

Singapore Management University

Institutional Knowledge at Singapore Management University

Dissertations and Theses Collection (Open Access)

Dissertations and Theses

5-2023

Connecting the dots for contextual information retrieval

Pei-Chi LO

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

Follow this and additional works at: https://ink.library.smu.edu.sg/etd_coll



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

Citation

LO, Pei-Chi. Connecting the dots for contextual information retrieval. (2023). 1-221.

Available at: https://ink.library.smu.edu.sg/etd_coll/494

This PhD Dissertation is brought to you for free and open access by the Dissertations and Theses at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Dissertations and Theses Collection (Open Access) by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

CONNECTING THE DOTS FOR CONTEXTUAL
INFORMATION RETRIEVAL

PEI-CHI LO

SINGAPORE MANAGEMENT UNIVERSITY

2023

Connecting The Dots for Contextual Information Retrieval

Pei-Chi Lo

Submitted to School of Computing and Information Systems in partial
fulfillment of the requirements for the Degree of Doctor of Philosophy in
Computer Science

Dissertation Committee:

Ee-Peng Lim (Supervisor/Chair)
Professor of Computer Science
Singapore Management University

Yuan Fang
Assistant Professor of Computer Science
Singapore Management University

Jing Jiang
Professor of Computer Science
Singapore Management University

Aixin Sun
Associate Professor of Computer Science and Engineering
Nanyang Technological University

Singapore Management University
2023

I hereby declare that this PhD dissertation is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree in any university previously.

A handwritten signature in black ink that reads "Pei-Chi Lo". The signature is written in a cursive style with a horizontal line underlining the text.

Pei-Chi Lo

23 May 2023

Abstract

There are many information retrieval tasks that depend on knowledge graphs to return contextually relevant result of the query. We call them **Knowledge-enriched Contextual Information Retrieval (KCIR)** tasks and these tasks come in many different forms including query-based document retrieval, query answering and others. These KCIR tasks often require the input query to be contextualized by additional facts from a knowledge graph, and using the context representation to perform document or knowledge graph retrieval and prediction. In this dissertation, we present a meta-framework that identifies *Contextual Representation Learning (CRL)* and *Contextual Information Retrieval (CIR)* to be the two key components in KCIR tasks.

We then address three research tasks related to the two KCIR components. In the first research task, we propose a **VAE-based contextual representation learning method using a co-embedding attributed network structure** that co-embeds knowledge and query context in the same vector space. The model shows superior downstream prediction accuracy compared to other baseline models using VAE with or without using external knowledge graph.

Next, we address the research task of solving a novel IR problem known as **Contextual Path Retrieval (CPR)**. In this task, a knowledge graph path relevant to a given query and a pair of head and tail entities is to be retrieved from the background knowledge graph. We develop a transformer-based model consisting of context encoder and path encoder to solve the CPR task. Our proposed models which include the proposed two encoders show promising ability to retrieve contextual paths.

Finally, we address the **Contextual Path Generation (CPG)** task which is

similar to CPR except that the knowledge graph path to be returned may require inferred relation edges since most knowledge graphs are incomplete in their coverage. For the CPG task, we propose both monotonic and non-monotonic approaches to generate contextual paths. Our experiment results demonstrate that the non-monotonic approach yields better-quality resultant knowledge graph paths.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Research Objectives and Framework	3
1.3	Summary of Dissertation Contributions	5
1.4	Dissertation Structure	6
2	Related Work	7
2.1	Text and Knowledge Graph Representation Learning	7
2.2	Contextual Retrieval of Knowledge	10
2.3	Contextual Retrieval of Text Documents	11
2.4	Contextual Item Recommendation	12
2.5	Contextual Question Answering	13
3	Co-Embedding Attributed Network Representation Learning	16
3.1	Research Objective	17
3.2	Related Work	18
3.2.1	HIN Embeddings	18
3.2.2	Variational Auto-encoder Embeddings	19
3.3	Co-Embedding with External Knowledge	21
3.3.1	Framework Overview	21
3.3.2	Attribute-to-Attribute Association	22
3.3.3	Input Matrices Construction	24
3.3.4	Co-Embedding Learning	26

3.4	Co-embedding Learning Using VAE	26
3.4.1	CAN	26
3.4.2	ECAN Models	29
3.4.3	Mixed Model	31
3.5	Experiments	33
3.5.1	Datasets	33
3.5.2	Baseline and Model Settings	35
3.5.3	Running Time	37
3.6	Results	37
3.6.1	Node Classification	37
3.6.2	Link Prediction	38
3.6.3	Choice of Dimension Size	39
3.7	Case Study	40
3.7.1	Movie Profiling	40
3.7.2	Error Analysis	41
3.8	Summary	43
4	Contextual Path Retrieval	45
4.1	Research Objectives	46
4.2	Contextual Path Retrieval	49
4.2.1	Definitions	49
4.2.2	Proposed ECPR Framework	52
4.3	Context Encoder	54
4.3.1	TF-IDF	55
4.3.2	Averaged Embeddings	55
4.3.3	Context-fused Entity Embeddings	56
4.3.4	Contextualized Embedding Representation	61
4.4	Path Encoder	62
4.5	Path Ranker	64
4.5.1	Binary Classifier	64

4.5.2	Learning to Rank	65
4.5.3	Training of Path Encoder and Ranker	66
4.6	Experiments	66
4.6.1	Model Settings	67
4.6.2	Evaluation Metrics	70
4.7	Experiment Results on Wikinews Datasets	74
4.7.1	Path Embeddings with PathVAE	75
4.7.2	AVG Embedding Encoders	77
4.7.3	Context-fused Entity Context Encoder	77
4.7.4	Contextualized Embeddings	78
4.7.5	Results of Rankers	79
4.7.6	Overall Results	79
4.7.7	Model Efficiency	81
4.7.8	Case Example Analysis	84
4.8	Analysis on Synthetic Datasets	89
4.8.1	Model Performance on Large-scale Dataset (Synthetic-L)	90
4.8.2	Similarity among Candidate Contextual Paths	90
4.8.3	Number of Candidate Paths	92
4.8.4	Length of Contextual Path	93
4.9	CPR And Other IR Tasks	94
4.10	Summary	95
5	Contextual Path Generation: A Monotonic Approach	98
5.1	Research Objective	98
5.1.1	Problem Formulation	98
5.1.2	Challenges	100
5.2	Proposed Architecture and Models	102
5.2.1	Overview of Model Architecture	102
5.2.2	Context Encoder	104
5.2.3	Controlled Path Generation	107

5.2.4	Penalization Scaling	111
5.2.5	Reward Scaling	114
5.2.6	CPG Models	117
5.3	Experiments on Real Datasets	119
5.3.1	Wikinews Dataset	119
5.3.2	Evaluation Metrics	120
5.3.3	Models for Comparison	123
5.3.4	Comparison of CPG Models	125
5.3.5	Effects of Penalization/Reward Scaling	128
5.3.6	Effects of Context Document Content Amount	129
5.3.7	Error Type Analysis	130
5.3.8	Inferring Relations in the Wiki-film Dataset	132
5.4	Experiment on a Synthetic Dataset	132
5.4.1	CPG Model Performance on the Synthetic-S Dataset	133
5.4.2	Coping with Incomplete Knowledge Graphs	134
5.5	Using a KGQA Model for CPG: A Model Adaptation Experiment	135
5.5.1	Modified EmbedKGQA Model	136
5.5.2	Experiment and Results	138
5.6	Summary	140
6	Contextual Path Generation: A Non-Monotonic Approach	143
6.1	Research Objective	144
6.1.1	Problem Definition	144
6.2	Proposed Framework and Contextual Path Generation Models	145
6.2.1	Two-Stage Framework	145
6.2.2	Context Extractor	147
6.2.3	Contextual Path Generator	152
6.2.4	Tree Serialization	159
6.3	Experiment Results	160
6.3.1	Dataset	160

6.3.2	Evaluation Metrics	160
6.3.3	Effectiveness of Context Extractor Methods	161
6.3.4	Performance in Contextual Path Generation	164
6.3.5	Comparison between Non-monotonic and Monotonic Path Generation	167
6.4	Discussion	169
6.5	Summary	171
7	Conclusion	174
	Appendices	179
A	Construction of Real and Synthetic Data Collections	180
A.1	Wikinews Datasets	181
A.1.1	P1: One-hop Path Annotation.	182
A.1.2	P2: Augmentation of Entity Network.	182
A.1.3	P2: Multi-hop Path Annotation	184
A.2	Synthetic Dataset	185
A.2.1	Knowledge Graph Construction	185
A.2.2	Generation of Paths	185
A.2.3	Generation of Context Documents	186
	Bibliography	187

List of Figures

1.1	Meta-Framework of KCIR Tasks	2
3.1	Co-embedding Attributed Networks with External Knowledge	22
3.2	CAN Model	27
3.3	ECAN (Non-Mixed Model)	29
3.4	ECAN (Mixed Model)	32
4.1	Annotation Interface	52
4.2	ECPR Framework	53
4.3	Visualization of PathVAE Embedding Model	75
5.1	Contextual Path Generation Example	100
5.2	Proposed CPG Architecture	103
5.3	Illustration of CPG-mixed Encoder and Decoder	106
5.4	Illustration of Relation and Entity Sampling Process	108
5.5	Reward Scaling	111
6.1	The Two-stage Framework for Contextual Path Generation	147
6.2	Context Extractors	148
6.3	Pretraining and Fine-tuning Steps of Non-Monotonic Contextual Path Generation with Pretrained Transformer (NMCPGT)	153

6.4	Binary Trees of Non-Monotonic Generation: (a) A terminal binary tree that shows the generation steps for a path (with generation order numbers and path order numbers shown in blue and green respectively); (b) Two binary trees with the same generated sequence.	154
6.5	Illustration of Non-Monotonic Path Generation: (a) Serialize an input tree with path traversal; (b) Generation of a Contextual Path	158
A.1	Annotation Interface	181
A.2	Star Network	183
A.3	Network with an Inserted Entity	183

List of Tables

3.1	Summary of Notations	18
3.2	Summary of Related Work	21
3.3	Dataset Statistics	34
3.4	Running Time (Second)	36
3.5	Node Classification Result (Macro-F1)	37
3.6	Link Prediction Result (AUC)	38
3.7	Node Classification Performance (Macro-F1) of Different Di- mension Sizes	39
3.8	Movie Node Profiling: <i>The Godfather</i> (words are underlined and entities are not)	41
3.9	Case Example Analysis	41
4.1	Table of Notations	50
4.2	Dataset Statistics	67
4.3	Performance Comparison among AVG Embedding Encoders (with PathVAE and LTR, CW only)	77
4.4	Performance Comparison among Context-fused Entity Context Encoders (with PathVAE and LTR)	78
4.5	Performance Comparison among Contextualized Word-Entity Embeddings (with PathVAE and LTR, CW only)	79
4.6	Performance Comparison among Binary Classification Ranker and LTR (with KEPLER and PathVAE, CW only)	79

4.7	Result on Wiki-film (Best Performance Bolded , Runner-up Performance <u>Underlined</u>)	82
4.8	Result on Wiki-music (Best Performance Bolded , Runner-up Performance <u>Underlined</u>)	83
4.9	Result on Synthetic-L (Best Performance Bolded , Runner-up Performance <u>Underlined</u>)	90
4.10	Retrieval Performance of Query Sets with Different Similarity Setting among Candidate Paths	91
4.11	Retrieval Performance of Query Sets with Different Number of Candidate Contextual Paths (Numbers in brackets are improvement over Random Guess)	92
4.12	Retrieval Performance of Query Sets with Different Length of Contextual Path	93
5.1	Table of Notations	102
5.2	Experiment Results (Best results are shown in boldface.)	126
5.3	Ablation of Penalization/Reward Scaling (Wiki-film)	127
5.4	Effects of Context Content Amount on Wiki-film with M-Wiki2vec(EW)	130
5.5	Error Type Analysis on Wiki-film dataset	131
5.6	Experiment Result: Synthetic-S Dataset	134
5.7	Experiment Result: CPG with Incomplete Knowledge Graphs (Synthetic-S, M-Wiki2vec)	135
5.8	Experiment Results: CPG with QA Model (Wiki-film)	139
6.1	Dataset Statistics	161
6.2	Performance in Context Entity Extraction (Wiki-film)	164
6.3	Path Generation Performance on Wiki-Film and Wiki-Music Datasets	165
6.4	Non-monotonic(Non-M) and Monotonic(M) Generation	167
A.1	Dataset Statistics	188

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Ee-Peng Lim. Prof. Lim has always been extremely generous and encouraging. His guidance and dedicated support have been invaluable in helping me overcome various challenges throughout my PhD journey. His mentorship will continue to inspire me in my future career.

I am also grateful to my dissertation committee members: Prof. Jing Jiang, Prof. Yuan Fang, and Prof. Aixin Sun. I sincerely appreciate the time and effort they have devoted to providing me with valuable insights to improve this dissertation.

This endeavor would not have been possible without my former colleagues at the Living Analytics Research Centre. I had the pleasure of collaborating with Dr. Yang-Yin Lee, Dr. Meng-Fen Chiang, Amila Silva, Lee-Hsun Hsieh, Agus Trisnajaya Kwee, and Hsien-Hao Chen, who offered unwavering support and shared their experiences and knowledge. I would also like to acknowledge the technical assistance rendered by the system administration team members: Soon Keat Fong, Desmond Yap, and Adrian Mendoza Alfonso.

Furthermore, I would like to extend my sincere thanks to Chew Hong Ong, Chui Ngoh Boo, Caroline Tan, and Pei Huan Seow for their assistance with administrative matters.

Finally, I would be remiss in not mentioning my parents, my sister, and friends. Your tremendous support and love throughout my PhD candidature played a crucial role in both of my study and personal well-being. I hope I make you proud. Special thanks to Lynn Chan for her companionship and comforting presence during my final year of study. I would also like to express my deep gratitude to my landladies, Ms. Kim and Ms. Ee, whose incredible kindness and warm hospitality made me feel at home throughout this endeavor.

Publications

This dissertation covers the content of the following publications,

1. Lo, Pei-Chi, & Lim, Ee-Peng (2020). Co-Embedding Attributed Networks with External Knowledge. In 2020 IEEE International Conference on Big Data (Big Data). IEEE, December 2020.
2. Lo, Pei-Chi, & Lim, Ee-Peng (2023). Contextual Path Retrieval: A Contextual Entity Relation Embedding-based Approach. ACM Transactions on Information Systems (TOIS), Volume 41, Issue 1, January 2023, Article No.: 1, pp 1–38.
3. Lo, Pei-Chi, & Lim, Ee-Peng (2023) (*Accepted for Publication*). A Transformer Framework for Generating Context-Aware Knowledge Graph Paths. Applied Intelligence.
4. Lo, Pei-Chi, & Lim, Ee-Peng (2023) (*Accepted for Publication*). Non-Monotonic Generation of Knowledge Paths for Context Understanding. ACM Transactions on Management Information Systems.

Chapter 1

Introduction

1.1 Motivation

In recent years, information retrieval applications increasingly depend on knowledge graph to return results relevant to query context which mention entities found in the knowledge graph. The query context here can be a question, query terms, text document, or even a multimedia file. The results can also be in different forms, such as a selected set of documents from a text corpus, extracts from a knowledge graph, or even generated text or knowledge structures. To distinguish the information retrieval tasks underlying the above applications from the traditional text retrieval tasks, we call the former the *Knowledge-enriched Contextual Information Retrieval (KCIR) tasks*.

One example KCIR task is semantic search using knowledge graphs [124] where the input query context is a piece of text (e.g., “show me NLP papers that use dynamic programming algorithms to solve the word segmentation problem”) that can be associated with knowledge graph entities (e.g., “NLP”, “dynamic programming”, and “word segmentation”), and the search results are selected documents from a large corpus. Another example KCIR task is query answering using knowledge graphs [50, 139] where the context is a textual question (e.g., “who is the director of the Lord of the Ring movie?”) in which “Lord of the

Ring” is also an entity in the knowledge graph. The result of this question should be a director entity extracted from the knowledge graph. Relation extraction with knowledge graphs can also be seen as another KCIR task that returns the relation edges between entities mentioned in a query context in the form of an input text document. The returned relation edges may or may not be observed in the given knowledge graph.

While the different KCIR tasks have their own task definitions, they share the common characteristics of having the input query context enriched by the knowledge graph and using the query context representation to perform retrieval and prediction. The end goal is to use the knowledge graph and query context to look for information, or to return parts of the knowledge graph relevant to the query context.

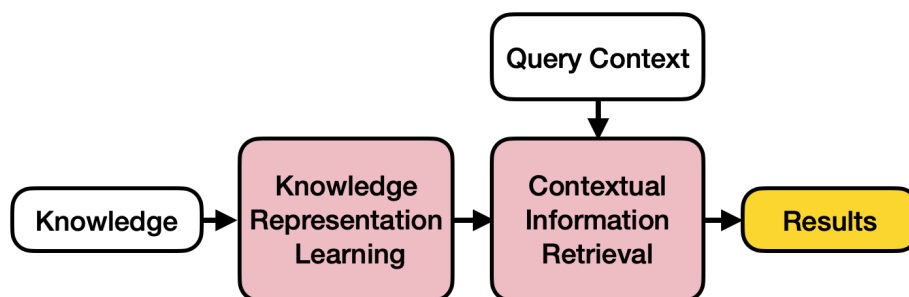


Figure 1.1: Meta-Framework of KCIR Tasks

Figure 1.1 depicts the meta-framework for conducting research on KCIR tasks. It shows the knowledge graph and query context as input of KCIR. The first component of this meta-framework is *Knowledge Representation Learning*, which takes in knowledge in the form of knowledge graph and derives the vector representations of knowledge graph entities and/or relation edges. The second component, *Contextual Information Retrieval*, takes in a query context, in the form of text or a set of knowledge graph entities, and the knowledge graph representations as input, then returns the result matching the query context. The knowledge graph may have entities connected by (i) relation edges constructed for specific domain, or (ii) simple associations based on co-occurrences or some similarity measures between entities. To leverage on the background knowl-

edge, the representation of query context should be enriched with the knowledge graph’s semantics. The contextual information retrieval component can be defined to return knowledge graph substructures as results or documents as results. In this dissertation, we will focus on the former.

1.2 Research Objectives and Framework

In this dissertation, we adopt the meta-framework in Figure 1.1 to categorize our research works.

- Under the knowledge representation learning component, we study the knowledge representation learning methods for *attributed networks* which can be viewed as a special form of knowledge graph. In an attributed network, every entity has a textual attribute. Our research thus focuses on developing new representation learning methods for the knowledge entities incorporating associations between entities and words in the attributed network.
- We then study a KCIR task, also known as *Contextual Path Retrieval (CPR)*, which takes an input query context consisting of two entities and a textual document so as to retrieve the path in the knowledge graph that explains the semantic connection between the two query entities within the document. In this work, we focus on the contextual information retrieval component of the meta-framework which encompasses the representation of query context and retrieval of relevant knowledge path, also known as the *contextual path*.
- The next KCIR task, known as *Contextual Path Generation (CPG)*, is similar to contextual path retrieval except that the path returned can include relation edges missing in the given knowledge graph.

In our knowledge representation learning work, we focus on developing a

novel variational autoencoder (VAE) architecture to combine entity and word attributes with data from external knowledge sources in the representation learning method. The research poses several challenges, including the extraction of the attributes' semantics from external knowledge sources, and to incorporate these different attribute semantics into both entity node and attribute embeddings within the same encoder. As the entity's attribute text contains both words and entity mentions, integrating knowledge about them with network links among context documents is thus important design consideration in the VAE architecture.

The remaining two works, CPR and CPG, are novel and useful in many new applications. They can be seen as a generalization of relation extraction which returns only one relation edge to connect a pair of entities mentioned in an query document context. CPR and CPG returns a path covering a set of relation edges connecting a pair of query entities mentioned in a query document context. The returned path explains the connections between the two query entities with the aid of knowledge graph semantics.

For CPR, the key technical challenges include capturing semantics of the input query context in the contextual representation, selecting candidate paths, and matching the contextual representation against the candidate paths in order to rank the latter. In the case of CPG, input query context representation learning remains to be a key challenge. It also faces the common challenge of identifying the relevant part of context for selecting or generating the correct relation edges for the resultant paths. Another major challenge is to generate the resultant paths between the two query entities using observed and newly inferred relation edges in the knowledge graph such that they are relevant to the input query context where the query entities are mentioned.

1.3 Summary of Dissertation Contributions

In this dissertation, we summarize our technical contributions as follows.

- For the first work, we study a network of entities with textual attributes and they are connected with one another by semantic edges. We propose a co-embedding attributed network (ECAN) structure that effectively co-embeds both entity and word attributes of such attributed networks in the same vector space. We demonstrate that our proposed co-embedding network model can effectively learn the representations of inter-linked entities if they are to be used as a form of knowledge graph.
- For the second work on contextual path retrieval, we propose several models following our proposed solution framework. Each model consists of one encoder for learning the representation of paths and another encoder for learning the representation of query context (query document text and its mentioned entities). We also propose different result ranking methods to order the retrieved paths according to their semantic similarities against the query context.
- For the third work on contextual path generation, we identify two different approaches to generate contextual paths, namely, monotonic and non-monotonic CPG. We propose for monotonic CPG a transformer-based encoder-decoder structure. The encoder jointly embeds the query document and entities in the query context, while the decoder generates the contextual path. To ensure that the generated paths can effectively connect the two query entities with minimal redundancy, we propose two scaling methods: penalization scaling and reward scaling. The former ensures that the generated path do not have loops, and the latter facilitates the model to generate complete paths that start and end with query entities.
- Under the CPG work, we further propose non-monotonic CPG models to

always generate a contextual path that connects the two query entities. Our proposed idea is to pre-train a path generation model to generate knowledge graph paths by unsupervised learning on a huge training path dataset. We then fine-tune this pre-trained model with query entities extracted from the query context document.

As we develop models to address our research problems, we also show how one could combine knowledge representation learning and contextual information retrieval components of the meta-framework. This hopefully will lead to more research on combining the two components to address new KCIR tasks. We also contribute to research on CPR and CPG by specially constructed real world datasets as well as algorithms to construct synthetic datasets. We establish a common set of performance metrics for evaluating the returned contextual paths against the ground truths. These datasets and evaluation metrics will also facilitate comparison against future research works as the KCIR research topic continues to grow.

1.4 Dissertation Structure

The rest of this dissertation is organized as follows. We present a survey of related work in Chapter 2. Chapter 3 describes the how we learn an attributed network embedding model that incorporates useful external knowledge. Chapter 4 describes our works on the CPR task. Chapter 5 introduces the CPG task and our proposed CPG methods which return the knowledge path monotonically. Chapter 6 addresses CPG task with a non-monotonic generation approach that generates well-formed knowledge paths as results. Finally, we conclude our research, and discuss future works in Chapter 7.

Chapter 2

Related Work

In this chapter, we survey previous works that are related to this dissertation research. We group them under: (a) text and knowledge graph representation learning, (b) contextual retrieval of knowledge, (c) contextual retrieval of text documents, (d) contextual item recommendation, and (e) contextual question answering. Topic (a) is related to the knowledge representation learning component of our KCIR meta-framework. Other than works addressing the representations of entities and relation edges of knowledge graphs, there are also works that cover the representation of textual descriptions of knowledge graph elements. Topics (b) to (e) cover different knowledge-enriched contextual information retrieval tasks. Among them, Topic (b) is most related to our proposed KCIR tasks: CPR and CPG.

2.1 Text and Knowledge Graph Representation Learning

As knowledge representation learning can be decoupled from the contextual information retrieval, we can adopt an unsupervised learning approach to encode a knowledge graph and its associated text data. In the following, we shall give a brief survey of the existing text and knowledge representation learning tech-

niques.

Text representation learning. The classical text representation has been based on one-hot encoding or TF-IDF vectorization which suffers from sparsity. With recent breakthrough in text representation learning, many dense word embeddings methods (e.g., GloVe [85], word2vec [80], and doc2vec [65]) have been developed to overcome the sparsity of terms. These works learn the word and document representations by ensuring that similar words will have similar word representations and the same for documents representations.

In recent years, many state-of-the-art text representation learning works have been introduced to derive representations using transformer structure [107]. For example, BERT utilizes multiple layers of transformer encoders that are trained to generate contextual word representations using large number of text sentences [29]. Different word representations are can be assigned to the same word when it appears in different sentences or different parts of a sentence.

Knowledge graph representation learning. There are several works on knowledge representation learning that lead to different methods. Given a relation edge (e_h, r, e_t) in a knowledge graph where e_h and e_t are entities, and r is a relation label, *Translation-based methods* such as TransE [12], TransR [72], and TransH [118] learn entity and relation embeddings z_* 's that satisfy the translation association $z_{e_h} + z_r \rightarrow z_{e_t}$. On the other hand, *multiplication-based methods* such as DISTMult [130] and ComplEx [105] represent a relation as a matrix W_r such that $z_{e_h}^T W_r z_{e_t} = 1$.

Hybrid text and knowledge graph representation learning. A hybrid representation learning method jointly learns the word and entity representations using a common vector space. For example, *Wikipedia2vec* jointly learns the representations of entities and words in a common vector space using skip-gram model [128]. There are works that extend contextualized word representation learning to include contextualized entity representation learning. For example, KG-BERT augments BERT with knowledge graph by fine-tuning a pre-trained

BERT with triplet identification or relation prediction loss functions [134]. Kbert [74] and KEPLER [117] further incorporates interaction between entity and word into a knowledge graph and text documents mentioning its entities. Kbert injects relation triplets extracted from the knowledge graph into a BERT-transformer to encode both words and entities in the context. KEPLER jointly optimizes knowledge graph loss (e.g., TransE) and masked language model loss to incorporate knowledge into contextualized word representations. The entity representation is then obtained by encoding the description from its Wikipedia page.

Knowledge Graph Completion. Knowledge graph completion task is a popular downstream task to evaluate knowledge graph representation methods. It is a prediction task that determines the tail entities (e.g., politicians) associated with a given head entity (e.g., USA) by a given relation label (e.g., president of). One major stream of research is to project the knowledge graph entities in a vector space such that the alignment between a pair of entities reflects a certain relation [52, 72, 95, 104]. Some knowledge graph completion works focus on modelling complex relations using reinforcement learning. For instance, Lin et al. model multi-hop knowledge graph relation with specially designed reward function that incorporates compositionality (i.e., $e_h + r \rightarrow e_t$) into the learned entity and relation embeddings [71].

Knowledge graph completion differs from CPR and CPG in two aspects: (a) it does not involve any input query context, (b) it returns inferred relation edges instead of paths, and (c) the relation label for the relation edges to be predicted is part of input while CPR and CPG do not require such input relation labels.

Relation/Link Prediction. Relation prediction or link prediction is another downstream task which utilizes knowledge graph representations [12]. It augments the knowledge graph with predicted relation edges in the knowledge graph without any given head entity and relation label. Relation/link prediction can be seen as a special type of knowledge graph reasoning that predicts missing

relation edges [28, 82, 105, 130]. Some important works in this line of research include translation-based methods which focus on representing relation as translation operation [12, 105, 130], and others which use CNN on the knowledge graph structure to predict whether a relation edge exists between a pair of entities [28, 82]. Compared with CPR and CPG, Relation/link prediction has two major differences: (1) it does not involve any input query context, and (2) it infers new relation edges using existing relation edges instead of a multi-edge paths between the two input entities.

2.2 Contextual Retrieval of Knowledge

In this section, we review previous studies on contextual information retrieval. Specifically, we focus on two streams of research that retrieve entities or relation edges from knowledge graphs using the semantics found in the query context.

Contextual Knowledge Graph Completion. This type of tasks extends the knowledge graph completion task, which we discuss in previous section, to include query context as input [122, 134]. For instance, West et al. first learn a pre-trained knowledge graph embedding to solve question answering task. They then augment the knowledge graph embedding with new relations that are inferred from the question [120]. Malaviya et al. propose to complete a commonsense knowledge graph with local graph structure as well as an input query text [76]. They infer the structural knowledge of ConceptNet using graph convolutional networks, and utilize BERT to model the semantic context of target entity nodes. They conclude that as BERT is effective in capturing taxonomic relations, the input query text provided by it significant boosts the performance of knowledge graph completion. While contextual knowledge graph completion shares many similarities with our CPG task, they differ in the following ways: (1) Knowledge graph completion focuses on inferring new relations to augment the knowledge graph, while CPG aims to retrieve existing knowledge graph rela-

tions and infer new relations at the same time. (2) Knowledge graph completion tasks predict whether a one-hop relation can be inferred from the query context, while CPG generates a multi-hop knowledge graph path given the query context.

Relation extraction. Relation extraction (RE) aims to extract relation edges from input query text. The task returns relation edges involving multiple entities mentioned in the given text [36, 108, 137]. Unlike CPR and CPG, RE does not return knowledge paths as result. Usually, RE also does not rely on any given knowledge graph. In some recent RE works, researchers studied a “knowledge-enriched” relation extraction task which involves a corpus of background text and a knowledge graph to learn how entities are related to one another with known relations and entities. From the given background text corpus, this task automatically selects sentences for relation extraction [51].

2.3 Contextual Retrieval of Text Documents

Unlike knowledge retrieval, contextual retrieval of text documents is another contextual information retrieval task which focuses on retrieving text documents from a given text corpus. This type of tasks retrieves text documents with a set of query entities and relations from a knowledge graph.

Passage retrieval. This task is defined to select relevant text passages to explain some given parts of knowledge graph or answers involving knowledge graph. For example, *entity relation explanation* answers why a relation edge exists in a knowledge graph by retrieving the relevant text passages [2, 9, 87, 109, 110].

Entity support passage retrieval. This task aims to return supporting passages to explain of why an given entity is connected to a query [10, 20, 21, 55].

Both the above passage retrieval tasks are very different from CPR and CPG as their results are text passages, not contextual paths. They also do not infer new relation edges for the given knowledge graph.

2.4 Contextual Item Recommendation

Given a knowledge graph and past user-item interactions as context, knowledge graph-enriched contextual recommendation predicts items a user may like, and provides explanation in the form of reasoning structure extracted from the given knowledge graph [24, 71, 116, 123, 125, 138].

In this section, we focus on **path-based recommendation methods**. These recommendation methods provide intuitive reasoning that is easy to comprehend by human using connectivity patterns or meta-paths inferred from previous user-item interaction history [22, 41, 45, 136, 141]. Some works also integrate both user-item interaction and item knowledge graph to generate explainable recommended items. For example, KPRN appends the user-item interaction to an item knowledge graph [116] and utilizes the paths between a user and an item in the “appended” 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 knowledge graph as an auxiliary data to enhance the user-item interaction modeling [17]. It augments the user-item data with knowledge graph relations, to explain why two items have interaction with the same user. Finally, PGPR performs a causal inference procedure using reinforcement learning to provide explanations [123]. Each recommendation is based on a set of knowledge sub-graphs.

Contextual recommendation works cannot be directly used to solve CPR nor CPG as the prediction targets of the two problems are vastly differ. CPR and CPG seek to identify a 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.

2.5 Contextual Question Answering

Contextual question answering (QA) is an information retrieval task that determines for a given question the important entities denoted by E_q , and traverses the given knowledge graph from E_q to find possible answers. The question itself is the context. There have been works on reasoning with knowledge graphs based on logic rules [6, 62], bayesian models [111], distributed representations [89, 103], neural networks [4, 26, 68, 70], and reinforcement learning [25, 37, 67, 73]. Here, we focus on those QA works which aim to find answers from knowledge graphs, text, or the combination of the two.

Knowledge graph-based question answering. There are several methods to perform multi-relation QA which reasons over multiple facts in the knowledge graph so as to determine the answers. [126, 135, 143]. For example, MetaQA traverses through a knowledge graph to retrieve the answer to a given question [139]. Although the traversed path could be seen as an explanation of answer's connection with the question entities, but it is not returned along with the answer, nor evaluated against the ground truth explanation path (which does not exist for MetaQA). Here, the various paths connecting question entities to answer are turned into answer features for relevance ranking [69, 113]. Moreover, the traversed paths are limited to involve the observed relation edges (not inferred relation edges) of the knowledge graph. MHQA-GRN addresses reading comprehension QA by constructing a directed acyclic graph from knowledge graph for each passage in the query article and using its representation as features to predict the answer [98]. It then summarizes the answers from all passages and aggregates the final answer. Nevertheless, MHQA-GRN does not generate a reasoning path for the query. Some works adopt reinforcement learning and propose to use policy-based agents to generate the most probable paths on a knowledge graph so as to determine the answers [71, 125].

Text-based question answering. Text-based QA reasons over multiple doc-

uments instead of knowledge graph to generate answers [88, 100, 119]. Finding semantic connections between entities has been studied in QA research to correctly determine an answer (usually an entity) with an explanation of how it is connected to the question. For example, HotpotQA conducts multi-hop reasoning over a given set of documents to find answers to questions [132]. Asai et al. proposed another retriever-reader framework on HotpotQA dataset that reasons over Wikipedia using graphical structure [5]. HotpotQA however does not involve a knowledge graph and hence is unaware of the types of entities and relations to construct the explanation. Instead, it relies on information from a set of input documents [64].

Hybrid question answering. Finally, we review QA works that are based on both text and knowledge graph. Collaborative Policy Learning (CPL) [34] uses two different agents to extract facts from a bag of noisy sentences which are turned into new relation edges for augmenting the given knowledge graph. A collaborative mechanism between the two agents is then learned using reinforcement learning. Unlike CPR and CPG, the knowledge graph reasoning works are not given the tail entity during the reasoning process. Ren et al. proposes the LEGO framework which alternates between expansion of query tree and reasoning in latent space to find entities which satisfy query tree conditions [92]. Starting from the initial set of entities in the question, LEGO determines a relevant relation (if any) from the question and add it to the query tree. It then finds entities from the latent space which satisfies the relation(s) in the query tree. The process is repeated until no further changes can be made to the query tree. Finally, a set of entities satisfying the query will be returned.

Recent studies in knowledge reasoning includes C-RNN [145], which combines a convolutional neural network for feature extraction with a recurrent neural network to predict the answer entity given a query entity and a knowledge graph which covers different paths from the query entity to candidate answer entities. For the same reasoning task, CogKR utilizes a framework inspired by

human cognition process to conduct multi-hop reasoning [31]. Specifically, it iterates between an expansion module and a reasoning module to predict the answer entity for a given query entity using a neighborhood knowledge subgraph. Nevertheless, none of the above models considers context when predicting the knowledge paths leading to the answer entities.

On the whole, all of the KGQA works we discuss above do not generate knowledge graph paths as result. The QA tasks are different from CPR and CPG 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 answers, which is similar to that of contextual item recommendation.

