized. This is to maintain the translation property of the entity embedding model  $e + \hat{r} \rightarrow \hat{e}$ . Note that we include this regularization term because we use TransE as our base entity embedding model. If entity embeddings that are not

based on translation property are used (e.g., DistMult [54], ComplEx [40], or RotatE [37]), then this relation preservation should be modified according to the assumption of the entity embedding method. RP follows the similar idea with VSP, and thus could be written as:

$$RP(z_e, z'_e) = \sum_{(e, \hat{r}, \hat{e}) \in L(e)} \tau((z_e + z_{\hat{r}} - z_{\hat{e}}) - (z'_e + z_{\hat{r}} - z'_{\hat{e}})).$$

**4.3.2 Retrofitting with Constraints.** With the constraints/regularization discussed previously, we retrofit all entities  $e \in E^d$  in a document  $d$ . Note that the constraints for two different mentions of the same entity in  $d$  might be different, thus they will be retrofitted differently. Likewise, the two mentions of the same entity in different context document will also be retrofitted differently. In this work, we propose three different optimization methods that use different constraint/regularization combination:

- **Retrofit with In-context Constraints Only:** We only retrofit  $e$  with CNA and SDA, plus the two regularization terms, VSP and RP:

$$C_d = k_1^{\text{rf}} \text{CNA} + k_2^{\text{rf}} \text{SDA} + k_3^{\text{rf}} \text{VSP} + k_4^{\text{rf}} \text{RP}, \sum_i^4 k_i^{\text{rf}} = 1.$$

$k_*^{\text{rf}}$  is a hyper-parameter that controls the contribution from CNA, SDA, VSP, and RP. Intuitively, we hope CNA to weigh more than SDA, as CNA entities are closer to the query entity in the context document, thus, we add a constraint  $k_1^{\text{rf}} > k_2^{\text{rf}}$ .

- **Retrofit with Both In-context and Out-context Constraints:** We retrofit  $e$  with CNA, SDA, NCR, plus regularization:

$$C_d = k_1^{\text{cf}} \text{NCR} + k_2^{\text{cf}} \text{CNA} + k_3^{\text{cf}} \text{SDA} + k_4^{\text{cf}} \text{VSP} + k_5^{\text{cf}} \text{RP}, \sum_i^5 k_i^{\text{cf}} = 1.$$

Similarly,  $k_*^{\text{cf}}$  is a hyper-parameter that controls the contribution from each term. We add a constraint  $k_2^{\text{cf}} > k_3^{\text{cf}}$  to make sure CNA's weight is higher than that of SDA.

We optimize with SGD and Adam Optimizer until converge. The best  $k$ s are searched using grid search by finding the best MRR.

#### 4.4 Contextualized Embedding Representation

This method is similar to Averaged Embedding method except for the use of contextualized representations for context encoding. Contextualized embeddings are proved to perform better than traditional embedding models (e.g., Word2vec) in tasks such as document classification [16].

**4.4.1 Contextualized Word Embeddings.** BERT is the state-of-the-art representation learning method, which utilizes transformer to learn a bi-directional language model [16]. Each token is embedded with respect to the surrounding tokens in the context. Thus, a word will have different representations when it appears in different contexts. The context of  $e_h(e_t)$  is encoded using BERT to be  $z_h^{cx, m}(z_t^{cx, m})$ . We use the BERT-small model<sup>2</sup> with three layers, and the dimension size of 512.

**4.4.2 KG Augmented Contextualized Word/Entity Embeddings.** Beyond contextualized word embeddings, we propose another context encoder option using both contextualized entity and word embeddings. This allows the query context to be more completely represented before matching it

<sup>2</sup><https://github.com/google-research/bert>.

against candidate paths. To incorporate the entity information in the encoding process, we propose to use contextualized word/entity embeddings as one of our context encoders. These embeddings jointly embed entities and words in a common vector space with respect to the context they are within. The context of  $e_h(e_t)$  is encoded using contextualized word/entity embeddings to be  $z_h^{cx,m}(z_t^{cx,m})$ . In this article, we use KG-BERT [56], K-BERT [28], and KEPLER [44]. While all three of them are contextualized word/entity embedding methods, KG-BERT only augments BERT with KG structural information while K-BERT and KEPLER further learn the entity-word interaction. K-BERT learns such interaction by injecting additional knowledge graph relations to the context. However, KEPLER jointly learns knowledge graph structure and language model. We train our own KG-BERT, K-BERT, and KEPLER model using our knowledge graph. All the three embeddings are based on the BERT-small model with three layers, and the dimension size of 512.

## 5 PATH ENCODER

For path encoding, we propose a PathVAE to encode paths into their latent representations that preserved as much semantics as possible for reconstructing the path. While many sequential embedding models exist, we choose to utilize a variational autoencoder structure for (i) VAE is unsupervised, so the learning of PathVAE could be separated from other components; (ii) the PathVAE is inductive, so new paths that are unseen from the training data can still be encoded. Unlike entities, path is a sequence of entities and relations. It is essential to preserve the ordering of the entities/relations as we generate the path representation. We first break up a path  $p_m = \langle e_h, r_1, e_2, r_2, \dots, r_{|p_m|-1}, e_t \rangle$  into a sequence of elements and feed each element to a **Long Short-Term Memory (LSTM)**, one at a time [19]. One could also choose to use other sequential encoding methods such as Bi-LSTM or transformer to replace the LSTM layer.

As each element is a single entity or relation, PathVAE encoder  $q_\phi$  takes each entity or relation embedding generated by a base embedding model and obtains the overall path embedding after processing the entire sequence of path elements using LSTM. In this work, we utilize TransE as the base embedding, which has been trained to generate embeddings of all entities and relations of the knowledge graph. With the LSTM returned path embeddings, PathVAE encoder generates a path representation  $Z_m^{pt}$  consisting of  $\mu_m^{pt}$  (mean) and  $\sigma_m^{pt}$  (covariance matrix).

The PathVAE decoder layer takes  $Z_m^{pt}$  and reconstructs a path  $\hat{p}_m = w_1, \dots, w_{|\hat{p}_m|}$  using LSTM (where  $w_i$  denotes the entity or relation of the  $i$ th element of the path) by computing its probability as follows [3]:

$$p_\theta(\hat{p}_m | Z_m^{pt}) = \prod_{i=1}^{|\hat{p}_m|} p_\theta(w_i | w_1 : w_{i-1}, Z_m^{pt}). \quad (1)$$

The loss function for PathVAE is then,

$$\mathcal{L}_{pt} \geq \mathbb{E}_{q_\phi} \sum_{p_m \in P} \log_\theta p(p_m | Z_m^{pt}) - D_{KL}(q_\phi(Z_m^{pt} | p_m) \| p(Z_m^{pt})), \quad (2)$$

where  $X_{p_m} = \{w_1, \dots, w_{|p_m|}\}$  ( $p_m = \langle w_1, \dots, w_{|p_m|} \rangle$ ), and  $Z_m^{pt}$  is the latent representation of  $p_m$  deriving from

$$Z_m^{pt} = \mu_m^{pt} + \sigma_m^{pt2} \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I). \quad (3)$$

$p_m$  and  $Z_m^{pt}$  are the raw form and learned latent representation, respectively, of path  $p_m$ .

With PathVAE, the path encoder can generate path representation for any paths in the knowledge graph even when they have not been included in model training. As long as the entities and relations in the path sequence can be found in the knowledge graph, PathVAE will always return

its path representation. We will later evaluate the efficacy of PathVAE in our experiments (see Section 9).

## 6 PATH RANKER

The third component of ECPR is a path ranker that matches the context representation with candidate paths to determine the contextual path. Binary classifier or learning to rank method can be used as path ranker as described below.

### 6.1 Binary Classifier

Our first path ranker is effectively a binary classifier that outputs the probability of a path being a contextual path. We assume that only one path will be the ground truth for each entity pair in a document. The path ranker is trained on a set of data triples  $Q = \{(q_m, p_k, y_{m,k})\}$  where  $q_m = (e_h, e_t, d)$ ,  $p_k$  and  $y_{m,k}$  ( $1$  : relevant,  $0$  : irrelevant) denote the query, candidate path, and class label, respectively. With query context and candidate path encoded as  $z_h^{cx,m}$  and  $z_t^{cx,m}$ , respectively, a data triple corresponds to the concatenation, i.e.,  $x_{m,k} = [z^{cx,m}, z_k^{pt,m}]$  where  $z^{cx,m} = [z_h^{cx,m}, z_t^{cx,m}]$ . For each data triple  $(q_m, p_k, y_{m,k} = 1) \in Q$ , we randomly sample at most five negative paths from its candidate paths. We learn a classifier  $f^{BI}$  such that  $\hat{y} = f^{BI}(x)$ , and simply use binary cross entropy as the loss function

$$\mathcal{L}_{BI}(y_{m,k}, \hat{y}_{m,k}) = \sum_{x_{m,k} \in H} \text{CrossEntropy}(y_{m,k}, \hat{y}_{m,k}), \quad (4)$$

where  $y_{m,k}$  are the ground truth labels of the candidate paths  $P_m^+ \cup P_m^-$  for the  $(e_h, e_t)$  pair, and  $\hat{y}_{m,k}$  are the prediction labels of the same candidate paths returned by the path ranker. We build the binary classifier as an two-layer inference neural networks in our experiments with 128-dimensional hidden layers.

### 6.2 Learning to Rank

While binary classifier provides a good method to rank the candidate paths, it only learns to identify the most plausible contextual path. It ignores the fact that learning to rank is a prediction task on list of options. Thus, our second path ranker, is a listwise ranker that aims to recover the ground truth ranking. The ranker is trained on a set of data triples  $Q = \{(q_m, p_k, s_{m,k})\}$ , where  $q_m$  and  $p_k$  are the queries and candidate path, respectively. The element  $s_{m,k}$  refers to the similarity between the ground truth contextual path and the candidate path  $p_k$ , i.e.,  $s_{m,k} = \text{CosineSimilarity}(z_k^{pt,m}, z_{GT}^{pt,m})$  for path  $p_k$ . We use  $P_m$  to denote the set of candidate paths (including the ground truth contextual path) for query  $q_m$ . The candidate paths are then ranked by their predicted scores  $s'_{m,k}$ 's. The path ranker then learns to minimize the difference between its predicted rank by  $s'_{m,k}$  values and ground truth rank by  $s_{m,k}$  values. Each query consists of  $|P_m|$  context representation is.

