University of California
Santa Barbara

Towards Effort-Saving Knowledge Mining and Reasoning over the Web

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Hanwen Zha

September 2021

The Dissertation of Hanwen Zha is approved.

Professor William Wang

Professor Tao Yang

Professor Xifeng Yan, Committee Chair

September 2021

Towards Effort-Saving Knowledge Mining and Reasoning over the Web

To my father, who provided unconditional love and endless support.

Acknowledgements

My deepest gratitude goes to my advisor, Prof. Xifeng Yan, without whom I would not start and finish my PhD. I am grateful he introduced me to the fantastic field of natural language processing. He is an insightful man but can also pay attention to details. During these years, he has deeply influenced my way of thinking not only about research but also about life. Until now, I can still remember his advice, encouragement, and criticism throughout my whole PhD life.

I would like to thank Prof. William Wang and Prof. Tao Yang for serving on my committee. They provide invaluable comments and feedback on every stage of PhD career over the years.

I'm also very lucky to work with many brilliant people. I owe many thanks to my collaborators: Keqian Li, Zhiyu Chen, Wenhu Chen, Yu Su, Honglei Liu, Jiaming Shen, Jiawei Han, Warren Greiff, and Michelle Vanni. They provides significant help in my work and I learned a lot from them.

During my internship, it was a great honor to work with my mentors and peers Shengqi Yang, Xi Chen and Kushal Lakhotia, and Scott Yih. They were patient and supportive to me and I learn a lot from these wonderful experiences.

I would like to thank all my friends. It is always fun to chat, eat and play with you, so I can relieve stress these years. I want to thank my girlfriend for her company and support through my PhD journey. Finally, I want to thank my parents and grandparents for supporting my PhD as always.

Curriculum Vitæ
Hanwen Zha

Education

2016 - 2021	Ph.D. in Computer Science, University of California, Santa Barbara.
2012 - 2016	B.S. in Computer Science, Nanjing University.

Publications

Hanwen Zha, Zhiyu Chen, Xifeng Yan, "Inductive Relation Prediction by BERT" (preprint arXiv' 21)

Wenhu Chen, **Hanwen Zha**, Zhiyu Chen, Wenhan Xiong, Hong Wang and William Wang, "HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data" (Findings of EMNLP' 20)

Zhiyu Chen, Wenhu Chen, **Hanwen Zha**, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, William Yang Wang, "Logic2Text: High-Fidelity Natural Language Generation from Logical Forms" (Findings of EMNLP' 20)

Hanwen Zha, Wenhu Chen, Keqian Li and Xifeng Yan, "Mining Algorithm Roadmap in Scientific Publications" (KDD'19, Research Track, Long Oral)

Keqian Li, Shiyang Li, Semih Yavuz, **Hanwen Zha**, Yu Su and Xifeng Yan, "HierCon: Hierarchical Organization of Technical Documents Based on Concepts" (ICDM'19 Long Paper, Best of ICDM 2019 Selection)

Zhiyu Chen, **Hanwen Zha**, Honglei Liu, Wenhu Chen, Xifeng Yan and Yu Su, "Global Textual Relation Embedding for Relational Understanding" (ACL'19 Short Paper)

Hanwen Zha, Jiaming Shen, Keqian Li, Warren Greiff, Michelle T. Vanni, Jiawei Han, Xifeng Yan, "FTS : Faceted Taxonomy Construction and Search for Scientific Publications." (KDD'18 Demo)

Keqian Li, Ping Zhang, Honglei Liu, **Hanwen Zha**, Xifeng Yan, "PoQaa: Text Mining and Knowledge Sharing for Scientific Publications." (KDD'18 Demo)

Keqian Li, **Hanwen Zha**, Yu Su, Xifeng Yan, "Concept Mining via Embedding." (ICDM'18 Long Paper)

Keqian Li, **Hanwen Zha**, Yu Su, Xifeng Yan, "Unsupervised Neural Categorization for Scientific Publications." (SDM'18 Long Paper)

Yufeng Li, **Hanwen Zha**, Zhihua Zhou, "Learning Safe Prediction for Semi-Supervised Regression." (AAAI'17 Long Oral)

Experience

Summer 2020	Research Intern, Facebook Assistant Topics: Question Answering over Dialogue
Summer 2019	Research Intern, Wework Topics: Automated Floor Design Generation