Chapter 3

Co-Embedding Attributed Network Representation Learning

In this chapter, we dive into the contextual representation learning component in our meta-framework. The goal of the contextual representation learning is to jointly embed the information from both text and knowledge, including the interaction between them so as to generate the embedding representations of objects for downstream information retrieval tasks. Here, we consider node objects of an attributed network which have words and entities as attribute values. This chapter thus proposes a new contextual representation models for this attributed network by incorporating the co-occurrence based association graphs of these entities and words.

Network embedding has been an important research topic in recent years due to the existence of massive amount of web and social network data as well as many applications on these networks, e.g., link prediction, recommendation, node classification, etc.. Many of these networks are attributed networks, a type of networks consisting of both nodes and their attributes. For example, in a citation network, nodes are publications and node attributes include paper title, author, and abstract. The node attributes themselves may also be semantically associated with one another. E.g., two keywords (e.g., “AI” and “deep learning”)

in abstract can be semantically related if one has the background research knowledge. Conventional network embedding techniques (e.g., deepwalk, node2vec) usually leave out node attributes in learning the node embeddings and this results in sub-optimal performance in the downstream prediction tasks. There are several efforts to incorporate node attributes into network embeddings such as [49] and [78] but they have not considered external attribute semantics in their models.

Knowledge covered in this chapter are in the form of an entity-entity knowledge graph or a word-word co-occurrence knowledge graph. These knowledge provide additional semantics of the nodes and attributes in the network. Such attribute semantics can be found in publicly available external knowledge sources. Examples of such external knowledge sources include Wikipedia, text corpuses, WordNet, etc.. From those sources, we can derive attribute semantics that come in the form of associations between words and entities for learning better embedding representations of the network nodes.

3.1 Research Objective

Consider an attributed network example with publications as nodes where v_1 and v_2 are two publication nodes with titles, “Mining Authority Nodes from Hyperlinks in WWW” and “The PageRank Citation Ranking: Bringing Order to the Web”. The two publications share similar topic but they have little overlap between their titles. We consider both entities and words in these publication titles. With external knowledge, we can determine that the entities that “Web” and “WWW” in the titles are highly similar, and another two entities “Authority” and “Pagerank” are similar. At the *word* level, “hyperlinks” and “web” are semantically similar. These semantic knowledge should bring the embeddings of v_1 and v_2 closer to each other.

Here, we specifically study this for the variational autoencoder-based (VAE-

Table 3.1: Summary of Notations

Symbol	Description
$\mathcal{G} = \{\mathcal{V}, \mathcal{A}, \mathcal{E}\}$	Undirected attributed network
$\mathcal{E}^{\mathcal{V}} / \mathcal{E}^{\mathcal{A}}$	Edges between nodes / between nodes and attributes
$N = \mathcal{V} $	Number of nodes
$F = \mathcal{A} $	Number of attributes
i / a	Index of nodes / attributes
D	Dimension size of latent representation vector
$\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{N \times D}$	Latent representation of nodes
$\mathbf{Z}^{\mathcal{A}} \in \mathbb{R}^{F \times D}$	Latent representation of attributes
$\mathbf{A} \in \mathbb{R}^{N \times N}$	Adjacency matrix of nodes
$\mathbf{X} \in \mathbb{R}^{N \times F}$	Attribute information matrix of nodes
$\mathbf{S} \in \mathbb{R}^{F \times F}$	Attribute association matrix
p_{a_1, a_2}	Proximity between two attributes a_1 and a_2
$\hat{A} / \hat{X} / \hat{S}$	Reconstructed matrix

based) models which has seen much development in recent research. One has to overcome a few technical challenges in considering attribute knowledge, namely: (a) extracting attribute semantics from external knowledge sources; (b) designing new VAE architecture(s) to incorporate different attribute semantics into both node and attribute embeddings; and (c) evaluation of the proposed VAE.

3.2 Related Work

3.2.1 HIN Embeddings

Heterogeneous Information Network (HINs) refers to networks with multiple types of nodes. HIN embeddings aim to learn low-dimensional representations of nodes that preserve the network structure. Although homogeneous network embedding approaches such as node2vec and deepwalk [39, 86], HIN embeddings considers the distinctive semantics of nodes. The early HIN embeddings works focus on how to fairly sample different types of nodes or combining the semantics of node types. Tang et al. proposed LINE that traverses all edge types, then samples one edge for each edge type at a time [101]. The more advanced

HIN embeddings consider meta-paths [30]. Aspem, as an HIN embedding technique of using meta-paths, defines aspect to be a sub-graph consisting of only some of the node types [96]. More recently, deep learning based HIN embeddings such as HNE and SDNE have been proposed [19, 112].

Beyond network structures, one can improve the quality of node embeddings by considering node attributes, node labels and text content associated with the nodes [48, 49, 114, 131]. For example, TriDNR considers both node content and node label in the learning of node embeddings [83]. It jointly optimizes a skip-gram model for network structure, and a coupled neural network model which maps the node content and node representation from skip-gram to the node label. ANRL utilizes an auto-encoder structure to learn node embedding that preserves both the network structure and node attribute proximity [140]. Nevertheless, even when auxiliary node information (e.g., node label and attributes) are modeled by the above embedding approaches, they have not considered: (1) the uncertainty of node representation, and (2) attribute representations. These two properties are important as the former avoids over-fitting and enables generative process, while the latter provides a common representation to compare nodes and attributes. In this paper, we utilize variational auto-encoder to address these shortcomings.

3.2.2 Variational Auto-encoder Embeddings

Variational auto-encoder (VAEs) consists of two computational neural networks (NN): the first NN f_ϕ works as an inference model that learns latent variables from observations, and the second NN g_θ works as a generative model that attempts to reconstruct the original observations from the latent variables [58, 61, 97]. Unlike other approaches such as node2vec or DeepWalk, the latent variables of VAEs consist of mean and variance of some Gaussian distribution to capture the uncertainty of each latent variable. This uncertainty could be very helpful in modeling heterogeneous information networks as properties of differ-

ent node types could be captured. For example, a popular attribute label that is shared among many nodes should have larger variance so all associated nodes could have generally low proximity to it.

VAE is a popular representation learning for text/image generation [38, 61, 129]. Previous works have shown that VAE embeddings outperforms several other traditional network embedding approaches [61]. Semi-supervised VAEs that incorporate partial label information [59, 79, 127], and VAEs that support interpretability using disentangled representation [13, 32] are among the works showing that VAE can be easily extended for different problem settings. The learned VAE representations are often used in node classification or network completion tasks [47, 53, 54, 66]. For example, there are works that aim to learn multi-modal network embedding using VAE as it can capture the correlation between different data modalities [47, 53, 66].

To co-embed the nodes and the auxiliary attributes in the same representation space, NetVAE employs a shared encoder to learn a unified representation for a nodes and its attribute, and separate decoders to decode them separately [54]. Another work that utilizes VAE to learn embeddings for nodes and attributes is CAN. It uses a double-VAE structure to jointly learn node and attribute representations [78]. Both NetVAE and CAN, however, do not consider external attribute semantics to improve node and attribute representations.

We compare six selected network embedding works in Table 3.2. VGAE is a baseline model that learn the embedding based solely on node adjacency matrix, whereas the others utilize attribute or label association in the learning process. While TriDNR and ANRL learn the node representations with the help of attributes, the attribute representations are not included in the learned embedding models. On the contrary, CAN and NetVAE model both node and attribute representations at the same time. Here, we aim to propose a model, ECAN, which further considers attribute association and attribute semantics from external knowledge sources.

Table 3.2: Summary of Related Work

Model	Method	Modeling of A	Encoding of				
			N Adj	N-L Assoc.	N-A Assoc.	A Adj	Ext. A Semantics
TriDNR [83]	Skip-gram	×	✓	✓	✓	×	×
ANRL [140]	Autoencoder	×	✓	×	✓	×	×
VGAE [61]	VAE	×	✓	×	×	×	×
CAN [78]	VAE	✓	✓	×	✓	×	×
NetVAE [54]	VAE	✓	✓	×	✓	×	×
ECAN (Our)	VAE	✓	✓	×	✓	✓	✓

N: Node. A: Attribute. L: Label.

3.3 Co-Embedding with External Knowledge

3.3.1 Framework Overview

To provide a common embedding solution approach, we propose a **Co-embedding Attributed Networks with External Knowledge (CANE)** framework. As a framework, CANE allows different methods to be used in each step to realise a co-embedding attributed network model. As illustrated in Figure 3.1, the CANE framework takes an attributed network \mathcal{G} as input where each node is linked to one or more attributes. From \mathcal{G} , we extract useful information that serves as input to the learning of co-embedding model. The $N \times N$ adjacency matrix \mathbf{A} , and the $N \times F$ node attribute matrix \mathbf{X} are derived from \mathcal{G} directly, while attribute association matrix \mathbf{S} is extracted from some external text corpus or knowledge graph.

Our proposed extraction of attribute associations from text corpora and knowledge graphs will be described in Section 3.3.2. Here, we assume that the external knowledge sources should be relevant to the given attributed network. With the derived attribute-to-attribute associations, the framework constructs the input matrices and learns the network and attribute embeddings as outlined in Sec-

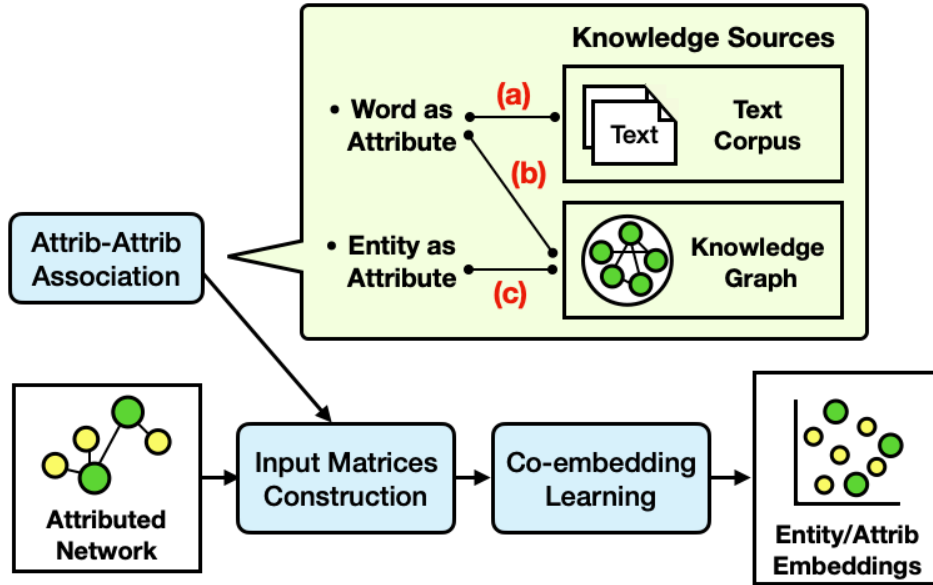


Figure 3.1: Co-embedding Attributed Networks with External Knowledge

tions 3.3.3 and 3.3.4. A complete summary of symbols used is given in Table 3.1.

3.3.2 Attribute-to-Attribute Association

Attribute Definition. The definition of attribute is important in our proposed framework as it determines how we extract attribute-to-attribute associations. Instead of defining attribute at the type-level, where different type of nodes are treated separately, our framework adopts an instance-level attribute definition. In other words, a node attribute is a concatenation of attribute type and attribute label(s). Using citation network as an example, a paper node can have (author, “John Smith”) as an attribute, (keyword, “search engine”), and (keyword, “data mining”) as two other attributes.

Attributes can be of many types, and should be handled differently to derive attribute-to-attribute associations. In this study, we categorize attributes into 4 main types: (a) numeric, (b) categorical, (c) set, and (d) multiset. The attribute-to-attribute associations of numeric attributes (e.g., age, salary, etc.) can be directly derived from their values. Hence, we focus on non-numeric attribute types in the following discussion.

Categorical attribute is a singleton set, and set is a special case of multiset.

Without loss of generality, we only consider multiset attribute, and a node with multiset attribute is mapped to node-attribute associations, one for each value of the multiset. For example, a paper node v_i with this sentence in the abstract, “Social networks have long tails.”, can be represented as a set of associations between v_i and each of (abstract, “social”), (abstract, “networks”), \dots , (abstract, “tails”). In the following, we present two kinds of attribute values: (a) word, and (b) entity.

Word as Attribute. For attributes that are textual, it is intuitive to use words as attribute values. As not all words are important, one can keep only the informative words and use them as attribute values. For example, we may select only words with high TF-IDF values.

Entity as Attribute. When the attribute has a category label or multiset of labels, each label can be viewed as an entity. Even when the attribute is textual, it is possible to extract entities from the attribute text and generate entities as attribute values. Such an approach will be illustrated in our experiment datasets (see Section 3.5.1).

With word and entity attributes defined, we now define measures for the association between two attributes. In this framework, we consider two types of external knowledge sources, (a) text corpus, and (b) knowledge graph.

(a) Text Corpus as External Knowledge. A text corpus consists of many text documents. The word-word associations can be derived from word co-occurrences, say, the number of time one word appears with another word in the same context. This is the same intuition behind popular word embedding approaches such as Skip-gram.

(b) Knowledge Graph as External Knowledge. Knowledge graphs are networks of entities connected by one or more semantic relation. From a knowledge graph, we could measure the association of two entities using the graph structure, the association between two words using their co-occurrences in knowledge graph’s text data, and the association between a pair of entity and word

using their co-occurrence. For instance, Milne and Witten propose to measure the similarity of two entities by their in-link edges originating from same entities [121]. If two entities share very similar set of in-links, they are highly associated. Knowledge graphs, in many cases, have text data describing their entities. For example, Wikipedia has one article for each entity, and these entity articles can serve as a text corpus to derive entity-to-entity associations.

3.3.3 Input Matrices Construction

The following are matrices to be constructed from network and associations from external knowledge sources.

Node-to-Node Association Matrix (A). The Node-to-Node Association Matrix captures the network structure of the attributed network using an $N \times N$ adjacency matrix. In the case of unweighted network, $\mathbf{A}_{ij} = 1$ if $(\mathcal{V}_i, \mathcal{V}_j) \in \mathcal{E}^{\mathcal{V}}$, and 0 otherwise. If the attributed network is weighted, \mathbf{A}_{ij} will simply be the weight of edge.

Node-to-Attribute Association Matrix (X). The Node-to-Attribute Association Matrix is constructed for an unweighted attributed network by setting $\mathbf{X}_{ia} = 1$ if $(\mathcal{V}_i, \mathcal{A}_a) \in \mathcal{E}^{\mathcal{A}}$, and 0 otherwise.

Attribute-to-Attribute Association Matrix (S). There are three possible Attribute-to-Attribute Association Matrices, denoted by $\mathbf{S}^{(a)}$, $\mathbf{S}^{(b)}$, and $\mathbf{S}^{(c)}$ as shown in Figure 3.1.

(a) Word-to-word associations learned from text corpus ($\mathbf{S}^{(a)}$). In this setting, we extract contextual closeness between two attributes which are words from a text corpus \mathcal{D} . The text corpus could be relevant to the attributed network (e.g., the abstracts of the papers in a citation attributed network) or a text collection that provides meaningful word co-occurrence information. We derive the contextual closeness from distance between the two words a_i and a_j within a

context window. For a window size w in a document d , the contextual closeness of one occurrence between a_i and a_j is $s_{ijd}^{\text{cx}} = w - \mathbf{d}_{ij}$ where \mathbf{d}_{ij} denotes the distance between a_i and a_j . This closeness score is symmetric. For example, given a sentence “*Discriminative learning for differing training and test distributions*” and a window size 4, the contextual closeness between the words “learning” and “training” in this context window is $4 - 2 = 2$. The closeness between “learning” and “distribution” would be 0 as “distribution” does not appear within “learning”’s context window. For all pairs of words found in the attributed-network, we construct a $F \times F$ matrix $\mathbf{S}^{(a)}$ as $\mathbf{S}_{ij}^{(a)} = \sum_{d \in \mathcal{D}} s_{ijd}^{\text{cx}}$ for all pairs of words a_i and a_j .

(b) Entity-to-entity associations learned from text corpus of knowledge graph ($\mathbf{S}^{(b)}$). Similar to (a), from the text corpus \mathcal{D} that covers entities in a knowledge graph (e.g., Wikipedia pages), we construct a $F \times F$ matrix $\mathbf{S}^{(b)}$ using context closeness score s_{ij}^{cx} between two entities a_i and a_j . $\mathbf{S}_{ij}^{(b)} = \sum_{d \in \mathcal{D}} s_{ijd}^{\text{cx}}$

(c) Entity-to-entity associations learned from knowledge graph ($\mathbf{S}^{(c)}$). The corresponding $F \times F$ matrix $\mathbf{S}^{(c)}$ matrix has each entry measuring the relatedness between two entities. In this work, we define two entity relatedness functions giving rise to two different matrices:

(c.1) *In-link similarity from knowledge graph:* We adopt the Wikipedia Link-based Measurement (WLM) proposed in [121] to measure entity relatedness. Entities sharing similar in-link neighbors will have higher similarity, $\mathbf{S}_{ij}^{(c1)}$. WLM between two entities a_i and a_j is defined as follows,

$$\mathbf{S}_{ij}^{(c1)} = 1 - \frac{\log \max(|\mathcal{C}_{a_i}|, |\mathcal{C}_{a_j}|) - \log (|\mathcal{C}_{a_i} \cap \mathcal{C}_{a_j}|)}{\log |\mathcal{E}| - \log \min (|\mathcal{C}_{a_i}|, |\mathcal{C}_{a_j}|)}$$

where \mathcal{E} is the set of entities (attributes) in the attributed network, \mathcal{C}_{a_i} is the set of entities having a link to a_i in the knowledge graph.

(c.2) *Cosine similarity from pre-trained knowledge graph embeddings:* This leverages on knowledge graph embedding methods such as Wikipedia2vec [128] to determine entity relatedness. Such embedding methods align entities and

words in the knowledge graph in the same vector space. The association between two entities is this the relatedness between their vector representation. With a pre-trained knowledge graph embedding, we therefore define the association between entities a_i and a_j as $\mathbf{S}_{ij}^{(c2)} = \cos(\mathbf{a}_i^{\text{kg}}, \mathbf{a}_j^{\text{kg}})$. where \mathbf{a}_i^{kg} denotes the vector representation of entity a_i in the pre-trained knowledge graph embeddings, and $\cos(\cdot)$ is the cosine similarity.

3.3.4 Co-Embedding Learning

The last step of our proposed framework is co-embedding learning. The co-embedding algorithm learns low-dimensional representation of nodes and attributes from the three aforementioned association matrices. After the embedding model is learned, we then use it in the subsequent downstream tasks. Our framework can accommodate any embedding learning algorithm that utilizes the same set of input matrices (i.e., node-node matrix, node-attribute matrix and attribute-attribute matrix). In the remaining sections, we will use variational auto-encoders (VAE) as the embedding algorithm as illustrated in Section 3.4.

3.4 Co-embedding Learning Using VAE

In this section, we introduce **ECAN (External Knowledge Aware Co-Embedding Attributed Network) model**, a multi-VAE structure that jointly model both nodes and attributes in the same semantic space. We further borrow the idea of mixed model to address the problem of imbalanced dimensions in multi-input VAEs [47].

3.4.1 CAN

Before we present our proposed ECAN, we introduce CAN, a multi-VAE model proposed in [78]. CAN however does not consider external knowledge in learning the embeddings, nor does it seeks to balance the dimension size of node and

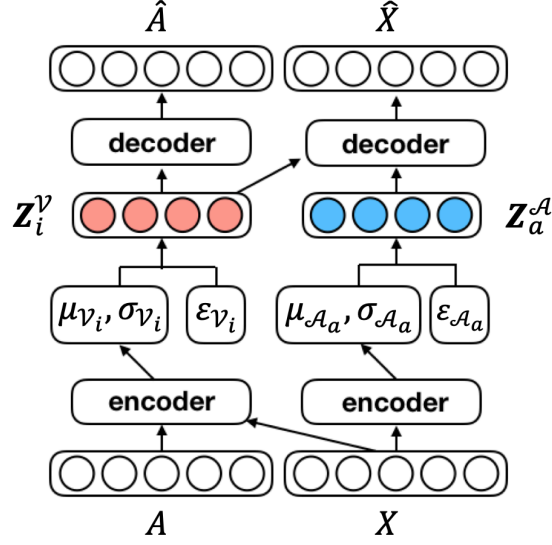


Figure 3.2: CAN Model

attribute representations. As shown in Figure 3.2, the first VAE of CAN encodes nodes using both \mathbf{A} and \mathbf{X} in a two-layer GCN [60]. The two-layer GCN is defined as:

$$\begin{aligned}
 H_{\mathcal{V}}^{(1)} &= \text{ReLU} \left(\tilde{\mathbf{A}} \mathbf{X} \mathbf{W}_{\mathcal{V}}^{(0)} \right) \\
 [\mu_{\mathcal{V}}, \sigma_{\mathcal{V}}^2] &= \mathbf{A} H_{\mathcal{V}}^{(1)} \mathbf{W}_{\mathcal{V}}^{(1)}
 \end{aligned} \tag{3.1}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ is a symmetrically normalized adjacency matrix, $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ is \mathcal{G} 's degree matrix with diagonal elements containing node degrees and 0 for the remaining elements, and $\text{ReLU} = \max(0, \cdot)$ is a non-linear activation function. This VAE outputs the mean $\mu_{\mathcal{V}}$ and variance $\sigma_{\mathcal{V}}^2$ as the learned Gaussian embedding of nodes.

The encoding of attribute follows similar process except that it replaces the two-layer GCN by two fully-connected layers. This encoder is defined as:

$$\begin{aligned}
 H_{\mathcal{A}}^{(1)} &= \tanh \left(\mathbf{X}^T \mathbf{W}_{\mathcal{A}}^{(0)} + b^{(0)} \right) \\
 [\mu_{\mathcal{A}}, \sigma_{\mathcal{A}}^2] &= H_{\mathcal{A}}^{(1)} \mathbf{W}_{\mathcal{A}}^{(1)} + b^{(1)}
 \end{aligned} \tag{3.2}$$

where b is the bias of fully-connected layer and \tanh is the non-linear activation function. Likewise, this VAE outputs the mean $\mu_{\mathcal{A}}$ and variance $\sigma_{\mathcal{A}}^2$ of the learned

Gaussian embedding of attributes. We denote the parameters in both encoders as $\phi = [\phi_1, \phi_2]$. $\phi_1 = [\mathbf{W}_{\mathcal{V}}^{(0)}, \mathbf{W}_{\mathcal{V}}^{(1)}]$ and $\phi_2 = [\mathbf{W}_{\mathcal{A}}^{(0)}, \mathbf{W}_{\mathcal{A}}^{(1)}, b^{(0)}, b^{(1)}]$ representing the parameters in the node and attribute encoders respectively.

The representation of a node i and an attribute a could then be derived from the latent variables:

$$\begin{aligned}\mathbf{Z}_i^{\mathcal{V}} &= \mu_{\mathcal{V}_i} + \sigma_{\mathcal{V}_i}^2 \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I) \\ \mathbf{Z}_a^{\mathcal{A}} &= \mu_{\mathcal{A}_a} + \sigma_{\mathcal{A}_a}^2 \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I)\end{aligned}\tag{3.3}$$

The generation model in CAN is very straightforward. To reconstruct an edge between two nodes i and j , we derive $\mu_{\mathcal{E}_{\mathcal{V}_i \mathcal{V}_j}}$ and $\sigma^2 \mathcal{E}_{\mathcal{V}_i \mathcal{V}_j}$ from g_{θ_1} ,

$$[\mu_{\mathcal{E}_{\mathcal{V}_i \mathcal{V}_j}, \sigma^2 \mathcal{E}_{\mathcal{V}_i \mathcal{V}_j}] = g_{\theta_1}(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}})$$

where $g_{\theta_1}(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}) = \text{sigmoid}(\mathbf{Z}_i^{\mathcal{V}T} \mathbf{Z}_j^{\mathcal{V}})$.

The edge between nodes i and j is then reconstructed by

$$p_{\theta_1}(\mathbf{A}_{ij} | \mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{V}}) = \mathcal{N}(\mu_{\mathcal{E}_{\mathcal{V}_i \mathcal{V}_j}, \sigma^2 \mathcal{E}_{\mathcal{V}_i \mathcal{V}_j} I)$$

where $\mathcal{N}(\cdot)$ is a Gaussian distribution. The edge generated by Gaussian carries a real-valued weight. In the case of binary edges, a Bernoulli distribution taking $\mu_{\mathcal{E}_{\mathcal{V}_i \mathcal{V}_j}}$ as input could be used for the reconstruction. The reconstruction of \mathbf{X}_{ia} follows the same procedure.

$$\begin{aligned}[\mu_{\mathcal{E}_{\mathcal{V}_i \mathcal{A}_a}, \sigma^2 \mathcal{E}_{\mathcal{V}_i \mathcal{A}_a}] &= g_{\theta_2}(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}) \\ g_{\theta_2}(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}) &= \text{sigmoid}(\mathbf{Z}_i^{\mathcal{V}T} \mathbf{Z}_a^{\mathcal{A}}) \\ p_{\theta_2}(\mathbf{X}_{ia} | \mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_a^{\mathcal{A}}) &= \mathcal{N}(\mu_{\mathcal{E}_{\mathcal{V}_i \mathcal{A}_a}, \sigma^2 \mathcal{E}_{\mathcal{V}_i \mathcal{A}_a} I)\end{aligned}\tag{3.4}$$

The loss function of VAEs aims to minimize (1) the reconstruction error and (2) KL-divergence between approximated and true latent variables. The reconstruction of \mathbf{A} and \mathbf{X} are covered in CAN, as well as KL-divergence between

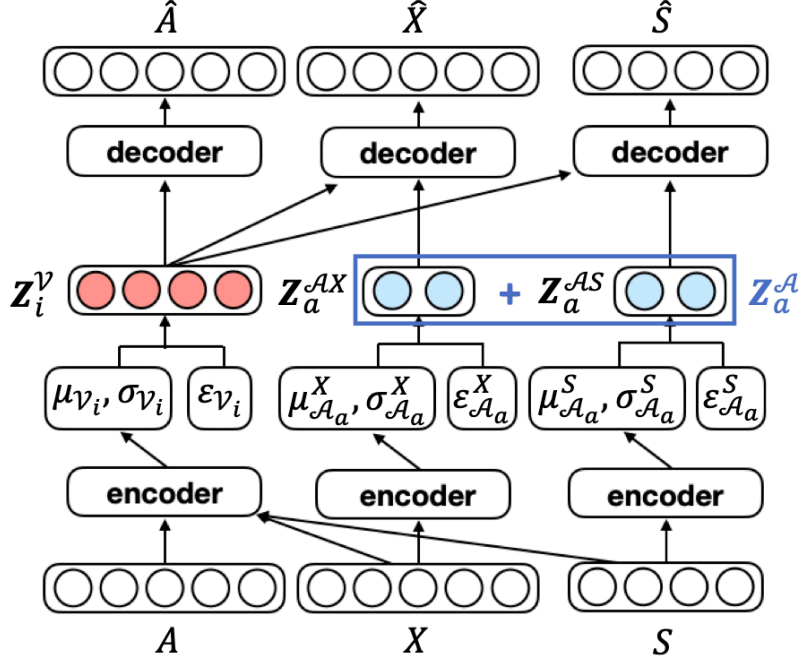


Figure 3.3: ECAN (Non-Mixed Model)

approximated and true prior $q_\phi(\mathbf{Z}^\mathcal{V}|\mathbf{A}, \mathbf{X})$, $p(\mathbf{Z}^\mathcal{V})$ and $q_\phi(\mathbf{Z}^\mathcal{A}|\mathbf{X}^T)$, $p(\mathbf{Z}^\mathcal{A})$:

$$\begin{aligned}
\log p(\mathbf{A}, \mathbf{X}) &\geq \mathbb{E}_{q_\phi} \sum_{(i,j) \in \mathcal{E}_\mathcal{V}} \log p_\theta(\mathbf{A}_{ij} | \mathbf{z}_i^\mathcal{V}, \mathbf{z}_j^\mathcal{V}) \\
&+ \mathbb{E}_{q_\phi} \sum_{(i,a) \in \mathcal{E}_\mathcal{A}} \log p_\theta(\mathbf{X}_{ia} | \mathbf{z}_i^\mathcal{V}, \mathbf{z}_a^\mathcal{A}) \\
&- D_{KL}(q_\phi(\mathbf{Z}^\mathcal{V} | \mathbf{A}, \mathbf{X}) \| p(\mathbf{Z}^\mathcal{V})) - D_{KL}(q_\phi(\mathbf{Z}^\mathcal{A} | \mathbf{X}^T) \| p(\mathbf{Z}^\mathcal{A})) \\
&\triangleq L(\theta, \phi; \mathbf{A}, \mathbf{X})
\end{aligned} \tag{3.5}$$

3.4.2 ECAN Models

Our proposed model **External Knowledge-Aware Co-Embedding Attributes Network (ECAN)** extends CAN with multiple VAEs used to encode and decode network edges \mathbf{A} , network attributes \mathbf{X} and the new attribute semantics \mathbf{S} . This generalizes the two-VAE structure in CAN. As shown in Figure 3.3, besides a VAE for encoding/decoding network structure and network attributes, a new VAE is constructed to learn attribute information from attribute-to-attribute

association matrix \mathbf{S} . The nodes are encoded by a two-layer GCN defined as:

$$\begin{aligned} H_{\mathcal{V}}^{(1)} &= \text{ReLU} \left(\tilde{\mathbf{A}} \mathbf{X} \mathbf{S} \mathbf{W}_{\mathcal{V}}^{(0)} \right) \\ [\mu_{\mathcal{V}}, \sigma_{\mathcal{V}}^2] &= \tilde{\mathbf{A}} H_{\mathcal{V}}^{(1)} \mathbf{W}_{\mathcal{V}}^{(1)} \end{aligned} \quad (3.6)$$

where $\text{ReLU}(\cdot)$ is the layer-1 non-linear mapping from $\tilde{\mathbf{A}} \mathbf{X} \mathbf{S}$ to a hidden layer, and another layer of mapping to a set of distribution parameters $\mu_{\mathcal{V}}$ and $\sigma_{\mathcal{V}}$. $\tilde{\mathbf{A}}$ represents the normalized version of \mathbf{A} .

To encode the attributes, we encode \mathbf{X} and \mathbf{S} separately in two VAEs, and concatenate the learned latent variables $\mu_{\mathcal{A}}$ and $\sigma_{\mathcal{A}}$ as attribute representation. The dimension size of these two VAEs are set to be $\frac{D}{2}$ so as to ensure the node and attribute representations could have the same length.

$$\begin{aligned} H_{\mathcal{A}}^{X(1)} &= \tanh \left(\mathbf{X}^T \mathbf{W}_{\mathcal{A}}^{X(0)} + b^{X(0)} \right) \\ H_{\mathcal{A}}^{S(1)} &= \tanh \left(\mathbf{S} \mathbf{W}_{\mathcal{A}}^{S(0)} + b^{S(0)} \right) \\ [\mu_{\mathcal{A}}^X, \sigma_{\mathcal{A}}^{X^2}] &= H_{\mathcal{A}}^{X(1)} \mathbf{W}_{\mathcal{A}}^{X(1)} + b^{X(1)} \\ [\mu_{\mathcal{A}}^S, \sigma_{\mathcal{A}}^{S^2}] &= H_{\mathcal{A}}^{S(1)} \mathbf{W}_{\mathcal{A}}^{S(1)} + b^{S(1)} \\ \mu_{\mathcal{A}} &= [\mu_{\mathcal{A}}^X, \mu_{\mathcal{A}}^S], \quad \sigma_{\mathcal{A}}^2 = [\sigma_{\mathcal{A}}^{X^2}, \sigma_{\mathcal{A}}^{S^2}] \end{aligned} \quad (3.7)$$

In decoding, \mathbf{A}_{ij} is generated from the product of $\mathbf{Z}_i^{\mathcal{V}}$ and $\mathbf{Z}_j^{\mathcal{V}}$. Attribute-wise, the node i 's attribute a , \mathbf{X}_{ia} , is derived from the product of $\mathbf{Z}_i^{\mathcal{V}}$ and $\mathbf{Z}_a^{\mathbf{A}\mathbf{X}}$, and the association between attributes a and b , \mathbf{S}_{ab} , is derived from the product of $\mathbf{Z}_a^{\mathbf{A}\mathbf{S}}$ and $\mathbf{Z}_b^{\mathbf{A}\mathbf{S}}$. With three VAEs in our model, the loss function minimizes the reconstruction errors of \mathbf{A} , \mathbf{X} , and \mathbf{S} as well as KL-divergence of the three approximated/true priors:

$$\begin{aligned}
\log p(A, X, S) &\geq \mathbb{E}_{q_\phi} \sum_{i,j \in \mathcal{V}} \log_{p_\theta} (A_{ij} | Z_i^\mathcal{V}, Z_j^\mathcal{V}) \\
&+ \mathbb{E}_{q_\phi} \sum_{i \in \mathcal{V}, a \in \mathcal{A}} \log_{p_\theta} (X_{ia} | Z_i^\mathcal{V}, Z_a^{\mathcal{A}\mathbf{X}}) \\
&+ \mathbb{E}_{q_\phi} \sum_{a,b \in \mathcal{A}} \log_{p_\theta} (S_{ab} | Z_a^{\mathcal{A}\mathbf{S}}, Z_b^{\mathcal{A}\mathbf{S}}) \\
&- D_{KL} (q_\phi(Z^\mathcal{V} | A, X) \| p(Z^\mathcal{V})) \\
&- D_{KL} (q_\phi(Z^{\mathcal{A}\mathbf{X}} | X) \| p(Z^{\mathcal{A}\mathbf{X}})) \\
&- D_{KL} (q_\phi(Z^{\mathcal{A}\mathbf{S}} | S) \| p(Z^{\mathcal{A}\mathbf{S}})) \\
&\triangleq L(\theta, \phi; A, X, S)
\end{aligned} \tag{3.8}$$

where

$$\begin{aligned}
\mathbf{Z}_i^\mathcal{V} &= \mu_{\mathcal{V}_i} + \sigma_{\mathcal{V}_i}^2 \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I) \\
\mathbf{Z}_a^{\mathcal{A}\mathbf{X}} &= \mu_{\mathcal{A}_a}^{\mathbf{X}} + \sigma_{\mathcal{A}_a}^{\mathbf{X}^2} \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I) \\
\mathbf{Z}_a^{\mathcal{A}\mathbf{S}} &= \mu_{\mathcal{A}_a}^{\mathbf{S}} + \sigma_{\mathcal{A}_a}^{\mathbf{S}^2} \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I) \\
\mathbf{Z}_b^{\mathcal{A}\mathbf{S}} &= \mu_{\mathcal{A}_b}^{\mathbf{S}} + \sigma_{\mathcal{A}_b}^{\mathbf{S}^2} \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I)
\end{aligned} \tag{3.9}$$

The encoding process of ECAN makes use of both internal (network structure) and external (text corpuses/knowledge graph) knowledge. However, the correlation between \mathbf{X} and \mathbf{S} is neglected in such VAE structure. Therefore, we propose a mixed model inspired by AMVAE to cope with the joint encoding of different sources [47].