Popular learning to rank methods include LAMBDAMART [9] and listNET [11]. In this article, we utilize  $\text{XE}_{NDCG}$ MART [8] as our learning to rank path ranker.  $\text{XE}_{NDCG}$ MART learns a scoring function such that  $f^{LTR} : X \rightarrow \mathbb{R}$  where  $X$  is the set of input data. Specifically, the scoring a candidate path  $p_k$  for the query  $(e_h, e_t)$  in  $d$  can be represented as  $f^{LTR}(x_{m,k}) = s'_{m,k}$  where  $x_{m,k} = [z^{cx,m}, z_k^{pt,m}]$ . The loss function consists of a score distribution  $\rho$  and a parameterized class of label distribution  $\phi$ :

$$\rho(s'_{m,k}) = \frac{e^{s'_{m,k}}}{\sum_{n=1}^{|P_m|} e^{s'_{m,n}}}, \quad \phi(s_{m,k}; \gamma) = \frac{2^{s_{m,k}} - \gamma_{m,k}}{\sum_{n=1}^{|P_m|} 2^{s_{m,n}} - \gamma_{m,n}}, \quad (5)$$



where  $\gamma$  is a bounding parameter and  $\gamma \in [0, 1]^{|P_m|}$ . The loss function is then the cross-entropy between the two distribution of all instances in the training set:

$$\mathcal{L}_{\text{LTR}} = \sum_{x_{m,k} \in H} \text{CrossEntropy}(\rho(s'_{m,k}), \phi(s_{m,k}; \gamma)). \quad (6)$$

We use the LightGBM package to implement this LTR ranker [24].<sup>3</sup>

### 6.3 Training of Path Ranker

In the training of ECPR framework, we could either jointly optimize path encoder and path ranker together, i.e.,  $\mathcal{L} = \lambda_2 \mathcal{L}_{\text{pt}} + \lambda_3 \mathcal{L}_{\text{pr}}$ ,  $\mathcal{L}_{\text{pr}} \in \{\mathcal{L}_{\text{BI}}, \mathcal{L}_{\text{LTR}}\}$ . This way, the error of path ranker will affect the alignment of path representations. It is, however, more complex to train the model unless a large set of training data is available. We optimize the context encoder, path encoder, and path ranker separately. While the training process involves little training time, the trained model may not guarantee good performance. We leave the discussion of joint optimization to future works.

## 7 DATA COLLECTION

As a novel research problem, there is no public available dataset for contextual path retrieval experiments and evaluation. Hence, we construct both a synthetic dataset and another two real datasets from WikiNews articles, and we utilized Amazon Mechanical Turk workers to annotate WikiNews<sup>4</sup> news articles. In both cases, we extract a subset of DBpedia data as the input knowledge graph. DBpedia<sup>5</sup> is a knowledge base extracted from Wikipedia. Each article entry in Wikipedia corresponds to an entity in DBpedia. Every attribute of a Wikipedia article entry (usually found within the infobox of the article) that is another Wikipedia entry is extracted as a DBpedia relation linking the two entities, and the attribute label is used as the relation label. In the following, we elaborate on how the knowledge graph is extracted from DBpedia, how the input entity pairs and context documents are obtained, and how the relevant paths for each entity pair are determined.

### 7.1 Wikinews Dataset

To construct a real dataset, we crowd-sourced annotations of contextual paths for two sets of Wikinews articles. Each Wikinews article serves as a context document. Wikinews is ideal for a number of reasons: (1) Wikinews articles are well written; (2) they are already classified into categories according to its topic; (3) they are Wikified, that is, the entity mentions are linked to the Wikipedia entries; and (4) the knowledge graph of entities and relations in Wikinews can be found in DBpedia. In this work, we select 40 articles under Film category and another 40 under Music category. We name the two datasets **Wiki-film** and **Wiki-music**. The Wikinews archive is publicly available.<sup>6</sup> Since not every AMT worker is familiar with films, the annotation of contextual paths in Wikinews article is non-trivial. Furthermore, an entity pair could have a lot of candidate contextual paths between them. We thus split the annotation into two phases: (P1) **one-hop path annotation** and (P2) **multi-hop path annotation**. To derive longer-hop paths for annotation, we also augment the Wikinews articles with additional sentences covering more entities between P1 and P2.

*7.1.1 P1: One-hop Path Annotation.* In this phase, we extract all one-hop relations from DBpedia between a pair of entities in an article. An annotator is asked to identify whether the

<sup>3</sup><https://github.com/microsoft/LightGBM>.

<sup>4</sup><https://en.wikinews.org>.

<sup>5</sup><https://wiki.dbpedia.org>.

<sup>6</sup><https://dumps.wikimedia.org/>.

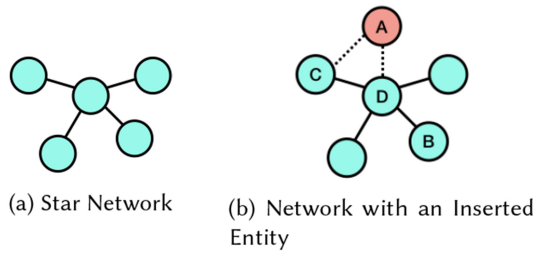


Fig. 3. Augmentation of Entity Network.

one-hop relation could explain the co-occurrence of the two entities. Figure 1 is a screenshot of our annotation interface showing a Wikinews article displayed as a set of sentences and the entity pair highlighted and underlined (i.e., *Alfonso Cuarón* and *Children of Men*). The annotator is required to determine if a relation **Alfonso Cuarón**  $\xrightarrow{\text{director}}$  **Children of Men** can explain the co-occurrence. To know more about the entities, the annotator can click on any highlighted entity and browse its Wikipedia page on the right. At the end of this annotation task, some entity pairs in an article have their contextual paths identified while other entity pairs have none. The latter can be due to either no contextual paths or longer contextual paths connecting them.

To further control the annotation quality, the annotator went through a brief online tutorial. We also designed a qualification test to exclude annotators who fail to get 8 correct answers out of 10 entity pairs. The annotator was also not allowed to give his/her answer until all sentences of the article are read (i.e., scrolling to the end of the article). For a one-hop relation to be used as a ground truth contextual path, we require it to be selected by at least two out of three annotators.

**7.1.2 P2: Augmentation of Entity Network.** After the one-hop path annotation, each article has a set of entities and their one-hop contextual paths. We then constructed a network connecting all these entities with the paths. From this entity network, we seek to generate longer contextual paths for other pairs of entities that co-occur in the article. However, we found the combined contextual paths form multiple star networks such that each star network involves a hub entity connecting to many other entities in the article as shown in Figure 3(a). This limits the longest path to be two-hops. To diversify the contextual paths, we manually add new entities to the entity networks to increase connectivity as well as to permit longer paths to be generated. For example, Figure 3(b) shows a newly inserted entity A to allow longer paths, e.g., from B to D to A and to C. To preserve the context, the added entities are required to be very relevant to the existing entities in the article.

Let the current entities of the document  $d$  be  $E_d$ . For each entity  $e$  in  $E_d$ , we consider augmenting the entity network with  $e$ 's neighboring entities currently not in  $E_d$  but exist in the knowledge graph, DBpedia in this case. For a neighboring entity of  $e$ , say,  $e_n$ , to be added to  $E_d$ ,  $e_n$  must have relation to at least another entity  $e' \in E_d$ . Moreover, we need to sample a sentence  $s$  from some paragraphs in  $e$ ,  $e'$ , and  $e_n$ 's Wikipedia articles that cover both  $e_n$  and at least one of  $e$  and  $e'$ .  $s$  is then inserted into  $d$  right after the sentence containing  $e'$ . Otherwise, we will not insert  $e_n$  into  $E_d$ . In total, we have added 85 additional entities to the dataset, while the number of articles remains to be 40. The algorithm is shown in Algorithm 1.

Consider the example Wikinews article in Figure 1<sup>7</sup>: “*Children of Men*, a movie based on a P.D. James book, has won the 2006 USC Scripter Award for its writing.... The winning screenwriters are

<sup>7</sup>[https://en.wikinews.org/wiki/%22Children\\_of\\_Men%22\\_wins\\_Scripter\\_Award\\_for\\_writing](https://en.wikinews.org/wiki/%22Children_of_Men%22_wins_Scripter_Award_for_writing).

**ALGORITHM 1:** Completion of Entity Network

---

```

input : Document  $d$ , Entity set  $E_d$  of  $d$ 
output: Document  $d'$ , Entity set  $E'_d$ 
initialization;
for entity  $e$  in  $E_d$  do
  for  $e_n$  in  $e$ 's neighbors in the knowledge graph and  $e_n \notin E_d$  do
    if  $e_n$  has relation with some other entity  $e'$  in  $E_d$  then
      Sample a sentence  $s$  from  $e$ ,  $e'$  or  $e_n$ 's Wikipedia page;
      Insert  $s$  into  $d$  after the first sentence containing  $e'$  in  $d$ ;
      Insert  $e_n$  to  $E_d$ ;
    end
  end
end

```

---

*Alfonso Cuarón, Timothy J. Sexton... Alfonso Cuarón Orozco was born in Mexico City, the son of Alfredo Cuarón. The Children of Men was James' 12th book, written in 1992.* Suppose **Children of Men** and **Alfonso Cuarón** are entities already in the article. Suppose **Mexico City** is a new common neighboring entity not in the original article, we may insert the sampled sentence  $s$  (underlined) right after the first sentence that sees the appearance of **Alfonso Cuarón** to have  $s$  work as background information of the new entity. This mechanism allows us to extend the size of the entity network of a document while maintaining a natural narrative.

7.1.3 *P2: Multi-hop Path Annotation.* After the augmentation of entity network from P1, we conduct another task to collect annotations for multi-hop paths. Consider the example in Figure 3(b), there are two paths from entity A to entity B:  $A \rightarrow C \rightarrow D \rightarrow B$  and  $A \rightarrow D \rightarrow B$ . In P2, we need annotators to decide which path is more likely to be the contextual path. We implemented a user interface similar to that of P1. Again, the annotators need to pass a qualification test including 10 questions with accuracy higher than 80%, and we derive the ground truth-annotated paths with majority vote. Eventually, we collected the contextual paths for 1,396 and 1,237 entity pairs for Wiki-film and Wiki-music dataset, respectively. The statistics of the two datasets are shown in Table 2.

## 7.2 Synthetic Dataset

While real datasets are useful for performance evaluation in real-world applications, they are costly to construct and hence too small to evaluate models with different task settings (e.g., queries with different number of candidate paths, queries with different length of contextual path...). This motivates us to construct a synthetic dataset with controllable dataset characteristics.

7.2.1 *Knowledge Graph Construction.* The first step to generating our synthetic dataset is to sample a subset of the knowledge graph that is dedicated to a certain domain. As one of our datasets is related to films, we have determined film related entities and relations in DBpedia to be included in our knowledge graph. The entities include DBpedia entries of types: Artist, Work, MovieDirector, TelevisionDirector, TheatreDirector, Writer, Person, and Film. From these entities, we find the DBpedia relations between them. In this manner, we use a knowledge graph  $G^{Film} = (E^{Film}, L^{Film}, R^{Film}, F^{Film})$ .