Honors and Awards

2013, 2014, 2015	First Prize of Elite Program Scholarship Award
2015	Meritorious Winner of Mathematical Contest in Modeling
2014	2nd Prize in the China Undergraduate Mathematical Contest in Modeling

Abstract

Towards Effort-Saving Knowledge Mining and Reasoning over the Web

by

Hanwen Zha

The web exposes modern humans to ever-growing information about the world. Meanwhile, knowledge is democratized and spread to a wider population, not just privileged elites. However, with broad knowledge buried deeply and diversely under the Internet, the potential of knowledge democratization is not fully exploited: knowledge can be hard to find and digest. Knowledge mining and reasoning techniques acquire such knowledge on unstructured and structured data to satisfy people's craving for knowledge. It pushes forward the democratization of knowledge, making the broader knowledge more accessible to a wider part of the world.

However, for practitioners to build a system and present it to final users, much more human effort is involved in the whole process. (1) From the system aspect, *human supervision* needs to be provided: For a new domain, high cost of data collection, the data-hungry nature of mainstream approaches like neural networks all pose challenges on label efficiency, i.e., to reduce the need for human supervision. (2) From the user aspect, certain *human intelligence* is needed: it takes time for users to digest, understand, accept the returned results. It suggests that the system should provide a global picture and has explainability. So the effort of human intelligence is saved. (3) Users may want intelligent *human-machine interaction*. It is probable they have a very vague query idea at the beginning and need some exploration until they have a clear mind. A human-in-the-loop intelligent system is demanded: it supports iterative query, exploration, refinement, and navigation.

In this dissertation, we propose complementary approaches targeting these aspects towards effort-saving knowledge mining and reasoning. We begin with knowledge mining, which directly harvests knowledge from massive unstructured text. We formulate and mine a graph describing a global picture of scientific development with free weak supervision. We also design a human-in-the-loop system to ease query development and facilitate intelligence exploration of a large text repository. Next, we propose a general-purpose textual relation embedding that is transferable for downstream relation-involving tasks. Finally, we focus on knowledge reasoning, leveraging strong and large pre-trained language models. We propose to use a pre-trained language model to incorporate both structural and textual information of knowledge graph. We also adopt a constrained decoding strategy to the pre-trained language model, successfully applying the generative model in commonsense knowledge base completion. Altogether, these allow a more effort-saving knowledge mining and reasoning, which accelerates the democratization of knowledge.

Contents

Chapter 1

Introduction

The web exposes modern humans to ever-growing information about the world in the past decades. By 2021, there are 6M English articles in Wikipedia [1] – a collectively edited encyclopedia, and 93M items in Wikidata [2] – a collaborative knowledge graph about entities and relations of the world. We have 209M research papers in Microsoft open academic graph [3], storing openly accessible knowledge in scientific publications. People can access daily COVID updates from CDC websites across different states and countries. Such data explosion happens almost in every area of society. Knowledge is democratized and spread to a wider population, not just privileged elites such as scholars and clergies with the digitized world. For example, a novice doctor may want to quickly learn about one disease and find related literature for the treatment. The growth of the data extends the potential of conveniently accessing desired knowledge.

With broad knowledge buried deeply and diversely under the Internet, the potential of knowledge democratization is not fully exploited: knowledge can be hard to find and digest. Non-experts are easily lost in the ocean of the Internet, struggling to acquire the desired piece of information. Knowledge mining and reasoning techniques extract information to satisfy people's craving for knowledge. It pushes forward the democratization of knowledge, making the broader knowledge more accessible to a wider part of the world.

Researches on knowledge mining and reasoning have spanned for several decades. They target knowledge acquisition in two different aspects: Knowledge mining mainly focuses on harvesting explicitly expressed knowledge from massive unstructured text. Knowledge reasoning mostly refers to inferring implicitly expressed missing facts from structured data like the knowledge graph.

Early rule-based knowledge mining methods [4] employ carefully designed patterns to extract desired knowledge. Those systems achieve high precision but are brittle to linguistic variations and usually limited to certain kinds of relations. The rise of machine learning models [5, 6, 7, 8] largely improves the performance especially in supervised settings where labeled data are given. Among them, neural network based models [5, 6, 7] allow accurate extraction in text with more linguistic variations. Knowledge graph reasoning research is recently dominated by embedding-based methods [9, 10, 11, 12, 13, 14] due to its superior performance. They learn a shallow embedding of entities and relations and a scoring function to compose such embeddings for candidate ranking.