3.4.3 Mixed Model

When there are multiple inputs to the joint learning of representations, one intuitive approach is to concatenate the two input matrices as one input to the VAE. Nevertheless, when one of the input matrices (say, S) has much larger dimension size than another (say, X), the encoding process might be dominated by the matrix with higher dimensions. To avoid this, the mixed model was proposed to jointly learn embedding from multimodal sources providing input data with different dimension sizes [47]. In the mixed mode, a two-encoder structure is

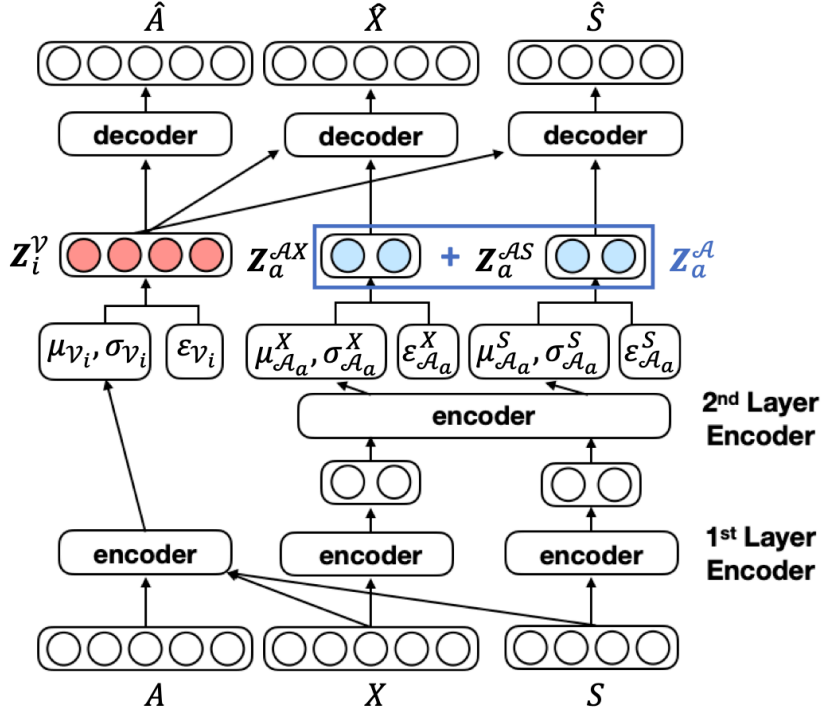


Figure 3.4: ECAN (Mixed Model)

deployed to balance the impact from different input.

We adopt the mixed model in our ECAN, and obtain a new ECAN structure called ECAN(mixed) (see Figure 3.4) which has an additional second-layer encoder that takes the concatenation of \mathbf{X} and \mathbf{S} 's representations from the first-layer encoders as input. The second-layer encoder outputs the latent variables $[\mu_{\mathcal{A}_a}^X, \sigma_{\mathcal{A}_a}^X]^2$ and $[\mu_{\mathcal{A}_a}^S, \sigma_{\mathcal{A}_a}^S]^2$. Finally, we concatenate \mathbf{z}_X and \mathbf{z}_S derived from the latent variables to be the final attribute representation. The loss function of ECAN(mixed) is almost identical to ECAN except for the KL-divergence terms. Since the encoding of \mathbf{Z}^{AX} and \mathbf{Z}^{AS} is through a shared encoder, the approximated prior q_ϕ for the two latent variables should be derived from both \mathbf{X} and \mathbf{S} . That is, the last two terms of Eq. 3.8 should be modified to $D_{KL}(q_\phi(Z^{AX}|X, S) \parallel p(Z^{AX}))$ and $D_{KL}(q_\phi(Z^{AC}|X, S) \parallel p(Z^{AC}))$ respectively. In this way, the correlation among input matrices is kept, and the learning is no longer dominated by the larger matrices.

3.5 Experiments

We conduct experiments on three real-world datasets to evaluate the performance of our proposed ECAN model against CAN. In the earlier paper [78], CAN has been shown to outperform other state-of-the-art network embedding models including AANE [48], ANRL [35] and GAE [61] in both link prediction and node classification tasks on seven real datasets. In this section, we thus focus on evaluating ECAN against CAN in similar tasks. Our goal is to determine: (a) if external knowledge about attribute semantics can improve the embedding representations for achieving more accurate downstream task results (i.e., node classification and link prediction); (b) if the mixed model can yield better performance than non-mixed model; and (c) if there are evidence that better attribute and node embeddings are learned with the help of external knowledge sources.

3.5.1 Datasets

Our experiments require attributed network datasets with words and entities as attributes. Associated with these datasets are suitable external knowledge sources covering associations between entities and between words. To the best of our knowledge, such datasets are not available and we decided to construct three new datasets specially for this research. While our models can handle multiple attributes, we keep the experiments simple and somewhat comparable to results in previous works by considering only one type of attribute. The dataset statistics are shown in Table 3.3. Note that the node labels are only used in node classification task.

- **Citation Network Dataset [102].** This is a subset of the citation network dataset from ArnetMiner which has papers as nodes, and title combined with abstract as an attribute type. Two papers are connected if one cites another. The papers are classified into six domains based on the publication venues: database, AI, hardware, system, theory, and programming lan-

Table 3.3: Dataset Statistics

Datasets	#Nodes	#Edges	#Attrib (Words)	#Attrib (Entities)	#Labels	L-M Ratio*
Citation	2,010	7,465	4,319	1,003	6	1
O*NET	974	11,342	18,119	1,174	6	0.872
IMDB	7,313	21,939	50,323	3,127	10	0.831

L-M Ratio: the ratio of two labels with the least and most number of instances. A balanced dataset has a L-M ratio of 1.

guages. Each domains consists of equal amount of papers in this dataset.

A text corpus is constructed as an external knowledge source by combining every paper’s title and abstract together to form a text document.

- **O*NET Occupation Dataset.** The Occupational Information Network (O*NET) (www.onetonline.org) is a free online occupation network created by the US government. Every occupation in O*NET is a node in the attributed network. Each occupation node has word attributes derived from its fields, namely: description, tasks and skills. An edge exists between two occupations in O*NET if they are related. In O*NET, every occupation is assigned a career preference profile according to the Holland Occupational Codes, also known as RIASEC [44]. There are six different RIASEC labels: **Realistic (R)**, **Investigative (I)**, **Artistic (A)**, **Social (S)**, **Enterprising (E)**, and **Conventional (C)**. Each occupation’s profile consists of two to four of the above labels. A text corpus is constructed by combining the description, task and skill requirements of every occupation into a document.
- **IMDB Movie Dataset.** We constructed an attributed network by extracting a subset of movies from Internet Movie Database (IMDB). In this network, each node represents a movie, and movies are linked to one another by “related movies”. Attributes are extracted from the movie synopsis. Every movie node is assigned one or more movie genre: adventure, action, comedy, action, animation, family, fantasy, drama, sci-fi, romance,

and mystery.

For each dataset, we construct both word attributes and entity attributes using their associated text corpora as follows. We extract word attributes from the text document associated with each network node (i.e., paper/occupation/movie node). For Citation dataset, we keep only the top 20 words with highest TF-IDF of each node as its word attributes. For both O*NET and IMDB datasets, we keep the top 30 words by TF-IDF as the node’s word attributes.

We next use spaCy (spacy.io) to recognize entities in the text document associated with a node and match them with Wikipedia entities using Wikipedia2Vec [128] followed by manual judgement to exclude obviously false entity linkage. The recognized entities include computing concepts and algorithm names for Citation dataset, and celebrity names and movie titles for IMDB dataset. For O*NET dataset, we link the skill entities in the text data of each occupations with Wikipedia entities by simple string matching. To select more important entities as attributes, we remove entities that appear in the text data of ≤ 10 nodes.

3.5.2 Baseline and Model Settings

We compare several variants of the ECAN model using different attribute associations and mixed/non-mixed encoder model components, as enumerated below.

- **CAN [78]**: This double-VAE model learns the latent representations of nodes and attributes using \mathbf{A} and \mathbf{X} without external knowledge. CAN has been shown to outperform other state-of-the-art network embedding models including AANE [48] and ANRL [35] in link prediction and node classification tasks. Therefore, we use it as the baseline model for comparison.
- **ECAN(a)**: This is ECAN using nodes, node adjacency matrix \mathbf{A} , node attributes which are words represented as \mathbf{X} , and word-word association $\mathbf{S}^{(a)}$ based on contextual closeness described in Section 3.3.3.

Table 3.4: Running Time (Second)

	CAN	ECAN (Non-Mixed Model)				
		(a)	(b)	(c1)	(c2)	(b+c)
Citation	55	68	71	66	75	76
O*NET	72	78	83	81	81	88
IMDB	153	203	227	198	248	240
		ECAN (Mixed Model)				
		(a)	(b)	(c1)	(c2)	(b+c)
Citation	55	76	81	87	91	89
O*NET	72	81	85	79	94	95
IMDB	153	239	271	244	257	267

- **ECAN(b)**: This is ECAN using nodes, node adjacency matrix \mathbf{A} , node attributes which are entities \mathbf{X} , and entity-entity associations $\mathbf{S}^{(b)}$ based on contextual closeness described in Section 3.3.3.
- **ECAN(c1)**: This is ECAN using nodes, node adjacency \mathbf{A} , node attributes which are entities \mathbf{X} , and entity-entity associations $\mathbf{S}^{(c1)}$ based on in-link similarities from Wikipedia knowledge graph as described in Section 3.3.3.
- **ECAN(c2)**: This is ECAN using nodes, node adjacency \mathbf{A} , node attributes which are entities \mathbf{X} , and entity-entity associations $\mathbf{S}^{(c2)}$ based on cosine similarity from pre-trained knowledge graph embeddings.
- **ECAN(b+c)**: This ECAN uses \mathbf{A} , \mathbf{X} , $\mathbf{S}^{(b)}$, and $\mathbf{S}^{(c2)}$. We use $\mathbf{S}^{(c2)}$ instead of $\mathbf{S}^{(c1)}$ because ECAN(c2) shows slightly better performance than ECAN(c1) (see Section 3.5).

We show the result using mixed model structure and non-mixed model structure for all ECANs. Our ECANs are implemented using Tensorflow [1]. Adam optimizer is used for optimization with learning rate = 0.01 [57]. For the optimization of VAEs, we use the same parameter settings as CAN to allow these model to be comparable. D is set to be 32 for all datasets and all methods while the dimension size for the hidden layers is 64.

Table 3.5: Node Classification Result (Macro-F1)

	CAN	ECAN (Non-Mixed Model)				
		(a)	(b)	(c1)	(c2)	(b+c)
Citation	0.817	0.818	0.820	0.828	0.828	0.841
O*NET	0.801	0.813	0.832	0.839	0.841	0.851
IMDB	0.732	0.758	0.767	0.781	0.783	0.789
		ECAN (Mixed Model)				
		(a)	(b)	(c1)	(c2)	(b+c)
Citation	0.817	0.821	0.820	0.833	0.837	0.849
O*NET	0.801	0.819	0.832	0.841	0.850	0.867
IMDB	0.732	0.766	0.770	0.785	0.785	0.804

3.5.3 Running Time

We show the time spent to learn the embedding model in Table 3.4. As running time is proportional to number of entries in the input matrices, it takes most time to learn the embeddings of IMDB dataset, followed by O*NET and Citation. Among all baselines, CAN spends the least amount of time because it consists of only two VAEs while the ECAN variants have three. Furthermore, due to the additional encoder layer, it may take longer time for mixed model to converge compared to non-mixed ECANs. We observe that the additional time required varies from 1% to 31%. We also observe that ECAN(c2) and ECAN(b+c) are the settings that consume most time for both the non-mixed and mixed models. This might be due to a dense input matrix S^{c2} derived by cosine similarity, while other S 's are more sparse. For instance, the number of non-zero entries in S^{c2} is about twice of that of S^{c1} in IMDB dataset (i.e., 72,311 in S^{c2} and 33,981 in S^{c1}). In general, all ECAN models can converge quickly in few minutes.

3.6 Results

3.6.1 Node Classification

In this task, we train logistic regression models to predict node labels using the node representation as input. We use OneVsRest classifier for Citation Network

Table 3.6: Link Prediction Result (AUC)

	CAN	ECAN (Non-Mixed Model)				
		(a)	(b)	(c1)	(c2)	(b+c)
Citation	0.941	0.941	0.945	0.951	0.955	0.960
O*NET	0.897	0.903	0.911	0.915	0.916	0.923
IMDB	0.903	0.905	0.913	0.917	0.921	0.924
		ECAN (Mixed Model)				
		(a)	(b)	(c1)	(c2)	(b+c)
Citation	0.941	0.941	0.947	0.953	0.961	0.964
O*NET	0.897	0.909	0.912	0.919	0.919	0.931
IMDB	0.903	0.905	0.918	0.925	0.930	0.937

Dataset as each node belongs to one class. For O*NET and IMDB datasets, we train 6 and 10 binary classifiers respectively as each node could have more than one class label. We report Macro-F1 using 10-fold cross validation following the previous works [48, 140]. As shown in Table 3.5, our ECAN models outperform CAN which does not consider external knowledge sources. Among the ECAN models using different inputs, it is generally the case that $(b+c) > (c2) \geq (c1) > (b) > (a)$. The result matches the intuition that (1) more attribute association information results in better performance; and (2) more clearly defined association (i.e., entity relatedness from knowledge graph) outperforms simple methods (e.g., contextual closeness among words). Finally, mixed models mostly outperform non-mixed ones. In particular, ECAN (Non-Mixed Model) can achieve 3% to 7.8% better macro-F1 than CAN, while ECAN (Mixed Model) can achieve 4% to 10% better macro-F1 than CAN. These results highlight the importance of learning the mixed model that balances the correlation among different inputs.

3.6.2 Link Prediction

Link prediction seeks to predict whether an edge exists between two given nodes. For each dataset, we train a binary Logistic Regression model that takes the Hadamard product of the two nodes' embeddings as input. Following the same procedure of positive/negative instances sampling as in the previous papers [60, 61, 78], we divide the existing node-node edges (i.e., positive instances) into

Table 3.7: Node Classification Performance (Macro-F1) of Different Dimension Sizes

Dataset / #dim	16	32	64	128
Citation	0.794	0.849	0.872	0.873
O*NET	0.8	0.867	0.87	0.871
IMDB	0.752	0.804	0.839	0.851

three subsets: training set (85%), validation set (5%), and testing set (10%). We then randomly sample same amount of non-existed edges as the negative instances. As shown in Table 3.6, we report the Area under ROC curve (AUC) which has been used in most previous works. The result is consistent with that of node classification. ECAN(c2) is the best model among ECAN models using one type of attribute association. ECAN(b+c), again, outperforms all other ECAN and CAN models. In particular, ECAN(b+c) mixed model outperforms CAN by 2.4% to 3.8%.

3.6.3 Choice of Dimension Size

The dimension size, D , is an important parameter in VAEs as it could heavily affect the performance of downstream tasks in some cases. Hence, we conduct an experiment to examine how sensitive ECAN is to D . Here, we focus only on node classification tasks with ECAN(b+c).

As shown in Table 3.7, among all three datasets, as D gets larger, the node classification performance improves. We also observe that the improvement quantum diminishes as the dimension size increases. However, the diminishing improvement quantum varies with datasets. For instance, the improvement quantum of Macro-F1 from $D = 32$ and $D = 64$ is significant for both Citation and IMDB, while the Macro-F1 only improves by 0.3% for O*NET. When D reaches 128, the performance cannot improve further for both Citation and O*NET datasets, but there is still a 1.2% increase in Macro-F1 for IMDB dataset. Hence, one has to choose a suitable D setting for each dataset. O*NET dataset consists of a small number of nodes, thus we could represent all nodes with rela-

tively small size of D (i.e., $D = 32$) and achieve good performance. As Citation and IMDB datasets are large, they need larger D values. In particular, IMDB dataset has the largest sets of nodes, attributes, and node labels. Therefore, even when the dimension size is large (i.e., $D = 128$), the performance is yet to converge. In other words, there is a room for ECAN(b+c) to achieve better node classification accuracy if D is set to be larger.

3.7 Case Study

3.7.1 Movie Profiling

One important goal of node embedding models is to capture the association among nodes. Hence, we expect a node’s representation should be close to those of its attributes. Moreover, similar nodes should also be close to one another in the embedding space. Therefore, we conduct a node profiling task on IMDB dataset. Consider the movie node “*The Godfather*” in the IMDB network. We return the top-5 attributes by cosine similarity between the node representation and the attribute representations obtained by different models. Ideally, the top ranked attributes are closely associated with the node and are representative. As shown in Table 3.8, ECAN mixed models appear to be better than the other models in returning associated attributes. The attributes returned by CAN and ECAN(a) are words, and attributes returned for the remaining embeddings, ECAN(b), ECAN(c2), and ECAN(b+c) are entities. Generally speaking, all the models return attributes that are closely related to the movie. However, the top attributes from CAN are mostly words linked to the movie with high frequency. We do not observe clear correlation among the attributes except for they are all related to the movie. For ECAN (a), (b), and (b+c) that consider contextual closeness, the attributes are more closely associated with one another. For example, ECAN(a) returns **mike** with last name **corleone** and he comes from a **mafia family** that **kills** people. Although ECAN(c2) does not explicitly

Table 3.8: Movie Node Profiling: *The Godfather* (words are underlined and entities are not)

Model	Nearest 5 Attributes
CAN	<u>michael</u> ; <u>vito</u> ; <u>corleone</u> ; <u>don</u> ; <u>kill</u>
ECAN(a)	<u>mike</u> ; <u>kill</u> ; <u>corleone</u> ; <u>mafia</u> ; <u>family</u>
ECAN(b)	Michael_Corleone; Marlon_Brando; The_Godfather; Sicilians; Francis_Ford_Coppola
ECAN(c2)	Corleone_family; Michael_Corleone; Francis_Ford_Coppola; The_Godfather; Al_Pacino
ECAN(b+c)	The_Godfather; Sicilians; Marfia; Michael_Corleone; Francis_Ford_Coppola

Table 3.9: Case Example Analysis

Node Classification - Genre Class Label		Genre Prediction Result		
Example		CAN	ECAN(a)	ECAN(b+c)
Movie	Class Label			
Coco	Animation	×	×	✓
Kimi no na wa ¹	Fantasy	✓	✓	×

Link Prediction		Link Prediction		
Example		CAN	ECAN(a)	ECAN(b+c)
Movie 1	Movie 2			
Ring	Ringu	×	×	✓

utilize context information as input, with the help from pre-trained knowledge graph embeddings (i.e., Wikipedia2vec), the attribute association from knowledge graph is still captured. Finally, by using both ECAN (b) and (c), we could extract attributes that are closely associated with each other both in context and in knowledge graph, which is very useful in the downstream tasks such as the genre classification task.

3.7.2 Error Analysis

In this analysis, we examine the erroneous cases returned by some of the evaluated models. In this work, we focus on IMDB dataset. Two selected case examples from node classification results and another example from link prediction

results are shown in Table 3.9. In Table 3.9, each row consists of an example, its ground truth label, and whether the models give the ground truth as prediction. If the prediction given by the model is consistent (or inconsistent) with the ground truth, we put down a tick (or cross) in the column corresponding to the model.

For node classification, the first mis-classified example is an animation movie “Coco”². Coco tells the story of a young boy accidentally visited the land of dead and met his dead ancestors. Unlike most animation movies using animals as main characters, Coco has the young boy as its main character. Since the movie node attributes are derived from the movie synopsis which is similar to that of non-animation movies, the embedding models that only consider contextual word associations from external text corpus, i.e., CAN and ECAN(a), fail to classify it correctly. On the other hand, when using entities as attributes and extracting entity associations from knowledge graphs, we observe that Coco movie is highly related to other animation films due to production company and movie description in Wikipedia. ECAN(b+c) thus learns the node representation of Coco to be closer to other animation movie nodes. This allows ECAN(b+c) to predict the correct genre label for Coco.

Our second example shows the limitation of our work. “Kimi no na wa”³ is a Japanese animation fantasy movie about a boy traveling back to the past and trying to save a village from exterminated. Both CAN and ECAN(a) are able to predict correctly using contextual closeness between words. Nevertheless, as the movie synopsis contains several Japanese names of people and locations, one could hardly find entities in the movie synopsis. This explains why ECAN(b+c) could not leverage on entity associations to predict the Fantasy genre label correctly.

Finally, we present the case example in link prediction results that involves

²<https://www.imdb.com/title/tt2380307>

³Japanese for “Your name”. <https://www.imdb.com/title/tt5311514>

two movies, “Ring” and “Ringu”. “Ring”⁴ is a horror movie with a story based on a urban legend in Japan which involves a ghost character called Sadako. The legend of Sadako is also adopted by another Japanese horror movie “Ringu”⁵. Due to this connection, the two movies should be related, but this related link has not been identified in IMDB. CAN and ECAN(a) predict the link to be unrelated because Ringu does not follow the exact same story of Ring. instead, it is about events happening after Sadako’s appearance. From the synopsis, we can hardly tell this connection. Such link between the two movies however can be inferred using a knowledge graph because the attributes (entities) of the two movies share many in-links reflecting their association. Hence, ECAN(b+c) managed to correctly predict Ring and Ringu are related.

3.8 Summary

In this chapter, we address the contextual representation learning research problem with ECAN. This study poses the following contributions,

- We propose a novel framework and several variants of External Knowledge-Aware Co-Embedding Attribute Network (ECAN) embedding models based on the proposed framework to incorporate external attribute semantics in the form of entity and word associations. We introduce two ways to measure the strength of attribute associations, i.e., (i) contextual distance between words in a text corpus, and (ii) similarity between entities in a knowledge graph. As these simple attribute semantics can be found in many attributed networks, ECAN can be easily applied to these networks.
- Our proposed ECAN models learn both entity and attribute representations in the same embedding space with a multi-VAE architecture. Specifically, we introduce a mixed model variant to merge the encoding of attribute semantics from different knowledge sources.

⁴<https://www.imdb.com/title/tt0498381/>

⁵<https://www.imdb.com/title/tt0178868/>

- We conduct extensive experiments on three real-world datasets and show that ECAN models, especially the mixed model variants, significantly outperform the state-of-the-art model. We also conduct case studies to illustrate the ability of ECAN to leverage on attribute semantics.

This work represents an early effort to incorporate external knowledge into attributed network embedding. Other combinations of external knowledge sources and attribute types can be considered in future work. One can explore alternative word embeddings (e.g., contextual word embeddings) and knowledge graph embedding models (e.g., TransE, TransH, RotatE, etc.) to extract the associations among attributes. Semi-supervised co-embedding models that can generate attributes of a node with some given class labels can also be explored. Additionally, it is worth considering encoding methods that take the graph structure into consideration, such as GNNs. Finally, ECAN is based on variational autoencoders, which offer little interpretability. In other words, one cannot infer the meaning of the dimensions of the learned latent space. We can also improve the interpretability of ECAN by incorporating attention.

Chapter 4

Contextual Path Retrieval

In this chapter, we focus on a specific knowledge-enhanced context information retrieval task, Contextual Path Retrieval (CPR). Contextual path retrieval is motivated by the emergence of knowledge graphs covering many different topical domains, as well as mentions of entities in documents which can be linked to knowledge graphs. One can perform retrieval tasks on a knowledge graph in several ways. For example, exploratory search can leverage knowledge graph to explore documents containing mentions of knowledge graph entities or relations. Knowledge graph completion task predicts entities connecting to a given query-entity via a specific relation. Knowledge-based 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 [72, 120, 126, 135].

CPR can be seen as a special type of contextual information retrieval task that returns relevant parts of the knowledge graph as answers or part of answers to some “how” or “why” queries given some query context consisting of text and entities. The derivation of CPR answers involves matching a query with path structures in knowledge graph. How to effectively learn the contextual representation of query incorporating the interaction between knowledge and text thus become crucial. In this chapter, we propose and compare different context encoders to learn contextual representation based on a common framework.

To the best of our knowledge, the CPR task has not yet been studied and it represents an early attempt to mimic the human wisdom in establishing contextual connections between two entities.

4.1 Research Objectives

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* $\xleftarrow{\text{Portrayer}}$ *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 entities¹. 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 [23, 77] or pre-trained embedding models [134].

CPR requires a query with three components: the context document, the head entity, and the tail entity. When only a partial query is provided, the CPR problem is simplified into other information retrieval (IR) problems. For example, if the context document is omitted, CPR is reduced to non-contextual knowledge graph retrieval, which returns all the existing knowledge graph paths between the head and tail entities. Conversely, when only one entity and the context document are provided, CPR retrieves all the knowledge graph paths related to the context that start with the query entity. These paths capture the semantic interactions between the query entity and all the other entities mentioned in the context document. In the most extreme scenario, where only the context document is provided, knowledge graph paths alone are insufficient to represent the complex contextual information embedded in the context. Instead, it may be necessary to retrieve a contextual graph composed of contextual paths between every pair of entities found in the context document.

CPR is also related to knowledge graph-based explainable recommendation which aims to discover user preference through investigating the underlying knowledge graphs constructed for items and users [3]. For instance, KGAT learns the importance of paths based on their abilities to predict the target items [115]. 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

¹<https://www.grid.ac/>

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), on the other hand, is more flexible as we can train the path and context representations separately. In this study, we choose the later and 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 so as 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.

4.2 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 4.1 shows the list of notations used in Sections 4.2-4.3.

4.2.1 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⁻¹*). 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., $e_{JohnCena} \xrightarrow{\text{participateIn}} e_x$, and $e_x \xrightarrow{\text{participateIn}^{-1}} e_{HeathSlater}$.

Table 4.1: Table of Notations

Symbol	Definition
General Notations	
G	Knowledge graph
E/e	Set of entities/Entity
L/l	Set of relation edges/Relation edge
R/r	Set of relation labels/Relation
$F(l)$	Function to return relation label of a relation edge l
D/d	Set of context documents/Document
Q/q	Set of query triplets $\langle e_h, e_t, d \rangle$ /Query triplet
P/p	Set of knowledge graph paths/Knowledge graph path
e_h/e_t	Head entity/Tail entity
P_m^+/P_m^-	Set of positive contextual paths/Set of negative contextual paths for query q_m
z^{cx}/z^{pt}	Context representation/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 relations with e but not present in d
k^*	Hyper-parameters in retrofitting cost function
Path Encoding	
w	An element in a path
q_ϕ/p_θ	VAE path encoder/VAE path decoder
Path Ranking	
$x_{m,k}$	Input to the path ranker $[z^{cx,m}, z_k^{pt,m}]$
$y_{m,k}$	Label for $x_{m,k}$, 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

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 [11], and AIDA CoNLL-YAGO Dataset [43], etc., and the increasingly more accurate NER and entity link methods capable of linking entity mentioned in documents to knowledge graphs. We use $|p|$ 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 4.1, *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 $\langle \textit{Alfonso Cuaón}, \mathbf{director}, \textit{Children of Men} \rangle$ 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 adopt a set of simple yet effective heuristics

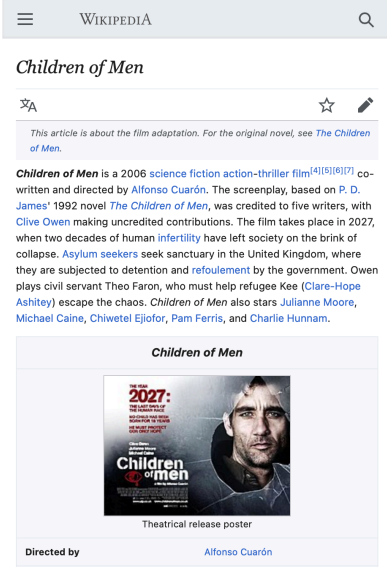
"Children of Men" wins Scriptor Award for writing

- 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



The screenshot shows the Wikipedia page for the film "Children of Men". It includes the title, a search bar, and a summary paragraph: "Children of Men is a 2006 science fiction action-thriller film 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." Below the text is a section for the theatrical release poster, showing the film's title and a portrait of Clive Owen, with the text "Directed by Alfonso Cuarón".

Figure 4.1: Annotation Interface

to constrain the set of candidate paths. In this study, 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 our solution framework below.

4.2.2 Proposed ECPR Framework

Our proposed the **E**mboding-based **C**ontextual **P**ath **R**etrieval (ECPR) framework, as shown in Figure 4.2, 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 context 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 queries, 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 connecting e_h with e_t which is denoted by P_m . Among the candidate paths in P_m , one is the contextual path and the remaining ones are the non-contextual paths.

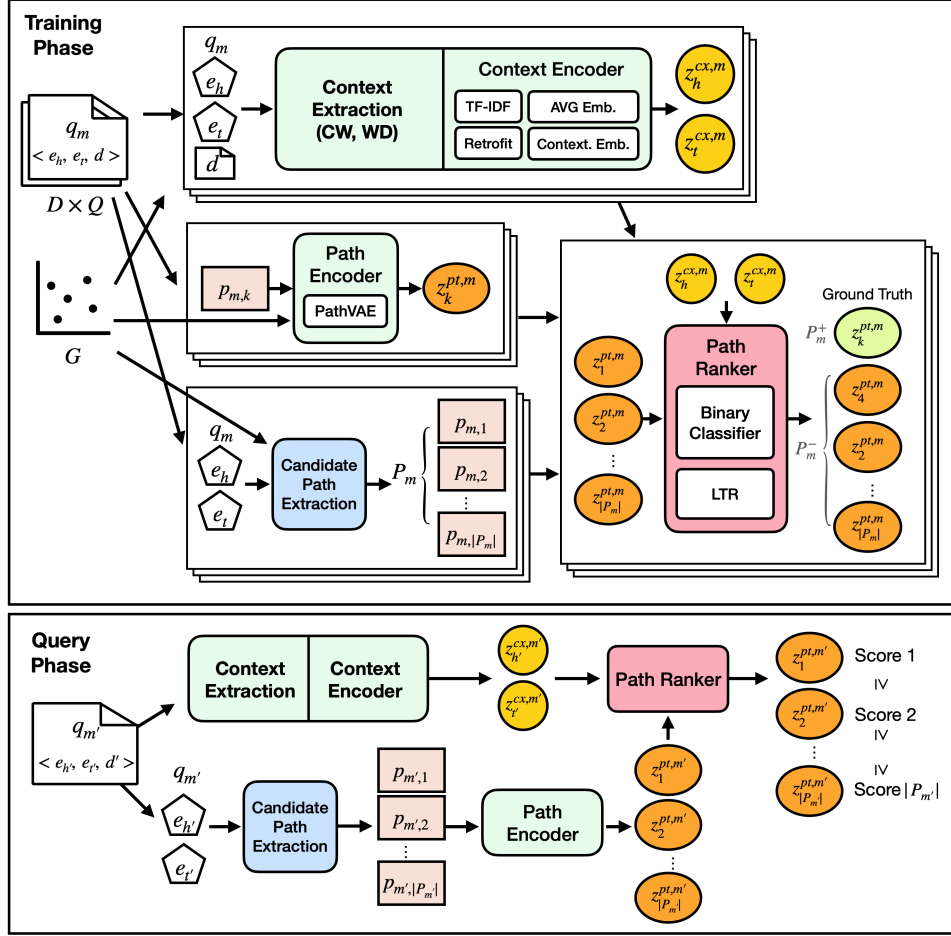


Figure 4.2: ECPR Framework

The sets of contextual and non-contextual candidate paths are denoted by P_m^+ and P_m^- respectively.

In the training phase, the **context encoder** is trained to return a *context representation* formed by concatenating the 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 from the relevant section of the context document as it generates $z_h^{cx,m}$ and $z_t^{cx,m}$.

The second component **path encoder** returns a *path representation* for each path found in G . In particular, it generates for query q_m the representation $\{z_k^{pt,m}\}$ for each candidate path $\{p_k^m\}$. 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 P_m^+ and non-contextual paths P_m^- .

The final component **path ranker** ranks a set of candidate paths against the query context during the query time using their representations (i.e., context representation and path representation). Path ranker is trained to differentiate the positive candidate path P_m^+ from the negative ones P_m^- . In this study, we use binary classifier and learning to rank as our path ranker.

In the query phase, given a query $q_{m'} = \langle e_{h'}, e_{t'}, d' \rangle$, the trained context encoder and path encoder (context encoding and path encoding) obtain the context representation $z_{h'}^{cx,m'} / z_{t'}^{cx,m'}$ and the representations $z_k^{pt,m'}$'s for candidate paths in $P_{m'}$. The path ranker will then return the ranking of candidate paths by comparing the context representation with each candidate path representation. In this work, we propose different options for context encoder, path encoder and path ranker. We shall elaborate the detail of the three components in the framework in Sections 4.3, 4.4, and 4.5 respectively.

4.3 Context Encoder

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. When encoding the context, we need to consider the context range as the context can be as loose as the whole context document d , also known as the **whole-context-document option (WD)**, or as tight as within some context window covering e_h and e_t in d , also known as the **context-window option (CW)**. In this section, we describe four different methods for context encoding, namely (i) TF-IDF, (ii) average embeddings, (iii) context-fused entity embeddings, and (iv) contextualized embedding. Among them, the context-fused entity embedding and contextualized embedding incorporate context constraints and background knowledge respectively to embed both local and global context information.

4.3.1 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 IDF found in the context document set D excluding stop-words. The selection of top 300 words is based on grid search on settings that yield the best CPR result accuracy in MRR. We subsequently use $z_h^{cx,m}$ and $z_t^{cx,m}$ to represent the TF-IDF vectors of e_h and e_t 's context respectively.

4.3.2 Averaged Embeddings

To address sparsity of words in the context which can be a problem for the TF-IDF method, we propose to use dense embedding methods which include the Averaged Embeddings method. Specifically, we encode the context using the averaged embeddings of tokens in the context. There are three different average embeddings methods, the entity-only, word-only, and word-entity embeddings.

- **Entity-only embeddings:** We can use any knowledge graph embeddings for entity-only embeddings. In this study, 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 [12]. The context representation parts $(z_h^{cx,m}$ and $z_t^{cx,m})$ are defined by the averaged TransE embeddings of all entities appeared in the context windows of e_h and e_t respectively. We train a TransE model with our knowledge graph G (see Appendix A for details about how our knowledge graph is constructed in our experiments).