7.2.2 *Generation of Paths.* We next generate a set of distinctive contextual paths that will be used for the construction of context documents. To generate a path  $p$ , we first sample an entity  $e_h$  from  $E^{Film}$  and assign  $\langle e_h \rangle$  to  $p$ . Subsequently, we sample a neighbor  $e_1$  of  $e_h$  with relation label  $r_1$  from  $L^{Film}$  and append  $\langle r_1, e_1 \rangle$  to  $p$ . The sampling repeats for the neighbors of  $e_1$  until  $|p|$  reaches

**ALGORITHM 2:** Synthetic Path Generation

---

```

input : Set of entities  $E^{Film}$ , Maximum length of path  $t$ 
output: Synthetic path  $p$ 
Uniformly sample an entity  $e_h$  from  $E^{Film}$ ;
 $p = \langle e_h \rangle$ ;
while  $length(p) \leq t$  do
    Sample a neighbor  $e_i$  of  $e_h$  related by label  $r$  from  $L^{Film}$ ;
     $p += \langle r, e_i \rangle$ ;
     $e_h \leftarrow e_i$ ;
    Throw a die  $di \in [0, 1]$ ;
    if  $di \leq 0.2$  then
        | break;
end

```

---

a length threshold  $t$  or when a path-termination event occurs with 20% chance. In each iteration, there is a chance of 20% that the sampling process will terminate with a complete contextual path before we move on to generate the next contextual path. The pseudo code for the path generation is shown in Algorithm 2. In this work, we empirically set  $t$  to be 6. We exclude duplicate paths in the generation process.

**7.2.3 Generation of Context Documents.** While the generation of contextual paths is solely based on sampling  $G^{Film}$ , we synthesize a context document  $d$  for each contextual path  $p$  by sampling sentences from Wikipedia articles covering the relations in  $p$ . Let a contextual path be denoted by  $p = \langle e_h, r_1, e_1, \dots, e_t \rangle$ . The context document generation steps are depicted in Algorithm 3. We begin by sampling a relation  $(e_i, r, e_j)$  from  $p$ . Let the Wikipedia articles of  $e_i$  and  $e_j$  be  $d_i$  and  $d_j$ , respectively. We then sample a sentence with a probability  $p_{sent}$  from a paragraph in either  $d_i$  or  $d_j$ . To enhance the “relevance” of the sampled sentence, we sample from paragraphs in the  $d_i$  and  $d_j$  that contain the mentions of both  $e_i$  and  $e_j$ . In addition, with a smaller probability  $p_{intro}$ , we sample from the introduction section of  $d_i$  (or  $d_j$ ) to mimic the natural writing style, which provides some background knowledge of  $e_i$  (or  $e_j$ ) as part of the context. Finally, we sample with very small probability from the remaining sentences of  $d_i$  and  $d_j$  to add noises to the context document  $d$ . In this work, we empirically set  $p_{sent}$  and  $p_{intro}$  to 0.6 and 0.3, respectively, as shown in Algorithm 3. We leave the comparison of different ways of generation to future work.

In this work, we generate two synthetic datasets of different scale, namely, Synthetic (S) and (L). We show the statistics of the two synthetic datasets and two real-world datasets in Table 2.

## 8 EXPERIMENTS

To determine the effectiveness of different ECPR-based models, we design a series of experiments to measure how accurate they retrieve the correct contextual paths. Specifically, the models cover the four proposed context encoders combined with pathVAE-based path encoder and the two path ranker options, binary classification, and learning-to-rank models. In addition, we include two non ECPR-based models that do not use pathVAE-based path encoding as baselines, and a shortest path heuristic method as another simple baseline. The experiments are conducted on both real-world and synthetic datasets, and the corresponding experiment results are given in Sections 9 and 10, respectively. In this work, four evaluation metrics are introduced to determine the performance of each model, including two ranking-based metrics and two semantic-based metrics.

Table 2. Dataset Statistics

	Synthetic		Wikinews	
	S	L	Wiki-film	Wiki-music
# Context Documents	2,000	80,000	40	40
# Entity Pairs	5,000	200,000	1,396	1,237
Max Path Length	6	6	6	6
# Entity in KG	59,173	91,364	59,173	44,886
# Relation in KG	651	651	651	513
AVG GT Path Length	4	4	3.87	3.62
# Distinct Entities in GT Path	19,173	33,142	563	471
# Distinct Relations in GT Path	648	648	139	108
Avg # Candidate Paths per Entity Pair	8.76	7.93	7.69	5.53
AVG Candidate Path Length (including GT)	4.83	4.77	3.92	3.58
# Distinct Entities in Candidate Paths (including GT)	53,382	72,163	7,264	5,994
# Distinct Relations in Candidate Paths (including GT)	651	651	163	122

**ALGORITHM 3:** Synthetic Document Generation

---

```

input : Path  $p$  generated using Algorithm 2, Maximum number of sentences in a document  $n$ 
output: Synthetic Document  $d$ 
 $d = \{\}$ ;
while  $|d| \leq n$  do
  Uniformly sample a relation  $(e_i, r, e_j)$  from  $p$ ;
   $d_i =$  Wikipedia article of  $e_i$ ;
   $d_j =$  Wikipedia article of  $e_j$ ;
  Throw a die  $di \in [0, 1]$ ;
  if  $0 \leq di < p_{sent}$  then
    | Sample a sentence  $s$  from paragraphs containing both  $e_i$  and  $e_j$  in  $d_i$  or  $d_j$ ;
  else if  $p_{sent} \leq di < p_{sent} + p_{intro}$  then
    | Sample a sentence  $s$  from the Introduction paragraph of  $d_i$  or  $d_j$ ;
  else
    | Sample a sentence  $s$  from the rest of the paragraphs of  $d_i$  or  $d_j$ ;
  end
   $d += s$ 
end

```

---

**8.1 Model Settings**

We compare the ECPR-based models with different component settings. Other than using PathVAE for path encoding, we experiment with two path ranker configurations, namely, binary classification and learning-to-rank methods. We also include four context encoder configurations mentioned in Section 4, namely:

- **TF-IDF (ECPR-TF-IDF)**: The context representation is the concatenation of TF-IDF vectors of the context of head and tail entities, i.e.,  $[z_h^{cx(\text{TF-IDF})}, z_t^{cx(\text{TF-IDF})}]$ .
- **AVG Embeddings (ECPR-AVG Emb)**: The context representation is the concatenation of head and tail entity's averaged embedding. We include the following AVG embedding options in the experiments: (a) context entity representations from entity-only embeddings (TransE) with dimension size of 64 ( $[z_h^{cx(\text{avgTransE})}, z_t^{cx(\text{avgTransE})}]$ ), (b) context word

representations from pre-trained word-only embeddings (Word2vec) with dimension size of 300 ( $[z_h^{cx(\text{avgWord2v})}, z_t^{cx(\text{avgWord2v})}]$ ), and (c) context word and entity representations from entity-word embeddings Wikipedia2vec ( $[z_h^{cx(\text{avgWiki2v})}, z_t^{cx(\text{avgWiki2v})}]$ ). We use the pre-trained enwiki\_20180420\_win10 model with parameter settings window = 10, iteration = 10, negative = 15, and dimension size = 300.<sup>8</sup>

- **Context-fused Entity Embedding (ECPR-Cxt-fused):** The context representation options experimented are: (a) No Retrofit: The context representation is the concatenation of head and tail entities' TransE representations without retrofitting as denoted by  $[z_h, z_t]$ , (b) Retrofit(I): The context representation is the concatenation of head and tail entities' TransE representations that are retrofitting with in-context constraints only as denoted by  $[z_h^{cx(\text{cf})}, z_t^{cx(\text{cf})}]$ , (c) Retrofit (I+O): The context representation is the concatenation of head and tail entities' TransE representations that are retrofitting with in-context and out-context constraint ( $[z_h^{cx(\text{cf})}, z_t^{cx(\text{cf})}]$ ). For the hyper-parameters in the cost function for options (b) and (c) (see Section 4.3.2), we search the best combination of hyper-parameters by grid search to optimize MRR. The hyper-parameters chosen for both Wiki-film and Wiki-music datasets are:  $k_1^{\text{rf}} = 0.45$ ,  $k_2^{\text{rf}} = 0.1$ ,  $k_3^{\text{rf}} = 0.25$ ,  $k_4^{\text{rf}} = 0.2$ , and  $k_1^{\text{cf}} = 0.2$ ,  $k_2^{\text{cf}} = 0.3$ ,  $k_3^{\text{cf}} = 0.1$ ,  $k_4^{\text{cf}} = 0.2$ ,  $k_5^{\text{cf}} = 0.2$ . Cosine similarity is used as the distance measurement  $d(\cdot)$ . When extracting CNA, the context window size is empirically set to be 15.
- **Contextualized Embedding (ECPR-Cxt Emb.):** The context representation is the concatenation of head and tail entity's context encoded by: (a) BERT ( $[z_h^{cx(\text{BERT})}, z_t^{cx(\text{BERT})}]$ ), (b) KG-BERT ( $[z_h^{cx(\text{KG-BERT})}, z_t^{cx(\text{KG-BERT})}]$ ), (c) K-BERT ( $[z_h^{cx(\text{K-BERT})}, z_t^{cx(\text{K-BERT})}]$ ), and (d) KEPLER ( $[z_h^{cx(\text{KEPLER})}, z_t^{cx(\text{KEPLER})}]$ ).

For all context encoders except for the context-fused entity encoder, we consider both **context window (CW)** and **whole document (WD)** context range options for context encoding. We experimented with different context window size settings and empirically set it to be 15, as it yields good results. Recall that in Section 4.3.1 the constraints are constructed using both context window and the whole document.

To further compare the ECPR-based models with simpler retrieval methods, we include two simple baselines that do not follow the ECPR framework. Both of them do not use embedding-based path encoding. We elaborate on them as follows:

- **Baseline: TF-IDF.** This baseline utilizes keywords in context and candidate path as TF-IDF features to retrieve the most relevant candidate path. We first derive the **inverse document frequencies (IDF)** of keywords using Wikipedia articles of entities in the knowledge graph. Then, we compute the TF-IDF of context documents and paths. A path's TF-IDF representation is defined by the weighted average  $\alpha Z_e + (1 - \alpha) Z_r$ , where  $Z_e$  is the average of TF-IDF vectors of the Wikipedia articles of entities on the path normalized by article lengths, and  $Z_r$  is the average of TF-IDF vectors of names of relations appearing in the path. In this work, we arbitrarily set  $\alpha$  to be 0.5. Finally, we rank the candidate contextual paths of an entity pair in a context article by (a) **supervised method:** a LTR model trained on the training set that takes the dot product of TF-IDF vectors of context document and candidate path as input to return the path with the highest prediction probability; and (b) **unsupervised method:** a method to return the candidate path with the higher cosine similarity between the TF-IDF vector of path and that of context document.

<sup>8</sup><https://wikipedia2vec.github.io/wikipedia2vec/>.

- **Baseline: AVG Embedding.** This serves as another baseline that does not use an embedding-based context encoder. We utilize a non-contextual embedding model, Wikipedia2vec, as the base representation [53]. Similar to TF-IDF baseline, we represent the context document by averaging the Wikipedia2vec embeddings of its words and entities. Each candidate path is a weighted average  $\alpha Z'_e + (1 - \alpha)Z'_r$ , where  $Z'_e$  and  $Z'_r$  are defined similar to the TF-IDF scheme except the use of Wikipedia2vec vectors of entities and relations in the candidate path. Again, we set  $\alpha = 0.5$ . We also include both supervised (using LTR model) and unsupervised (cosine similarity) versions of this baseline method.