However, for practitioners to build a system and present it to final users, much more human effort is involved in the whole process. (1) From the system aspect, *human supervision* needs to be provided: For a new domain, high cost of data collection, the data-hungry nature of mainstream approaches like neural networks all pose challenges on label efficiency, i.e., to reduce the need for human supervision. (2) From the user aspect, certain *human intelligence* is needed: it takes time for users to digest, understand, accept the returned results. It suggests that the system should provide a global picture and has explainability. So the effort of human intelligence is saved. (3) Users may want intelligent *human-machine interaction*. It is probable they have a very vague query idea at the beginning and need some exploration until they have a clear mind. A human-in-the-loop intelligent system is demanded: it supports iterative query, exploration, refinement, and navigation. Those all pose significant challenges to knowledge mining and reasoning, especially for complex domains.

In this dissertation, we study effort-saving knowledge mining and reasoning, a system that satisfies people's knowledge-seeking needs realized by light human effort. We propose complementary approaches towards effort-saving knowledge mining and reasoning. Next, we introduce each direction in more detail.

1.1 Knowledge Mining

To reduce the need for human supervision in collecting labeled data, semi-supervised learning [8] and distant supervision [15] based methods are proposed for knowledge mining. The former explores consistency on unlabeled data, while the latter explores free supervision from existing knowledge bases. Two paradigms partially reduce the human effort in knowledge mining. Orthogonally, we target saving human intelligence in corpus exploration and digesting the global picture. In chapter 2, we formulate a new task of mining an evolution graph describing scientific development. We realize this goal through exploring diverse free weak supervision. In chapter 3, we design an human-in-the-loop intelligence search system. It is a network-based, unified search and navigation platform to ease query development and facilitate intelligence exploration of a large text repository. In chapter 4, we propose a general-purpose embedding of textual relations by capturing global co-occurrence statistics between text and knowledge base. Such knowledge is transferable to download relation-involved tasks to alleviate the need for human supervision.

1.2 Knowledge Reasoning

To save human intelligence for understanding and digesting results, a line of knowledge graph reasoning approaches [16, 17, 18, 19, 20, 21] leverage rules and knowledge graph paths. These approaches usually have the advantage of explainability, by presenting rules and reasoning paths to humans. However, it used to achieve worse performance compared with embedding-based methods in the literature.

We explore the usage of pre-trained language models in the setting of knowledge base completion, which reduces human effort in creating supervision. In chapter 5, we focus on handling inductive learning where unseen entities and relations are present. It eases users when the knowledge graph is dynamically updated. We propose to incorporate both structural and textual information into the pre-trained language model. The model thus enjoys the benefits of both sides. In chapter 6, we study knowledge base completion in the commonsense setting, where the entities and relations are sparse and unnormalized. We show a simple pre-trained generative language model can effectively and efficiently outperform sophisticated methods with a constrained decoding strategy.

Chapter 2

Mining Technique Evolution via Relation Extraction

2.1 Introduction

The number of scientific publications is ever increasing. According to the prominent STM report [22], the number of journal articles published in 2014 alone approached 2.5 million and this number is still increasing on a yearly basis. The long time to digest a scientific paper posts great challenges on the number of papers a researcher can digest. Experienced researchers may be familiar to identify the demanded papers. However, the problem becomes much severe for intelligence analysts who need to browse papers and quickly grasp the major activities in new research areas. The novice researchers may have a similar obstacle in finding out papers related to their research. They usually take plenty of time to come up with keywords, retrieve and read relevant papers and iterate this process.

One step assisting with this process is taxonomy construction [4, 23, 24, 25], which extracts concepts from a collection of documents and builds a tree structure to describe the hierarchical relation between different concepts. Analysts and researchers can follow this concept hierarchy to quickly identify more desired keywords or documents. However, previous taxonomy construction methods mostly focus on isA relation. They either rely on pattern-based methods [4, 23] which extract hierarchical relation leveraging linguistic features, or clustering-based methods [24, 25], which cluster concepts to induce an implicit hierarchy.

In this chapter, we generate a graph called *Algorithm Roadmap*, focusing on a special type of concept – "algorithms", and its specific form – "abbreviations". Given a scientific corpus, we mine comparative algorithms (described in section 2.3), and construct a graph connecting mined algorithms. In Figure 2.1, for example, a roadmap for algorithm Generative Adversarial Network (GAN) [26], describes its successors and competitors in the scientific literature. The generated *algorithm roadmap* captures the development of algorithms, sketches the undergoing research, and models the dynamics of an area. It serves as a tool for analysts and researchers to locate the successors and families of algorithms when doing analysis and survey.