- **Word-only embeddings:** In this method, the context representation components $z_h^{cx,m}$ and $z_t^{cx,m}$ are the averaged Word2vec embeddings [80] of all words appeared in e_h and e_t 's context windows respectively.
- **Word-Entity embeddings:** The context of e_h (or e_t), in this method, is the averaged Wikipedia2vec embeddings of all words and entities appeared in e_h 's (or e_t 's) context window as denoted by $z_h^{cx,m}$ (or $z_t^{cx,m}$). Wikipedia2vec jointly embeds words and entities in a common vector space [128]. It models word-to-word, word-to-entity, and entity-to-entity interaction using skip-gram model.

4.3.3 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 to the context. For instance, consider a query with *(Daniel Craig, Casino Royale)* as (e_h, e_t) 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 in the context 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 [33]. 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 [81]. At the beginning of retrofitting (or 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 (or 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 entity embeddings, we use retrofitting and counterfitting to augment the original word and entity embeddings with context constraints. In other words, given the head (or tail) entity embedding z_h^{KG} (or z_t^{KG}) from a knowledge graph embedding such as TransE, we retrofit it with constraints made up of an entity set E^* extracted from the context. The resultant entity embedding $z_h^{cx,m}$ and $z_t^{cx,m}$ will then be concatenated as the context representation. For the rest of this section, for simplicity, we overload the entity embedding notation of an entity e by z_e and the retrofitted representation of z_e by z'_e .

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

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

We omit the parameters for the constraint functions in this formula. 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).

Firstly, 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:

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

where $z'_e := z_e$ says that z'_e is initialized with z_e at the beginning; and γ is the ideal maximum distance between e and $e' \in E_{d,e}^{CW}$. 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

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

The **out-context constraints**, on the other hand, 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 which do not appear in the context of e . z_e will learn to repel itself from 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 negative context includes $E_{\text{Daniel Craig}}^{NC} = \{\text{Knives Out, Logan Lucky, ...}\}$. Intending to push e away from e^\dagger 's, NCR seeks to minimize the following cost function,

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

where δ 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 regularization terms:

- **Vector Space Preservation (VSP):** Given the n neighboring vectors $N(e)$ within a certain radius ρ 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

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

- **Relation Preservation (RP):** This regularization method randomly samples m edges $(e, \hat{r}, \hat{e}) \in KG$ to be $L(e)$, and minimizes the distance between $e + \hat{r}$ and \hat{e} . This is to maintain the translation property of the entity embedding model $e + \hat{r} \rightarrow \hat{e}$. If underlying entity embeddings are not based on translation (e.g., DistMult [130], ComplEx [105], or RotatE [99]), this relation preservation should be modified accordingly. RP follows the similar idea with VSP, and thus could be written as:

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

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 retrofitting methods each with a 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:

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

k_*^{rf} is a hyper-parameter that controls the contribution from CNA, SDA, VSP, and RP. Intuitively, we want CNA to be weighted 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:

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

Similarly, k_*^{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 ensure that CNA’s weight is higher than that of SDA.

We optimize with SGD and Adam Optimizer until convergence. The best k_* ’s are determined using grid search for the best MRR.

4.3.4 Contextualized Embedding Representation

This method is similar to Averaged Embedding method except for the use of contextualized representations for context encoding. Contextualized embeddings are proven to perform better than traditional embedding models(e.g., word2vec) in tasks such as document classification [29]. We propose two contextualized embedding representation schemes:

- **Contextualized Word Embeddings.** BERT is the state-of-the-art representation learning method, which utilizes transformer to learn a bi-directional language model [29]. 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 (or e_t) is encoded using BERT to be $z_h^{cx,m}$ (or $z_t^{cx,m}$). We use the BERT-small model² with 3 layers, and the dimension size of 512.
- **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 against candidate paths. To incorporate the entity information in the

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

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 (or e_t) is encoded using contextualized word/entity embeddings to be $z_h^{cx,m}$ (or $z_t^{cx,m}$). In this study, we use KG-BERT [134], k-bert [74], and KEPLER [117]. 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 learns the entity-word interaction. k-bert learns such interaction by injecting additional knowledge graph relations to the context. On the other hand, 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 3 layers, and the dimension size of 512.

4.4 Path Encoder

For path encoding, we propose 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 because (i) VAE is unsupervised, and the learning of PathVAE can be separately trained; (ii) PathVAE is inductive, so new paths that are not covered in 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 [42]. 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_ϕ 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 μ_m^{pt} (mean) and σ_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 [8].

$$p_\theta(\hat{p}_m | Z_m^{pt}) = \prod_{i=1}^{|\hat{p}_m|} p_\theta(w_i | w_1 : w_{i-1}, Z_m^{pt}) \quad (4.1)$$

The loss function for PathVAE is then,

$$\mathcal{L}_{pt} \geq \mathbb{E}_{q_\phi} \sum_{p_m \in P} \log_\theta p(p_m | Z_m^{pt}) - D_{KL}(q_\phi(Z_m^{pt} | p_m) \| p(Z_m^{pt})) \quad (4.2)$$

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

$$Z_m^{pt} = \mu_m^{pt} + \sigma_m^{pt2} \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I) \quad (4.3)$$

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 4.7).

4.5 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.

4.5.1 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 $y_{m,k}$ denote the query, candidate path, and class label respectively. The class label $y_{m,k} = 1$ when $p_k \in P_m^*$, and 0 otherwise. 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 P_m^- . We choose not to include all negative paths to keep a more balanced dataset, and to avoid noises. We learn a classifier f^{BI} such that $\hat{y} = f^{BI}(x)$, and simply use binary cross entropy as the loss function,

$$\mathcal{L}_{BI}(y_{m,k}, \hat{y}_{m,k}) = \sum_{x_{m,k} \in H} \text{CrossEntropy}(y_{m,k}, \hat{y}_{m,k}) \quad (4.4)$$

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.

4.5.2 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 which 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 [16] and listNET [18]. In this study, we utilize $\text{XE}_{ND\text{CG}}\text{MART}$ [15] as our learning to rank path ranker. $\text{XE}_{ND\text{CG}}\text{MART}$ 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 ρ and a parameterized class of label distribution ϕ :

$$\rho(s'_{m,k}) = \frac{e^{s'_{m,k}}}{\sum_{n=1}^{|P_m|} e^{s'_{m,n}}}, \quad \phi(s_{m,k}; \gamma) = \frac{2^{s_{m,k}} - \gamma_{m,k}}{\sum_{n=1}^{|P_m|} 2^{s_{m,n}} - \gamma_{m,n}} \quad (4.5)$$

where γ 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:

$$\mathcal{L}_{\text{LTR}} = \sum_{x_{m,k} \in H} \text{CrossEntropy}(\rho(s'_{m,k}), \phi(s_{m,k}; \gamma)) \quad (4.6)$$

We use the LightGBM package to implement this LTR ranker [56]³.

4.5.3 Training of Path Encoder and Ranker

In the training of ECPR framework, we could either jointly optimize path encoder and path ranker together, i.e., $\mathcal{L} = \lambda_2\mathcal{L}_{pt} + \lambda_3\mathcal{L}_{pr}$, $\mathcal{L}_{pr} \in \{\mathcal{L}_{BI}, \mathcal{L}_{LTR}\}$. 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 less time, the trained model may not guarantee good performance. We leave the discussion of joint optimization to future works.

4.6 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 which do not use pathVAE-based path encoding as baselines, and a shortest path heuristic method as another simple baseline. The experiments are conducted on two real world datasets (Wiki-film and Wiki-music) and another two synthetic datasets (Synthetic-S and Synthetic-L) specially constructed for CPR and other subsequent research with steps outlined in Appendix A. Table 4.2 shows the statistics of the four datasets. The experiment results on the real world datasets and synthetic datasets are given in Sections 4.7 and 4.8 respectively. In this work, four evaluation metrics are introduced to determine the performance of each model including two ranking-based metrics and

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

Table 4.2: 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 (L_{MAX})	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

two semantic-based metrics.

4.6.1 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 ⁴.

- **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 retrofitted with in-context constraints only as denoted by $[z_h^{cx(cf)}, z_t^{cx(cf)}]$, **(c)** Retrofit(I+O): The context representation is the concatenation of head and tail entities' TransE representations that are retrofitted with in-context and out-context constraint ($[z_h^{cx(cf)}, z_t^{cx(cf)}]$). For the hyper-parameters in the cost function for options (b) and (c) (see Section 4.3.3), 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: $k_1^{rf} = 0.45$, $k_2^{rf} = 0.1$, $k_3^{rf} = 0.25$, $k_4^{rf} = 0.2$, and $k_1^{cf} = 0.2$, $k_2^{cf} = 0.3$, $k_3^{cf} = 0.1$, $k_4^{cf} = 0.2$, $k_5^{cf} = 0.2$. 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 ($[z_h^{cx(BERT)}, z_t^{cx(BERT)}]$), **(b)** KG-BERT ($[z_h^{cx(KG-BERT)}, z_t^{cx(KG-BERT)}]$), **(c)** k-bert ($[z_h^{cx(k-bert)}, z_t^{cx(k-bert)}]$), and **(d)** KEPLER ($[z_h^{cx(KEPLER)}, z_t^{cx(KEPLER)}]$).

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 4.3.3 the constraints are constructed using both context window and the

⁴<https://wikipedia2vec.github.io/wikipedia2vec/>

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 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 α 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 which 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.
- **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 [128]. 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 which randomly select a candidate path as prediction, and a *shortest path* baseline which 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 [12, 72]. 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 ⁵.

4.6.2 Evaluation Metrics

During the query phase, each ECPR-based model returns for an query entity pair and context the probability of every candidate path being the contextual path. 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.

⁵<https://pytorch.org/>

Mean Reciprocal Rank (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:

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

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.

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 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 which 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 [142]. 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:

$$GEO(p, p_{gt}) = \sum_{op_j \in OP(p, p_{gt})} c_{op_j} \quad (4.7)$$

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:

$$\begin{aligned}
&\text{Semantic Entity Insertion: } c_{ei}(e) = \text{dist}(e, e_0) \\
&\text{Semantic Entity Deletion: } c_{ed}(e') = \text{dist}(e', e_0) \\
&\text{Semantic Entity Substitution: } c_{es}(e, e') = \text{dist}(e, e') \\
&\text{Semantic Relation Insertion: } c_{ri}(r) = \text{dist}(r, r_0) \\
&\text{Semantic Relation Deletion: } c_{rd}(r') = \text{dist}(r', r_0) \\
&\text{Semantic Relation Substitution: } c_{rs}(r, r') = \text{dist}(r, r')
\end{aligned} \tag{4.8}$$

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

$$\text{dist}(e_1, e_2) = \frac{|(p_{e_1}^o \cup p_{e_2}^o) - (p_{e_1}^o \cap p_{e_2}^o)|}{|p_{e_1}^o \cup p_{e_2}^o|} \tag{4.9}$$

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 $(e_1, r_1, \dots, r_{L-1}, e_L)$, a 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 NGE0 for entity and relation path separately as NGE0(Ent) and NGE0(Rel) so we could make better observations about the dissimilarity between predicted path and ground truth path.

Path Embedding Distance (PED)

While N GEO 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:

$$PED(p, p_{gt}) = 1 - |\text{Rescaled Cosine Similarity}(z_p^{\text{pt}}, z_{gt}^{\text{pt}})| \quad (4.10)$$

where *Rescaled Cosine Similarity* is cosine similarity (originally ranged $[-1, 1]$) rescaled to be in the range of $[0, 1]$, i.e.,

$$\text{Rescaled Cosine Similarity} = \frac{\text{Cosine Similarity} + 1}{2}$$

In this work, we use the same pathVAE in ECPR to learn the path representation to obtain PED.

We report PED and N GEO between the highest ranked path returned by the path ranker and ground truth path. Both PED and N GEO return 0 when the returned path is identical to the ground truth, and value closer to 1 otherwise.

4.7 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-

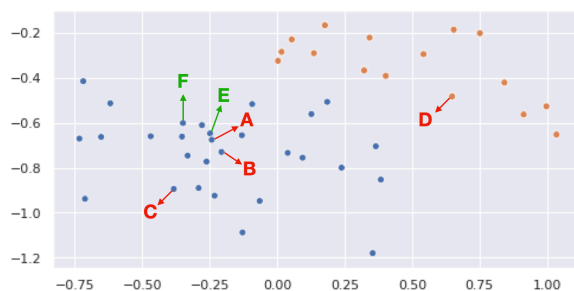


Figure 4.3: Visualization of PathVAE Embedding Model

fold cross validation is used to report the average MRR, $\text{hit}@k$, NGeo, 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 4.7.6, we leave out results from models using whole document as the context range option in view of their poorer results than that those model using the context window option. For easy reading, the best results in the result tables are boldfaced.

4.7.1 Path Embeddings with PathVAE

Before we present other results, we first present results of path encoding with PathVAE. In Figure 4.3, 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*,
- **(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 4.3. Path D, on the other hand, is far from the rest as it describe a different relation. Figure 4.3 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 4.3 match the above judgements. 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)*, where *The Godfather(film series)* 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 the embeddings of 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 4.3.

Table 4.3: Performance Comparison among AVG Embedding Encoders (with PathVAE and LTR, CW only)

Dataset	Model	Context Enc	Evaluation Metrics						
		AVG Emb Model	MRR	hit@k			NGEO		PED
				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	0.382	0.2177	0.5069	0.6998	0.16	0.15	0.278
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	0.513	0.2108	0.6223	0.9117	0.13	0.12	0.216

4.7.2 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 4.3, 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 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.

4.7.3 Context-fused Entity Context Encoder

Table 4.4 shows the 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-

Table 4.4: Performance Comparison among Context-fused Entity Context Encoders (with PathVAE and LTR)

Dataset	Model	Context Enc	Evaluation Metrics						
		Context-fused Entity	MRR	hit@k			NGEO		PED
				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	0.16	0.15	0.241
		Retrofit (I+O)	0.414	0.2587	0.5432	0.7116	0.16	0.15	0.227
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	0.13	0.12	0.214
		Retrofit (I+O)	0.521	0.2547	0.6679	0.9136	0.13	0.12	0.203

context constraints helps to augment the query entity representations with knowledge that are relevant to the context. Table 4.4 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.

4.7.4 Contextualized Embeddings

Finally, we compare ECPR-based models that utilize contextualized embeddings. As shown in Table 4.5, 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, we shall use KEPLER as the representative contextualized word-entity embeddings in subsequent experiments.

Table 4.5: Performance Comparison among Contextualized Word-Entity Embeddings (with PathVAE and LTR, CW only)

Dataset	Model	Context Enc	Evaluation Metrics						
			Context. Embeddings	MRR	hit@k			NGEO	
		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	0.12	0.1	0.173
		KEPLER	0.558	0.3786	0.729	0.8339	0.12	0.09	0.171
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	0.11	0.09	0.173
		KEPLER	0.653	0.4447	0.7561	0.93	0.11	0.09	0.165

Table 4.6: 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	0.558	0.3786	0.729	0.8339	0.12	0.09	0.171
Wiki-music	ECPR-Cxt Emb	Binary	0.648	0.4431	0.7499	0.9291	0.12	0.09	0.171
		LTR	0.653	0.4447	0.7561	0.93	0.11	0.09	0.165

4.7.5 Results of Rankers

In addition to different context encoders, we also 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 4.6, we found that LTR path ranker outperforms binary classifier ranker. This findings 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.

4.7.6 Overall Results

In the previous sections, we evaluate the performance of the ECPR-based models using the same path encoding method, and the different configurations of

context encoding methods using the context window option. This allows us to determine the best configuration of each context encoding method using the context window option. We now show the results of the different context encoding method with the **best configuration** using the context window and whole document options, and compare against the various baseline models. We conduct this evaluation on the Wiki-film and Wiki-music datasets and the results are shown in Tables 4.7 and 4.8 respectively.

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 which 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), on the other hand, 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 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 seems small, we conduct significance test on the results and concluded significant difference ($p\text{-value} < 0.01$) between our best and runner-up models.

We also compare models when using context window and whole document as context range (except ECPR-Cxt-fused). Tables 4.7 and 4.8 show that those using context window outperforms 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, NGeo and PED. We observe that the similarity-based metrics generally capture performance differences more clearly. In particular, NGeo reflects how much modification should be made to a top ranked path for it to be converted to the ground truth path, and PED indicates the similarity of the two paths in a embedding space. This result suggests 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 this in our case studies in Section 4.7.8.

There might be concerns about whether our models favor shorter paths over longer paths. While we find our model 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 ($=3.837$).

4.7.7 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.

Table 4.7: 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
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
				k-bert	CW	0.546	0.3682	0.7238	0.8261	0.12	<u>0.10</u>
ECPR-Cxt Emb			KEPLER		CW	0.558	0.3786	0.729	0.8339	0.12	0.09
				WD	<u>0.507</u>	<u>0.3243</u>	<u>0.6717</u>	<u>0.7953</u>	0.12	<u>0.10</u>	0.180
Other Baselines	TF-IDF	Cos	TF-IDF	CW	0.360	0.1362	0.392	0.6422	0.18	0.18	0.325
				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

Path Enc.: Path encoder, **Path Rnk.:** Path ranker, **Cxt Enc.:** Context encoder

Cxt Rng: Context range, **CW:** Context Window Only, **WD:** Whole Document

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

Table 4.8: 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)	-†	0.521	0.2547	0.6679	0.9136	0.13	0.12	0.203
				k-bert	CW	0.626	0.4002	0.7378	0.9251	0.11	0.09
ECPR-Cxt Emb			KEPLER		CW	0.653	0.4447	0.7561	0.93	0.11	0.09
				WD	<u>0.627</u>	<u>0.4361</u>	<u>0.7522</u>	<u>0.9295</u>	0.11	<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

Path Enc.: Path encoder, **Path Rnk.:** Path ranker, **Cxt Enc.:** Context encoder

Cxt Rng: Context range, **CW:** Context Window Only, **WD:** Whole Document

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

Candidate path extraction could cost a lot of time as one has to conduct random walk on every possible entity that could be on the path from the head entity to the tail entity. To improve efficiency when extracting candidate paths, for a head entity e_h we keep a dictionary of list of entities it can reach in one to L_{MAX} 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. In 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.

4.7.8 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.

Comparison among All Context Encoders.

In the following, we illustrate the differences among different models using their top ranked candidate paths for the same context document and query entities. These models 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...”⁶

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, on the other hand, is returned by ECPR-Cxt Emb:

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

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

⁶https://en.wikinews.org/wiki/La_La_Land_receives_record-equalling_fourteen_Oscar_nominations;_Hacksaw_Ridge_gets_six

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...”⁷

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*

In fact, one might argue that the path returned by ECPR-Cxt Emb is actually better. It has not been include 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 4.8 which involves experiments using a synthetic dataset. On the other hand, misled by the mention of *Buckinghamshire* in the context window, ECPR-AVG Emb ranks the following path the highest.

Casino Royale $\xrightarrow{\text{WikiPageLink}}$ *Buckinghamshire* $\xrightarrow{\text{country}}$ *United Kingdom* $\xrightarrow{\text{country}}$
Charlie and the Chocolate Factory

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 N GEO and PED.

⁷https://en.wikinews.org/wiki/James_Bond_set_at_Pinewood_Studios_destroyed_by_fire

Models using Context Window versus Models using Whole Document.

‘ Our earlier experiment results show that context defined by words/entities within same context window outperforms those which 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...”⁸

The ground truth path is

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

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

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

It ranked the ground truth path at the 4th position.

⁸https://en.wikinews.org/wiki/Nine_of_Alfred_Hitchcock%27s_films_are_restored;_30_years_since_his_death

ECPR-BERT versus ECPR-Cxt Emb Models using KEPLER.

Generally, our experiment results show that ECPR-Cxt Emb methods (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...”⁹

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:

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

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:

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

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, on the other hand, does not have such background knowledge embedded in it. It therefore fails to retrieve the ground truth contextual path.

⁹https://en.wikinews.org/wiki/Five_United_States_ex-presidents_raise_relief_funds_at_hurricane_event

4.8 Analysis on Synthetic Datasets

In this section, we analyze our proposed models from different aspects using synthetic datasets. As described in Section A.2, we generate synthetic context documents and their contextual paths from a sampled knowledge graph built on DBpedia. As the size of synthetic dataset is much larger than Wikinews datasets, we have 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 does the model perform on large-scale datasets?
- How does 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 does the number of candidate paths affect the performance? Queries with many candidate paths should be more difficult than those with few candidate paths.
- How does 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 take the average embeddings of both words and entities in context encoding.

Table 4.9: 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
ECPR-Cxt Emb			k-bert	CW	<u>0.528</u>	<u>0.3547</u>	<u>0.6947</u>	<u>0.8149</u>	0.12	<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
			KEPLER	CW	0.532	0.3614	0.7182	0.8305	0.12	0.1	0.187
				WD	0.516	0.3114	0.6552	0.7772	0.12	<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.

4.8.1 Model Performance on Large-scale Dataset (Synthetic-L)

As Wiki-film and Wiki-music datasets are relatively small-sized, we experimented selected ECPR models with exact same setting as described in Section 4.6 on the much larger Synthetic-L dataset. As shown in Table 4.9, the results observed using Synthetic(L) are similar to those in Tables 4.7 and 4.8. 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.

4.8.2 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,

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

Table 4.10: 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	0.547	0.3599	0.569	0.3808	0.022	0.0209

(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 4.10, we first verify that query set A is more difficult than query set B. Furthermore, not only does ECPR-Cxt Emb with KEPLER outperforms ECPR-AVG Emb with Wikipedia2vec on both query sets, the performance gap between two query sets for ECPR-Cxt 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.

Table 4.11: Retrieval Performance of Query Sets with Different Number of Candidate Contextual Paths (Numbers in brackets are improvement over Random Guess)

Model	A (N=50) AVG #Candidate=12.3		B (N=50) 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	0.356 (+0.095)	0.141 (+0.056)	0.703 (+0.104)	0.429 (+0.088)

4.8.3 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-L, 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 4.11. Additionally, we report the performance of a random baseline where a randomly selected candidate path is returned. We show the change 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 improvement 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.

Table 4.12: 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 ≥ 5		Length of GT Path ≤ 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	0.543	0.3596	0.561	0.3791	0.539	0.3484

4.8.4 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 ≥ 5 , and **(Query set B)** involving ground truth paths with length ≤ 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 4.12. Firstly, 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 A. Furthermore, the change in performance between 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.

4.9 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 so as 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.

4.10 Summary

Contextual path retrieval (CPR) is a novel and important information retrieval task when knowledge graphs are available for explaining the connections between entities found in some common context. In this study, 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. Our contribution in this chapter can be summarized as follows.

- We formally define the Contextual Path Retrieval Problem (CPR) and propose a novel ECPR framework to determine contextul 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 which effectively leverage on all semantics in context and to incorporate background knowledge using pre-trained model.
- We propose PathVAE which utilize 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.

Moving forward, we believe that there are still ample 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.

One major limitation of CPR is that it requires the inclusion of entity pairs and context in a query, assuming that the user issuing the query has a certain level of understanding or specific expectations from the knowledge graph. The requirement for users to understand the knowledge graph can be alleviated by implementing a user interface that visually represents the knowledge graph and the retrieved contextual path. For example, the interface can display a sampled knowledge graph containing 1 to n -hop neighbors of the query entities, along with the relations between them. All candidate contextual paths should be highlighted in this sampled knowledge graph. Furthermore, the interface should include explanations of the meaning of knowledge graph relations. This approach ensures that all necessary information to comprehend the contextual paths is provided, eliminating the need for users to possess any prior knowledge about the entities or relations.

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 that 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.

Chapter 5

Contextual Path Generation: A Monotonic Approach

Nowadays, knowledge-based applications, such as question answering and information searches, increasingly depend on complex reasoning over knowledge graphs for result generation and explanation. Finding a knowledge path that explains the semantic connection between two entities mentioned in a given piece of text is thus an important task.

5.1 Research Objective

5.1.1 Problem Formulation

Example: Consider the example in Figure 5.1. We have an input text d containing a set of sentences $\{S_1, S_2, \dots\}$ and it covers news about Daniel Craig replacing Pierce Brosnan in the James Bond movie series. Suppose we want to know the semantic connection between Martin Campbell and Daniel Craig mentioned in d and both Campbell and Craig already exist as entities in a given film-related knowledge graph G . The semantic connection that is required is a path of entities and relations: Campbell $\xrightarrow{\text{direct}}$ Casino Royale $\xrightarrow{\text{has actor}}$ Craig that says Campbell is the director of the movie Casino Royale which stars Craig.

This path is known as the *contextual path* as it accurately captures the connection between the two persons in the news context. Another path Campbell $\xrightarrow{\text{director}}$ GoldenEye $\xrightarrow{\text{country}}$ UK $\xrightarrow{\text{birth place}}$ Craig is not the contextual path for the same news as this path is irrelevant to the news content. In this example, we call Campbell and Craig the *head entity* and *tail entity* and denote them by e_H , e_T respectively. The news document is called the *context document*.

We call the above knowledge graph path inference task *Contextual Path Generation (CPG)*. In the following, we give a formal definition of CPG.

Problem Definition (Contextual Path Generation): Formally, the input of a contextual path generation (CPG) task is a query q consisting of head and tail entities denoted by e_H and e_T respectively and a context document d , i.e., $q = \langle e_H, e_T, d \rangle$. A background knowledge graph G that covers a large set of entities and relations connecting them is also given. CPG returns a contextual path between e_H and e_T , $p^q = \langle e_H, r_1, e_2, r_2, \dots, e_T \rangle$. p^q is composed of entities e_i and relation labels r_k of a *knowledge graph* G relevant to d which mentions both e_H and e_T . All entities and relation labels on the path p^q are expected to exist in the knowledge graph G .

A knowledge graph G is made up of a set of entities E . Entities are connected with each other through a set of relations L . Each relation $l = (e_i, r_k, e_j)$ ($l \in L$) connects two entities e_i and e_j by a relation label $r_k \in R$ where R denotes the set of relation labels in G . A chain of relations makes up a path $p = l_1, l_2, \dots, l_{|p|}$, where $|p|$ represents the number of relations exists in p . In CPG, we use the expression $e_1 \xrightarrow{r_1} e_2$ to represent a single-relation path, or a one-hop path. A n -hop path is thus represented by $e_1 \xrightarrow{r_1} e_j \xrightarrow{r_k} e_2 \dots \xrightarrow{r_n} e_n$.

For the purpose of establishing connections between entities, we assume that every relation label r and its inverse r^{-1} exist in R . Specifically, for every relation (e_i, r_k, e_j) in L , its inverse (e_j, r_k^{-1}, e_i) also exists in L . Given a pair of head and tail entities, there may be no path, one path, or multiple paths connecting them in G . The contextual path could be one of these paths, or a path that

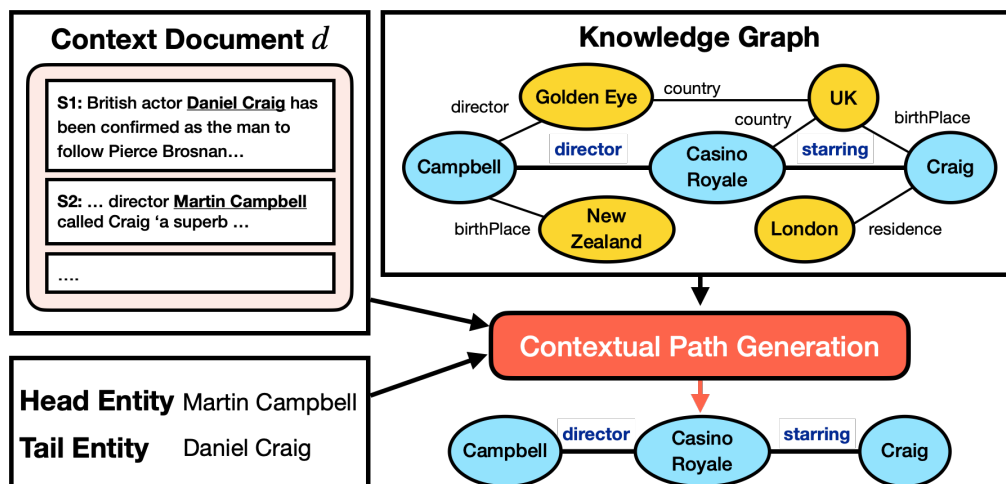


Figure 5.1: Contextual Path Generation Example (See footnote¹ for more content in d)

does not exist in G as it involves relation(s) inferred from G and the context document d . This makes CPG different from other problems that adopt a closed world assumption [91] about the knowledge graph relations, i.e., non-existent relations are invalid by default.

A solution to this CPG task problem will substantially benefit knowledge-based content analysis and search. Contextual paths not only provide the background knowledge to understand d , but also offer a new knowledge layer to search d and other documents (e.g., to find documents that cover directors directing movies which feature Daniel Craig).

5.1.2 Challenges

In this chapter, we aim to propose a model that is capable of generating the correct contextual path given a pair of head and tail entities, a context document, and a background knowledge graph. There are mainly three technical challenges in CPG tasks, namely: (a) sparse and noisy content in the context document which could mislead the choice of entities and relations forming the contextual

¹British actor Daniel Craig has been confirmed as the man to follow **Pierce Brosnan** as the sixth James Bond. Producer Barbara Broccoli and director **Martin Campbell** called Craig 'a superb actor who has all the qualities needed to bring a contemporary edge to the role'... The next Bond film **Casino Royale** is due to film in Italy, the Bahamas, the Czech Republic and Pinewood Studios. The film is due for release in 2006.

path, (b) missing relations in the knowledge graph that are required to form part of the contextual path, and (c) well-formedness of the generated contextual path.

The first challenge is caused by context documents containing potentially sparse and noisy information due to its short length and some parts of the content being irrelevant to the input query. Without a good semantic representation of the context, the solution model will be misled into returning the incorrect contextual paths.

Secondly, the knowledge graph may not provide all the relations required to form the contextual path for a given query. That is, some relations may be missing from the knowledge graph. This can happen in many practical scenarios where the knowledge graph is incomplete. Hence, CPG has to infer missing relation edges when generating the contextual paths. These generated relations not only have to be semantically “reasonable”, but also relevant to the input query.

Thirdly, when generating a contextual path, we need to ensure its well-formedness which includes ensuring the path to be loopless and to successfully connect e_H with e_T . Loops are redundant information in the resultant path. A contextual path p^q is loopless if every entity $e \in p^q$ appears only once. The generation of p^q is finished if it starts and ends with e_H and e_T respectively. To the best of our knowledge, both criteria have not yet been considered in the sequence generation research if we consider paths as a special type of sequence [?, 34, 116].

To develop models for the CPG task, we need to consider the above three challenges. Moreover, we also seek to create datasets with ground truth contextual paths as well as performance metrics for measuring different aspects of accuracy (e.g., semantic relevance and well-formedness of the resultant contextual paths) in the model results.

Table 5.1: Table of Notations

Symbol	Definition
$G = (E, R, L)$	Knowledge graph
$E = \{e_1, \dots, e_{ E }\}$	Set of entities
$R = \{r_1, \dots, r_{ R }\}$	Set of relation labels
$L = \{l_1, \dots, l_{ L }\}$	Set of relations
$q = \langle e_H, e_T, d \rangle$	Query triplet
d	Context document
e_H	Head entity
e_T	Tail entity
p^q	Contextual path of query q
$\mathbf{S}_d = \{\mathbf{s}_1, \dots, \mathbf{s}_{ S_d }\}$	Sentences of the context document d
s_i	Embedding of a sentence s_i
z_e, z_r	Entity and relation representations
H_d	Base and BERT-encoded context document
H_B	BERT-encoded context document for mixed-encoder
H_{e_H}/H_{e_T}	Representations of e_H and e_T
S_t	Transformer decoder output at step t

5.2 Proposed Architecture and Models

In this section, we describe our proposed transformer-based model architecture and its modules for deriving different CPG solution models. The important notation used throughout this chapter is listed in Table 5.1.

5.2.1 Overview of Model Architecture

Our proposed CPG architecture is based on the transformer encoder-decoder model [107]. It sees the generation of contextual paths as a decoding step with the input condition encoding the context document and the pair of head and tail entities. To the best of our knowledge, this is the first work to attempt conditional knowledge graph path generation using a transformer approach. We design our CPG model architecture that consists of a *context encoder*, and a *controlled path generator* as shown in Figure 5.2. By instantiating this architecture with different module options, we can create different solution models for CPG tasks (see Section 5.2.6).

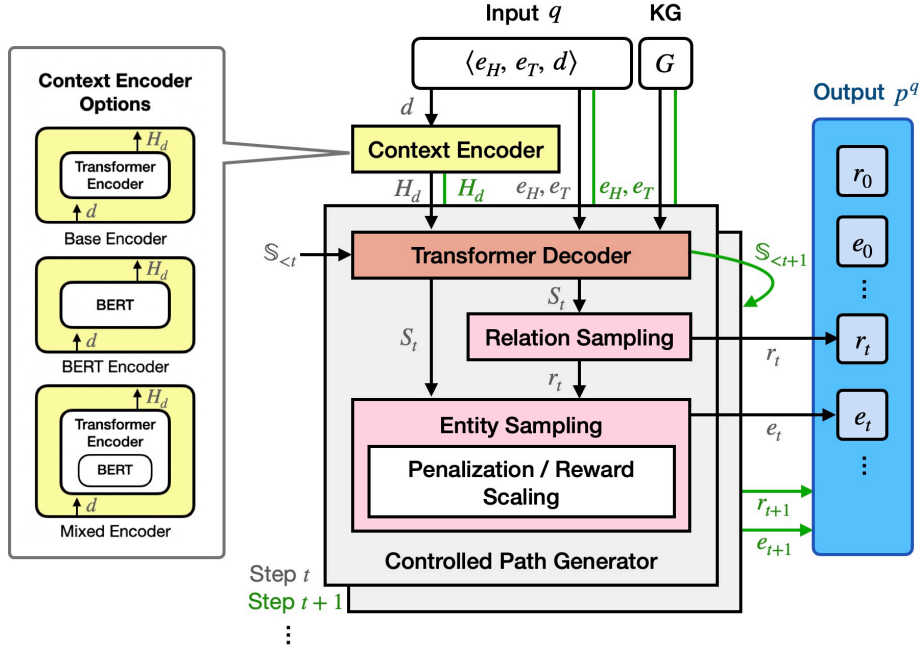


Figure 5.2: Proposed CPG Architecture

Given a query $q = \langle e_H, e_T, d \rangle$, the context encoder converts the context document d into a *context representation* H_d with a knowledge graph providing the background knowledge. The controlled path generator uses H_d , H_{e_H} and H_{e_T} as input to generate a contextual path p^q starting from e_H in multiple steps. To ensure the quality of the generated contextual paths, we propose two novel scaling methods, namely *penalization* and *reward scaling*. These scaling methods guide the generation process to avoid resampling entities already included in the path, but to enhance the likelihood of generated path reaching the tail entity e_T .