Finally, we include a *random guess* baseline that randomly selects a candidate path as prediction and a *shortest path* baseline that always predicts the shortest path (randomly choose one when there are multiple shortest paths). The performances of these two baselines serve as lower bound references for the others. We report the performance of the six types of context encoders combined with path encoder and the two path rankers (i.e., binary classifier and LTR ranker).

To obtain base entity and relation embeddings using TransE in both path and context encoders, We fix the embedding dimension size  $d^{\text{bg}} = 64$ , which has also been used in previous works [7, 27]. We train the model for 500 epochs with early stopping. Adam optimizer is used with initial learning rate of  $10^{-3}$ . All deep network structures are constructed using Pytorch.<sup>9</sup>

## 8.2 Evaluation Metrics

During the query phase, each ECPR-based model returns the probability of every candidate path being the contextual path for an query entity pair and context. We then rank the candidate paths by probability in decreasing order and report the **Mean Reciprocal Rank (MRR)** and **hit@k**. Both MRR and hit@k give high (or low) performance score when the ground truth paths are ranked at the top (or bottom) or near the top (or bottom). Instead of focusing solely on ground truth path retrieval, we also want to measure performance based on how similar the top-ranked candidate path(s) is similar to the ground truth path. We therefore introduce two path difference measures, **NGEO** and **PED**, to compare how the highest ranked path is similar to the ground truth path. As NGEO and PED are error-based measures, the smaller they are, the better the model performs.

**8.2.1 MRR.** MRR measures how highly ranked is the ground truth path among the candidate paths returned by a model. It is widely used in ranking and recommendation tasks. For each query triplet  $q_m = \langle e_h, e_t, d \rangle$ , let  $rank$  be the ranking of the ground truth path  $p_{gt}$  in the descending ranking list, we define the reciprocal rank of  $p_{gt, m}$  as  $\frac{1}{rank(p_{gt})}$ . The MRR of a set of test queries  $Q$  is thus defined by:

$$MRR = \frac{1}{|Q|} \sum_{q_m \in Q} \frac{1}{rank(p_{gt, m})}.$$

**8.2.2 Hit@k.** Hit@k measures if a model ranks the ground truth path among the top  $k$  predicted paths. For a set of test queries  $Q$ , we define Hit@k as:

$$\text{Hit@k} = \frac{1}{|Q|} \sum_{q_m \in Q} f_{hit@k}(rank(p_{gt, m})),$$

where  $f_{hit@k}(x) = 1$  if  $x \leq k$ , and 0 otherwise.

**8.2.3 Normalized Graph Edit Distance (NGEO).** When the retrieved contextual path does not match the ground truth, the degree of similarity between ground truth path and the retrieved

<sup>9</sup><https://pytorch.org/>.

one can be measured. High degree of similarity suggests that the two match well and hence contributing positively to the result accuracy. We therefore introduce other similarity-based performance metrics. Instead of using these new metrics to optimize model training, we use them to compare two results that fail to rank ground truth path at the top. The result with top path most similar to the ground truth path should be more superior.

Our first similarity-based metric, the **Graph Edit Distance (GEO)**, is originally designed to measure graph similarity by the number of operations needed to transform one graph to another [60]. We adapt it to measure the similarity between a top-ranked path  $p$  and the ground truth path  $p_{gt}$ .

Let  $OP(p, p_{gt})$  be the shortest sequence of edit operations for converting the former to the latter. The GEO is defined by:

$$GEO(p, p_{gt}) = \sum_{op_j \in OP(p, p_{gt})} c_{op_j}, \quad (7)$$

where  $c_{op_j}$  is the cost of an edit operation  $op_j$ .

There are six types of operations defined: semantic entity insertion, semantic entity deletion, semantic entity substitution, semantic relation insertion, semantic relation deletion, and semantic relation substitution. The cost of each edit operation is defined by difference the operation makes to entity type or relation label distance as determined by the ontology structure underlying the entities and relation labels. Here, the ontology structure defines the subclass relationships among entity types and relation labels from the knowledge graph. When an entity type (or relation label)  $e_v$  (or  $r_v$ ) is a subclass of another entity type (or relation label)  $e_w$  (or  $r_w$ ), we denote it by  $e_w \rightarrow e_v$  (or  $r_w \rightarrow r_v$ ). For example, the ontology of DBpedia defines the entity type of *The Godfather* to be  $dbo : Film$ .  $dbo : Film$  is a subclass of  $dbo : Work$ , and  $dbo : Work$  is the highest level of class in this ontology. To compare across all types of entities, we add a common root to the ontology, which serves as the common parent class of all highest level classes. For example, we denote the ontology path  $p^o$  for the entity *The Godfather* by  $root \rightarrow dbo : Work \rightarrow dbo : Film \rightarrow The\ Godfather$ . Let  $e$  ( $r$ ) and  $e'$  ( $r'$ ) be the inserted entity's type (inserted relation label) and deleted entity's type (deleted relation label), respectively, and  $e_0$  ( $r_0$ ) be the root in the knowledge ontology. For each operation  $op$  performed on a path, the semantic cost  $c_{op}$  incurred is defined as follows:

$$\begin{aligned} \text{Semantic Entity Insertion: } c_{ei}(e) &= dist(e, e_0) \\ \text{Semantic Entity Deletion: } c_{ed}(e') &= dist(e', e_0) \\ \text{Semantic Entity Substitution: } c_{es}(e, e') &= dist(e, e') \\ \text{Semantic Relation Insertion: } c_{ri}(r) &= dist(r, r_0) \\ \text{Semantic Relation Deletion: } c_{rd}(r') &= dist(r', r_0) \\ \text{Semantic Relation Substitution: } c_{rs}(r, r') &= dist(r, r'), \end{aligned} \quad (8)$$

where semantic distance is defined as the co-topic distance for  $e_1$  and  $e_2$  in the ontology:

$$dist(e_1, e_2) = \frac{|(p_{e_1}^o \cup p_{e_2}^o) - (p_{e_1}^o \cap p_{e_2}^o)|}{|p_{e_1}^o \cup p_{e_2}^o|}. \quad (9)$$

For instance, the semantic distance between *The Godfather* (with ontology path:  $root \rightarrow dbo : Work \rightarrow dbo : Film \rightarrow The\ Godfather$ ) and *Yungblud* (with ontology path:  $root \rightarrow dbo : Person \rightarrow dbo : Artist \rightarrow dbo : MusicalArtist \rightarrow Yungblud$ ) is  $\frac{7}{8}$ . The GEO of a candidate path is then the summation over costs of all semantic operation needed to convert the path into the ground truth path. The above definitions of semantic distance and GEO apply to relations as well.



In this work, we report the normalized GEO:

$$NGEO(p, p_{gt}) = \min\left(\frac{GEO(p, p_{gt})}{|p_{gt}|}, 1\right),$$

where  $|p_{gt}|$  is the length of ground truth path. We normalize GEO so it does not bias against long paths and limits the path distance from ground truth to 1. The path here could be defined as a full path  $(e_1, r_1, \dots, r_{L-1}, e_L)$ , an entity path where we only focus on entities in a path  $(e_1, \dots, e_L)$ , or a relation path  $(r_1, \dots, r_{L-1})$ . In this work, we report the NGENO for entity and relation path separately as NGENO(Ent) and NGENO(Rel) so we could make better observations about the dissimilarity between predicted path and ground truth path.

**8.2.4 Path Embedding Distance (PED).** While NGENO measures the differences between two paths, it lacks explicit semantic comparison between entities or relations from the two paths. Thus, we propose another similarity-based metric: **Path Embedding Distance (PED)** that quantify the semantic discrepancies between two paths using embeddings.

The key idea of PED is to measure the distance between the generated path and ground truth path using their embedding representations. With a PathVAE that encodes paths into latent representations, the alignment of the path representations in the embedding space reflect that paths of similar semantics will have similar representations. Using our path encoder, we could obtain the representation of generated path  $z_p^{pt}$  as well as the ground truth path  $z_{gt}^{pt}$ . The path embedding distance could thus be defined by the cosine distance of the two:

$$PED(p, p_{gt}) = 1 - \left| \text{Rescaled Cosine Similarity} \left( z_p^{pt}, z_{gt}^{pt} \right) \right|, \quad (10)$$

where *Rescaled Cosine Similarity* is cosine similarity (originally ranged  $[-1, 1]$ ) rescaled to be in the range of  $[0, 1]$ , i.e.,  $\text{Rescaled Cosine Similarity} = \frac{\text{Cosine Similarity} - (-1)}{1 - (-1)}$ . In this work, we use the same pathVAE in ECPR to learn the path representation to obtain PED.

We report PED and NGENO between the highest ranked path returned by the path ranker and ground truth path. Both PED and NGENO return 0 when the two are identical, and larger values as the two become more different from each other.

## 9 EXPERIMENT RESULTS ON WIKINEWS DATASETS

In this section, we present the performance results of ECPR-based models and other baselines on the two Wikinews datasets, Wiki-film and Wiki-music. Five-fold cross validation is used to report the average MRR,  $\text{hit}@k$ , NGENO, and PED of all the models. As the ECPR framework is complex, we first present a series of retrieval accuracy results for evaluating the model components and options. We then present selected case examples to show the differences among context encoder options, between context window and whole document options in context encoding, and between different contextualized embedding-based context encoder options. Other than the result analysis (in Section 9.6), we leave out results from models using whole document as the context range option in view of their poorer results than those models using the context window option. For easy reading, the best results in the result tables are boldfaced.

### 9.1 Path Embeddings with PathVAE

Before we present other results, we first present results of path encoding with PathVAE. In Figure 4, we show the t-SNE visualization of PathVAE embeddings of a random sample of paths including four selected paths. These four selected paths share the same head entity *Francis Coppola*:

- (A) *Francis Coppola*  $\xleftarrow{\text{director}}$  *The Godfather*,
- (B) *Francis Coppola*  $\xleftarrow{\text{director}}$  *The Godfather Part II*,

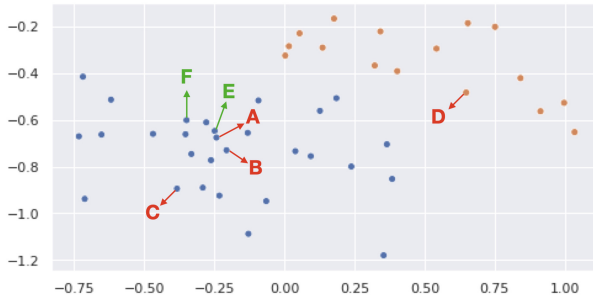


Fig. 4. Visualization of PathVAE embedding model.

- (C) *Francis Coppola*  $\xleftarrow{\text{director}}$  *Apocalypse Now*, and
- (D) *Francis Coppola*  $\xrightarrow{\text{child}}$  *Sofia Coppola*.

We want to examine if PathVAE demonstrates the property of placing similar paths close to one another and different paths from from one another. This way, one can determine if clusters form among paths. Furthermore, we want to evaluate if PathVAE can effectively handle new paths.

Among the selected paths, paths A, B, and C are similar to one another, as they share the same relation label, director. We observe their mutual closeness by PathVAE in Figure 4. Path D, however, is far from the rest, as it describes a different relation. Figure 4 also depicts paths A, B, and C in the same cluster (colored blue) while D is in another (colored orange) when we cluster the paths using K-means. This empirically illustrates that PathVAE can encode paths effectively.