Relation Type	Instance
Single Sentence	We train models using different GAN methods : WGAN-GP , WGAN with weight clipping and DCGAN.
Single Sentence	In almost all experiments BayesGAN outperforms DCGAN and W-DCGAN.
Cross Sentence	LapGAN proposed a Laplacian pyramid GAN. ... DCGAN used a deeper convolutional network.
Cross Sentence	GDL focuses on unsupervised learning. ... GAN and DCGAN show results for unsupervised learning.

Table 2.1: Examples of comparative algorithms.

Conclusively, there exist three major challenges for mining the *algorithm roadmap* in the scientific literature, corresponding to the label, entity, and relation respectively.

Label Scarcity: Collecting in-domain algorithm entities and relation labels in scientific publications are prohibitively expensive. Existing datasets or curated in-domain

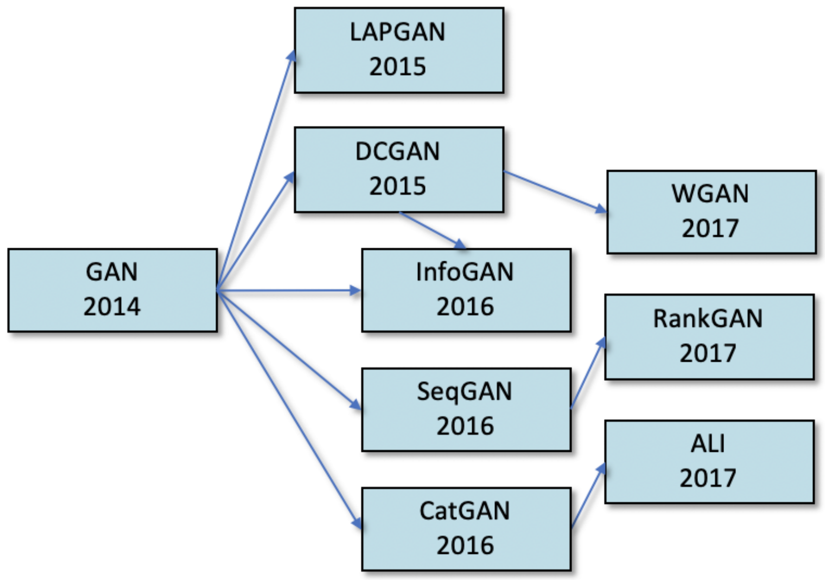


Figure 2.1: A pedagogical example of the *algorithm roadmap* for "GAN" algorithm.

knowledge bases [27, 28] are rather small and frequently outdated with the development of science. Moreover, a newly invented algorithm probably only appears in a single paper. This scarcity raises a challenge for supervised and distantly supervised entity extraction methods like [29, 30] or weakly supervised phrase extraction approaches relying on frequency [31]. The low coverage of knowledge base can also influence the availability of relation labels when using distant supervision [15].

Entity: General entity recognition does not directly separate the algorithms with the others. Though using abbreviations as the representation of algorithms alleviates the problem of considering all types of entities, few types other than algorithm exists. In addition, the abbreviation, as a short form of text, is prone to ambiguity. Word sense disambiguation methods [32] have been studied to disambiguate word senses, however, deciding the sense for the abbreviation in the scientific domain is still challenging when lack of labeled data.

Relation: The narrations of two comparative algorithms either lie in a single sentence or are distributed across sentences. For example, in Figure 2.1, the comparative relation may be described in one sentence, e.g., “Algorithm A outperforms Algorithm B ...,” or in multiple-sentences, e.g., “Algorithm A ... ; Algorithm B ....” Moreover, it is likely more than two algorithms are compared or more than two abbreviations appear in a paragraph. Additional abbreviations may convey a meaning related to comparative relation. Unsupervised pattern-based methods such as [4] focus on isA relation, which are not suitable for finding compared algorithms. Most existing researches for the supervised relation extraction focus on single sentence relation extraction with an exception of [33, 34], which focus on general documents while not targeting on a specific narration of algorithm abbreviations and comparative relation. On the other hand, these supervised approaches require annotated corpora.