Algorithm 1 shows the procedural view of our proposed CPG architecture. In the following, we use an example to walk through Algorithm 1. A more detailed description of the context encoder and controlled path generator will be given in Sections 5.2.2 and 5.2.3 respectively.

Example of Contextual Path Generation. We illustrate how we generate the contextual path $\text{Campbell} \xrightarrow{\text{direct}} \text{Casino Royale} \xrightarrow{\text{has actor}} \text{Craig}$ with the example query in Figure 5.1. Firstly, we encode d with the context encoder and obtained H_d . At the beginning, the contextual path p^q is empty, and the initial state S_0 given to the transformer decoder consists of the initial entity $e_0 = [\text{start}]$ and

initial relation $r_0 = [\text{start}]$. $[\text{start}]$ is a special token to mark the start of the path. We also pass H_d , $e_H = \text{Campbell}$, and $e_T = \text{Craig}$ to the transformer decoder module. In the first step ($t = 1$), the transformer decoder takes S_0 , H_d , and the two entities as input and outputs the hidden state S_1 . With S_1 , the relation sampling module is to sample the next relation $r_1 = [\text{start}]$. With both r_1 and \tilde{S}_1 , the entity sampling module is to sample the entity $e_1 = \text{Campbell}$. We append (r_1, e_1) to the contextual path p^q . Following the sample procedure, we sample $r_2 = \text{direct}$ and $e_2 = \text{Casino Royale}$ at $t = 2$. At $t = 3$, we sample $r_3 = \text{has actor}$ and $e_3 = \text{Craig}$. The path generation process terminates as it reaches the tail entity $e_T = \text{Craig}$. Finally, the model outputs $p^q = \{[\text{start}], [\text{start}], [\text{start}], \text{Campbell}, \text{direct}, \text{Casino Royale}, \text{has actor}, \text{Craig}\}$.

5.2.2 Context Encoder

We denote a context document d by a sequence of sentences $\mathbf{S}_d = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{|\mathbf{S}_d|}\}$. We first encode each sentence \mathbf{s}_i into an embedding s_i . Other than textual content, d may contain mentions of entities including the head and tail entities. To derive the context vector representation H_d that leverages on pre-trained background knowledge, we propose three transformer-based encoder models to summarize $\mathbf{S}_d = \{\mathbf{s}_1, \dots, \mathbf{s}_{|\mathbf{S}_d|}\}$, namely: *Base-encoder*, *BERT-encoder* and *Mixed-encoder* as shown in Figure 5.1.

Base-encoder. For Base-encoder, we use entity and/or word embeddings of the input knowledge graph G to first obtain each sentence’s embedding. Suppose a sentence \mathbf{s}_i contains a set of entities E_i and words W_i . The sentence embedding s_i is defined as the average word and/or entity embeddings learned using *knowledge graph embedding* methods, such as TransE [12], and Wikipedia2vec [128]. Other embedding schemes could also have been used but these will be left to be investigated in our future work. We average all sentence embeddings to obtain the context embedding H_d , i.e., $H_d = \frac{1}{|\mathbf{S}_d|} \sum_{\mathbf{s}_i \in \mathbf{S}_d} s_i$. This encoding process maps the original sentences which may include sparse vocabulary to a dense

Algorithm 1: Contextual Path Generation**input :** Query $q = \langle e_H, e_T, d \rangle$, Knowledge Graph G **output:** Contextual path p^q // **Context Encoder** $H_d \leftarrow Enc(d)$ // Applicable to all context encoder options $H_B \leftarrow BERT(d)$ // Only applicable to mixed-encoder option// **Controlled Path Generation** $t = 1; e_t = e_0; r_t = r_0; p^q = \{\}$ // Initialize $t, e_t, r_t,$ and p^q $S_0 \leftarrow e_0$ // Initialize S_0 **while** $t \leq L_{MAX}$ and $e_t \neq e_T$ **do**

// Transformer Decoder (Base/BERT Encoder)

 $S_t \leftarrow Dec(S_{<t}, H_d, H_{e_H}, H_{e_T})$

// Transformer Decoder (Mixed encoder)

 $S_t \leftarrow Dec(S_{<t}, H_d, H_{e_H}, H_{e_T}, H_B)$ // Non-linear transformation layer $FFN_R(\cdot)$ $S_t^R \leftarrow FFN_R(S_t)$

// Obtain relation sampling distribution using softmax

 $\mathbf{P}_t^R \leftarrow Softmax(S_t^R)$ // Sample r_t $r_t \leftarrow Sample(\mathbf{P}_t^R)$ // Non-linear transformation layer $FFN_E(\cdot)$ $S_t^E \leftarrow FFN_E(S_t, r_t)$

// Obtain entity sampling distribution using softmax

 $\mathbf{P}_t^E \leftarrow Softmax(S_t^E)$ $\mathbf{P}'_t^E \leftarrow PRScale(\mathbf{P}_t^E, G)$ // Penalization and Reward Scaling $e_t \leftarrow Sample(\mathbf{P}'_t^E)$ // Sample e_t $p^q \leftarrow p^q + (r_t, e_t)$ // Append r_t and e_t to p^q $t++$ **end****return** p^q

embedding.

BERT-encoder. For BERT-encoder, we use the *contextualized word embedding* of a pre-trained BERT to derive a context document embedding H_d . We first concatenate all sentences of S_d and insert a [CLS] token before the concatenated sentences. The output representation vector corresponding to the [CLS] token captures the meaning of the entire document d . We denote this represen-

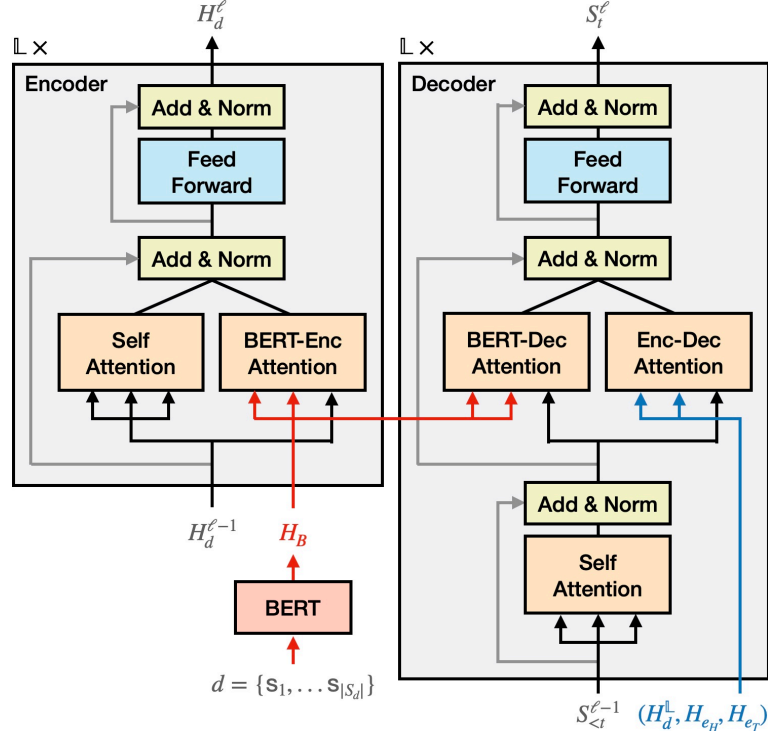


Figure 5.3: Illustration of CPG-mixed Encoder and Decoder

tation vector by $H_d = BERT(d)$. BERT itself is already a transformer. Other more advanced encoders such as k-BERT, and RoBERTa [29, 74, 146] can also be used but we will leave investigation of them to future work.

Mixed-encoder. For Mixed-encoder, we combine the BERT-encoded context document embeddings, denoted by $H_B = BERT(d)$, with the base-encoded sentence embeddings H_d (for TransE or Wiki2vec). This is achieved using attention-based fusion models as shown in the context encoder component of Figure 5.3.

Let H_d^ℓ denote the document representation at the ℓ -th layer of the mixed encoder for all $\ell \in [\mathbb{L}]$, where \mathbb{L} is the number of layers in the mixed encoder. We denote the 0-th layer embedding of d by $H_d^0 = H_d$, and the i -th sentence embedding in H_d^ℓ as h_i^ℓ for $i \in [|\mathbf{S}_d|]$. The output representation in the ℓ -th layer of i -th sentence is:

$$\tilde{h}_i^\ell = \frac{1}{2} (attn_S(h_i^{\ell-1}, H_d^{\ell-1}, H_d^{\ell-1}) + attn_B(h_i^{\ell-1}, H_B, H_B))$$

where $attn_S$ and $attn_B$ are two separate attention models. $h_i^{\ell-1}$ is the representation of sentence i from the previous layer. Each \tilde{h}_i^ℓ is then processed by a non-linear transformation layer $FFN(\cdot)$, which is defined as follows,

$$FFN(x) = W_2(\max(W_1x + b_1, 0)) + b_2$$

where x is the input, W_* and b_* are trainable parameters, and $\max(\cdot)$ is an element-wise operation. We then obtain the output of the ℓ -th layer by:

$$H_d^\ell = (FFN(\tilde{h}_1^\ell), \dots, FFN(\tilde{h}_{|d|}^\ell))$$

The mixed encoder outputs the context embedding $H_d^{\mathbb{L}}$ at the last layer. In this way, $H_d^{\mathbb{L}}$ captures both information from the self-attention block, and semantic information from the BERT embeddings.

5.2.3 Controlled Path Generation

If we treat context document, head and tail entities as conditions and the contextual path as a sequence, CPG can be regarded as some kind of conditional sequential generation. Most works on conditional sequential generation focus on generating text sequences conditioned on a single-valued signal, such as tense, sentiment polarity or formality [27, 46, 94, 133]. For example, BERT-infused translation uses a pre-trained encoder to first obtain the abstract representation of a condition to be used as the input of the text generation model [144]. KG-BART generates a commonsense sentence given a set of concepts from the commonsense knowledge graph input [75]. There are also works that generate sentences conditioned on a context document following a given syntax of an exemplar [63, 84]. Nevertheless, none of the existing works on conditional sequential generation address the generation of knowledge graph paths which connect a head entity with a tail entity with a sequence of relations with relevant labels.

Depending on the choice of context encoding option, the controlled path

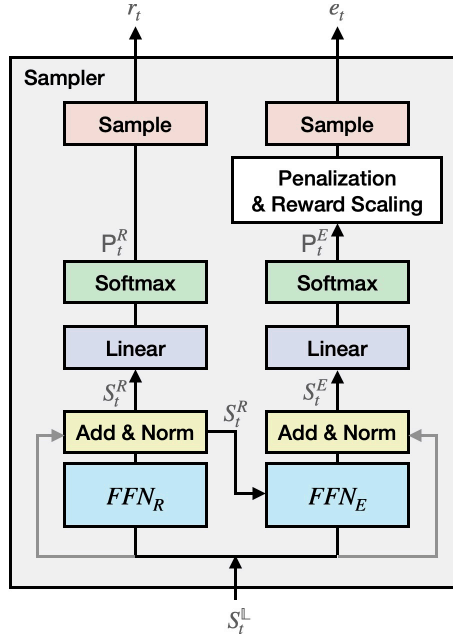


Figure 5.4: Illustration of Relation and Entity Sampling Process

generation process is slightly different. In the following, we describe the decoding step of the controlled path generation process for Base-Encoder and BERT-Encoder, followed by that for Mixed Encoder. Finally, we cover the common relation and entity sampling step using softmax.

Decoding with Base-encoded and BERT-encoded context. With the context representation H_d (in the case of Base-encoder and BERT-encoder), the contextual path generation module iteratively generates the relation edges to form a contextual path starting from e_H . As shown in Figure 5.2, the module consists of a transformer decoder that takes H_d , H_{e_H} , and H_{e_T} as input for all its layers. $H(\cdot)$ is an encoding function for entities. At step t , the transformer decoder also takes the relation and entity sampled from the previous generation step denoted by $S_{<t}$. Suppose r_t and e_t denote the relation and entity respectively generated at step t . For generation step $t = 1$, the sampled relation and entity from the previous step r_0 and e_0 are assigned zero start vectors as input to the transformer decoder. That is, $S_{<1} = S_0$ is a zero vector.

For each generation step t ($t \geq 1$), the transformer decoder generates an output vector $S_t^{\mathbb{L}}$ after \mathbb{L} layers of decoding. For generation step t , we use $S_{<t}^{\ell} =$

$(S_1^\ell, \dots, S_{t-1}^\ell)$ to denote the hidden state of the ℓ -th layer in the decoder up to step t . S_1^ℓ indicates the start of a sequence, and S_t^ℓ is the embedding of the element to be generated at step $t - 1$. At each ℓ -th layer, we have the output of the self-attention model $attn_S$:

$$\hat{S}_t^\ell = attn_S(S_t^{\ell-1}, \mathbb{S}_{<t+1}^{\ell-1}, \mathbb{S}_{<t+1}^{\ell-1})$$

$S_t^{\ell-1}$ is the encoder output from previous layer at step t .

$$\hat{S}_t^\ell = attn_S(S_t^{\ell-1}, \mathbb{S}_{<t}^{\ell-1}, \mathbb{S}_{<t}^{\ell-1})$$

We then obtain the output of the encoder-decoder attention model $attn_E$:

$$\tilde{S}_t^\ell = attn_E(\hat{S}_t^\ell, H_d^L, H_d^L)$$

We then obtain S_t^ℓ by passing \tilde{S}_t^ℓ through a non-linear transformation layer:

$$S_t^\ell = FFN(\tilde{S}_t^\ell)$$

The final output of the transformer decoder at step t is S_t^L .

Decoding with Mixed-encoded context. For the controlled path generator coupled with mixed-encoder, we specially design the transformer decoder to incorporate a BERT-Decoder attention block on H_B and another Encoder-Decoder attention block on H_d , H_{e_H} , and H_{e_T} which form the context representation output of transformer encoder H_d^L . These attention blocks control how each path element to be generated interacts with the context representations from the BERT-encoder and mixed-encoder in every transformer layer.

$$\tilde{S}_t^\ell = \frac{1}{2} \left(attn_B(\hat{S}_t^\ell, H_B, H_B) + attn_E(\hat{S}_t^\ell, H', H') \right)$$

where

$$H' = (H_d^{\mathbb{L}}, H_{e_H}, H_{e_T})$$

We then obtain S_t^ℓ by passing \tilde{S}_t^ℓ through a non-linear transformation layer:

$$S_t^\ell = FFN(\tilde{S}_t^\ell)$$

The final output of the transformer decoder at step t is $S_t^{\mathbb{L}}$.

Relation and Entity Sampler. $S_t^{\mathbb{L}}$ is fed to the relation and entity sampler to sample a relation-entity pair linking the previously generated entity e_{t-1} with e_t via a relation label r_t . For relation sampling, we pass $S_t^{\mathbb{L}}$ through a feed forward network FFN_R to obtain S_t^R . We then feed S_t^R to a linear mapping function and softmax layer to obtain the probability distribution over candidate relations \mathbf{P}_t^R , and finally sample r_t from \mathbf{P}_t^R . Subsequently, we pass $S_t^{\mathbb{L}}$ together with S_t^R through a feed forward network FFN_E to obtain S_t^E . S_t^E goes through a similar process to generate \mathbf{P}_t^E . We then apply penalization and reward scaling on \mathbf{P}_t^E . Finally, we sample e_t from \mathbf{P}_t^E .

To reduce the candidate pool and to ensure the generated path is well-formed, we force the model to generate only relations (or entities) that have at least one link to some entity of the type identical to the previous entity (or relation). For instance, when e_{t-1} is an entity of type **Person**, we force the model to not predict relations such as *foundingDate* or *headquarters*, as these relations have not been linked to any of the Person type entities in the knowledge graph.

Let R^* and E^* be the candidate relation set and entity set, respectively. The generation of path ends when the module samples the tail entity e_T from E^* , or when the path reaches the maximum length L_{MAX} , a user specified parameter. Finally, we obtain the contextual path $p^q = \langle e_H, r_1, e_1, \dots, r_l, e_l \rangle$ where $l \leq L_{MAX}$ is the length (or number of relations) of p^q . Note that e_l should ideally be e_T .

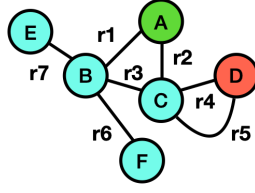


Figure 5.5: Reward Scaling

The generation of an entity/relation tuple at generation step t is as follows: (1) a relation softmax takes $\text{concat}(e_H, e_T, h_t^o, r_{t-1}, e_{t-1})$ as input, and generates probabilities of all candidate relations $\{p_{r'} | r' \in R^*\}$ where S_t^L is the output of the Transformer decoder; (2) a relation r_t is then sampled from R^* ; (3) r_t is then fed to an entity softmax together with e_H, e_T , and h_t^o to obtain the probability distribution of candidate entities $\{p_{e'} | e' \in E^*\}$; (4) entity e_t is sampled from E^* based on the distribution, and repeat the above for step $t + 1$ until e_T or L_{MAX} is reached.

In the above path generation, two issues need to be addressed, namely (1) *loopy paths* and (2) *unfinished paths*. A loopy path is one that traverses the same entity more than once. An unfinished path is one that fails to reach the desired tail entity e_T . We therefore propose penalization scaling and reward scaling to address them in Sections 5.2.4 and 5.2.5 respectively.

Following the transformer structure, we use the loss function: $\mathcal{L} = \sum_{t=1}^{|p^q|} \text{CE}(r_t, r_t^*) + \text{CE}(e_t, e_t^*)$ where CE denotes the cross-entropy loss, r_t^* and e_t^* are the ground truth relation and entity respectively at step t .

5.2.4 Penalization Scaling

The penalization scaling scheme addresses the loopy path issue. Consider the example path: $A \xrightarrow{r1} B \xrightarrow{r3} C \xrightarrow{r2} A \xrightarrow{r1} B \xrightarrow{r3} C \xrightarrow{r4} D$ which contains a loop, $A \xrightarrow{r1} B \xrightarrow{r3} C \xrightarrow{r2} A$. Loops are redundant sub-paths that not only harm the accuracy of the contextual path but may also result in unfinished paths when we limit the maximum length of path to be generated, i.e., L_{MAX} .

The idea of penalization scaling is simply to avoid re-visiting any previously

generated entity. For example, when predicting the next relation/entity of $A \xrightarrow{r^1} B \xrightarrow{r^3} C$, we should force the model not to sample A again. To achieve this, after entity softmax is applied and before the sampling of next entity, we multiply the prediction probability of previously sampled entities A , B , and C by a hyperparameter $0 \leq \beta < 1$. That is, at step t , $p_{e'} = \beta \cdot p_{e'}$ if $e' \in \{e_H, e_T, e_1, \dots, e_{t-1}\}$.

To determine how penalization scaling will reduce loops in the generated path, we formulate Theorem 5.2.1 which determines the probability of generating a unique path of length t without revisiting any entity.

Theorem 5.2.1. *Suppose m is the number of entities in E , and the probability of sampling every entity is initially the same. The probability of generating a unique path sequence of length t without revisiting any entity is:*

$$p = \prod_{i=1}^t \left(1 - \frac{(i-1)\beta}{m - (i-1)(1-\beta)} \right) \quad (5.1)$$

Proof. We first prove that Equation 5.1 holds when $t = 1$ as $p = 1$.

Suppose Equation 5.1 holds for $t = k$. That is,

$$p = \prod_{i=1}^k \left(1 - \frac{(i-1)\beta}{m - (i-1)(1-\beta)} \right)$$

For $t = k + 1$, the probability to add another distinct entity from $m - k$ distinct entities is

$$\frac{m - k}{m - k + k\beta}$$

The probability of $k + 1$ distinct entities in the sequence is thus:

$$\begin{aligned} p &= \left[\prod_{i=1}^k \left(1 - \frac{(i-1)\beta}{m - (i-1)(1-\beta)} \right) \right] \cdot \frac{m - k}{m - k + k\beta} \\ &= \left[\prod_{i=1}^k \left(1 - \frac{(i-1)\beta}{m - (i-1)(1-\beta)} \right) \right] \cdot \left(1 - \frac{k\beta}{m - k(1-\beta)} \right) \\ &= \prod_{i=1}^{k+1} \left(1 - \frac{(i-1)\beta}{m - (i-1)(1-\beta)} \right) \end{aligned} \quad (5.2)$$

By induction, we prove that Theorem 5.2.1 holds. \square

Based on Theorem 5.2.1, we can derive two special cases:

- For special case $\beta = 0$ (i.e., revisiting any previously sampled entity is not possible), the probability of generating a unique sequence of t entities is

$$p = 1$$

- For special case $\beta = 1$ (i.e., any previously sampled entity can be visited as if they were just like any other not yet sampled entity), the probability of generating a unique sequence of t entities is

$$p = \prod_{i=1}^t \left(1 - \frac{i-1}{m}\right)$$

When m is large compared with t , the denominator $m - (i-1)(1-\beta)$ becomes very closed to m . As a result, p becomes smaller as we increase β . Nevertheless, the following corollary provides a lower bound to p .

Corollary 5.2.1.1. *Suppose $\beta < 0.5$, and $m \gg K \cdot (1-\beta)$ for very large integer \mathbb{K} . The probability of generating a unique sequence of t entities p will then be larger than $\left(\frac{\mathbb{K}-2t}{\mathbb{K}}\right)^{t-1}$.*

Proof.

$$\begin{aligned}
p &= \prod_{i=1}^t \left(1 - \frac{(i-1)\beta}{m - (i-1)(1-\beta)} \right) \\
&\text{Since } \beta < 0.5, \text{ we have } (1-\beta) > \beta. \\
p &> \prod_{i=1}^t \left(1 - \frac{(i-1)(1-\beta)}{m - (i-1)(1-\beta)} \right) \\
&\text{With } m \gg \mathbb{K} \cdot (1-\beta) \\
p &\gg \prod_{i=1}^t \left(1 - \frac{(i-1)(1-\beta)}{\mathbb{K} \cdot (1-\beta) - (i-1)(1-\beta)} \right) \tag{5.3} \\
&= \prod_{i=1}^t \left(1 - \frac{(i-1)}{\mathbb{K} - (i-1)} \right) \\
&= \prod_{i=0}^{t-1} \frac{\mathbb{K} - 2i}{\mathbb{K} - i} \\
&> \left(\frac{\mathbb{K} - 2t}{\mathbb{K}} \right)^{t-1}
\end{aligned}$$

Equation 5.3 shows the lower bound of p . As $\mathbb{K} \gg t$, p remains to be quite close to 1. For example, when $t = LMAX = 6$ and $\mathbb{K} = 500$, $p > 0.88$. That is, with more than 88% chance, we will obtain a sequence of distinct entities of length 6. \square

5.2.5 Reward Scaling

An unfinished path is one that does not end with the tail entity e_T or with the length of L_{MAX} . Two reasons could contribute to unfinished paths: (1) there is a loop(s) in the path, or (2) the intermediate entity (or entities) steers the path away from e_T . Since the loop problem has been addressed by penalization scaling, we propose a reward scaling scheme to avoid case (2).

Reward scaling scheme prefers the entity softmax to favor entities that are on the paths leading to the tail entity e_T as well as relevant to the given context. For instance, in Figure 5.5, when predicting the next relation/entity after $A \xrightarrow{r1} B$ with $e_T = D$, C would be a better choice than E and F because it is on a path between B and D .

Our proposed reward scaling therefore introduces a multiplicative factor $\alpha_i \geq 1$ to the prediction probability of each entity e_i in entity softmax. We first estimate how likely each entity will reach e_H by a *personalized random walk* over a weighted graph $G' = (E', R')$ derived from G . E' are the entities within the distance of n^{rw} hops from the tail entity e_T , and $R' \subseteq L$ are the relations among entities of E' . Here we set $n^{rw} = 6$ according to the maximum length of contextual path to be generated. We assign each relation $r \in R'$ a weight based on its semantic similarity to the context document d by measuring the cosine distance between BERT-encoded embedding of r 's relation label and that of d . We turn this cosine distance into a transition probability of r . Next, we conduct K iterations of a random walk starting from e_T with restarting probability p^{rw} on G' . Let n_i be the number of times an entity $e_i \in E'$ is visited, the reward factor of e_i is defined as $\alpha_i = 1 + \frac{n_i}{\sum_{e' \in E'} n_{e'}}$. We empirically set $K = 20$.

Assume the probability distribution of sampling entities is **uniform**, and the number of entities in E' to be sampled is m' . Theorem 5.2.2 shows the probability of a path with t steps ending with e_T .

Theorem 5.2.2. *With reward scaling, the probability of a path with t hops ending with the tail entity e_T is:*

$$\left(\frac{n^{rw} + 1}{n^{rw}} \times \frac{1}{m'} \right) \left(1 - \frac{n^{rw} + 1}{n^{rw}} \times \frac{1}{m'} \right)^{t-1} \quad (5.4)$$

Proof. The proof assumes that entity softmax and relation softmax return equal uniform probabilities for entities and relation label respectively in every iteration of softmax computation. It also assumes no penalization scaling is performed.

Before applying reward scaling, the probability of sampling e_T at any step is $\frac{1}{m'}$. The probability of e_T sampled at step t is:

$$\frac{1}{m'} \left(1 - \frac{1}{m'} \right)^{t-1} \quad (5.5)$$

With reward scaling, the probability of sampling e_T is changed to $\frac{1}{\alpha_{e_T}}$. Hence,

the probability of e_T sampled at step t is:

$$\frac{\alpha_{e_T}}{m'} \left(1 - \frac{\alpha_{e_T}}{m'}\right)^{t-1} \quad (5.6)$$

where α_{e_T} is derived from the number of random walk visits of e_T . As there are K random walks,

$$\alpha_{e_T} = 1 + \frac{\text{Num of random walk visits of } e_T}{\text{Num of random walk visits of all entities in } E'} \quad (5.7)$$

$$= 1 + \frac{K}{n^{rw} K} \quad (5.8)$$

$$= \frac{n^{rw} + 1}{n^{rw}} \quad (5.9)$$

This is because e_T will be sampled for at least K times as it is the starting entity of random walks. Finally, the probability of sampling e_T at t^{th} step is thus:

$$\left(\frac{n^{rw} + 1}{n^{rw}} \times \frac{1}{m'}\right) \left(1 - \frac{n^{rw} + 1}{n^{rw}} \times \frac{1}{m'}\right)^{t-1} \quad (5.10)$$

□

Note that reward scaling does not affect the inference of missing relations as we do not rule out missing relations during path generation as R' is derived from the random walk and relations that are more likely to exist (i.e., more related to d) are assigned higher weight. For instance, although the link between e_i and e_j is unobserved in the knowledge graph, the model learns to infer this link because e_j can be visited by the random walk and thus is promoted by the reward scaling. We assign $\alpha_i = 1$ for entities not in E' .

Theorem 5.2.3. *The probability of generating any path **not** ending with e_T reduces with reward scaling.*

Proof. Suppose a path not ending with e_T requires t hops to generate. Without

reward scaling, its probability is:

$$\left(1 - \frac{1}{m'}\right)^t \quad (5.11)$$

With reward scaling, its probability is:

$$\left(1 - \frac{\alpha_{e_T}}{m'}\right)^t \quad (5.12)$$

As $\left(1 - \frac{1}{m'}\right) > \left(1 - \frac{\alpha_{e_T}}{m'}\right)$, Equation 5.11 is larger than Equation 5.12 for any t . This implies that it is less likely to sample paths not ending with e_T after applying reward scaling. □

5.2.6 CPG Models

With different context encoder options coupled with the above controlled path generation module, we derive the following variations of the CPG models:

CPG-Base. These models use Base-encoder as the context encoder. Specifically, we develop two CPG-Base models: (a) *CPG-Base(TransE)* and (b) *CPG-Base(Wiki2vec)*. We describe the implementation of these two models in Section 5.3.3.

CPG-BERT. This is the only CPG model using BERT-encoder. Other than that, CPG-BERT and the various CPG-Base models share the same transformer decoder, relation softmax, and entity softmax module designs.

CPG-Mixed. This group of CPG models uses a base-encoder and the transformer decoder of a controlled path generator with an added BERT-Decoder attention as shown in Figure 5.3. We call such models CPG-Mixed and there are two CPG-Mixed variants depending on the choice of base-encoded sentence embedding. *CPG-Mixed(TransE)* and *CPG-Mixed(Wiki2vec)* adopt TransE and Wiki2vec as their base-encoded sentence embeddings, respectively.

Model Complexity. Here, we compare the number of trainable parameters of the CPG models. Let \mathbf{D}^s denote the embedding dimension, and \mathbf{D}^{FFN} denote the inner-layer dimensionality of the FFN layer. \mathbb{L} is the number of layers of encoder/decoder in the model. For simplicity, we exclude bias in the following analysis.

We start by analyzing the number of parameters in each component included in a transformer structure:

- Multi-head Self Attention: $\mathbb{N}_{sa} = 4 \times (\mathbf{D}^s \times \mathbf{D}^s)$
- Layer Normalization: $\mathbb{N}_{norm} = \mathbf{D}^s$
- Feed-Forward Layer: $\mathbb{N}_{ff} = \mathbf{D}^s \times \mathbf{D}^{FFN}$

CPG-Base uses the original encoder-decoder transformer structure. The number of trainable parameters in the encoder is thus $\mathbb{L} \times (1 \times \mathbb{N}_{sa} + 2 \times \mathbb{N}_{norm} + 2 \times \mathbb{N}_{ff})$. The number of trainable parameters in the decoder is $\mathbb{L} \times (2 \times \mathbb{N}_{sa} + 3 \times \mathbb{N}_{norm} + 2 \times \mathbb{N}_{ff})$.

CPG-BERT utilizes a pre-trained BERT model as the encoder and only trains the decoder. The number of parameters of its decoder is identical to that of CPG-Base.

CPG-Mixed is the most computationally expensive model as it introduces an additional BERT-Enc attention component in each layer. As a result, the encoder contains $\mathbb{L} \times (2 \times \mathbb{N}_{sa} + 2 \times \mathbb{N}_{norm} + 2 \times \mathbb{N}_{ff})$ trainable parameters, while the decoder has $\mathbb{L} \times (3 \times \mathbb{N}_{sa} + 3 \times \mathbb{N}_{norm} + 2 \times \mathbb{N}_{ff})$ parameters.

All of the models share the same entity/relation sampling process, as shown in Figure 5.2. When sampling the entity/relation, we use a feed-forward network with a linear layer which projects the output of the feed-forward network to the dimension of vocabulary. The number of trainable parameters for entity and relation sampling is thus $\mathbb{N}_{samp} = 2 \times (\mathbf{D}^s \times \mathbf{D}^{FFN}) + \mathbf{D}^s \times |R^*| + \mathbf{D}^s \times |E^*|$.

Overall, the different CPG models do not differ much in number of model parameters. CPG-Mixed has more parameters for the self-attention modules.

5.3 Experiments on Real Datasets

5.3.1 Wikinews Dataset

We have constructed Wiki-film and Wiki-music datasets following the annotation approach in Appendix A. Each dataset consists of 40 Wikinews articles as context documents. We then identify 563 and 471 entities mentioned in these two sets of context documents respectively and which can be found in a knowledge graph extracted from DBpedia². In the DBpedia knowledge base, each entity in the knowledge graph corresponds to an article in Wikipedia. Every attribute within the infobox of the article is extracted as a DBpedia relation linking to another entity corresponding to another Wikipedia article. The attribute label in the infobox is used as the relation label. To derive the set of (e_H, e_T) entity pairs, we identify 1396 and 1237 entity pairs from the context documents of the Wiki-film and Wiki-music datasets such that both entities of each pair can be found in a context document with paths connecting them. Since each entity pair is associated with a context document, the number of queries (d, e_H, e_T) for the datasets are 1396 and 1237, respectively. Each query is associated with a ground truth contextual path. The detailed statistics of the datasets are shown in Table 4.2.

We crowd-source the ground truth contextual path for each (e_H, e_T) entity pair using AMT workers annotating the path using a custom-developed web-based annotation interface and the annotation approach described in Appendix A. To make the annotation reasonable to most annotators, we limit the contextual paths to have a maximum length of six.

²<https://wiki.dbpedia.org>

5.3.2 Evaluation Metrics

In our experiment, we divide the entity pairs into five folds. One-fold is used for testing, and the remaining four folds are used for model training. We measure the model performance using two types of metrics.

The first set of metrics measures the quality of generated paths not using ground truth contextual paths. Instead, they measure the proportion of generated paths that are unfinished (**%Uf**), and proportion of generated paths with loops (**%Lp**).

The second set of metrics measures accuracy with respect to the ground truth paths. The percentage of ground truth paths (**%Gt**) measures the proportion of generated paths matching the ground truth ones. The second metric *normalized graph edit distance* **NGED** measures the *structural similarity* between generated path and ground truth path. This metric prefers the generated path most structurally similar to the ground truth contextual path as they can be different due to length and ordering of elements.

Normalized Graph Edit Distance (NGED). The Graph Edit Distance (GED) is a metric originally designed to measure the similarity of two graphs by counting the number of operations needed to transform one graph to another [142]. We adapt it to measure the similarity between a generated path p_i and the ground truth path p_i^{gt} . In this work, we define the **normalized GED** between p_i and p_i^{gt} as:

$$NGED(p_i, p_i^{gt}) = \min\left(\frac{GED(p_i, p_i^{gt})}{|p_i^{gt}|}, 1\right) \quad (5.13)$$

where $|p_i^{gt}|$ is the length of ground truth path. A path p here refers to a sequence of both entities and relations, i.e., $p = \langle e_1, r_1, \dots, r_{L-1}, e_L \rangle$. $GED(p_i, p_i^{gt})$ is defined by:

$$GED(p_i, p_i^{gt}) = \sum_{op_j \in OP(p_i, p_i^{gt})} c_{op_j} \quad (5.14)$$

where $OP(p_i, p_i^{gt})$ is the sequence of edit operations for converting the for-

mer to the latter. The symbol c_{op_j} denotes the semantic cost of operation op_j which is of one of the following six operation types, namely: *semantic entity insertion* (vi), *semantic entity deletion* (vd), *semantic entity substitution* (vs), *semantic relation insertion* (ri), *semantic relation deletion* (rd), and *semantic relation substitution* (rs).