Next, we check if distance between paths reflects similarity. Among paths A, B, and C, the first two are more similar to each other, as they share related tail entities, i.e., (*The Godfather Part II* is a sequel to *The Godfather*). Path C’s tail entity (*Apocalypse Now*) is less related to that of A and B. The locations of paths A, B, and C in Figure 4 match the above judgments. The distance between embeddings of paths A and C is larger than that between A and B. This indicates that our trained pathVAE embedding model captures path similarity well.

In addition to paths A–D, in the figure, we also show path (E) *Francis Coppola*  $\xleftarrow{\text{director}}$  *The Godfather(film series)*, which is a randomly sampled neighbor surrounding paths A–D. The visualization displays the embeddings of path E within the same cluster as paths A to C. Furthermore, as the entity *The Godfather(film series)* is related to *The Godfather* and *The Godfather Part II*, E is very near paths A and B in the PathVAE embedding space.

Finally, to demonstrate PathVAE’s ability to induce new paths, we show a path that does not exist in the training path set: (F) *Francis Coppola*  $\xleftarrow{\text{director}}$  *The Godfather Saga*. Since *The Godfather Saga* is a television miniseries that combines *The Godfather* and *The Godfather Part II* into one film, PathVAE has correctly placed path F near paths A and B, as shown in Figure 4.

## 9.2 AVG Embedding Encoders

We evaluate the CPR results of ECPR-based models using different AVG embedding context encoders while using PathVAE for path encoding and learning-to-rank for path ranking. For simplicity, we assume **context window (CW)** to be the default context range option. As shown in Table 3, entity-word embeddings (i.e., Wikipedia2vec) performs the best among all AVG embedding encoders. Entity-only (TransE) and Word-only (Word2vec) AVG embedding options share similar poor performance. The same result is observed for both Wiki-film and Wiki-music datasets. It shows that by combining both entity and word embeddings, Wikipedia2vec can more effectively

Table 3. Performance Comparison among AVG Embedding Encoders (with PathVAE and LTR, CW only)

Dataset	Model	Cxt Enc.	Evaluation Metrics						
			Emb Model	MRR	hit@k			NGEO	
		k = 1			k = 3	k = 5	Rel	Ent	
Wiki-film	ECPR-AVG Emb	TransE	0.372	0.1999	0.4897	0.6887	0.17	0.17	0.301
		Word2vec	0.375	0.2005	0.4897	0.6891	0.17	0.16	0.299
		Wikipedia2vec	<b>0.382</b>	<b>0.2177</b>	<b>0.5069</b>	<b>0.6998</b>	<b>0.16</b>	<b>0.15</b>	<b>0.278</b>
Wiki-music	ECPR-AVG Emb	TransE	0.495	0.2054	0.6163	0.9068	0.14	0.14	0.221
		Word2vec	0.504	0.2089	0.6207	0.9092	0.14	0.13	0.219
		Wikipedia2vec	<b>0.513</b>	<b>0.2108</b>	<b>0.6223</b>	<b>0.9117</b>	<b>0.13</b>	<b>0.12</b>	<b>0.216</b>

Table 4. Performance Comparison among Context-fused Entity Context Encoders (with PathVAE and LTR)

Dataset	Model	Cxt Enc.	Evaluation Metrics						
			Cxt-fused Ent.	MRR	hit@k			NGEO	
		k = 1			k = 3	k = 5	Rel	Ent	
Wiki-film	ECPR-Cxt Emb	No Retrofit	0.363	0.1443	0.3992	0.6515	0.18	0.18	0.324
		Retrofit (I)	0.391	0.2204	0.5079	0.6834	<b>0.16</b>	<b>0.15</b>	0.241
		Retrofit (I+O)	<b>0.414</b>	<b>0.2587</b>	<b>0.5432</b>	<b>0.7116</b>	<b>0.16</b>	<b>0.15</b>	<b>0.227</b>
Wiki-music	ECPR-Cxt Emb	No Retrofit	0.459	0.1832	0.5429	0.9043	0.17	0.16	0.247
		Retrofit (I)	0.515	0.2203	0.6354	0.9121	<b>0.13</b>	<b>0.12</b>	0.214
		Retrofit (I+O)	<b>0.521</b>	<b>0.2547</b>	<b>0.6679</b>	<b>0.9136</b>	<b>0.13</b>	<b>0.12</b>	<b>0.203</b>

capture the context semantics than TransE and Word2vec. When there is no other entity mentioned in the context window, entity-only AVG embedding option will reduce to random guess. This may explain why entity-only AVG embedding performs slightly poorer than word-only AVG embedding. As AVG embedding using Wikipedia2vec yields the best performance, we will leave out TransE and Word2Vec options in the subsequent experiment results and discussions.

### 9.3 Context-fused Entity Context Encoder

Table 4 shows the CPR results of ECPR-based models using context-fused entity embeddings, models that use constraints for retrofitting, i.e., Retrofit (I+O), outperforms Retrofit (I), and No Retrofit. The inclusion of both in-context and out-context constraints helps to augment the query entity representations with knowledge that are relevant to the context. Table 4 shows that the No Retrofit option yields the worst performance. This is reasonable, as the input in this setting is basically  $[z_{e_i}, z_{e_j}]$ , which do not provide any additional information and will result in performance similar to random guess. The Retrofit (I+O) option yields the best performance followed by the Retrofit (I) option. Henceforth, we will use Retrofit (I+O) as the representative context-fused entity encoding option in the subsequent experiment results.

### 9.4 Contextualized Embeddings

Finally, we compare ECPR-based models that utilize contextualized embeddings. As shown in Table 5, ECPR-KEPLER and ECPR-BERT are the best- and worst-performing models. ECPR-Cxt Emb(K-BERT) is the second-best-performing model, followed by ECPR-Cxt Emb(KG-BERT). Both KEPLER and K-BERT incorporate descriptive knowledge of entities in addition to relation structure of knowledge graph during training. KG-BERT, in contrast, incorporates only relation structure of knowledge graph. This may explain the observed performance differences. Henceforth,

Table 5. Performance Comparison among Contextualized Word-entity Embeddings (with PathVAE and LTR, CW Only)

Dataset	Model	Cxt Enc.	Evaluation Metrics							
			Cxt Emb.	MRR	hit@k			NGEO		PED
					k = 1	k = 3	k = 5	Rel	Ent	
Wiki-film	ECPR-BERT	BERT	0.483	0.2864	0.6422	0.7828	0.13	0.11	0.189	
	ECPR-Cxt Emb	KG-BERT	0.487	0.2954	0.6467	0.7866	0.13	0.12	0.179	
		K-BERT	0.546	0.3682	0.7238	0.8261	<b>0.12</b>	0.1	0.173	
		KEPLER	<b>0.558</b>	<b>0.3786</b>	<b>0.729</b>	<b>0.8339</b>	<b>0.12</b>	<b>0.09</b>	<b>0.171</b>	
Wiki-music	ECPR-BERT	BERT	0.574	0.3389	0.6999	0.9212	0.12	0.1	0.183	
	ECPR-Cxt Emb	KG-BERT	0.598	0.3578	0.713	0.9234	0.12	0.1	0.179	
		K-BERT	0.627	0.4002	0.7378	0.9251	<b>0.11</b>	<b>0.09</b>	0.173	
		KEPLER	<b>0.653</b>	<b>0.4447</b>	<b>0.7561</b>	<b>0.93</b>	<b>0.11</b>	<b>0.09</b>	<b>0.165</b>	

Table 6. Performance Comparison among Binary Classification Ranker and LTR (with KEPLER and PathVAE, CW Only)

Dataset	Model	Path Rnk.	Evaluation Metrics						
			MRR	hit@k			NGEO		PED
				k = 1	k = 3	k = 5	Rel	Ent	
Wiki-film	ECPR-Cxt Emb	Binary	0.553	0.3772	0.7187	0.8331	0.14	0.11	0.182
		LTR	<b>0.558</b>	<b>0.3786</b>	<b>0.729</b>	<b>0.8339</b>	<b>0.12</b>	<b>0.09</b>	<b>0.171</b>
Wiki-music	Wiki-film	Binary	0.648	0.4431	0.7499	0.9291	0.12	<b>0.09</b>	0.171
		LTR	<b>0.653</b>	<b>0.4447</b>	<b>0.7561</b>	<b>0.93</b>	<b>0.11</b>	<b>0.09</b>	<b>0.165</b>

we shall use KEPLER as the representative contextualized word-entity embeddings in subsequent experiments.

### 9.5 Results of Pank Rankers

In addition to different context encoders, we compare the path rankers: binary classifier and **learning-to-rank (LTR)** model. We evaluate the two with ECPR-Cxt Emb using KEPLER context encoder and PathVAE path encoder. As shown in Table 6, we found that LTR path ranker outperforms binary classifier ranker. This finding is more significant in the NGEO and PED results. LTR’s list-wise ranking mechanism is more superior than binary classifier, which only has been optimized to predict the ground truth contextual path. Hence, it is appropriate to use LTR as default.

### 9.6 Overall Results

We show the result of baselines and the representative context encoders in Tables 7 and 8 and the results using the whole document option for context encoding. For MRR and hit@k, the best-performing model is ECPR-Cxt Emb with KEPLER using context window as context encoder, PathVAE and LTR ranker as path encoder and path ranker, respectively. For both datasets, ECPR-Cxt Emb outperforms ECPR-Cxt-fused followed by others. ECPR-AVG Emb and ECPR-TF-IDF share similar performance and outperform the baseline-AVG Emb and Baseline-IF-IDF by a small margin. The baseline TF-IDF with cosine similarity path ranking performs so poorly that its MRR is only 0.003 better than random guess. Generally, the two no-PathVAE baselines (TF-IDF and AVG Emb.) also perform poorly when learning to rank is used.

In summary, we conclude that **ECPR-Cxt Emb** > **ECPR-Cxt-fused** > {**ECPR-AVG Emb**, **ECPR-TF-IDF**} where > denotes “outperforms.” This suggests that context encoders that embed

more information perform better. Although ECPR-AVG Emb already considers words and entities in the query context, they do not differentiate the importance of words and entities to the query context. ECPR-Cxt-fused with Retrofit (I+O), however, treats entities in and outside context window differently. It can therefore learn to exclude negative context from the entity representation by using NCR constraints and achieve better performance results. Finally, ECPR-Cxt Emb encodes context in a way that the more important information weighs more in the context representation. It distinguishes every token no matter if it is an entity or a word and is able to represent the context use both background knowledge from the pre-trained model and contextual information from the context document. Therefore, it is not a surprise for ECPR-Cxt Emb to outperform the others. Although the gaps in performance seem small, we conduct significance test on the results and concluded significant difference ( $p\text{-value} < 0.01$ ) between or best and runner-up models.

We also compare models when using context window and whole document as context range (except ECPR-Cxt-fused). The two tables show that those using context window outperform those using the whole context document. This suggests that the whole document content dilutes the focus on query entities. It is therefore better to derive context encoding using words and entities nearby the query entities.