We propose a framework to mine the *algorithm roadmap* in scientific publications to tackle the previous raised challenges. It first extracts abbreviations with specific pattern as algorithm candidates. Then it leverages weak supervision from tables and text to create training data for comparative relation identification and entity typing. Next, it applies our proposed relation extraction method *Cross-sentence Attention NeTwork for cOmparative Relation* (CANTOR) to extract comparative algorithms in the text. It leverages words and abbreviations in the context, and jointly predicts the candidate types for addressing ambiguity during the roadmap construction. Finally, it connects the compared algorithms into a graph with time and frequency information.

Extensive experiments on three real-world datasets demonstrate our superior performance in finding the comparative relation. Our CANTOR model outperforms supervised and unsupervised baseline methods by a large margin. We perform case studies on the constructed *algorithm roadmaps* to further visualize the effectiveness of the construction.

2.2 Related Work

Knowledge base construction is a known technique for harvesting knowledge and storing facts in a structured format. The constructed knowledge base plays an important role in downstream applications such as information retrieval, question answering, and document analysis, etc. Most existing automatically constructed knowledge bases focus on general domain, which either extract facts from Wikipedia info-boxes [35, 36] or harvest knowledge with specific linguistic patterns [37, 38]. Taxonomy can be viewed as a tree-structure knowledge graph, where linked nodes have hierarchical relation. Plenty of methods have been proposed to extract these hierarchical relation, either leveraging linguistic patterns [4] or hierarchical clustering of concepts which implicitly captures the hierarchical relation [24]. These methods mainly focus on the general domain, harvesting common knowledge with pattern or statistics.

Many works focus on for mining scientific publications, for example, [27] proposed a keyphrase and relation extraction competition for scientific publications, [28] collected a dataset for scientific taxonomy construction, [39] studied the evolution of scientific topics through dynamic topic models [40] modeling implicit topics and obscure relations, and some technical reports [41, 42] manually analyzed the development of areas such as artificial intelligence. Some of these works collected datasets for scientific publications, but the process is known to be expensive and the collected datasets are normally small in size.

Word sense disambiguation [32] is a type of technique used to distinguish ambiguous word senses. They either disambiguate word senses with a sense inventory or distinguish super senses by clustering words. Inspired by methods using super senses, we use types as evidence to distinguish abbreviations. To leverage the constraint of abbreviations, we use predefined types as super senses for abbreviations.

Another line of work related to ours is relation extraction, which has attracted much attention from the community, while most of the works focus on news and web data [43, 44]. Recent neural network based methods have achieved great success in relation extraction, including CNN-based approaches [5, 7] and LSTM-based approaches [45]. These approaches all consider relations lying in a single sentence. On the other hand, most relation extraction works assume entities and relation sets are given in the datasets, while others apply distant supervision to link entity mentions [15, 46] in the text to the knowledge base entities [30] and acquire relation labels. Their weaknesses lie in the fact that they require either annotated corpora or well-covered knowledge bases.

Beside single-sentence relation approaches, some previous works exist on cross-sentence relation extraction. [47] proposes to construct cross-sentence relation data for entities with minimal-span assumption. [33] proposes to use a Graph-LSTM to encode the shortest path in the extracted dependency parse tree, where the tree roots of different sentences are linked together. [34] proposes a method using self-attention [48] and bi-affine scoring algorithm to predict biological relations between all mention pairs in the abstract simultaneously. Our work differs from them in three key ways. First, we leverage weak supervision from the paper rather than using annotated corpus or distant supervision from an external knowledge base. Second, we consider typing of entities for abbreviation ambiguity and roadmap construction. Third, we model both single-sentence and cross-sentence comparative relations with words and abbreviations in the context.

2.3 Preliminaries

In this paper, we mainly target at mining *algorithm roadmap* in scientific publications. In order to provide a better understanding of our paper, we first give definitions related to *algorithm roadmap* and then briefly overview our proposed method.

Algorithm Roadmap. It is a directed acyclic graph $\mathcal{G}$, where each node of the graph is an algorithm term in abbreviation form. Each directed edge $e_1 \rightarrow e_2$ in the graph $\mathcal{G}$ represents a directed evolutionary relation between two algorithm nodes e_1 and e_2. For example, in the computer science domain, there are algorithms such as GAN (Generative Adversarial Networks) [26] and DCGAN (Deep Convolutional Generative Adversarial Networks) [49]. A directed edge $GAN \rightarrow DCGAN$ represents "DCGAN" is a successor and is evolved from "GAN".