We then define the semantic cost of each operation type as follows:

$$\begin{aligned}
\text{Semantic Entity Insertion: } c_{vi} &= dist(t, t_0) \\
\text{Semantic Entity Deletion: } c_{vd}(t') &= dist(t', t_0) \\
\text{Semantic Entity Substitution: } c_{vs}(t, t') &= dist(t, t') \\
\text{Semantic Relation Insertion: } c_{ri}(r) &= dist(r, r_0) \\
\text{Semantic Relation Deletion: } c_{rd}(r') &= dist(r', r_0) \\
\text{Semantic Relation Substitution: } c_{rs}(r, r') &= dist(r, r')
\end{aligned} \tag{5.15}$$

Where t , t' and t_0 refer to the inserted entity type, deleted entity type, and the root entity type in the ontology, respectively. r , r' and r_0 refer to the inserted relation label, deleted relation label, and the root relation label in the ontology, respectively. Given two entity types (or relation labels) t_1 and t_2 , we define $dist(t_1, t_2)$ to be the distance between t_1 and t_2 in the ontology structure of the knowledge graph.

$$dist(t_1, t_2) = 1 - \frac{|S(t_1) \cap S(t_2)|}{|S(t_1) \cup S(t_2)|} \tag{5.16}$$

where $S(t_i)$ is the set of supertypes of t_i in the ontology. For example, let Actor and MovieDirector be two entity types in the ontology structure and the relevant supertype relationships in the ontology are $\{\text{Agent} \rightarrow \text{Person}, \text{Person} \rightarrow \text{Artist}, \text{Artist} \rightarrow \text{Actor}, \text{Person} \rightarrow \text{MovieDirector}\}$ where \rightarrow denotes the supertype relationship. The distance between Actor and MovieDirector (which share two entity types) is thus $1 - \frac{2}{5} = \frac{3}{5}$.

From the above definition, we also derive **normalized GED for the entity**

paths of generated path p_i and ground truth path p_i^{gt} as $NGED^{Ent}(Ent(p_i), Ent(p_i^{gt}))$, and **normalized GED for the relation paths** of p_i and p_i^{gt} as $NGED^{Rel}(Rel(p_i), Rel(p_i^{gt}))$.

The entity path involves only the entities of a path $Ent(p) = (e_1, \dots, e_L)$, and the relation path involves only the relations of a path $Rel(p) = (r_1, \dots, r_{L-1})$.

Path Embedding Distance. While NGED measures the differences between two contextual paths referring to a knowledge graph structure, it lacks an explicit semantic comparison between two different entities or relations. Thus, we propose the **Path Embedding Distance (PED)** metric to measure the semantic discrepancies between two paths using embeddings.

The key idea of PED is to learn the embedding representation of each path and measure the embedding distance between the generated path and ground truth path. We train an LSTM-VAE to encode paths into their embedding representations independent of their path lengths. The path representations generated by LSTM-VAE are aligned semantically such that paths of similar semantics will have similar representations. Using our pre-trained LSTM-VAE, we could obtain the representation of generated path Z_i as well as the ground truth path Z_i^{gt} . The path embedding distance could thus be defined by the cosine distance of the two:

$$PED(p_i, p_i^{gt}) = 1 - |Cosine\ Similarity(Z_i, Z_i^{gt})| \quad (5.17)$$

Learning of LSTM-VAE. To encode a path $p_k = \langle e_h, r_1, e_2, r_2, \dots, r_{L-1}, e_T \rangle$, we first break up p_k into a sequence of (entity, relation) tuples and feed each tuple to a Long Short-Term Memory (LSTM), one at each step. That is, p_k will be divided into $(e_h, r_1), (e_2, r_2), \dots, (e_{L-1}, r_{L-1})$, and e_t tuples which are then fed to LSTM. In each step, the LSTM layer encodes a tuple of the path, and generates the representation to be passed on to the next step. The representation generated at the final step after feeding e_t to the LSTM is the overall path representation Z_k^{pt} .

Let p_k be an input path and its path elements be represented by $X_{p_k} = \{w_1, \dots, w_L\}$ where w_t is the path element of step t . The decoder of LSTM-VAE works as a standard language model [8] as follows:

$$\mathbf{p}_\theta(p_k | Z_k^{pt}) = \prod_{l=1}^L \mathbf{p}_\theta(w_l | w_1 : w_{l-1}, Z_k^{pt}) \quad (5.18)$$

The loss function for LSTM-VAE \mathcal{L}_{pt} is defined to consist of a reconstruction loss and a regularizer term as shown below:

$$\mathcal{L}_{\text{pt}} \geq \mathbb{E}_{\mathbf{q}_\phi} \sum_{p_k \in P} \log_{\mathbf{p}_\theta}(X_{p_k} | Z_k^{pt}) - D_{KL}(\mathbf{q}_\phi(Z^{pt}|X) \parallel p(Z^{pt})) \quad (5.19)$$

where θ and ϕ are the parameters of the decoder and encoder, respectively. Z_k^{pt} is the latent representation of p_k deriving from

$$\mathbf{Z}_k^{pt} = \mu_k^{pt} + \sigma_k^{pt2} \odot \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, I) \quad (5.20)$$

5.3.3 Models for Comparison

We include all the three categories of CPG models in our experiments, namely: (1) CPG-Base, (2) CPG-BERT, and (3) CPG-Mixed. The dimensionality of sentence embedding used in these models is $\mathbf{D}^s = 256$. Each transformer encoder/decoder consists of $\mathbb{L} = 6$ layers. For the attention models $attn_*$, the dimensionality of key and value are both 32. For all non-linear transformation layers $FFN_*(\cdot)$ we set the input and output dimension to be 256, and the inner-layer dimension to be 2048. We set the penalization factor $\beta = 0.5$ as this yields the best performance during tuning. We empirically set $K = 20$, $p^{rw} = 0.4$, $d^{rw} = 10$ for the random walk in the reward scaling scheme.

- **CPG-Base.** As baselines, we have CPG-Base(TransE) and CPG-Base(Wiki2vec) represented as Trans(E) and Wiki2vec(EW) respectively.
 - TransE is a knowledge graph embedding method that learns the em-

beddings of entities in G by modeling relations as translation operations. A sentence’s embedding is derived by averaging the embeddings of entities in the sentence. That is, $s_i = \text{Avg}_{e \in E_i} \text{emb}_{\text{TransE}}(e)$ where E_i denotes the entities in sentence \mathbf{s}_i , and $\text{emb}_{\text{TransE}}(e)$ denotes the TransE embedding of entity e .

- **Wikipedia2vec** (or **Wiki2vec**) is specially designed for the knowledge graph constructed from Wikipedia data. It jointly models words and entities of a knowledge graph G in the same embedding space. The sentence embedding is the averaged embedding of both entities and words presented in the sentence. That is, $s_i = \text{Avg}_{e \in E_i \cup W_i} \text{emb}_{\text{Wiki2vec}}(e)$ where W_i denotes the set of words in sentence \mathbf{s}_i , and $\text{emb}_{\text{Wiki2vec}}(e)$ denotes the Wiki2vec embedding of entity e .

We train the TransE model using the input knowledge graph (DBpedia). We use the publicly available pre-trained Wiki2vec [128]. The parenthesized suffix ‘(E)’ (or ‘(EW)’) indicate if the information used covers entity embedding only (or both word and entity embeddings).

- **CPG-BERT.** Here, we have our CPG-BERT models using contextual word embeddings. We use BERT-mini from <https://github.com/google-research/bert> (256 dimensions with 4 layers) for the BERT-encoder to generate sentence embeddings. For simplicity, we use BERT(W) to denote this model.
- **CPG-Mixed.** We have two CPG-Mixed models, CPG-Mixed(TransE) and CPG-Mixed(Wiki2vec), denoted by M-TransE(EW) and M-Wiki2vec(EW) respectively. The two models use BERT-mini for BERT-encoded sentence embedding.

We also introduce two baseline models that do not follow our proposed CPG architecture for comparison, namely:

- **GPT.** We fine-tune a GPT model to generate knowledge graph paths. We provide the head and tail entities as prefix input in the form of “⟨soe⟩ e_H

$\langle \mathbf{SEP} \rangle e_T \langle \mathbf{eoe} \rangle$ ” to GPT and fine-tune it to generate the contextual paths between the head and tail entities. Since there is no context given to this model, the generation is prone to random generation.

- **GPT+Context.** We further fine-tune another GPT model (denoted by GPT+Context) that takes both query entities and the context document as input in the form of “ $\langle \mathbf{soe} \rangle e_H \langle \mathbf{SEP} \rangle e_T \langle \mathbf{eoe} \rangle \langle \mathbf{sod} \rangle d \langle \mathbf{eod} \rangle$ ”. This GPT model is also fine-tuned to generate a random path between the head and tail entities.

5.3.4 Comparison of CPG Models

In the first set of experiment results shown in Table 5.2, the various CPG models with both penalization and reward scaling schemes are evaluated and compared. The best results are indicated in **boldface**. We discuss the result findings below.

Overview of Model Performances. Table 5.2 shows that CPG-Mixed with Wikipedia2vec (M-Wiki2vec(EW)) is the best-performing model by all metrics recovering 63.7% and 69.9% of ground truth paths for datasets Wiki-film and Wiki-music, respectively. This is followed by CPG-Mixed with TransE as the base embedding (M-TransE(EW)) which achieves 63.3% and 67.4% for Wiki-film and Wiki-music datasets, respectively. M-Wiki2vec(EW) and M-TransE(EW) also outperform the other models in other metrics, i.e., NGED, PED, %Uf, and %Lp. Overall, CPG-Mixed models outperform the CPG-BERT model, BERT(W), and BERT(W) in turn outperforms CPG-Base models, TransE(E) and Wiki2vec(EW). Fine-tuned GPTs, GPT and GPT+Context, perform the worst as they do not have a mechanism to ensure always generate a relevant path. TransE(E) recovers only 34.5% and 36.7% of ground truth paths for Wiki-film and Wiki-music datasets, respectively. This is reasonable since BERT-encoder and mixed encoder consider contextualized word embeddings unlike the base encoder treating every instance of the word or entity the same.

Table 5.2: Experiment Results (Best results are shown in boldface.)

Models	NGED		PED	%Gt	%Uf	%Lp
	Ent	Rel				
Wiki-film						
Fine-tuned GPTs						
GPT	0.46**	0.35**	0.39**	8.1**	9.3**	38.4**
GPT+Context	0.33**	0.29**	0.37**	13.4**	15.4**	32.1**
CPG-Base						
TransE(E)	0.23**	0.20**	0.34**	34.5**	19.4**	12.1**
Wiki2vec(EW)	0.21**	0.18**	0.29**	40.5**	19.2**	11.8**
CPG-BERT						
BERT(W)	0.17**	0.14*	0.25**	46.5**	14.6**	10.1**
CPG-Mixed						
M-TransE(EW)	0.14	0.13*	0.22*	63.3	11.6	7.3
M-Wiki2vec(EW)	0.14	0.12	0.20	63.7	10.3	7.3
Wiki-music						
Fine-tuned GPTs						
GPT	0.43**	0.31**	0.37**	9.2**	10.4**	27.4**
GPT+Context	0.36**	0.27**	0.37**	14.8**	17.2**	29.5**
CPG-Base						
TransE(E)	0.24**	0.19**	0.31**	36.7**	17.6**	11.5**
Wiki2vec(EW)	0.20**	0.18**	0.31**	42.4**	17.3**	10.9**
CPG-BERT						
BERT(W)	0.18**	0.14**	0.24**	48.2**	13.9**	8.7**
CPG-Mixed						
M-TransE(EW)	0.14	0.14	0.23*	67.4*	10.8	6.8
M-Wiki2vec(EW)	0.13	0.13	0.21	69.9	10.9	6.7
<p>W: Word only. E: Entity only. EW: Use both entity and word. NGED: Normalized Graph Edit Distance, %Gt: % ground truth paths recovered. %Uf: % of unfinished paths, %Lp: % of loopy paths. **: Significance test between M-Wiki2vec(EW) and other models with p-value ≤ 0.05 *: Significance test between M-Wiki2vec(EW) and other models with $0.05 < \text{p-value} \leq 0.1$</p>						

We conduct paired t-test to compare the results of each baseline model with the best performing model, i.e., M-Wiki2vec(EW). As shown in Table 5.2, in

Table 5.3: Ablation of Penalization/Reward Scaling (Wiki-film)

Models	NGED		%Gt	%Uf	%Lp
	Ent	Rel			
CPG-Base					
TransE(E)	0.23	0.20	34.5	19.4	12.1
TransE(E) w/o PR	0.28	0.25	23.7	36.7	24.6
TransE(E) w/o R	0.28	0.25	25.1	33.2	17.3
TransE(E) w/o P	0.27	0.25	27.1	23.3	22.9
Wiki2vec(EW)	0.21	0.18	40.5	19.2	11.8
Wiki2vec(EW) w/o PR	0.25	0.24	29.7	34.9	23.8
Wiki2vec(EW) w/o R	0.25	0.24	33.1	35.7	17.2
Wiki2vec(EW) w/o P	0.25	0.23	34.2	22.6	24.1
CPG-BERT					
BERT(W)	0.17	0.14	46.5	14.6	10.1
BERT(W) w/o PR	0.23	0.18	29.3	31.7	19.9
BERT(W) w/o R	0.22	0.17	35.3	28.5	15.4
BERT(W) w/o P	0.21	0.17	33.9	22.6	17.8
CPG-Mixed					
M-TransE(EW)	0.14	0.13	63.3	11.6	7.3
M-TransE(EW) w/o PR	0.19	0.16	45.2	24.9	14.2
M-TransE(EW) w/o R	0.18	0.16	48.8	22.7	11.2
M-TransE(EW) w/o P	0.17	0.15	50.1	16.2	13.7
M-Wiki2vec(EW)	0.14	0.12	63.7	10.3	7.3
M-Wiki2vec(EW) w/o PR	0.19	0.15	47.3	23.4	14.0
M-Wiki2vec(EW) w/o R	0.18	0.14	49.1	21.1	9.5
M-Wiki2vec(EW) w/o P	0.17	0.15	49.7	14.3	12.5

W: Word only. **E:** Entity only. **EW:** Using both entity and word.
P: Penalization scaling. **R:** Reward scaling. **w/o:** without.

both Wikinews datasets, M-Wiki2vec(EW) performs better than all other non CPG-Mixed models with p-value ≤ 0.05 . Among the CPG-Mixed models, M-Wiki2vec(EW) still outperforms M-TransE(EW), albeit not significantly in most of the metrics. NGED, PED, and %Gt are the only three metrics that M-Wiki2vec(EW) significantly outperforms with p-value ranges from 0.5 to 0.1.

Choices of entity, word, and BERT embeddings. In this analysis, we evaluate how the choice of embeddings in context encoding could affect model performance. We focus on CPG models with both penalization and reward scaling schemes only as shown in the top section of Table 5.2. The results show that

models using embeddings trained on both entity and word embeddings (i.e., Wiki2vec(EW), M-TransE(EW), and M-Wiki2vec(RW)) outperform those using entity embeddings only (i.e., TransE(E)).

For example, CPG-Base using Wiki2vec (Wiki2vec(EW)) outperforms that using TransE (TransE(E)). The same findings are observed between CPG-Mixed models (i.e., M-Wiki2vec(EW) outperforms M-TransE(EW)). The NGED and %GT results of M-TransE(EW) are poorer than M-Wiki2vec(EW) but the difference is less obvious compared to that between TransE(E) and Wiki2vec(EW).

5.3.5 Effects of Penalization/Reward Scaling

Here, we compare the models with penalization and reward scaling with those without (i.e., without penalization scaling **w/o P**, those without reward scaling **w/o R**, and those without both scaling **w/o PR**). We only show the experiment results conducted on the Wiki-film dataset, since the results for the Wiki-music dataset are similar. Table 5.3 shows the results of the ablation of penalization and reward scaling. For easy reading, we also show the results with both scaling schemes in boldface.

The results show that by not applying these scaling schemes (i.e., w/o PR), both %Uf and %Lp deteriorate substantially for all the models. The largest increase for %Uf and %Lp are 117% (from 14.6% to 31.7%) and 110% (from 10.1% to 19.9%) respectively for BERT(W) w/o PR. Smaller deteriorations in %Uf and %Lp can still be observed if only one of the scaling schemes is applied. Models without scaling also suffer substantial decline in %GT. This result shows that our scaling schemes effectively direct the models to generate paths toward the tail entities without loops. As a side effect, we also see improvement in NGED.

5.3.6 Effects of Context Document Content Amount

In the earlier experiments, contextual paths are generated using the entire context document as input. However, many real-world use cases expect sparse and noisy content in the context documents. Thus, in this section, we examine how a CPG model behaves when varying the amount of data in the context document using Wiki-film dataset. In this study, we only report the results using the best performing model M-Wiki2vec(EW) with both penalization and reward scalings. We experiment with four different settings, namely:

- (a) **x**: a zero vector as context representation simulating no context is given;
- (b) **ment_sent**: we only use sentences covering the head and tail entities e_H and e_T as context; and
- (c) **rand50**: we randomly sample 50% of the news article as context.
- (d) **full**: we use the full news article as the context document.

Note that both settings (b) and (c) simulate a scenario of limited context with relevant content, and a scenario of half the amount of context, respectively.

Table 5.4 shows the NGED and %Gt results of M-Wiki2vec(EW) under different settings. We also illustrate the paths generated by M-Wiki2vec(EW) for an example input consisting of $e_H = \text{Campbell}$, $e_T = \text{Brosnan}$ and our James Bond context document example in Figure 5.1. Under the **x** context setting, M-Wiki2vec(EW) tends to choose popular relations as there is no context information at all. On the other hand, entities that are more likely to lead to the tail entity are preferred due to the reward scaling. Therefore, the model usually generates shortest paths with popular relations/entities. In our James Bond example, M-Wiki2vec(EW) generates a shortest path between the entities Campbell and Brosnan, which is unfortunately not the ground truth.

The **rand50** setting yields better performance than no context setting (**x** in Table 5.4). Nevertheless, if the context sampled is irrelevant to the ground truth,

Table 5.4: Effects of Context Content Amount on Wiki-film with M-Wiki2vec(EW)

Ground Truth: Campbell \xrightarrow{d} Casino Royale \xrightarrow{s} Daniel Craig \xrightarrow{pb} James Bond \xrightarrow{pb} Brosnan			
Context	NGED(R/E)	%Gt	Path Generated
x	0.35 / 0.32	12.1	Campbell \xrightarrow{d} GoldenEye \xrightarrow{s} Brosnan
rand50	0.31 / 0.28	14.9	Campbell \xrightarrow{d} Casino Royale \xrightarrow{p} Barbara Broccoli \xrightarrow{p} GoldenEye \xrightarrow{s} Brosnan
ment_sent	0.18 / 0.21	44.2	Campbell \xrightarrow{d} Casino Royale \xrightarrow{s} Daniel Craig \xrightarrow{pb} James Bond \xrightarrow{pb} Brosnan
full	0.14 / 0.12	63.7	Campbell \xrightarrow{d} Casino Royale \xrightarrow{s} Daniel Craig \xrightarrow{pb} James Bond \xrightarrow{pb} Brosnan

d= director, s= hasactor, p= producer, pb= portrayBy

irrelevant entities or relations will also be included in the generated paths. For the example query, the generated path contains some irrelevant entities (i.e., Barbara Broccoli and GoldenEye) and is longer.

Finally, under the **ment_sent** setting, the paths generated are quite similar to **full** context setting although the % of ground truth paths recovered by the **ment_sent** setting (i.e., 44.2%) is still clearly lower than that of **full** (i.e., 63.7%). Since only sentences with entity mentions are covered in this setting, there is a drop in overall accuracy. Nevertheless, we still observe few cases where the generated path with **ment_sent** setting is identical to that generated with the **full** setting because some noisy information in the context might have been removed. In our example, the model using the **ment_sent** setting manages to generate the ground truth path.

5.3.7 Error Type Analysis

To better understand the difference among the CPG models, we analyze the errors in the generated contextual paths using Wiki-film dataset. We randomly sample 100 erroneous paths generated from each model and conduct a qualitative study which involves inspecting the paths manually. We assign one or more of the following error types to each path:

- E1: *Loopy path* (see Section 5.2.4 for definition)
- E2: *Unfinished path* (see Section 5.2.5 for definition)

Table 5.5: Error Type Analysis on Wiki-film dataset

Models		E1	E2	E3	E4
GPT Baseline	GPT	34%	11%	47%	9%
	GPT+Context	29%	16%	42%	7%
CPG-Base	TransE(E)	20%	30%	35%	16%
	Wiki2vec(EW)	20%	32%	31%	20%
CPG-BERT	BERT (W)	19%	27%	27%	25%
CPG-Mixed	M-TransE(EW)	23%	25%	13%	35%
	M-Wiki2vec(EW)	20%	28%	15%	32%

- E3: *Non-loopy path that still contains extra relations upon human judgement.*

E.g., The path $Actor \xrightarrow{\text{has actor}} Movie_A \xrightarrow{\text{director}} Director \xrightarrow{\text{director}} Movie_B$ is of E3-type as the ground truth is $Actor \xrightarrow{\text{has actor}} Movie_B$.

- E4: *Path with exactly one erroneous relation or entity.*

E.g., $Actor \xrightarrow{\text{has actor}} Movie_A \xrightarrow{\text{director}} Director$ instead of $Actor \xrightarrow{\text{has actor}} Movie_B \xrightarrow{\text{director}} Director$

Table 5.5 shows the distribution of E1-E4 errors among the 100 erroneous paths per model. Note that there are other forms of errors we do not consider in this error analysis, and more than one error type may be assigned to each path. As a result, the numbers in Table 5.5 will not add up to 100%.

From Table 5.5, we make the following observations. Firstly, the erroneous paths generated by TransE(E) and Wiki2vec(EW) are more likely to have non-loopy path with extra relations (E3) compared to the ones generated by BERT(W), M-Trans(EW) and M-Wiki2vec(EW). However, they have less E4 errors compared to the rest. Secondly, among the CPG-Base models, Wiki2vec(EW) enjoys a smaller proportion of E3 errors (31%) than that of TransE(E) (35%). This is due to Wiki2vec(EW) being semantically enriched by knowledge graph embedding. BERT(W) also sees some improvement in E3 due to its pre-trained contextual word embeddings. Using both entities and words for sentence embedding, M-Wiki2vec(EW) reduces E3 error to 15% only. Finally, it is interesting to find

the proportion of E4 errors higher for models with better CPG performance. Hence, it is not a surprise to see M-Wiki2vec(EW) yielding the highest E4 error rate as the model has already proportionally done well in other easier error types (E1 to E3).

5.3.8 Inferring Relations in the Wiki-film Dataset

We next conduct a qualitative study on the Wiki-film dataset to examine inferred relations in the generated paths. Using our James Bond context document example, the M-Wiki2vec(EW) model generates a path for $e_H = \text{BBC}$ and $e_T = \text{Craig}$ which co-occur in the context document segment “*Craig’s film credits include... his major breakthrough was a starring role in the 1996 **BBC** drama Our Friends in the North.*” The generated contextual path is:

$$\text{BBC} \xrightarrow{\text{studio}} \text{Our Friends in the North} \xrightarrow{\text{hasactor}} \text{Craig}$$

This path explains the connection between these two entities perfectly, and the relation $\text{BBC} \xrightarrow{\text{studio}} \text{Our Friends in the North}$ does not exist in our knowledge graph but is semantically appropriate.

Note that another path that connects BBC and Craig exists in the knowledge graph:

$$\begin{aligned} \text{BBC} \xrightarrow{\text{owningCompany}} \text{BBC_Two} \xrightarrow{\text{channel}} \text{Our Friend in the North} \\ \xrightarrow{\text{hasactor}} \text{Craig} \end{aligned}$$

However, it does not match the context as well as our generated contextual path as the relations involved are less direct.

5.4 Experiment on a Synthetic Dataset

While the two Wikinews datasets are constructed with ground truth contextual paths, they are small and do not allow us to study the CPG model performance at scale. In this section, we therefore conduct another set of experiments using

a synthetic dataset, Synthetic-S. Synthetic-S is about 50 times larger by number of context documents and 4 times larger by number of head-tail entity pairs. We describe the process of constructing the synthetic dataset in Appendix A.

5.4.1 CPG Model Performance on the Synthetic-S Dataset

With the synthetic dataset, we conduct the evaluation of our proposed CPG-models as well as the GPT baseline models. We report their NGED, %Gt, %Uf, and %Lp performance in Table 5.6. The results are generally consistent with that on the Wiki-film and Wiki-music datasets. M-Wiki2vec(EW) again outperforms all other model variants across all metrics except for %Lp. M-TransE(EW) and BERT(W) remain to be the second and third best-performing models, respectively. Significance test shows that M-Wiki2vec(EW) has significantly better performance than non CPG-Mixed models with p value ≤ 0.05 . Unsurprisingly, GPT and GPT+context are the worst performing models. On the other hand, our proposed CPG models still generate decent contextual paths that are complete, loopless, and matching the context.

Nevertheless, we observe that the performance results of all models are generally lower for the synthetic dataset compared with those for Wiki-film dataset. For instance, the best %Gt result from M-Wikipedia(EW), 53.3%, is substantially lower than what the same model can achieve for the Wiki-film dataset (%Gt=63.7). This is because the generation of paths may be affected by the noisy sentences sampled during the document generation process.

As in our experiments on Wiki-film and Wiki-music datasets, we also conduct significance tests to compare the results of each baseline model with that of M-Wiki2vec(EW). This significance test results show that M-Wiki2vec(EW) performs significantly better than all baselines with p -value ≤ 0.05 . Among the CPG-Mixed models, M-Wiki2vec(EW) outperforms M-TransE(EW) but their difference is not significant.

Table 5.6: Experiment Result: Synthetic-S Dataset

Models	NGED		PED	%Gt	%Uf	%Lp
	Ent	Rel				
GPT Baseline						
GPT	0.53**	0.37**	0.4**	11.4**	7.7**	31.4**
GPT+context	0.41**	0.25**	0.38**	15.4**	14.6**	30.9**
CPG-Base						
TransE(E)	0.26**	0.21**	0.31**	38.6**	19.6**	13.7**
Wiki2vec(EW)	0.22**	0.18**	0.28**	40.3**	18.2**	12.7**
CPG-BERT						
BERT(W)	0.18**	0.15**	0.23**	45.6**	15.3**	9.1*
CPG-Mixed						
M-TransE(EW)	0.15	0.14	0.20	51.4*	15.3*	8.8
M-Wiki2vec(EW)	0.15	0.13	0.20	53.3	14.7	8.9

W: Word only. **E:** Entity only. **EW:** Use both entity and word.

** : Significance test between M-Wiki2vec(EW) and other models with p-value ≤ 0.05

* : Significance test between M-Wiki2vec(EW) and other models with $0.05 < \text{p-value} \leq 0.1$

5.4.2 Coping with Incomplete Knowledge Graphs

In this experiment, we examine how a proposed CPG model copes with incomplete knowledge graphs. We specifically want to study knowledge graphs with an incomplete set of relation edges. Under this setting, a CPG model will have to infer the missing relation edges as it seeks to generate the contextual path results. Hence, we divide this section into two parts. The first part involves an experiment to evaluate how a CPG model performs for different extents of missing edges in the knowledge graph. The second part performs a qualitative evaluation of the inferred relation edges. We will use M-Wiki2vec(EW) as our representative CPG model.

In the first part, we evaluate the performance of M-Wiki2vec(EW) by randomly removing k percent of the edges ($10 \leq k \leq 50$) from the knowledge graph. We use the synthetic dataset in this evaluation and report the perfor-

Table 5.7: Experiment Result: CPG with Incomplete Knowledge Graphs (Synthetic-S, M-Wiki2vec)

k	NGED		PED	%GT	% CP Edges Removed	% Removed CP Edges Recovered
	Ent	Rel				
0 (Full KG)	0.15	0.13	0.2	53.3	-	-
10	0.15	0.14	0.21	45.2	5.37	63.7
30	0.19	0.19	0.25	36.4	19.31	56.3
50	0.28	0.21	0.32	27.7	38.45	43.4

CP Edges: Edges exist in the ground truth contextual path

mance of M-Wiki2vec(EW) in Table 5.7. The results show that compared with the knowledge graph with $k = 0\%$ missing edges, M-Wiki2vec(EW) sees poorer performance as k increases. We also note that when k is small, as much as 63.7% of the edges removed from contextual paths can be correctly recovered. For larger k settings, the proportion of correctly recovered contextual path edges removed from the knowledge graph decreases as expected.

In the second part, we empirically observe that M-Wiki2vec(EW) is more likely to recover edges between entities which are logically or semantically related. For instance, we find that a movie m is inferred to be made in country c (i.e., $m \xrightarrow{\text{country}} \text{France}$) if it mostly stars actors and actresses from c . As a result, M-Wiki2vec(EW) successfully recovers Top Gun $\xrightarrow{\text{country}}$ United States, and Amour(film) $\xrightarrow{\text{country}}$ France. On the other hand, the model fails to recover edges like Ridley Scott $\xrightarrow{\text{producer}}$ American Gangster(film), Crimson Tide $\xrightarrow{\text{starring}}$ Viggo Mortensen, and Clint Eastwood $\xrightarrow{\text{artist}}$ Bar Room Buddies. These are edges with specific semantics and are thus more difficult to infer.

5.5 Using a KGQA Model for CPG: A Model Adaptation Experiment

CPG has some resemblance to some question answering (QA) tasks if we view the input context document, head and tail entities forming the question, question

entity, and answer entity, respectively. CPG is however significantly different from QA tasks (including knowledge graph QA) as the latter seek knowledge graph entities as answers to a given question but not the correct path connecting entities in the input context document.

In this section, our goal is to examine how a state-of-the-art QA model that derives answers from a knowledge graph performs in the CPG task after some adaptation. Among the many QA tasks, QA over knowledge graphs or KGQA is most related to CPG tasks. KGQA aims to find an answer entity from a knowledge graph given a query. The query includes a question as the context and a question entity mentioned in the context. While the question entity is analogous to the head entity of CPG tasks, KGQA does not involve any tail entity. Instead of returning a contextual path, KGQA requires all candidate answer entities to be ranked and the highest ranked one will be returned as the predicted answer. We choose to adapt EmbedKGQA, a state-of-the-art KGQA model proposed by Saxena, Tripathi, and Talukdar for CPG [93]. In the following, we outline the changes made to EmbedKGQA model and our experiment results.

5.5.1 Modified EmbedKGQA Model

EmbedKGQA is an embedding-based model trained on the MetaQA dataset [139]. In the original KGQA setting, a training instance consists of an entity e_h and a piece of text q as the question, and another entity e_t as the answer. EmbedKGQA assumes that there is a learned knowledge graph embedding model $Z = \{Z_e, Z_r\}$ where $z_e \in Z_e$ is the entity embedding of entity e and $z_r \in Z_r$ is the relation embedding of relation r . To predict the answer to a question, EmbedKGQA learns to embed the question q as a fixed dimension vector z_q by ROBERTa so that the answer score $\phi(z_e^h, z_q, z_e^t)$ is maximized. EmbedKGQA uses ComplEx as the base knowledge graph embedding model, and adopts its

scoring function as the answer scoring function:

$$\phi(z_e^h, z_q, z_e^t) = \text{Re}\left(\sum_{k=1}^n z_e^{h(k)} z_q^{(k)} \bar{z}_e^{t(k)}\right) \quad (5.21)$$

where $z_e^h, z_e^t, z_q \in \mathbb{C}^n$, $z^{(k)}$ refers to the k -th dimension of z , and $\bar{z} = \text{Re}(z) - i\text{Im}(z)$. $\text{Re}()$ is a function that only returns the real value of the input complex number and $\text{Im}()$ returns the imaginary value. By maximizing this scoring function, EmbedKGQA learns to represent the question as a relation in the knowledge graph embedding space that connects the question entity and answer entity. The trained EmbedKGQA model can then be used to determine the answer entity of a new question entity $e_{h'}$ and question q' by selecting the answer entity with the highest $\phi()$ score as follows:

$$e_{t'} = \max_{t'} \phi(z_e^{h'}, z_{q'}, z_e^{t'}) \quad (5.22)$$

Although EmbedKGQA is easy to generalize to other QA datasets, it cannot be directly utilized to solve the CPG task. Hence, we made three major modifications to the model. First, as the question q in KGQA is similar to the context document d in CPG, we use the question embedding module in EmbedKGQA to encode the context document. The context document embedding module encodes the context document to an n -dimension vector following the same process as the original question embedding module. We use the exact parameter setting as in EmbedKGQA [93].

Second, EmbedKGQA uses all entities in the knowledge graph as candidate answer entities, but it is unclear how it performs the negative sampling process. We therefore employ a pre-filtering process to sample negative answers. Let E_h denote the set of 1- to 3-hop neighbors of e_h . We define two types of negative samples: (1) entities outside E_h , and (2) entities within E_h but which do not exist in the ground truth contextual path. For each query, we sample 20 negative answers from these two sets of entities with a 1:1 ratio.

Finally, although EmbedKGQA tries to identify entities in the knowledge graph that are most relevant to the question, it does not generate a path between the question entity and answer entity. Therefore, we propose a simple contextual path construction algorithm which connects entities to form a path. As shown in Algorithm 2, the path construction process starts with ranking entities and relations among 1-hop neighbors of e_h . It then chooses the most context-relevant entity e^* to be the next element in the path. To connect e_h and e^* , we find the most context-relevant relation among relations between them in the knowledge graph. The algorithm then uses e^* as new e_h and repeats itself until it reaches e_t or when the path length reaches the maximum.

Algorithm 2: Construction of Contextual Path with the Modified EmbedKGQA

input : Head entity e_h , Tail entity e_t , Context document d , Maximum length of path L_{MAX}
output: Contextual Path p

```

 $p = \{e_h\}$ 
while  $length(p) \leq L_{MAX}$  or  $e_h \neq e_t$  do
    Extract  $G_h = \{E_h, R_h\}$ , the 1-hop subgraph surrounding  $e_h$ 
     $e^* = \mathit{argmax}_{e' \in E_h} \phi(e_h, d, e')$ 
     $r^* = \mathit{argmax}_{r' \in R_h, e_h \xrightarrow{r'} e^*} S(r', d)$ 
     $p.insert(r^*)$  // to insert  $r^*$  into  $p$ 
     $p.insert(e^*)$  // to insert  $e^*$  into  $p$ 
     $e_h \leftarrow e^*$ 
end

```

5.5.2 Experiment and Results

We compare EmbedKGQA with the GPT baselines and our proposed CPG models on the Wiki-film dataset only. Additionally, we introduce a model variant “EmbedKGQA no-revisit” where revisiting of entities is strictly prohibited. When constructing the contextual path with Algorithm 2, we exclude every entity that exists in p from E_h to avoid creating loops. As a result, this model will not generate any loop and potentially improve the quality of generated paths.