Next, we examine the differences between the traditional performance metrics, MRR and  $\text{hit}@k$ , with our proposed path similarity-based metrics, NGE0 and PED. We observe that the similarity-based metrics generally capture performance differences more clearly. Furthermore, NGE0 reflects how much modification should be made to a path to be converted to the ground truth path, and PED indicates the similarity of paths in an embedding space. This result implies that when a good model (e.g., ECPR-Cxt-Emb using KEPLER) does not rank the ground truth path at the top, it would still predict a path that is similar. We will elaborate on this in our case studies in Section 9.8.

There might be concerns about whether our models favor shorter paths over longer paths. While we find our models favor shorter paths, the averaged length of paths selected by the model for Wiki-film dataset ( $=3.842$ ) are longer than that of shortest candidate paths ( $=2.34$ ). We have the same observation for the averaged length of paths selected by the AMT human annotators ( $=3.837$ ).

## 9.7 Model Efficiency

In ECPR models, we use pre-trained context encoders and path encoders. In both training and querying, context and path encoding incur very little time ( $< 1$  ms per context/path). Thus, we spent most of the time on candidate path extraction and learning of path ranker.

Candidate path extraction could cost a lot of time, as one might need to conduct random walk on every possible entity that could be on the path from head to tail entity. To improve efficiency when extracting candidate paths, for an head entity  $e_h$  we keep a dictionary of list of entities it can reach in 1–6 steps. Before generating candidate paths between  $e_h$  and a tail entity  $e_t$  with random walk, we eliminate every  $i$ -hop neighbor of  $e_h$  if it cannot reach  $e_t$  in  $L_{MAX} - i$  steps. By doing so, we significantly reduce the time spent in candidate path extraction. On average, it takes 3.42 seconds to extract all candidate paths given a pair of head and tail entities in the query phase.

To learn a binary classifier path ranker, we spent 412 and 339 seconds for Wiki-film and Wiki-music dataset. In the query phase, it only takes  $< 1$  ms to provide prediction to a query. Compared to binary classifier path ranker, LTR path rankers take more time, as LTR involves a more complicated optimization process. Still, it only takes 14.3 and 11.2 minutes to train LTR path rankers for Wiki-film and Wiki-music datasets, and around 30 ms to give prediction in the query phase.

## 9.8 Case Example Analysis

Here, we illustrate the model differences using a few case examples. In the examples, entity mentions are underlined. Mentions of query entities are in bold and underlined.

Table 7. Result on Wiki-film (Best Performance **Bolded**, Runner-up Performance Underlined)

Settings					Evaluation Metrics							
Model	Path Enc.	Path Rnk.	Cxt Enc.	Cxt Rng	MRR	hit@k			NGEO		PED	
						k = 1	k = 3	k = 5	Rel	Ent		
Baseline	Random Guess				0.354	0.13	0.3901	0.6410	0.19	0.18	0.329	
	Shortest Path Baseline				0.363	0.1537	0.405	0.641	0.18	0.17	0.324	
ECPR-TF-IDF	PathVAE	LTR	TF-IDF	CW	0.368	0.1801	0.4548	0.6744	0.17	0.17	0.313	
				WD	0.366	0.1723	0.4313	0.6529	0.18	0.17	0.321	
ECPR-AVG Emb			Wiki2vec	CW	0.382	0.2177	0.5069	0.6998	0.16	0.15	0.278	
				WD	0.371	0.1988	0.4889	0.6862	0.17	0.16	0.306	
ECPR-Cxt-fused			Retrofit (I+O)	-†	0.414	0.2587	0.5432	0.7116	0.16	0.15	0.227	
ECPR-Cxt Emb			K-BERT	CW	0.546	0.3682	0.7238	0.8261	<b>0.12</b>	<u>0.10</u>	0.173	
				WD	0.501	0.315	0.6623	0.7892	0.13	0.11	0.176	
			KEPLER	CW	<b>0.558</b>	<b>0.3786</b>	<b>0.729</b>	<b>0.8339</b>	<b>0.12</b>	<b>0.09</b>	<b>0.171</b>	
				WD	<u>0.507</u>	<u>0.3243</u>	<u>0.6717</u>	<u>0.7953</u>	<b>0.12</b>	<u>0.10</u>	<u>0.180</u>	
Other Baselines			TF-IDF	Cos	TF-IDF	CW	0.360	0.1362	0.392	0.6422	0.18	0.18
	WD	0.357				0.1341	0.3918	0.6403	0.18	0.18	0.329	
	LTR	CW		0.365		0.1541	0.423	0.6469	0.18	0.17	0.316	
		WD		0.364		0.1539	0.411	0.6411	0.18	0.17	0.319	
	AVG Emb.	Cos	Wiki2vec	CW		0.367	0.1663	0.4308	0.6572	0.18	0.17	0.320
				WD		0.366	0.1642	0.4256	0.6533	0.18	0.17	0.323
		LTR		CW		0.368	0.1792	0.4421	0.6791	0.17	0.17	0.315
				WD		0.368	0.1784	0.4304	0.6786	0.17	0.17	0.317

CW: Context Window Only, WD: Whole Document.

†Retrofit (I+O) does not apply to either CW or WD setting.

**9.8.1 Comparison among All Context Encoders.** In the following, we illustrate the differences of these models using the top-ranked candidate paths they return for specific context documents and query entities. By default, the models are assumed to use context window option, PathVAE path encoder, and learning-to-rank path ranker.

**Case Example 1:** Consider the example query entities *Emma Stone* and *Andrew Garfield* in the following context from Wiki-film:

“The movie, featuring Ryan Gosling and **Emma Stone**, received nominations in all major categories. Gosling and Stone received nominations for Best Actor and Actress, respectively... **Andrew Garfield**, who previously starred in The Amazing Spider-Man along with Emma Stone, competes with Gosling for his role in Hacksaw Ridge...”<sup>10</sup>

Both of ECPR-AVG Emb and ECPR-Cxt-fused with Retrofit (I+O) predict the path:

*Emma Stone*  $\xleftarrow{\text{starring}}$  *The Amazing Spider-Man*  $\xrightarrow{\text{starring}}$  *Andrew Garfield*

as the contextual path. This is because both context encoders are somewhat misled by the mention of *The Amazing Spider-Man* appearing near that of *Andrew Garfield*.

The ground truth path, however, is returned by ECPR-Cxt Emb:

*Emma Stone*  $\xleftarrow{\text{starring}}$  *La La Land*  $\xrightarrow{\text{WikiPageLink}}$  *Academy Awards*  $\xrightarrow{\text{WikiPageLink}}$  *Andrew Garfield*.

<sup>10</sup>[https://en.wikinews.org/wiki/La\\_La\\_Land\\_receives\\_record-equalling\\_fourteen\\_Oscar\\_nominations;\\_Hacksaw\\_Ridge\\_gets\\_six](https://en.wikinews.org/wiki/La_La_Land_receives_record-equalling_fourteen_Oscar_nominations;_Hacksaw_Ridge_gets_six).

Table 8. Result on Wiki-music (Best Performance **Bolded**, Runner-up Performance Underlined)

Settings					Evaluation Metrics							
Model	Path Enc.	Path Rnk.	Cxt Enc.	Cxt Rng	MRR	hit@k			NGEO		PED	
						k = 1	k = 3	k = 5	Rel	Ent		
Baseline	Random Guess				0.458	0.1808	0.5425	0.9042	0.17	0.16	0.245	
	Shortest Path Baseline				0.475	0.1978	0.6013	0.9044	0.17	0.16	0.234	
ECPR-TF-IDF	PathVAE	LTR	TF-IDF	CW	0.488	0.2038	0.6154	0.9067	0.14	0.14	0.225	
				WD	0.484	0.2022	0.6111	0.9065	0.15	0.14	0.228	
Wiki2vec			CW	0.513	0.2108	0.6223	0.9117	0.13	0.12	0.216		
			WD	0.509	0.2053	0.6204	0.9073	0.14	0.13	0.22		
ECPR-Cxt-fused			Retrofit (I+O)	-†	CW	0.521	0.2547	0.6679	0.9136	0.13	0.12	0.203
					WD	0.521	0.2547	0.6679	0.9136	0.13	0.12	0.203
ECPR-Cxt Emb			K-BERT	CW	0.626	0.4002	0.7378	0.9251	<b>0.11</b>	<b>0.09</b>	0.173	
				WD	0.591	0.3889	0.7123	0.9172	<b>0.11</b>	0.11	0.182	
	CW	<b>0.653</b>		<b>0.4447</b>	<b>0.7561</b>	<b>0.93</b>	<b>0.11</b>	<b>0.09</b>	<b>0.165</b>			
	WD	<u>0.627</u>		<u>0.4361</u>	<u>0.7522</u>	<u>0.9295</u>	<b>0.11</b>	<u>0.1</u>	<u>0.171</u>			
Other Baselines	TF-IDF	Cos	TF-IDF	CW	0.472	0.1981	0.5939	0.9051	0.17	0.16	0.239	
				WD	0.471	0.1968	0.5935	0.9044	0.17	0.16	0.241	
		LTR	CW	0.478	0.1993	0.6023	0.9053	0.16	0.16	0.233		
			WD	0.475	0.1981	0.6012	0.9045	0.17	0.16	0.235		
	AVG Emb.	Cos	Wiki2vec	CW	0.477	0.1998	0.6010	0.9054	0.16	0.15	0.231	
				WD	0.474	0.198	0.6009	0.9051	0.16	0.15	0.232	
		LTR		CW	0.487	0.2029	0.6139	0.9063	0.15	0.14	0.228	
				WD	0.483	0.2013	0.6107	0.9052	0.16	0.14	0.231	

**CW**: Context Window Only, **WD**: Whole Document.

†Retrofit (I+O) does not apply to either CW or WD setting.

Since the co-star Ryan Gosling appears in the context window of *Emma Stone*, KEPLER is able to tell that this context is about the movie *La La Land* instead of *The Amazing Spider-Man*, which is less relevant to the awards nomination context. ECPR-Cxt-Emb with KEPLER is therefore able to retrieve the correct contextual path, even when *La La Land* is not found in the context window of both head and tail entities.

**Case Example 2:** When ECPR-Cxt-Emb with KEPLER does not predict the ground truth path successfully, it still returns paths that are very similar to the ground truth. Consider the following context document and the query entity pair (*Casino Royale*, *Charlie and the Chocolate Factory*):

“Firefighters have confirmed that the large James Bond sound stage at Pinewood Studios has been destroyed by fire. It is thought eight fire engines were called to the scene near Iver Heath in Buckinghamshire on Sunday morning, where filming for Casino Royale, the latest Bond movie...and high-budget movies like Harry Potter and Charlie and the Chocolate Factory have since been filmed there...”<sup>11</sup>

The ground truth path is:

*Casino Royale*  $\xrightarrow{\text{WikiPageLink}}$  *Pinewood Studios*  $\xrightarrow{\text{WikiPageLink}}$  *Charlie and the Chocolate Factory*.

While ECPR-Cxt Emb does not predict the same, it returns a path that is very similar to the ground truth:

*Casino Royale*  $\xrightarrow{\text{subject}}$  *Films shot at Pinewood Studios*  $\xleftarrow{\text{subject}}$  *Charlie and the Chocolate Factory*.