Comparative Relation. It is a relation between two algorithms, which means two terms were compared with each other in some papers. For example, pair $(GAN, DCGAN)$ having comparative relation means "DCGAN" was compared to "GAN" in some papers, but there is no direction information implies which technique is a successor.

Roadmap Construction. We are the first to mine algorithm pairs with comparative relation using weak supervision harvested from tables and texts. Moreover, we connect the compared algorithms into a directed graph $\mathcal{G}$ by deriving order with time and frequency information.

2.4 Extracting Comparative Relation

In this section, we present a framework to extract comparative algorithm pairs from papers. The framework consists of three steps: i) Extracting abbreviations as algorithm candidate mentions; ii) Applying weak supervision from tables and texts to create training data for comparative relation and typing; iii) Learning to predict the relation for candidate mention pairs.

2.4.1 Candidate Mention Extraction

We use abbreviations as our algorithm candidates. The intuition of using abbreviations as algorithm candidates lies in two folds: entity and relation label availability.

Lack of annotated corpus and well-covered in-domain knowledge bases, general entity recognition methods [29, 30] do not fit our candidate mention extraction. With low occurrence frequency, phrase extraction approaches do not satisfy the job as well.

We observed that abbreviation is a commonly used representation of algorithm terms. With a unified form, it is easy to harvest from the corpus. More importantly, using abbreviations as candidates provides a possibility to gather supervision from tables for comparative relation, which we will show in section 2.4.3.

Abbreviations follow specific patterns and may refer to several types of meanings. For example, Table 2.2 shows algorithms such as CNN (Convolutional Neural Network), datasets such as MNIST (Modified National Institute of Standards and Technology dataset), and metrics such as AUC (Area under curve). Types of a few abbreviations can be distinguished by checking the signal words following the abbreviation. For example, an algorithm abbreviation may be followed by algorithm, method, model etc. in the text.

We use regular expression with pattern consists of *capital letters*, *lowercase letters*, *numbers*, and *hypen*, to unsupervisedly harvest abbreviations as algorithm mention candidates from the text. We extract type of a few abbreviations identified by signal words to provide weak supervision for entity typing in section 2.4.2, unidentified abbreviations are randomly sampled as type *Others*.

Type	Abbreviations	Signal Word
Algorithm	CNN, LSTM, GAN	algorithm
Metric	AUC, MAP, MAE	metric
Dataset	MNIST, CIFAR10, SQuAD	dataset
Others	NP, VP, POS	/

Table 2.2: An example of different types of abbreviations.

2.4.2 Cross-Sentence Relation Extraction

We designed our model to incorporate both single-sentence and cross-sentence information, and consider all abbreviations in a paragraph. To this end, our model consists of a single-sentence module with Piecewise CNN [7], and a cross-sentence module which leverages self-attention to attend to all words capturing the paragraph-level relation information, and abbreviation-attention to attend to all abbreviations helping describe the relation of the candidate pair. Moreover, typing is jointly done on the attended candidates to assist downstream roadmap construction. Mention pair predictions are pooled on single-sentence module and cross-sentence module for the entity pair prediction. Finally, the predictions of the two modules are interpolated with weights learned simultaneously with other parameters.

Inputs

Both the single-sentence module and the cross-sentence module take a sequence of N token embeddings in $\mathbb{R}^d$. The input embedding of each token is x_i, which is a concatenation of word embedding and positional embedding [7].