Table 5.8: Experiment Results: CPG with QA Model (Wiki-film)

Models	NGED		PED	%GT	%Uf	%Lp
	Ent	Rel				
GPT Baseline						
GPT	0.45	0.35		10.2	11.3	35.5
GPT+Context	0.32	0.27		16.2	15.8	29.4
QA Model						
EmbedKGQA	0.28	0.23		25.3	39.2	19.5
EmbedKGQA No-revisit	0.27	0.22		28.8	37.7	0
CPG-Base						
TransE(E)	0.21	0.19		39.7	19.2	10.2
Wiki2vec(EW)	0.2	0.18		42.9	18.4	9.4
CPG-BERT						
BERT(W)	0.16	0.14		53.1	14.5	8.3
CPG-Mixed						
M-TransE(EW)	0.13	0.12		68.8	12.2	5.9
M-Wiki2vec(EW)	0.12	0.12		69.3	10.7	5.6

W: Word only. **E:** Entity only. **EW:** Use both entity and word.

We show the results in Table 5.8. EmbedKGQA outperforms the GPT baselines in NGED, PED, and %Ground truth path. However, it underperforms all other CPG models in % unfinished path (%Uf) due to the lack of a proper path construction method. %Loops wise, EmbedKGQA’s performance lies between GPT baselines and CPG models. EmbedKGQA is able to identify relations and entities that match the context better, resulting in lower NGED and PED. However, it does not guarantee a high-quality contextual path even with our proposed path construction algorithm. The proportion of unfinished paths %Uf and loopy paths %Lp are thus higher compared to our CPG models. While we reduce %Lp to zero in EmbedKGQA no-revisit by introducing the no-revisit policy, it does not significantly improve the model performance in other metrics such as NGED and %Uf.

The experiment results suggest the uniqueness of CPG makes it difficult for both the vanilla and modified QA models to be used as solutions. Specifically,

CPG requires the construction of high-quality contextual paths, which is often not considered in the existing QA works.

5.6 Summary

In this chapter, we formally define the Contextual Path Generation (CPG) problem and proposed a transformer-based architecture to address three challenges: (a) sparse and noisy context document, (b) incomplete knowledge graph, and (c) well-formedness of the generated path. Our contribution in this chapter can be summarized as follows.

- To address the first challenge, we design a transformer-based model architecture to encode a context document d for input to the generation of a contextual path. In particular, we develop context encoders using knowledge graph entity and word embeddings, and pre-trained contextualized word embeddings.
- We develop an autoregressive controlled path generation approach to generate contextual paths with the encoded d , e_H and e_T . As this approach can infer missing relations in the knowledge graph, it addresses the second challenge. We also introduce penalization and reward scaling strategies to reduce the sampling of already generated entities and to increase the likelihood of reaching e_T respectively in path generation. The two strategies address the challenge of generating well-formed contextual paths,
- Based on our proposed model architecture, we derive several CPG models with different combinations of context encoder and controlled path generation options.
- We also construct two real datasets and a large synthetic dataset. Each dataset consists of a set of queries and their “ground truth” contextual

paths for model training and experiment evaluation. These datasets will be made publicly available along with the publication of this chapter.

- Our experiments show that several of our proposed CPG models perform reasonably well in returning the ground truth contextual paths. The result also shows that the penalization and reward scaling strategies can effectively reduce unfinished paths by as much as 52.6% and loops by as much as 53.5%. We also show the model performance under different input context settings.
- To show the difference between CPG and QA tasks, we modify a state-of-the-art QA model, EmbedKGQA, for CPG tasks [69, 139]. Our experiment results show that while CPG shares some similarities with knowledge graph QA, the QA solution models are not likely to perform well for CPG even after some adaptation.

While there is still room for performance improvement, this CPG research also opens up a number of new research directions. Firstly, CPG creates a new way of knowledge graph reasoning induced by external textual content. The inferred relations in the generated contextual paths can be used to supplement the existing knowledge graph. Nevertheless, there are still interesting research questions on how these inferred relations can be aid the other downstream information retrieval tasks to offer explanations or to improve the tasks' performance; and how these inferred relations can be aggregated from the query results on large number of context documents to create a knowledge graph from scratch or to update an existing knowledge graph.

Secondly, our proposed approach handles both unfinished path and looping problems by scaling strategies. Further research can be conducted to allow the controlled path generation step to directly produce loopless and finished contextual paths. This may further enhance the quality of resultant contextual paths.

Finally, CPG can be further extended to generate contextual subgraphs, as

opposed to contextual paths. This will require the input to be a set of entities and the generation process to consider the graph structure. To our best knowledge, research on controlled graph or subgraph generation is still at its nascent stage and will likely become an important research topic in the future.

Chapter 6

Contextual Path Generation: A Non-Monotonic Approach

In previous chapter, we introduce a generation model that address the contextual retrieval task CPG. Still, how to generate a high-quality contextual path that is both complete and loopless remains a critical issue. In this chapter, we attempt to address CPG with a novel generation model, named Non-Monotonic Contextual Path Generation (NMCPG). As the name suggests, NMCPG generates a knowledge graph non-monotonically. This ensures that the generated contextual paths always start and end with the query entities. Unlike the monotonic CPG, NMCPG is based on a pre-trained generation model that learns to generate high quality knowledge graph path. To cope with the NMCPG model, we also design a two-stage framework that incorporate knowledge graph and context document differently compared to CPG. The pre-trained generation model is fine-tuned to embed the interaction between knowledge and text in the representation space.

6.1 Research Objective

6.1.1 Problem Definition

There are mainly three challenges we face when solving the CPG problem, namely (a) *noisy and sparse context document*, (b) *incompleteness of knowledge graph*, and (c) *well-formedness of generated contextual paths*.

The first challenge is caused by the document containing information that is both relevant and irrelevant to the query entities. A method that fails to extract the relevant information will thus generate low-quality paths. The challenge is further complicated by context documents that are short and is thus sparse in content. To address this challenge, we need to extract relevant semantics from the context to used as input condition to path generation.

Secondly, most knowledge graphs in the real world are often incomplete with missing relation edges. To cope with this issue, the framework is expected to infer new relation edges.

Thirdly, most models for sequence generation are designed for *single object-type left grounded* sequences. Single object-type sequence is one that contains elements of the same type. Left-grounded sequence is one that has leftmost element is fixed but not the remaining elements. Hence, these models could not generate contextual paths with entities and relations as alternating path elements, and always end with the tail entities, the two well-formedness criteria of CPG. Given a query entity pair (e_h, e_t) , a well-formed path is a path that start with e_h and ends with e_t . As non well-formed paths are considered incorrect, our goal is thus to design a generation model that always generates well-formed contextual paths.

To address the challenges and obtain high-quality contextual paths, we propose a two-stage framework that extracts context-relevant entities and generates the contextual path non-monotonically from a given initial state. As shown in

Figure 6.1, the framework consists of a *context extractor* and a *contextual path generator*. The former derives context entities $E(q)$ from the input knowledge graph G and query represented by a tuple (e_h, e_t, d) where d is the context document. The context entities capture the relevant query’s information, enrich the sparse content in context document with knowledge graph’s entities, and minimize noises found in the original context document. We propose two different methods for context extractor to extract from knowledge graph contextually relevant entities, one based on binary classification and another based on learning-to-rank on multi-head self attention of candidate entities. The latter leverages interaction between entities to enhance the relevance of resultant entities addressing the noisy and sparse context document challenge.

In addition, we introduce the contextual path generator which generates the contextual path $p_q = \langle e_q^1, r_q^1, \dots, e_q^{N_q} \rangle$ conditioned on the context entities. p_q is a sequence of entities and relations connecting from $e_q^1 = e_h$ to $e_q^{N_q} = e_t$. This generator is capable of inferring new relation edges which addresses our second challenge. For the contextual path generator module, we propose a novel method which generates the contextual path non-monotonically and also guarantees the generated contextual paths are well-formed addressing the final challenge.

6.2 Proposed Framework and Contextual Path Generation Models

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

6.2.1 Two-Stage Framework

Before we present the framework, we give the definition of knowledge graph. We define the *knowledge graph* G to be a tuple (E, L, R) where E denotes a

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

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

As shown in Figure 6.1, our proposed framework consists of a *context extractor* and a *contextual path generator*. The former derives *context entities* $E(q) = \{e_1, e_2, \dots, e_{|E(q)|}\}$ from the query q and input knowledge graph G . Context entities are entities relevant to the query and they may be mentioned in d . To enrich the sparse content of the context document, one should also select relevant entities from the knowledge graph to be context entities thereby addressing part of the path relevance challenge (as the remaining part is to be addressed by the contextual path generator).

The context entities together with the query are then provided to the contextual path generator to construct the resultant contextual path(s) leveraging the interaction between context entities to finally determine the relevant entities and

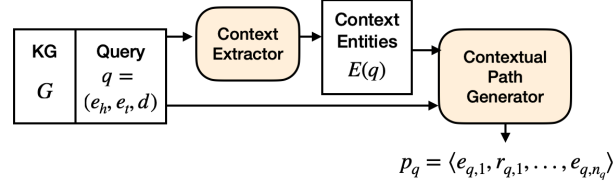


Figure 6.1: The Two-stage Framework for Contextual Path Generation

relation edges and sequence them to form the resultant path. In this process, new relation edges may be inferred to address the incomplete knowledge graph challenge. We further propose a non-monotonic path generation process to ensure that the resultant path is both semantically relevant and well-formed addressing the path relevance and path well-formedness challenges respectively.

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

6.2.2 Context Extractor

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

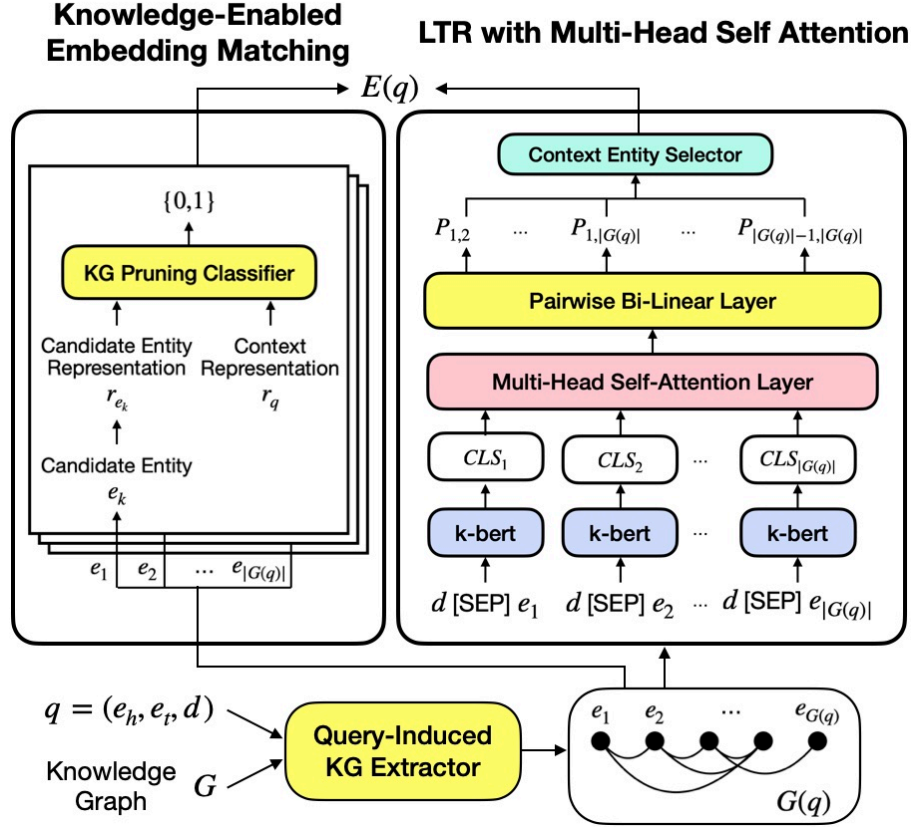


Figure 6.2: Context Extractors

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

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

KEMatch: Knowledge-enabled Embedding Matching

In KEMatch, we train a classifier to decide whether a candidate entity is related to the context or not. Thus, we introduce a simple binary classifier which takes an entity and the context as input, and outputs a probability between 0-1 that suggests the relatedness between the two inputs. Given $E^r(q)$ already derived

by the query-induced entity extractor, the classifier performs matching of every candidate entity e from $E^r(q)$ with the vector representation of the query z_q . We define this matching model by: $f_{kem}(z_e, z_q) \rightarrow \{0, 1\}$ where z_e and z_q denote the candidate entity representation and query representation respectively. The candidate entities are predicted with label 1 (or matching) if they are deemed as semantically relevant, and label 0 (or non-matching) otherwise. Only the predicted matching entities will be returned as the set of context entities $E(q)$. We denote the size of $E(q)$ by m .

KEMatch uses k-bert [74], a knowledge-enabled contextualized word embeddings, to derive z_q and z_e . We introduce four different query representation schemes:

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

ENC. z_q is then defined as the average k-bert embedding of word tokens in e_h, e_t and d .

The matching model of KEMatch f_{kem} is trained with a set of queries with their corresponding positive and negative entity sets denoted by $E(q)^+$ and $E(q)^-$ respectively. For a query q , we use p_q^* to denote the ground truth contextual path. $E(q)^+$ is then assigned the (positive) entities in p_q^* . To avoid false negative entities, $E(q)^-$ is assigned with a set of hard negative entities sampled from entities with few or no common neighbor with entities of $E(q)^+$ in the knowledge graph G . That is, $E(q)^- = \{e | e \in E(q) - E(q)^+ \text{ and } \sum_{e' \in E(q)^+} \text{co-occur}(e, e') \leq \delta\}$, where $\text{co-occur}(\cdot)$ is a function that returns the number of common neighbors of the two input entities and δ is a threshold parameter. In this paper, we use the averaged $\text{co-occur}(\cdot)$ between positive pairs in the training set as δ , and choose $E(q)^-$ to be 10 times the size of $E(q)^+$.

We learn the matching function f_{kem} as a logistic regression classifier which takes z_q and z_e as input, and outputs the prediction probability \hat{y} . In this paper, we consider three input feature combinations, namely: (i) *concatenation* $[z_q, z_e]$, (ii) *hadamard product* $z_q \odot z_e$, and (iii) *subtraction* $z_q - z_e$. Note that when $z_q = [ENC(p_1), ENC(p_q)]$, hadamard product and subtraction methods are not applicable due to dimension mismatch between z_q and z_e . f_{kem} is learned with cross entropy loss. Once f_{kem} is learned, given a query q , we select top n^{CXT} entities in $G(q)$ with highest prediction probability as context entities $\hat{E}(q)$. In our experiment, we empirically set $n^{CXT} = 10$.

LTRMHSA: Learning-to-rank with Multi-head Self-attention

While KEMatch is easy to train with low computation overheads, it suffers from a major shortcoming of not considering how an entity determines the relevance of other entities when it co-occurs with different entities. For instance, when the entity *Ben Affleck* (an actor) is mentioned together with *Rosamund Pike* (an actress) in the context, the candidate movie entity *Gone Girl* will become relevant

and thus should be included in $E(q)$, as they both starred in this movie. On the other hand, when the entity *Ben Affleck* appears with *Matt Damon* (another actor) in the context, another movie entity *Good Will Hunting* should be included in $E(q)$ instead. As a result, we propose LTRMHSA, an attention-based learning-to-rank model which considers the interaction among candidate entities when determining their relevance. We adapt the multi-head self-attention model from Tu et. al [106], to model interaction among entities of $E^r(q)$, and propose a Multi-Head Self-Attention (MHSA) layer and Pairwise Bi-Linear layer to select context entities.

As depicted in Figure 6.2, LTRMHSA also uses the query-induced entity selector to obtain $E^r(q)$. It then concatenates all its entities with the context document d and encodes these pairs using k-bert [74]. Each document-candidate entity pair (d, e_i) is first represented as a sequence of word tokens started with a “[cls]” tag token followed by d ’s word tokens, “[sep]” tag token, the word tokens of e_i ’s entity name, and “[sep]” tag token. The output representation of “[cls]” from k-bert for each (d, e_i) pair, denoted by $z_{cls,i}$, then summarizes the semantics of the context document d and the corresponding entity e_i . The MHSA layer takes the sequence of m_q output representations z_1^{cls} to $z_{m_q}^{cls}$ and allows them to interact using multi-head self-attention:

$$\begin{aligned}
 Multihead(Q, K, V) &= Concat(head_1, \dots, head_n)W^0 \\
 head_h &= Attention(QW_h^Q, KW_h^K, VW_h^V), \quad h \in \{1, \dots, n_h\} \\
 Attention(Q_h, K_h, V_h) &= softmax\left(\frac{Q_h K_h^T}{\sqrt{d_k}}\right)V_h
 \end{aligned} \tag{6.1}$$

where Q, K and V are $m_q \times d_z$ matrices where d_z is the dimensionality of $z_{cls,i}$. Q_h, K_h , and V_h are $m_q \times d_k$ matrices where d_k is the inner dimension of self attention. W_h^Q, W_h^K , and W_h^V are learnable $d_z \times d_k$ projection matrices of the sequence of $z_{cls,i}$ ’s for different heads. For n_h heads, d_k is determined by d_z

divided by n_h . In this work, we set $n_h = 8$,

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

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

$$\mathbb{S}(e_i) = \begin{cases} 2 & \text{if } e_i \text{ can be found in } p_q^* \\ 1 & \text{if } e_i \in E(d) \\ 0 & \text{otherwise} \end{cases} \quad (6.2)$$

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

$$y_{i,j} = \begin{cases} 1 & \text{if } \mathbb{S}(e_i) > \mathbb{S}(e_j) \\ 0 & \text{if } \mathbb{S}(e_i) \leq \mathbb{S}(e_j) \end{cases} \quad (6.3)$$

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

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

6.2.3 Contextual Path Generator

Contextual path generator is responsible for generating a path p_q given $E(q)$ and for inferring missing relations when needed. We propose a **Non-Monotonic**

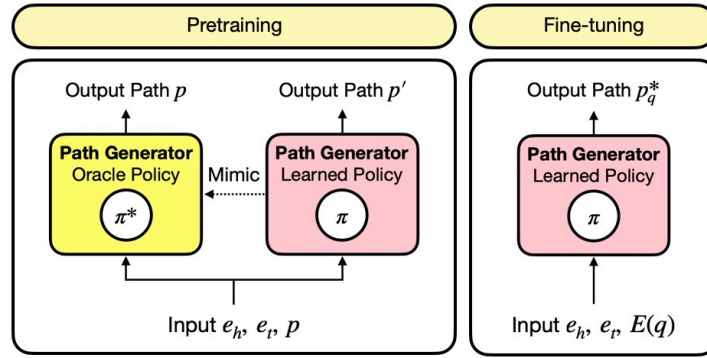


Figure 6.3: Pretraining and Fine-tuning Steps of Non-Monotonic Contextual Path Generation with Pretrained Transformer (NMCPGT)

Contextual Path Generation with Pretrained Transformer (NMCPGT) method

as shown in Figure 6.3. NMCPGT addresses the two limitations of traditional monotonic generation models such as n-gram [7] or neural language models [90] where the order of element generation is always from left to right. The first limitation is the generation of *unfinished paths* when the model generates a path that starts with e_h but could not ends with e_t . The second limitation is the difficulty to leverage on both e_h and e_t to determine the next element to generate in the monotonic manner. Thus, in this paper, we propose the non-monotonic path generation model which substantially increases the odd of finished paths and is trained to generate the more likely path elements using e_h, e_t and the other already generated path elements in each generation step.

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

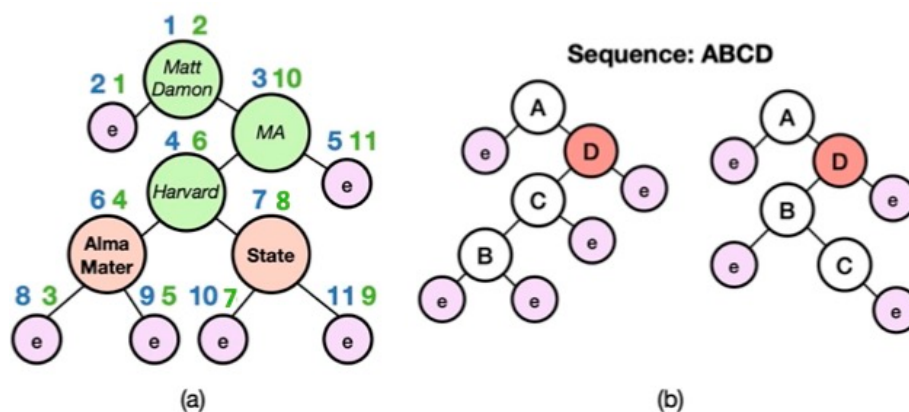


Figure 6.4: Binary Trees of Non-Monotonic Generation: (a) A terminal binary tree that shows the generation steps for a path (with generation order numbers and path order numbers shown in blue and green respectively); (b) Two binary trees with the same generated sequence.

Pretraining of Non-Monotonic Path Generation Model

Our non-monotonic path generation model learns to generate path elements in any order. Borrowing the idea from an earlier work [14], we represent the generation order of all the path elements (i.e., entities and relations of the contextual path) using a binary tree in which each tree node is either an entity/relation or an “end” (or “E”) item. The model generates these elements in a top-down and left-to-right manner, until the binary tree has “E” items generated for all the leaf nodes. Once the generation process is completed, the generated contextual path is the sequence of path elements (excluding the “E” items) determined by a traversal method, called path order, which is a modified Depth First Search (DFS).

Consider the binary tree output example of our non-monotonic path generation module in Figure 6.4(a). The generation order starts with the root node, followed by its left child node, and then right child node. The generation order numbers of the three nodes are thus assigned 1, 2 and 3 as shown in blue in the figure. Following that, the nodes to be generated next are those at the third and fourth layers with generation order numbers assigned from 4 to 11. For each node, we generate a path element or “E” item. As shown in Figure 6.4(a), we finally

obtain a contextual path from the binary tree by constructing a sequence of the path elements (excluding the “E” items) following the path order numbers shown in **green**. The generated path is: $e_{MattDamon} \xrightarrow{\text{almaMater}} e_{HarvardUniversity} \xrightarrow{\text{state}} e_{Massachusetts}$.

Our non-monotonic path generation model is based on reinforcement learning. We let V be the set of all entities and relations, i.e., $V = E \cup R$. Let $Y = (w_1, \dots, w_N)$ denote a sequence of path elements where $w_i \in V$ and \tilde{V} be $\{V \cup \langle e \rangle\}$ where $\langle e \rangle$ denote the terminal or “E” item. Let D denote the collection of Y 's. The generation process is regarded as deterministically navigating a state space \mathcal{S} . A state $s \in \mathcal{S}$ corresponds to a binary tree of nodes from \tilde{V} . For instance, the example we show in Figure 6.4(a) has an initial state $s_1=(Matt\ Damon)$, and a final state $s_{11}=(Matt\ Damon, \langle e \rangle, MA, \dots, \mathbf{State}, \dots, \langle e \rangle)$. The subscript of state s_i is the generation order number. An action a is an element of \tilde{V} that is chosen to be added to the tree for the next available generation order index. As mentioned previously, when every leaf node in the binary tree is the terminal node $\langle e \rangle$, the generation reaches the terminal state s_T . $T = 2N + 1$ denotes the number of nodes (or states) that exist in the terminal binary tree. We use $\tau(i)$ to represent the generation order index of the generated element with path order number = i , e.g., $\tau(10) = 3$ in Figure 6.4(a).

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

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

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

$$\mathbb{E}_{Y \sim D} \mathbb{E}_{t \sim U[T]} \mathbb{E}_{s_t \sim d_{\pi^{\text{in}}}^t} [\mathcal{C}(\pi; \pi^{\text{out}}, s_t)] \quad (6.4)$$

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

The non-monotonic generation model can be implemented with LSTM units model, or with transformer structure [107]. In this paper, we build the model with the latter.

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

$$\pi^*(a|s_t) = \begin{cases} 1 & \text{if } a = \langle e \rangle \text{ and } Y_t = \langle \rangle \\ P(a) & \text{if } a \in Y_t \\ 0 & \text{otherwise} \end{cases} \quad (6.5)$$

where $P(a)$ is defined such that $\sum_{a \in Y_t} P(a) = 1$. For every generation step t , the oracle policy always generates valid action a (i.e., if $a \in Y_t$) with positive probabilities, invalid actions with zero probabilities, and $\langle e \rangle$ when no other elements are required to be generated. We can design different oracle policies

by defining $P(a)$ differently. Let g and h be the nearest left parent and nearest right parent respectively of the tree node corresponding to step t based on the path order. Let the actions generated at steps g and h be $a_g = w_g$ and $a_h = w_h$ respectively. We now derive Y_t as follows:

$$Y_t = \begin{cases} \{w_{g+1}, \dots, w_{h-1}\} \cup \{\langle e \rangle\} & \text{if } a_g \text{ and } a_h \text{ are found} \\ \{w_1, \dots, w_{h-1}\} \cup \{\langle e \rangle\} & \text{if only } a_h \text{ is found} \\ \{w_{g+1}, \dots, w_n\} \cup \{\langle e \rangle\} & \text{if only } a_g \text{ is found} \end{cases} \quad (6.6)$$

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

Cost Functions. Since the generation is non-monotonic, the generation order does not necessarily match the path order, and hence there can be multiple terminal binary trees that share the same generated sequence of path elements. For a desired sequence to be finally generated based on a partially generated binary tree, we need to determine if the element to be generated could violate the desired sequence. Therefore, we consider all entities and relations that can be generated as correct generation. We show two valid binary trees of a sequence $ABCD$ generated by the non-monotonic generation model in Figure 6.4(b). When deciding the left child of D, any token that locates before it in the sequence can lead to

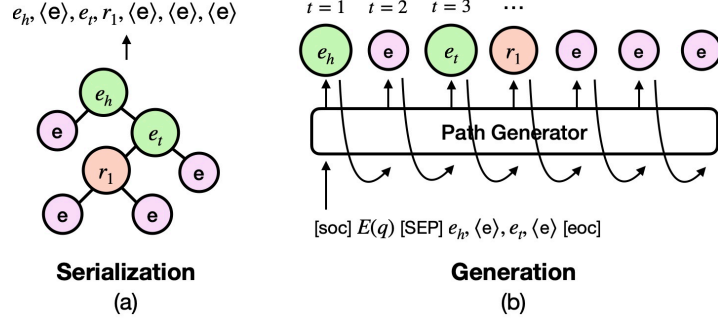


Figure 6.5: Illustration of Non-Monotonic Path Generation: (a) Serialize an input tree with path traversal; (b) Generation of a Contextual Path

generation of a valid binary tree. For instance, C is chosen in the left tree and the right tree chooses B . As a result, we consider B or C as correct generation when computing the cross entropy. Any other tokens are considered incorrect, including the termination node $\langle e \rangle$. We define the **Cross Entropy Loss** as follows,

$$\begin{aligned}
 \mathcal{C}(\pi; \pi^{\text{out}}, s_t) = & \\
 - \left(\sum_{w \in V^+} y_{\pi, s_t} \log(p_{\pi, s_t}) + (1 - y_{\pi, s_t}) \log(1 - p_{\pi, s_t}) \right) & \quad (6.7)
 \end{aligned}$$

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

In our experiment, we generated 100,000 paths from the knowledge graph for the pretraining. The paths are generated by first sampling e_1 from the graph as the head entity of the path, followed by sampling the next entity e_2 connected to e_1 with an edge $e_1 \xrightarrow{r_1} e_2$ and adding that to the path. This process is repeated until we have sampled L edges, or stops with a chance of 20%. We will use the first and last entities of the path as e_h and e_t respectively.

Fine-tuning Non-Monotonic Contextual Path Generation

To direct the NMCPGT model to generate a path according to a context, we fine-tune our pretrained path generation model with additional context entities

$E(q)$.

The context entities are arbitrarily ordered and separated by spaces. They are then concatenated with the query entities before given to the pretrained transformer. We then fine-tune the transformer to return the correct contextual path. The fine-tuning step is very similar to that of pretraining except that the pretraining step involves sampled paths instead of ground truth contextual paths. The fine-tuning step is also different due to additional context entities. Its input sequence is in the format of “[soc] $e_1 e_2 \dots e_{|E(q)|}$ [sep] e_h [e] e_t [e] [eoc]” where [sep] is a special token separating the context entities and the initial tree. We illustrate the contextual path generation process in Figure 6.5(b).

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

6.2.4 Tree Serialization

During both the pretraining and fine-tuning phases, we need to feed the initial generation state represented as a binary tree to the transformer for training. Since the transformer can only generate a sequence of objects, it is trained to decode or generate the sequence of actions that represents the serialized binary tree of the training path. We serialize a path by traversing it in a top-down, left-to-right manner. The serialization process is shown in Figure 6.5(a). For each time step t , the transformer decoder generates (or predicts) the next action of binary tree which is in turn used as input to the transformer decoder for generating the next action. This repetitive process ends when the binary tree is terminal.

6.3 Experiment Results

6.3.1 Dataset

Similar to the experiments discussed in previous chapters, we use the Wiki-film and Wiki-music dataset described in Section A.2. The detailed statistics of the datasets are shown in second and third columns of Table 6.1.

6.3.2 Evaluation Metrics

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

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

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

$$\mathbf{s}(p, p^*) = \frac{\{E(p) \cup L(p)\} \cap \{E(p^*) \cup L(p^*)\}}{\{E(p) \cup L(p)\} \cup \{E(p^*) \cup L(p^*)\}} \quad (6.8)$$

where $E(p)$ and $L(p)$ represent the set of entities and relation labels exists in p respectively. We then compute the average pairwise similarity for all the generated paths. While this metric can effectively determine how close the generated

Table 6.1: Dataset Statistics

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

path is to ground truth, it does not consider the ordering of the entities and relations. Moreover, it does not take into consideration the semantic relatedness between the two paths. The next normalized graph edit distance metric thus addresses these limitations.

Last but not least, we also include **Normalized Graph Edit Distance (NGEO)** which has been covered in Section 5.3.

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

6.3.3 Effectiveness of Context Extractor Methods

Our first set of experiments evaluates the effectiveness of context extraction models returning the ground truth contextual path entities as context entities for a set of queries. Other than our proposed context extractor methods KEMatch

and LTRMHSA, we also include a context window baseline method described below.

- **Context window baseline.** We introduce a context extractor baseline method to compare against our proposed KEMatch and LTRMHSA methods. In this baseline method, we extract the entities mentioned in the text surrounding e_h and that in the text surrounding e_t within a distance of cw word tokens in context document d . In this paper, we empirically use $cw = 20$.
- **KEMatch.** KEMatch prunes the query-induced entities graph with a classifier, and keep only the top n^{CXT} entities with highest prediction probability as context entities. As described in Section 6.2, we experiment three different methods of query representation z_q , namely (1) *average entity representation*, (2) *mention paragraph representation*, (3) *title and mention paragraph representation*, and (4) *contextual query entity representation*. We also propose three feature combination schemes to combine z_q and z_e , the representation of entity e from $G(q)$: (a) *concatenation* $+$, (b) *hadamard product* \odot , and (c) *subtraction* $-$, and an additional (d) *all* which concatenates (a), (b), and (c) together as feature. The parameters $k = 2$, $\delta = 4$, and $n^{CXT} = 10$ are chosen based on grid search.
- **LTRMHSA.** We employ a learning-to-rank with multi-head self-attention model to extract the most context-relevant entities as context entities. We set $n^{CXT} = 10$ to stay consistent with KEMatch.

We use a pretrained DBpedia k-bert model as the context encoder [74]. The model uses the following configuration: $L = 12$, $A = 12$, $H = 768$.

We measure the performance of context entity extraction by precision and recall defined by **Precision** = $\frac{|E(q) \cap E^*(q)|}{|E(q)|}$ and **Recall** = $\frac{|E(q) \cap E^*(q)|}{|E^*(q)|}$ respectively. $E(q)$ denotes the set of context entities extracted by the extractor and $E^*(q)$ is the set of ground truth context entities. We utilize the negative sampling process

described under Section 6.2 and conduct 10-fold cross validation. As we have consistent observations on Wiki-film and Wiki-music and due to limitation in space, we only show the experiment result of Wiki-film dataset in Table 6.2. In this comparison, we additionally include the context window baseline.

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

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

While Table 6.2 shows that LTRMHSA enjoys higher accuracy than all of the KEMatch variants, the former involves a computational complexity of $O(m_q^2)$ compared with KEMatch’s $O(m_q)$. When m_q is large, the computational overhead of LTRMHSA could be significantly higher which adversely affects the time needed for generating a contextual path. As a result, one should take into consideration the trade-off between performance and execution time when de-

Table 6.2: Performance in Context Entity Extraction (Wiki-film)

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

cluding which context extractor to use together with NMCPGT in practice.

6.3.4 Performance in Contextual Path Generation

Next, we conduct experiments to evaluate the models’ performance in generating the contextual paths. We compare our proposed models combining non-monotonic contextual path generation method (NMCPGT) with different context extractor options, namely: KEMatch and its variants, LTRMHSA, context window entities, and random context entities. The *random context entities* method randomly selects n^{CXT} entities from $G(q)$ as the context entities. We set $n^{CXT} = 10$ which is consistent with the n^{CXT} for KEMatch and LTRMHSA. For simplicity, we name our models by “NMCPGT+⟨context extractor method⟩” where the ⟨context extractor method⟩ options are Random Context, Context Window, KEMatch, and LTRMHSA.

To evaluate the importance of the context document, we additionally include a **non-contextual non-monotonic path generation (NCNMPG)** baseline which generates the contextual path from path generator that only takes e_h and

Table 6.3: Path Generation Performance on Wiki-Film and Wiki-Music Datasets

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

e_t as input, without any knowledge of the context document. In other words, this model does not use any context entities as condition, and only relies on the pretrained transformer when generating paths.

For the path generation, we only include the annealed coaching oracle $\pi_{annealed}^*$ as it has been shown to outperform the uniform and coaching oracles in [14]. The model is optimized with cross entropy loss. The non-monotonic path generator uses a 4-layer transformer structure with, 4 attention heads, hidden dimension of 256, and feed-forward dimension of 1024. Adam optimizer is used with the initial learning rate of 10^{-5} . The model is trained with 100 epochs. After the 50 burn-in epochs, β is linearly annealed from 1.0 by 0.01 in each epoch. The implementation of all neural structures is based on Pytorch.