<sup>11</sup>[https://en.wikinews.org/wiki/James\\_Bond\\_set\\_at\\_Pinewood\\_Studios\\_destroyed\\_by\\_fire](https://en.wikinews.org/wiki/James_Bond_set_at_Pinewood_Studios_destroyed_by_fire).

In fact, one might argue that the path returned by ECPR-Cxt Emb is actually better. It has not been included for human annotation (i.e., to be considered for ground truth), because it includes an entity not mentioned in the context document (i.e., *Films shot at Pinewood Studios*). Recall that our annotation process assumes that all contextual paths are derived from an entity network involving entity mentions in the context document. We leave the discussion of queries with such ground truth paths in Section 10, which involves experiments using a synthetic dataset. However, misled by the mention of *Buckinghamshire* in the context window, ECPR-AVG Emb ranks the path

$$\textit{textitCasinoRoyale} \xrightarrow{\text{WikiPageLink}} \textit{Buckinghamshire} \xrightarrow{\text{country}} \textit{United Kingdom} \xrightarrow{\text{country}} \\ \textit{Charlie and the Chocolate Factory}$$

the highest. While both ECPR-AVG and ECPR-Cxt-Emb fail to return the ground truth, the path returned by ECPR-Cxt Emb is more contextual than that returned by ECPR-AVG Emb as measured by both NGeo and PED.

**9.8.2 Context Range: Context Window vs. Whole Document.** Our earlier experiment results show that define context defined by words/entities within same context window outperforms those that use the whole context document. This is not surprising, as there might be irrelevant information or noises in the document. Here, we focus on ECPR-Cxt-Emb with KEPLER using whole document or context window. While we do not report case studies on other models, the result is consistent.

**Case Example 3:** Consider the query *Alfred Hitchcock* and *United Kingdom* in the context: “At least nine of **Alfred Hitchcock**’s rare silent films, made at the beginning of his career, will be staged in 2012 in many public screenings... Hitchcock was born in Leytonstone, London, United Kingdom on August 13, 1899... and one of his most successful movies during his Hollywood stay was the 1958 film Vertigo...”<sup>12</sup>

The ground truth path is

$$\textit{Alfred Hitchcock} \xrightarrow{\text{birthPlace}} \textit{Leytonstone} \xrightarrow{\text{country}} \textit{United Kingdom}.$$

ECPR-Cxt-Emb with KEPLER that only focuses on context window surrounding the query entities successfully returns the correct contextual path. As several mentions of movies directed by Hitchcock are mentioned in the context document, ECPR-Cxt-Emb with KEPLER using whole document option returns a wrong path as follows:

$$\textit{Alfred Hitchcock} \xleftarrow{\text{director}} \textit{Vertigo} \xrightarrow{\text{country}} \textit{United Kingdom}.$$

It ranked the ground truth path at the fourth position.

**9.8.3 BERT vs. KEPLER.** Generally, ECPR-Cxt Emb (i.e., KG-BERT, K-BERT, and KEPLER) outperform ECPR-BERT, although BERT already shows promising improvement over other baseline models, especially in context documents where query entities are not given much description.

**Case Example 4:** For instance, when retrieving the contextual path between *Barack Obama* and *Bill Clinton* in the following context:

“On Saturday night, former United States presidents **Barack Obama**, **Bill Clinton**, Jimmy Carter and father and son George H.W. Bush and George W. Bush attended a concert at the Reed Arena in Texas to raise funds for hurricane relief...”<sup>13</sup>

<sup>12</sup>[https://en.wikinews.org/wiki/Nine\\_of\\_Alfred\\_Hitchcock%27s\\_films\\_are\\_restored;\\_30\\_years\\_since\\_his\\_death](https://en.wikinews.org/wiki/Nine_of_Alfred_Hitchcock%27s_films_are_restored;_30_years_since_his_death).

<sup>13</sup>[https://en.wikinews.org/wiki/Five\\_United\\_States\\_ex-presidents\\_raise\\_relief\\_funds\\_at\\_hurricane\\_event](https://en.wikinews.org/wiki/Five_United_States_ex-presidents_raise_relief_funds_at_hurricane_event).



Since the context windows of the two query entities overlap each other, we only have one context window to extract information from. ECPR-BERT, returns the path:

$$\textit{Barack Obama} \xleftarrow{\text{subject}} \textit{Presidents of the United States} \xrightarrow{\text{subject}} \textit{Bill Clinton},$$

as many other ex-presidents of the United States are mentioned in the context window. The ground truth path is returned by ECPR-Cxt Emb:

$$\textit{Barack Obama} \xleftarrow{\text{WikiPageLink}} \textit{One America Appeal} \xrightarrow{\text{WikiPageLink}} \textit{Bill Clinton}.$$

ECPR-Cxt Emb appears to know that when the keywords Texas, hurricane, and funds appear together with *Barack Obama* and *Bill Clinton* in the same context, the story is about the establishment of *One\_America\_Appeal*. ECPR-BERT, however, does not have such background knowledge embeds in it. It therefore fails to retrieve the ground truth contextual path.

## 10 ANALYSIS ON SYNTHETIC DATASETS

In this section, we analyze our proposed models from different aspects using synthetic datasets. As described in Section 7.2, we generate synthetic context documents and their contextual paths from a sampled knowledge graph built on DBpedia. As the size synthetic dataset is much larger than Wikinews datasets, we are able to find sufficient number of queries and their candidate paths to evaluate how well a model copes with queries of varying levels of difficulty.

Through this analysis on synthetic dataset, we aim to answer the following research questions:

- How will the model perform on large-scale datasets?
- How will the similarity among candidate paths affect the performance? When candidate paths are similar, it will naturally be more difficult for a model to determine the most contextual path among them.
- How will the number of candidate paths affect the performance? Queries with many candidate paths should be more difficult than those with few candidate paths.
- How will the length of the contextual path affect the performance? When the ground truth contextual path involves many entities and relations, it will be more difficult to encode its semantics and match with the query context.

For the first research question, we use the Synthetic (L) dataset. For the second research question onwards, we use Synthetic (S) dataset and compare the performance on synthetic dataset of two selected models: (a) ECPR-Cxt Emb + PathVAE + LTR (context window only) and (b) ECPR-AVG Emb + PathVAE + LTR (context window only). (a) is our best-performing model, and (b) is a simple model that takes the average embeddings of both words and entities in context encoding.

### 10.1 Model Performance on Large-scale Dataset

As Wiki-film and Wiki-music datasets are relatively small-sized, we experimented selected ECPR models with exact same setting as described in Section 8 on the much larger Synthetic (L) dataset. As shown in Table 9, the results observed using Synthetic (L) are similar to those in Tables 7 and 8. The best-performing model is ECPR-Ext Emb with KEPLER followed by ECPR-Ext Emb with K-BERT. The results also show that all ECPR models outperform the two baselines. This result suggests that ECPR models can effectively handle large-scale datasets.

Table 9. Result on Synthetic (L) (Best Performance **Bolded**, Runner-up Performance Underlined)

Settings					Evaluation Metrics						
Model	Path Enc.	Path Rnk.	Cxt Enc.	Cxt Rng	MRR	hit@k			NGEO		PED
						k = 1	k = 3	k = 5	Rel	Ent	
Baseline	Random Guess				0.348	0.126	0.378	0.631	0.19	0.2	0.332
	Shortest Path Baseline				0.357	0.1472	0.401	0.638	0.18	0.19	0.329
ECPR-TF-IDF	PathVAE	LTR	TF-IDF	CW	0.366	0.1743	0.4429	0.6737	0.17	0.17	0.317
				WD	0.36	0.1721	0.4176	0.6503	0.18	0.19	0.325
Wiki2vec			CW	0.378	0.1882	0.4837	0.6729	0.16	0.16	0.283	
			WD	0.371	0.1849	0.4518	0.6643	0.18	0.18	0.31	
ECPR-Cxt-fused			Retrofit (I+O)	-†	0.408	0.2639	0.5634	0.7263	0.15	0.16	0.264
			K-BERT	CW	<u>0.528</u>	<u>0.3547</u>	<u>0.6947</u>	<u>0.8149</u>	<b>0.12</b>	<u>0.11</u>	<u>0.198</u>
WD				0.483	0.2999	0.6364	0.7436	<u>0.14</u>	0.12	0.203	
ECPR-Cxt Emb			KEPLER	CW	<b>0.532</b>	<b>0.3614</b>	<b>0.7182</b>	<b>0.8305</b>	<b>0.12</b>	<b>0.1</b>	<b>0.187</b>
	WD	0.516		0.3114	0.6552	0.7772	<b>0.12</b>	<u>0.11</u>	0.191		

CW: Context Window Only, WD: Whole Document.

†Retrofit (I+O) does not apply to either CW or WD setting.

Table 10. Retrieval Performance of Query Sets with Different Similarity Setting among Candidate Paths

Model	A (N = 300) $PED^P < 0.3$		B (N = 300) $PED^P > 0.5$		B-A	
	MRR	hit@1	MRR	hit@1	MRR	hit@1
	ECPR-AVG Emb	0.363	0.1563	0.392	0.2245	0.029
ECPR-Cxt Emb	<b>0.547</b>	<b>0.3599</b>	<b>0.569</b>	<b>0.3808</b>	<b>0.022</b>	<b>0.0209</b>

## 10.2 Similarity among Candidate Contextual Paths

Our second research question studies how the models perform when the candidate paths are very similar. Consider the three example paths,

(1) Francis Coppola  $\xleftarrow{\text{director}}$  The Godfather  $\xrightarrow{\text{starring}}$  Al Pacino,

(2) Francis Coppola  $\xleftarrow{\text{director}}$  The Godfather Part II  $\xrightarrow{\text{starring}}$  Al Pacino, and

(3) Francis Coppola  $\xleftarrow{\text{director}}$  The Godfather Part III  $\xrightarrow{\text{starring}}$  Al Pacino.

These paths are similar, as they only differ by their intermediate entities. A query with such candidate paths is considered difficult, as it requires the semantics of query context and candidate paths to be accurately represented for the models to determine the correct contextual path.

To conduct this evaluation, we construct two sets of queries from Synthetic (S) based on the degree of similarity among the candidate paths of these queries. For each query in the synthetic dataset, we compute the pairwise PED among all its candidate paths  $PED^P$ . Small  $PED^P$  suggests high path similarity. We then derive two query sets: (**Query set A**) consisting of 300 queries with  $PED^P < 0.3$ , and (**Query set B**) consisting of another 300 queries with  $PED^P > 0.5$ . The average number of candidate paths per query in both query sets A and B is 8. We then evaluate the model trained on Wikinews-film on the two query sets constructed using the synthetic dataset.

Based on the results in Table 10, we first verify that query set A is more difficult than query set B. Furthermore, not only does ECPR-Cxt Emb with KEPLER outperform ECPR-AVG Emb with Wikipedia2vec on both query sets, the performance gap between two query sets for ECPR-Cxt

Table 11. Retrieval Performance of Query Sets with Different Number of Candidate Contextual Paths (Numbers in Brackets Are Improvement over Random Guess)

Model	A (N = 50)		B (N = 50)	
	AVG #Candidate = 12.3		AVG #Candidate = 3.4	
	MRR	hit@1	MRR	hit@1
Random Guess	0.261	0.085	0.599	0.341
ECPR-AVG Emb	0.339 (+0.078)	0.102 (+0.017)	0.624 (+0.025)	0.397 (+0.056)
ECPR-Cxt Emb	<b>0.356</b> (+0.095)	<b>0.141</b> (+0.056)	<b>0.703</b> (+0.104)	<b>0.429</b> (+0.088)

Emb is also smaller than that for ECPR-AVG Emb. This suggests that ECPR-Cxt Emb could handle query set A with accuracy similar to query set B.

### 10.3 Number of Candidate Paths

Queries with more candidate paths are likely to be more challenging than those with few candidate paths. Among the queries of Synthetic (S), we construct two subsets of 50 queries each: (**Query set A**) has on average 12.3 candidate paths per query, and (**Query set B**) has an average of 3.4 candidate paths per query. We use the model trained on Wikinews-film and evaluate on the two query sets on the synthetic dataset. We report MRR and hit@1 in Table 11. Additionally, we report the performance of a random baseline where a randomly selected candidate path is returned. We show the delta in performance between the models and this random guess baseline in brackets.

The performance of query set A is much lower than that of query set B for both ECPR-Cxt-fused and ECPR-Cxt-Emb, confirming our hypothesis that queries with more candidate paths are more difficult. Moreover, while both models significantly outperform the random baseline, ECPR-Cxt Emb consistently achieves better improvement as opposed to ECPR-AVG Emb. The increment in MRR is almost similar for both query sets A and B, suggesting that ECPR-Cxt Emb's performance in both difficult and simple tasks are very much alike.

### 10.4 Length of Contextual Path

Finally, we answer our third research question by examining how ECPR-Cxt-fused and ECPR-Cxt-Emb perform on queries with longer contextual paths. Contextual path with more hops means that more entities and relations are needed in describing the relation between head and tail entities. When the contextual path is long, we may not find the mentions of every entity in the contextual path within the context document. There may be cases where some entities in the contextual path are not even found in the context document. Thus, such queries are considered difficult tasks.

Here, we construct two query sets from Synthetic (S) each with 300 queries: (**Query set A**), involving ground truth paths with length  $\geq 5$ , and (**Query set B**), involving ground truth paths with length  $\leq 3$ . In addition, we extract a subset of A (**Query set A'**) in which not every path entity exists in the context document. Queries in A' are considered the most difficult tasks. When constructing the query set, we control the average number of candidate paths per query to be 8 to avoid performance being affected by number of candidate paths. The average length of candidate paths for query sets A and B are 3.63 and 3.67, respectively. The path length difference is considered small. We show the performance in Table 12. First, query set A is clearly more difficult compared to B to both ECPR-Cxt Emb and ECPR-AVG Emb. The models perform less accurately for query set

Table 12. Retrieval Performance of Query Sets with Different Length of Contextual Path

Model	A (N = 300)		B (N = 300)		A' (N = 93)	
	Length of GT Path $\geq 5$		Length of GT Path $\leq 3$		Subset of A	
	MRR	hit@1	MRR	hit@1	MRR	hit@1
ECPR-AVG Emb	0.376	0.2033	0.383	0.218	0.369	0.1963
ECPR-Cxt Emb	<b>0.543</b>	<b>0.3596</b>	<b>0.561</b>	<b>0.3791</b>	<b>0.539</b>	<b>0.3484</b>

A. Furthermore, the delta between performance on A' and A is much smaller for ECPR-Cxt Emb compared to that for ECPR-AVG Emb. This observation suggests that ECPR-Cxt Emb copes with these difficult queries better.

## 11 CPR AND OTHER IR TASKS

In this section, we discuss how CPR could benefit other IR tasks. We have elaborated the similarities and disparities between CPR and IR tasks such as recommendation and question-answering in Section 2. While our proposed models cannot be directly utilized to address these IR tasks due to the disparities, they can support others by providing additional information.

Explainable recommendation systems often use purchase history to infer user preferences to determine items to be recommended. For example, one may recommend items from a company that sold many items to the user beforehand. The systems may also find other users with similar purchase histories to use these users' item for recommendation.

Textual context, or information context, is often overlooked during the recommendation. The information context could come from a product's description or an article the user just read. After linking mentions in the information context to entities in knowledge graph, one can apply CPR to return the contextual paths linking the entities. When there are product items linked to these entity mentions, the system could use the contextual paths to find other product items as candidates for the recommendation. For example, suppose a user reads an article about a book he has purchased. The article is about a movie story adapted from the book, and there exists another book adapted by the same director. Such an indirect relation between the two books is hard for current recommendation systems to learn. However, through figuring out the contextual path,

$$Book A \xrightarrow{\text{adapt}} Movie A \xrightarrow{\text{director}} Director \xrightarrow{\text{director}^{-1}} Movie B \xrightarrow{\text{adapt}^{-1}} Book B$$

CPR can help to explain the actual reason the second book should be recommended to the user. In summary, CPR helps to find the context-dependent connection between two entities. Any IR systems that have textual data as input can benefit from this additional information provided by CPR.

## 12 CONCLUSION

**Contextual path retrieval (CPR)** is a novel research task that is becoming very important when knowledge graphs are available for explaining the connections between entities found in some common context. In this article, we propose an ECPR framework to solve the task with modularized functional components and several proposed models for these components. We show that our ECPR model with KEPLER contextualized embedding outperforms baseline models through a series of experiments on two real datasets. Case study analysis has been conducted to compare the characteristics of different model settings. Furthermore, we analyze how selected models perform with different types of queries using synthetic datasets.

Still, we believe that there is still room for future CPR research. First, the accuracy of CPR task can be improved further through using much larger training data and larger knowledge graphs.

Second, we can further extend the ECPR framework, say, applying more advanced path encoding and path-ranking methods. Third, we believe more work can be conducted to develop highly efficient models for CPR tasks, as the LSTM component of ECPR does not scale very well for very large knowledge graphs and many candidate paths.

Other future directions include extensions of CPR task. In particular, CPR task with multiple ground truth paths per query should be further studied. This will allow CPR to be used in more real-world applications. As relations in knowledge graphs may not be complete, the future research of CPR should focus on generation of contextual paths instead of retrieval. At present, CPR task assumes each query consists of two entities. One can extend CPR to consider more entities and to retrieve contextual graphs connecting these entities instead of paths. Finally, there are many downstream applications and IR tasks, which require knowledge extraction and reasoning, could benefit from CPR. For example, to allow fact checking to classify a claim as fact or hoax, one could extend CPR to verify the connections of entities mentioned in the claim. CPR can also be used in a question-answering scenario where explanation text or answer can be generated from a contextual path.

## 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] Nitish Aggarwal, Sumit Bhatia, and Vinith Misra. 2016. Connecting the dots: Explaining relationships between unconnected entities in a knowledge graph. In *ESWC*.
- [2] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018).
- [3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3 (2003), 1137–1155.
- [4] Sumit Bhatia, Purusharth Dwivedi, and Avneet Kaur. 2018. That’s interesting, tell me more! Finding descriptive support passages for knowledge graph relationships. In *ISWC*.
- [5] Roi Blanco and Hugo Zaragoza. 2010. Finding support sentences for entities. In *SIGIR*.
- [6] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*.
- [7] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.
- [8] Sebastian Bruch. 2019. An Alternative Cross Entropy Loss for Learning-to-Rank. arXiv:1911.09798 [cs.LG]
- [9] C. J. Burges. 2010. *From RankNet to lambdaRank to LambdaMART. An Overview*. Microsoft Research Technical Report MSR-T R-2010-82.
- [10] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*.
- [11] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *ICML*. 129–136.
- [12] Shubham Chatterjee and Laura Dietz. 2019. Why does this entity matter? Support passage retrieval for entity retrieval. In *SIGIR*.
- [13] Hongxu Chen, Yicong Li, Xiangguo Sun, Guandong Xu, and Hongzhi Yin. 2021. Temporal meta-path guided explainable recommendation. In *WSDM*.
- [14] Ryan Clancy, Ihab F. Ilyas, and Jimmy Lin. 2019. Scalable knowledge graph construction from text collections. In *Workshop on Fact Extraction and VERification*.
- [15] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *AAAI*.
- [16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]
- [17] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL*.

- [18] Benjamin Heitmann and Conor Hayes. 2010. Using linked data to build open, collaborative recommender systems. In *AAAI Spring Symposium Series*.
- [19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computat.* 9, 8 (1997), 1735–1780.
- [20] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.
- [21] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*.
- [22] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI*.
- [23] Amina Kadry and Laura Dietz. 2017. Open relation extraction for support passage retrieval: Merit and open issues. In *SIGIR*.
- [24] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.* 30 (2017), 3146–3154.
- [25] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.
- [26] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. arXiv:1808.10568 [cs.AI]
- [27] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.
- [28] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2019. K-BERT: Enabling Language Representation with Knowledge Graph. arXiv:1909.07606 [cs.CL]
- [29] Aman Mehta, Aashay Singhal, and Kamalakar Karlapalem. 2019. Scalable knowledge graph construction over text using deep learning based predicate mapping. In *WWW*.
- [30] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]
- [31] Nikola Mrksić, Diarmuid Ó. Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL*.
- [32] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL*.
- [33] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.
- [34] Giuseppe Pirrò. 2015. Explaining and suggesting relatedness in knowledge graphs. In *ISWC*.
- [35] Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *ACL*.
- [36] Baoxu Shi and Tim Wenerger. 2017. ProjE: Embedding projection for knowledge graph completion. In *AAAI*.
- [37] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.
- [38] Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*.
- [39] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *J. Mach. Learn. Res.* 18, 130 (2017), 1–38.
- [40] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.
- [41] 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 *NeurIPS*.
- [42] Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 2017. Generating descriptions of entity relationships. In *ECIR*.
- [43] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *ACL*.
- [44] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. arXiv:1911.06136 [cs.CL]
- [45] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *KDD*.
- [46] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*.
- [47] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.
- [48] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Trans. Assoc. Computat. Linguist.* 6 (2018), 287–302.

- [49] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based QA. In *WWW*.
- [50] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*.
- [51] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. arXiv:1707.06690 [cs.CL]
- [52] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *ACL*.
- [53] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*.
- [54] Bishan Yang, Wen-Tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.
- [55] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.
- [56] Liang Yao, Chengsheng Mao, and Yuan Luo. 2020. KG-BERT: BERT for knowledge graph completion. In *AAAI*.
- [57] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple QA by attentive convolutional neural network. In *COLING*.
- [58] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. In *IJCAI HINA*.
- [59] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *KDD*.
- [60] Weiguo Zheng, Lei Zou, Wei Peng, Xifeng Yan, Shaoxu Song, and Dongyan Zhao. 2016. Semantic SPARQL similarity search over RDF knowledge graphs. *Proc. VLDB Endow.* 9, 11 (July 2016), 840–851.
- [61] Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *COLING*.

Received 16 June 2021; revised 2 October 2021; accepted 24 November 2021