We report the % ground truth recovered(**%Recov**), averaged pairwise similarity(**AVG PW Sim**), **NGEO(E)**, and **NGEO(R)** of the generated contextual paths by different models in Table 6.3. We have similar observations based on the experiments on the two real-world datasets. The contextual path genera-

tion performance is also consistent with the results in Table 6.2. The model NMCPGT+LTRMHSA outperforms other models for all metrics, followed by NMCPGT+KEMatch with context query entity representation and mention paragraph representation. Among the NMCPGT+KEMatch model variants, Hadamard product is the best feature combination method while subtraction being the worst. The simple baselines which do not extract context entities, including Non-Contextualized Generation and Random Context baseline, perform the worst. This indicates the importance of a good context extractor – the better we extract context entities, the more accurate the generated contextual paths will be. In the following, we use two case examples to illustrate the difference between the different models.

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

*“Produced by an Igloolik, Nunavut company, the film is titled *The Journals of Knud Rasmussen*, and co-directed by **Zacharias Kunuk** of Igloolik and Norman Cohn of Montreal. The company received critical acclaim for their first film, *Atanarjuat*, ... The film portrays the pressures on traditional **Inuit** culture....¹”*

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

¹https://en.wikinews.org/wiki/Film_from_Nunavut_in_Canada%27s_north_to_open TIFF

Table 6.4: Non-monotonic(Non-M) and Monotonic(M) Generation

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

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

*Firefighters have confirmed that the large **James Bond** sound stage at **Pinewood Studios** has been destroyed by fire... where filming for *Casino Royale*, the latest Bond movie, has been completed... Pinewood, which was created in 1935, was the filming ground for *Dr No*, the first ever James Bond movie in 1962.²*

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

6.3.5 Comparison between Non-monotonic and Monotonic Path Generation

In this section, we compare our non-monotonic path generator NMCPGT+LTRMHSA with a left-to-right counterpart by evaluating them on the Wiki-film dataset. As discussed in Section 6.2.3, we choose to use a non-monotonic generation model because left-to-right generation models are likely to generate unfinished paths.

To conduct this experiment, we replace $\pi_{annealed}^*$ of NMCPGT by a Left-

²https://en.wikinews.org/wiki/James_Bond_set_at_Pinewood_Studios_destroyed_by_fire

to-Right Oracle π_{L-R}^* which always assign probability of 1 to the leftmost to-be-generated path element of the sequence. As suggested in [14], π_{L-R}^* results in maximum likelihood learning of an autoregressive sequence model, which makes the generation process identical to neural sequence models such as GPT-2 [90]. We use the same input sequence, “[*soc*] $e_1 e_2 \dots e_{|E(q)|}$ [*sep*] e_h [*e*] e_t [*e*] [*eoc*]”, for fine-tuning the non-monotonic and monotonic models. In this experiment, we include a new metric representing the percentage of **%unfinished path** in the result (%Unf) to evaluate the model’s ability of completing the path. We define a unfinished path as one that starts with e_h but could not end with e_t within $L_{MAX} = 6$ relation edges. Note that the generated paths, finished or not, may involve inferred relation edges. We report the experiment result on Wiki-film dataset for NMCPGT+LTRMHSA in Table 6.4. Since NMCPGT+LTRMHSA is required to generate path elements between e_h and e_t , it has %Unf = 0. The monotonic model, however, sees 29.73% of the generated paths unfinished. With fewer finished paths, the %path recovered of the monotonic model is also substantially less than NMCPGT+LTRMHSA.

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

*...Ledger starred in the 2005 movie **Brokeback Mountain** where he was nominated for the Academy Award and the Golden Globe Award for Best Actor. He also starred in the 2000 movie *The Patriot* with **Mel Gibson**...*³

The ground truth contextual path of this query should be $e_{\text{BrokebackMountain}} \xrightarrow{\text{starring}}$ $e_{\text{Ledger}} \xrightarrow{\text{starring}}$ $e_{\text{ThePatriot}} \xrightarrow{\text{starring}}$ e_{Gibson} . The non-monotonic model using $\pi_{annealed}^*$ generate the ground truth path perfectly. On the other hand, the monotonic

³https://en.wikinews.org/wiki/Australian_actor_Heath_Ledger_found_dead_in_New_York_City

model using π_{L-R}^* generates a path that is unfinished: $e_{\text{BrokebackMountain}} \xrightarrow{\text{subject}}$
 $e_{\text{AcademyAward}} \xrightarrow{\text{subject}} e_{\text{AcademyAwardForBestActor}} \xrightarrow{\text{subject}} e_{\text{Ledger}} \xrightarrow{\text{starring}} e_{\text{ThePatriot}} \xrightarrow{\text{pageLink}}$

e_{Ledger} . Nevertheless, the generated path is still very relevant to the context. In fact, if we do not force the model to stop at the length of 6, it will eventually reach $e_{\text{Mel Gibson}}$ at the 8th hop. This example explains why we obtain relatively high pairwise similarity and NGeo for the monotonic model.

6.4 Discussion

In this paper, we formally define the CPG task and propose a two-stage CPG framework consisting of a context extractor and a non-monotonic path generation with pretrained transformer (NMCPGT). Through experiments, we show that our NMCPGT combined with various context extractor models generate knowledge graph paths that are more similar to the ground truths than the baseline models.

In the research aspect, this paper addresses the inherent challenges of knowledge graph reasoning induced by external textual content. Knowledge graph reasoning refers to inferring new facts in the knowledge graph from textual data. CPG represents a new approach to generate new relation instances as facts from input context documents. It can be combined with information retrieval research to offer explanations to search or prediction results using both existing and inferred relations of knowledge graphs. This will pave the foundational work for future research in information retrieval with explainable AI.

NMCPGT also brings several technical advancements to the field of information retrieval. One major distinction between this work and previous KGIR works is the use of a generative pretrained model. Previous works that use the knowledge graph to reason, such as MHQA-GRN [98] and LEGO [92], mostly generate knowledge paths by traversing the knowledge graph. Such methods suffer from limitations, including (1) only being able to generate ob-

served knowledge graph relations, (2) having long inference times, and (3) not guaranteeing finished and loopless paths. In contrast, NMCPGT uses a pre-trained knowledge graph path generation model that is trained to generate high-quality knowledge paths in a very short inference time. Furthermore, after being fine-tuned with the context entities as a part of the prompt, the generation model gains the ability to infer unseen relations from the context. To the best of our knowledge, we are the first to approach the knowledge graph path generation task with a pretrained generation model.

NMCPGT has introduced another advancement, which is the ability to generate a knowledge path in a non-monotonic manner. As explained in Section 6.3.5, this non-monotonic generation order greatly enhances the path quality and accuracy by reducing the generation of incomplete paths. Conversely, traversal-based knowledge path generation methods are limited to left-to-right or right-to-left generation, where the next relation and entity to be generated depend on the currently visited entity in the knowledge graph. The knowledge path generation model in NMCPGT is optimized with a policy learning-based objective, which enables it to generate entities without depending on the knowledge graph structure. This approach ensures that the model always generates a complete path that begins and ends with the query entities, providing better flexibility to the generation process as it is less restricted by the existing knowledge graph.

In the practical aspect, CPG overlays a context document with contextual paths. This simplifies content understanding as both the known and inferred relations of contextual paths between entities in the context can be highlighted to the readers. For instance, a content analysis system could process documents with CPG and obtain contextual paths of entity pairs mentioned in the documents. The users can therefore access background information of the entity pairs from the contextual paths, regardless it being explicit or implicit to them in the documents. Given an entity pair, one can refer to other entities exist in the contextual path to gather more understanding of the relation between them. Moreover, the

user can filter documents that cover a specific event between this entity pair (i.e., with contextual path consisting specific knowledge graph relation).

In addition, it is possible to automatically enrich a knowledge graph with CPG. Knowledge graphs can be incomplete and outdated, which is a common limitation faced by many knowledge graph-based applications. CPG may help to address such issue by inferring new relations during generation of contextual paths. With a large enough set of documents, one could augment the knowledge graph with newly inferred relation so the knowledge within it is more up-to-date. This can be especially helpful for ISs that require latest information to reach the best solution, such as decision-making systems or competitor analysis systems.

To summarize, the method proposed in this work showcases a new way to infer and represent contextual relations between two entity mentions in a text document. We presented such contextual relations as knowledge graph paths (i.e., contextual paths), which can be easily utilized by any information system or other downstream tasks. To the best of our knowledge, we are the first to define contextual path and the problem of CPG. CPG contributes to knowledge in IR from both research and practical aspects. It also opens various future research directions, which we discuss in the next section.

6.5 Summary

In this chapter, we aim to enhance the understanding of textual documents by deriving knowledge paths among the mentions of entities. To address the challenges in path generation, we proposed a two-stage contextual path generation framework that can handle (i) noisy context information, (ii) missing relation edges in knowledge graphs, and (iii) generate well-formed paths. We improve context entity extraction and develop non-monotonic generation with pretraining to overcome these challenges.

In the following, we describe our contribution in this chapter:

- We design a two-stage framework consisting of a context extractor and path generator to develop CPG solution models. Based on this framework, we propose two context extraction methods, *knowledge-enabled embedding matching* and *learning-to-rank with multi-head self attention*, to determine a set of context entities relevant to the query entity pair and context document. For contextual path generation, we propose a *non-monotonic path generation method with pretrained transformer* to generate high quality paths.
- We show in the experiments on two real-world datasets that our best performing CPG model recovers nearby 85% of ground truth paths, which is 14.8% higher than a context window baseline.
- We show in the experiments on a synthetic dataset that our models can cope with large datasets. We also show that our model using non-monotonic path generation returns 42% more ground truth contextual paths and 29.7% less incomplete paths than the model using monotonic generation approach.
- We evaluate and demonstrate our proposed model’s ability to recover contextual paths with different proportions of missing relation edges in the knowledge graph.

Due to the high cost of annotation efforts, we only conducted experiments on a dataset built with Wikinews articles. However, with more datasets annotated in the future, we will be able to conduct experiments on larger datasets to further validate the current findings. Moreover, we assume that only one ground truth contextual path exists between a query entity pair in this dataset, which can be vastly different from a real-world scenario. Therefore, we shall further refine both the dataset and experiment design to address this limitation.

There are several interesting future directions for CPG research. First, it offers a general framework that can be adapted to different knowledge graph-enhanced information systems. For example, in a recommendation system, we

can recommend items by constructing a path out of the user’s product purchase and browsing history. A search engine can use contextual paths in a set of text documents to summarize how they are interconnected with entities and relations. Second, CPG can be viewed as an interesting means to derive new relations not found in the existing knowledge graphs. For this to work at scale, we require many context documents of the same knowledge domain to be provided. It should also be compared and evaluated against other knowledge completion methods. Third, an extension of CPG to find contextual subgraphs (instead of contextual paths) connecting a set of input entities in a context document is also another interesting research direction. Finally, it is worthwhile to explore how to further expand the scale of the pretrained knowledge path model used in NM-CPGT. With the recent advancements in large pretrained language models, they can generate natural language with high quality and accuracy. Researchers have even identified new abilities that emerged during large-scale pretraining, such as in-context learning and reasoning with chain-of-thought. We hypothesize that a pretrained large knowledge path model of larger scale will have better generation and reasoning ability, which can further improve the accuracy of downstream KG-based IR tasks like CPG.

Chapter 7

Conclusion

This dissertation aims to address the common challenge underlying contextual information retrieval research, that is finding the semantic connections among entities embedded in context of different forms (e.g., text, image, etc.) This challenge is akin to connecting dots in a puzzle with the help of background knowledge. We first confine the context to be textual and the background knowledge to be some knowledge graph or association graph that captures relationships between knowledge entities which may be found in the given context. For example, the context could be a piece of news mentioning some named entities, or a classroom question about biological concept entities,

With the above-mentioned context definition, one can define different types of contextual information retrieval tasks. In this dissertation, Contextual Path Retrieval (CPR), Contextual Path Generation (CPG), and Non-Monotonic Contextual Path Generation (NMCPG) are example tasks which return some path(s) as results. One can define other contextual information retrieval tasks that return text or graphs instead of paths. To address these contextual information retrieval tasks, we first present a meta-framework that considers the interaction between knowledge and textual content of context when learning the context representation. This learned contextual representation will be subsequently used in the downstream contextual information retrieval task(s) which is usually formulated

as a prediction task (e.g., prediction of path's relevance in the case of CPR, or prediction of path's elements).

Before we embark on the research CPR, CPG and NMCPG, we first study the contextual representation learning for textual documents as contexts, and these contexts are connected with one another in a large document network. By treating entities and words mentioned in these textual documents as attribute values, we seek to learn the contextual representations of these textual documents using our proposed multi-VAE approach called ECAN. ECAN combines network embeddings with attribute embeddings such that contextual representations, entity and attribute representations share the same space.

From the technical standpoint, this dissertation has contributed the following:

- It introduces different interesting forms of context representation learning, including contexts in network and contexts covering entities and words where the context entities can be found in some knowledge graph. It further demonstrates how contextual representations can be effectively used for contextual path retrieval and generation tasks.
- In contextual path retrieval, this dissertation proposes a solution framework in which new contextual representation methods and path embedding methods are developed to support effective retrieval of contextual paths. It is also in this work we create the first few datasets constructed from real world data for CPR and CPG research, as well as datasets using synthetic algorithms.
- In contextual path generation, the dissertation focuses on utilizing the knowledge and text representations to infer the relation edges required to generate the path connecting the two query entities guided by the contextual representation of context document. Our proposed monotonic CPG model employs an encoder-decoder transformer. The encoder learns to

combine knowledge graph and text representations into one single contextual representation vector. The decoder generates contextual paths given the contextual representation from the encoder. In non-monotonic CPG, the dissertation proposes a generation model consisting of a path decoder. We first pre-train path decoder to generate knowledge graph paths from initial query entities' representations. This model is then fine-tuned to generate the contextual path conditioned on the query entities and the contextual representation.

While this dissertation provides both datasets and models for conducting research in contextual information retrieval, it is subject to several limitations:

- The datasets utilized in this study are relatively small in scale. We utilized the Wiki-film and Wiki-music datasets in our experiments conducted in Chapters 4 to 6. Compared to existing knowledge graphs such as the comprehensive DBPedia knowledge graph¹ with over 3.5 million entities, Scopus² with more than 90 million entities, and WordNet³ covering more than 110,000 entities, the scale of the knowledge graphs employed by these two datasets are relatively small. While we have created synthetic datasets that are much larger, we are not able to experiment on large real world datasets to demonstrate the scale generalizability of our model performance.
- For CPR and CPG, we assume that the input query has undergone appropriate entity linking. During this process, each entity has been linked to its respective entity ID in the knowledge graph. However, the accuracy of CPR and CPG heavily depends on the accuracy of entity linking. In other words, it is crucial to ensure that the entity mentions are correctly linked to the corresponding knowledge graph entities.

¹<https://yago-knowledge.org/>

²<https://www.elsevier.com/solutions/scopus>

³<https://wordnet.princeton.edu/>

- In this dissertation, our emphasis lies in the retrieval and generation of contextual information through multi-hop knowledge graph paths. Although this representation of contextual information effectively captures the semantic connection between two entities, it may overlook the valuable interactions among other entities mentioned within the given context.
- The methods presented in this dissertation are specifically designed to process textual input. However, it is important to note that other forms of data, including images, videos, and sensor data, have not been considered within the scope of this research. Embracing multimodal data has the potential to offer intriguing and distinctive perspectives, enabling a more profound comprehension of the context.

With these limitations in mind, this dissertation opens up several exciting research directions that can be studied in the future:

- Firstly, our study can be extended to improve the generalizability of our proposed frameworks. There are two aspects of generalizability we can improve upon, namely: (a) *domain generalizability*, and (b) *scale generalizability*. In this dissertation, we have conducted our research on film and music news domains only. To generalize the results to other domains, we need to construct more datasets covering other domains. An important future direction is to develop new ways to collect much larger datasets for contextual information retrieval research.
- Secondly, our proposed methods can be extended to handle inputs without entity linkage. Given the existing research suggesting that the inclusion of contextual information enhances entity linking accuracy [40], it is worthwhile to investigate the potential of jointly linking entities and retrieving/generating contextual paths. Such an exploration could potentially lead to improvements in CPR/CPG accuracy as well.

- Thirdly, there is potential to explore contextual information retrieval tasks beyond CPR and CPG. For instance, instead of returning contextual paths, researchers can investigate tasks that involve retrieving graph structures or text results as part of the retrieval process. Utilizing more complex structure to represent contextual information can offer a more holistic view of the interaction among entities mentioned in the contextual document.
- Fourthly, our proposed meta-framework can be adapted to contexts beyond text. For example, the meta-framework can be extended to consider multimedia or more complex context. Considering the advancements in multimedia and NLP, we envisage that new context definitions and contextual information retrieval tasks will pave the way to many novel and interesting future works.

Finally, to conclude, this dissertation is clearly one early attempt to connect one entity to another entity with information embedded in context. In the process of establishing connections, we may retrieve existing connections from knowledge graphs or generate new connections to enrich the existing knowledge graphs. This demonstrates the symbiotic relationship between contextual information retrieval and knowledge graph. At the higher level, entities are not the only dots to be connected. With further research, we hopefully will be able to connect more complex information structures such as relation edges, subgraphs, and beyond.

Appendices

Appendix A

Construction of Real and Synthetic Data Collections

There were no publicly available datasets for contextual path retrieval and contextual path generation experiments and evaluation. Hence, we construct three datasets, two real datasets and one synthetic dataset from WikiNews articles, and recruit Amazon Mechanical Turk workers to annotate WikiNews¹ news articles. We extract a subset of DBpedia data as the input knowledge graph. DBpedia² is a knowledge base created for Wikipedia articles. Each article entry in Wikipedia corresponds to an entity in DBpedia. The entity's attributes are usually found within the infobox of the corresponding article entry. The attribute value is another DBpedia entity corresponding to another Wikipedia entry. The attribute label corresponds to the relation label. In the following, we elaborate on how our knowledge graph is extracted from DBpedia, how the input entity pairs and context documents are obtained, and how the contextual paths for entity pairs are determined.

¹<https://en.wikinews.org>

²<https://wiki.dbpedia.org>

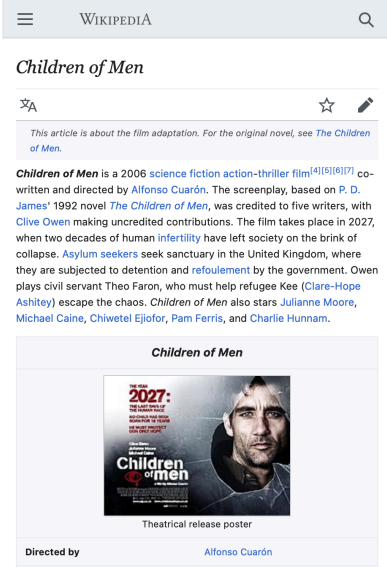
"Children of Men" wins Scriptor Award for writing

- '**Children of Men**', a movie based on a P.D. James book, has won the USC Scriptor Award 2006 for its writing.
- Both the original author, James, and the screenwriting team will be honored by the University of South California for their work.
- The winning screenwriters are **Alfonso Cuarón**, Timothy J. Sexton, David Arata, Mark Fergus, and Hawk Ostby. "
- The Children of Men" was James' 12th book, written in 1992.
- 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."
- 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 *director* → Children of Men

Yes
 No



The screenshot shows the Wikipedia page for "Children of Men". It includes the title, a search bar, and a summary paragraph: "Children of Men is a 2006 science fiction action-thriller film^{[4][5][6][7]} 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." Below the text is a theatrical release poster for the film and the text "Directed by Alfonso Cuarón".

Figure A.1: Annotation Interface

A.1 Wikinews Datasets

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 available³. 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.

³<https://dumps.wikimedia.org/>

A.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 one-hop relation could explain the co-occurrence of the two entities. Figure A.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.

A.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-occurring 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

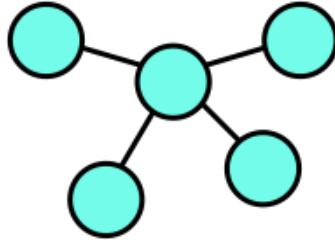


Figure A.2: Star Network

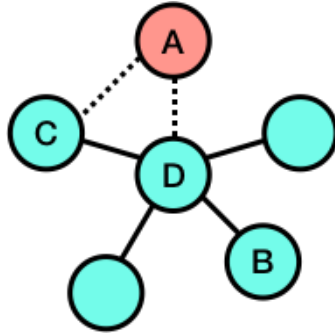


Figure A.3: Network with an Inserted Entity

as shown in Figure A.2. 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 A.3 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 3.

Algorithm 3: 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

```

Consider the example Wikinews article in Figure A.1⁴: ‘**Children of Men**, a movie based on a P.D. James book, has won the 2006 USC Scriptor Award for its writing.... The winning screenwriters are **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 which see the appearance of **Alfonso Cuarón** so as to have s works 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.

A.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 A.3, 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

⁴https://en.wikinews.org/wiki/%22Children_of_Men%22_wins_Scriptor_Award_for_writing

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 1396 and 1237 entity pairs for Wiki-film and Wiki-music dataset, respectively. The statistics of the two datasets are shown in Table A.1.

A.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 for 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.

A.2.1 Knowledge Graph Construction

The first step to generating our synthetic dataset is to sample a subset of the knowledge graph that are dedicated to a certain domain. As one of our datasets are 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 $G^{Film} = (E^{Film}, L^{Film}, R^{Film}, F^{Film})$.

A.2.2 Generation of Paths

We next generate a set of distinctive contextual paths which 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

Algorithm 4: 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

sampling repeats for the neighbors of e_1 until $|p|$ reaches 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 4. In this work, we empirically set t to be 6. We exclude duplicate paths in the generation process.

A.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 5. 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_2) to mimic the natural writing style, that provide some background knowledge of e_i (or e_j) as part of the context. Finally, we sample with very small probability

Algorithm 5: Synthetic Document Generation

input : Path p generated using Alg 4, 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

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 5. 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 A.1.

Table A.1: 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
# Entities in Knowledge Graph	59,173	91,364	59,173	44,886
# Relations in Knowledge Graph	651	651	651	513
AVG Ground Truth (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

Bibliography

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.
- [2] Nitish Aggarwal, Sumit Bhatia, and Vinith Misra. Connecting the dots: Explaining relationships between unconnected entities in a knowledge graph. In *ESWC*, 2016.
- [3] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9), 2018.
- [4] KM Annervaz, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing. In *NAACL*, 2018.
- [5] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*, 2020.
- [6] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *The Journal of Machine Learning Research*, 18(1):3846–3912, Jan 2017.

- [7] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, 1983.
- [8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003.
- [9] Sumit Bhatia, Purusharth Dwivedi, and Avneet Kaur. That’s interesting, tell me more! finding descriptive support passages for knowledge graph relationships. In *ISWC*, 2018.
- [10] Roi Blanco and Hugo Zaragoza. Finding support sentences for entities. In *SIGIR*, 2010.
- [11] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.
- [12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*. 2013.
- [13] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI*, 2018.
- [14] Kianté Brantley, Kyunghyun Cho, Hal Daumé, and Sean Welleck. Non-monotonic sequential text generation. In *Workshop on Widening NLP*, 2019.
- [15] Sebastian Bruch. An alternative cross entropy loss for Learning-to-Rank, November 2019.
- [16] C J Burges. From ranknet to lambdarank to lambdamart. 2010.

- [17] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*, 2019.
- [18] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, pages 129–136, June 2007.
- [19] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *KDD*, 2015.
- [20] Shubham Chatterjee and Laura Dietz. Why does this entity matter? support passage retrieval for entity retrieval. In *SIGIR*, 2019.
- [21] Shubham Chatterjee and Laura Dietz. Entity retrieval using fine-grained entity aspects. In *SIGIR*, 2021.
- [22] Hongxu Chen, Yicong Li, Xiangguo Sun, Guandong Xu, and Hongzhi Yin. Temporal meta-path guided explainable recommendation. In *WSDM*, 2021.
- [23] Ryan Clancy, Ihab F Ilyas, and Jimmy Lin. Scalable knowledge graph construction from text collections. In *Workshop on Fact Extraction and VERification*, 2019.
- [24] Zhihong Cui, Hongxu Chen, Lizhen Cui, Shijun Liu, Xueyan Liu, Guandong Xu, and Hongzhi Yin. Reinforced kgs reasoning for explainable sequential recommendation. *World Wide Web*, 25:631–654, 2022.
- [25] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*, 2018.

- [26] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2017.
- [27] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *ICLR*, 2020.
- [28] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.
- [29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.
- [30] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. Metapath2Vec: Scalable representation learning for heterogeneous networks. In *KDD*, 2017.
- [31] Zhengxiao Du, Chang Zhou, Jiangchao Yao, Teng Tu, Letian Cheng, Hongxia Yang, Jingren Zhou, and Jie Tang. CogKR: Cognitive graph for multi-hop knowledge reasoning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, page 1–1, 2021.
- [32] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem van de Meent. Structured disentangled representations. In *PMLR*, volume 89, pages 2525–2534, 2019.
- [33] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *NAACL*, 2015.

- [34] Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. Collaborative policy learning for open knowledge graph reasoning. In *EMNLP*, 2019.
- [35] Hongchang Gao and Heng Huang. Deep Attributed Network Embedding. In *IJCAI*, pages 3364–3370, 2018.
- [36] Zhiqiang Geng, Yanhui Zhang, and Yongming Han. Joint entity and relation extraction model based on rich semantics. *Neurocomputing*, 429:132–140, 2021.
- [37] Frédéric Godin, Anjishnu Kumar, and Arpit Mittal. Learning when not to answer: a ternary reward structure for reinforcement learning based question answering. In *NeurIPs Workshop in Relational Representation Learning*, 2019.
- [38] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.
- [39] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *KDD*, 2016.
- [40] Nitish Gupta, Sameer Singh, and Dan Roth. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*, 2017.
- [41] Benjamin Heitmann and Conor Hayes. Using linked data to build open, collaborative recommender systems. In *2010 AAAI Spring Symposium Series*, 2010.
- [42] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [43] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenaу, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and

- Gerhard Weikum. Robust disambiguation of named entities in text. In *EMNLP*, 2011.
- [44] John L Holland. A theory of vocational choice. *Journal of counseling psychology*, 6(1):35, 1959.
- [45] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*, 2018.
- [46] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *ICML*, 2017.
- [47] Feiran Huang, Xiaoming Zhang, Chaozhuo Li, Zhoujun Li, Yueying He, and Zhonghua Zhao. Multimodal network embedding via attention based multi-view variational autoencoder. In *ICMR*, 2018.
- [48] Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *SDM*, 2017.
- [49] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *WSDM*, 2017.
- [50] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *WSDM*, 2019.
- [51] Wiradee Imrattana-trai, Makoto P. Kato, and Masatoshi Yoshikawa. Identifying entity properties from text with zero-shot learning. In *SIGIR*, 2019.
- [52] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Knowledge graph completion with adaptive sparse transfer matrix. Feb. 2016.
- [53] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

- [54] Di Jin, Bingyi Li, Pengfei Jiao, Dongxiao He, and Weixiong Zhang. Network-specific variational auto-encoder for embedding in attribute networks. In *IJCAI*, 2019.
- [55] Amina Kadry and Laura Dietz. Open relation extraction for support passage retrieval: Merit and open issues. In *SIGIR*, 2017.
- [56] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.*, 30:3146–3154, 2017.
- [57] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.
- [58] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2013.
- [59] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NeurIPS*, 2014.
- [60] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [61] Thomas N Kipf and Max Welling. Variational graph auto-encoders. 2016.
- [62] Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Veronika Thost. Attributed description logics: Reasoning on knowledge graphs. In *IJCAI*, 2018.
- [63] Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. Syntax-Guided Controlled Generation of Paraphrases. *TACL*, 8:330–345, 06 2020.

- [64] Souvik Kundu, Tushar Khot, Ashish Sabharwal, and Peter Clark. Exploiting explicit paths for multi-hop reading comprehension. In *ACL*, 2019.
- [65] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, 2014.
- [66] Hang Li, Haozheng Wang, Zhenglu Yang, and Haochen Liu. Effective representing of information network by variational autoencoder. In *IJCAI*, 2017.
- [67] Zixuan Li, Xiaolong Jin, Saiping Guan, Yuanzhuo Wang, and Xueqi Cheng. Path reasoning over knowledge graph: A multi-agent and reinforcement learning based method. In *ICDMW*, 2018.
- [68] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutionary representation learning. In *SIGIR*, 2021.
- [69] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. Kagnet: Knowledge-aware graph networks for commonsense reasoning. In *EMNLP-IJCNLP*, 2019.
- [70] Qika Lin, Jun Liu, Yudai Pan, Lingling Zhang, Xin Hu, and Jie Ma. Rule-enhanced iterative complementation for knowledge graph reasoning. *Inf. Sci.*, 575:66–79, 2021.
- [71] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, 2018.
- [72] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. 2015.

- [73] Hao Liu, Shuwang Zhou, Changfang Chen, Tianlei Gao, Jiyong Xu, and Minglei Shu. Dynamic knowledge graph reasoning based on deep reinforcement learning. *Knowledge-Based Systems*, 241:108235, Apr 2022.
- [74] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: Enabling language representation with knowledge graph. In *AAAI*, 2020.
- [75] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S Yu. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *AAAI*, 2021.
- [76] Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. *AAAI*, 34(03):2925–2933, Apr 2020.
- [77] Aman Mehta, Aashay Singhal, and Kamalakar Karlapalem. Scalable knowledge graph construction over text using deep learning based predicate mapping. In *WWW*, 2019.
- [78] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. Co-embedding attributed networks. In *WSDM*, 2019.
- [79] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. Semi-supervisedly co-embedding attributed networks. In *NeurIPS*, 2019.
- [80] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, January 2013.
- [81] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*, 2016.

- [82] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL*, 2018.
- [83] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *IJCAI*, 2016.
- [84] Hao Peng, Ankur P. Parikh, Manaal Faruqui, Bhuwan Dhingra, and Das Dipanjan. Text generation with exemplar-based adaptive decoding. In *NAACL*, 2019.
- [85] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.
- [86] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.
- [87] Giuseppe Pirrò. Explaining and suggesting relatedness in knowledge graphs. In *ISWC*, 2015.
- [88] Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. In *ACL*, 2019.
- [89] Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *NeurIPS*, 2019.
- [90] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [91] Raymond Reiter. On closed world data bases. In *Readings in artificial intelligence*, pages 119–140. Elsevier, 1981.
- [92] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. Lego:

- Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *ICML*, volume 139 of *PLMR*, pages 8959–8970, 2021.
- [93] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL*, 2020.
- [94] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from Non-Parallel text by Cross-Alignment. In *NeurIPS*, 2017.
- [95] Baoxu Shi and Tim Wenginger. Proje: Embedding projection for knowledge graph completion. Feb. 2017.
- [96] Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han. Aspem: Embedding learning by aspects in heterogeneous information networks. In *SDM*, 2018.
- [97] Kihyuk Sohn, Honglak Lee, and Xinchun Yan. Learning structured output representation using deep conditional generative models. In *NeurIPS*, 2015.
- [98] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. Evidence integration for multi-hop reading comprehension with graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(2):631–639, Feb 2022.
- [99] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.
- [100] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *NAACL*, June 2018.

- [101] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *TheWebConf*, 2015.
- [102] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *KDD*, 2008.
- [103] Xing Tang, Ling Chen, Jun Cui, and Baogang Wei. Knowledge representation learning with entity descriptions, hierarchical types, and textual relations. *IP&M*, 56:809–822, 2019.
- [104] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research*, 18(130):1–38, 2017.
- [105] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, 2016.
- [106] Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. Select, answer and explain: Interpretable Multi-Hop reading comprehension over multiple documents. In *AAAI*, 2020.
- [107] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [108] Severine Verlinden, Klim Zaporozhets, Johannes Deleu, Thomas Demeester, and Chris Develder. Injecting knowledge base information into end-to-end joint entity and relation extraction and coreference resolution. In *ACL-IJCNLP*, 2021.

- [109] Nikos Voskarides, Edgar Meij, and Maarten de Rijke. Generating descriptions of entity relationships. In *ECIR*, 2017.
- [110] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. Learning to explain entity relationships in knowledge graphs. In *ACL*, 2015.
- [111] Guojia Wan and Bo Du. Gaussianpath:a bayesian multi-hop reasoning framework for knowledge graph reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4393–4401, May 2021.
- [112] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, 2016.
- [113] Peifeng Wang, Nanyun Peng, Pedro Szekely, and Xiang Ren. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *EMNLP*, 2020.
- [114] Suhang Wang, Charu Aggarwal, Jiliang Tang, and Huan Liu. Attributed signed network embedding. In *CIKM*, 2017.
- [115] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *KDD*, 2019.
- [116] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*, 2019.
- [117] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

- [118] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. 2014.
- [119] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.
- [120] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based QA. In *WWW*, 2014.
- [121] Ian H Witten and David N Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, 2008.
- [122] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L. Hamilton. Temp: Temporal message passing for temporal knowledge graph completion. In *EMNLP*, 2020.
- [123] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*, 2019.
- [124] Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*, pages 1271–1279, 2017.
- [125] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, 2017.
- [126] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question answering on Freebase via relation extraction and textual evidence. In *ACL*, 2016.

- [127] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *AAAI*, 2017.
- [128] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*, 2016.
- [129] Xinchun Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016.
- [130] Bishan Yang, Wen-Tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.
- [131] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *IJCAI*, 2015.
- [132] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.
- [133] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. In *NeurIPS*. 2018.
- [134] Liang Yao, Chengsheng Mao, and Yuan Luo. Kg-bert: Bert for knowledge graph completion. In *AAAI*, 2020.
- [135] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. Simple QA by Attentive Convolutional Neural Network. In *COLING*, 2016.

- [136] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. Collaborative filtering with entity similarity regularization in heterogeneous information networks. In *IJCAI HINA*, 2013.
- [137] Ningyu Zhang, Xiang Chen, Xin Xie, Shumin Deng, Chuanqi Tan, Mosha Chen, Fei Huang, Luo Si, and Huajun Chen. Document-level relation extraction as semantic segmentation. In *IJCAI*, 2021.
- [138] Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *NAACL*, 2019.
- [139] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *AAAI*, 2018.
- [140] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. Anrl: Attributed network representation learning via deep neural networks. In *IJCAI*, 2018.
- [141] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. Meta-Graph based recommendation fusion over heterogeneous information networks. In *KDD*, 2017.
- [142] Weiguo Zheng, Lei Zou, Wei Peng, Xifeng Yan, Shaoxu Song, and Dongyan Zhao. Semantic sparql similarity search over rdf knowledge graphs. *PVLDB*, 9(11):840–851, 2016.
- [143] Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. An interpretable reasoning network for Multi-Relation question answering. In *COLING*, 2018.
- [144] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. Incorporating bert into neural machine translation. *ICLR*, 2020.

- [145] Qiannan Zhu, Xiaofei Zhou, Jianlong Tan, and Li Guo. Knowledge base reasoning with convolutional-based recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):2015–2028, May 2021.
- [146] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized bert pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, 2021.