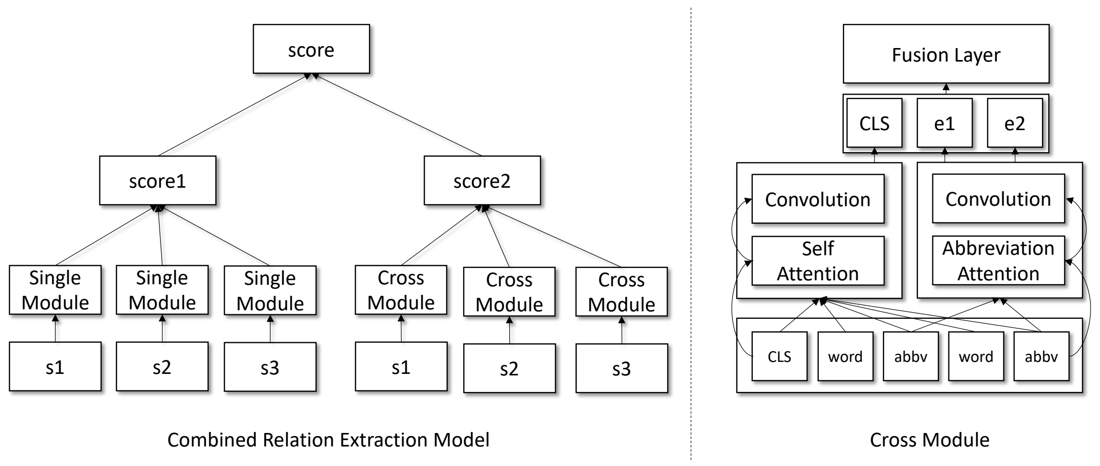


Figure 2.2: Architecture of our CANTOR model.

Single-Sentence Module

We use PCNN (piecewise convolutional neural networks) [7] as our single-sentence relation extractor, which is a well-performed model for short-context relation extraction.

PCNN is a variation of CNN that adopts piecewise max pooling in relation extraction. It divides the sentence into three segments: part before first entity, part between two entities and part after second entity. Thus each convolutional filter q_i is divided into three segments (q_{i1}, q_{i2}, q_{i3}). The max-pooling is performed on three segments separately, which is defined as

$$p_{ij} = max(q_{ij}) \quad 1 \leq i \leq n, \quad 1 \leq j \leq 3 \tag{2.1}$$

where n is the number of convolutional filters, and p_i is equal to the concatenation of p_{ij} over all segments j, which aggregates information from different parts. A non-linear layer is added on top of the sentence relational encoding which is represented by all filters $p_{1:r}$, to get the relation prediction:

$$o_1 = W_1 tanh(p_{1:r}) + b_1. \tag{2.2}$$

Cross-Sentence Module

Our cross-sentence module focuses on finding paragraph-level comparative relation, where two algorithm mention candidates lie across sentences. We base on recent Transformer architecture [48, 50] to build this module, due to its better performance in encoding long-distance context compared to Long Short Term Memory Networks (LSTMs) [51] and Convolutional neural networks (CNNs).

Self-Attention We adapt Transformer [48] to encode word sequences in a paragraph, where we calculate the self-attention of the words, and use a convolutional layer in self-attention blocks similar to [34] to alleviate the burden on the model to attend to local features. We add residual connections [52] to both multi-head attention and convolutional layers. The Transformer contains stacked layers of Transformer block, which contains its own set of parameters. The token embedding $X = \{x_1, ..., x_N\}$ is fed to the first-layer transformer block and the output of the kth-layer block $\widehat{A}_k$ is calculated by

$$\widehat{A}_k = A_k + A_k \times softmax(\frac{A_k^T A_k}{\sqrt{d_{Ak}}}) \tag{2.3}$$

where $softmax(\cdot)$ is a column-wise normalizing function, and d_{Ak} is the dimension of the input token embedding of kth transformer block used for self-attention. A convolutional layer $Conv$ with residual connection follows the self-attention layer:

$$H_{Ak} = \widehat{A}_k + Conv(\widehat{A}_k). \tag{2.4}$$

We follow BERT [50] which recently achieves great success in multiple NLP tasks, to add a special $< CLS >$ token at the start of the paragraph and a special $< SEP >$

token at the end of each sentence in the paragraph. The representation of $< CLS >$ is used for gathering relational information in a paragraph. With self-attention layer, all other tokens in a paragraph attend to this $< CLS >$ token. $< SEP >$ is a special token stands for the end of a sentence, which is used to incorporate the sentence boundary information in the model.

Abbreviation-Attention The abbreviation-attention layer calculates attention over all abbreviations in the sentences. When additional algorithms are also compared or share a similar relation to two candidates, two candidate mentions may have a high probability to be comparative.

Different from the self-attention mechanism, the abbreviation attention is calculated based on all abbreviations in a paragraph. Denoting all token embeddings of abbreviations as B, transformer blocks with a new set of parameters are applied. Similar to self-attention, with kth-layer input embedding B_k, the kth-layer output of abbreviation attention $\widehat{B_k}$ is calculated as

$$\widehat{B_k} = B_k + B_k \times softmax(\frac{B_k^T B_k}{\sqrt{d_{Bk}}}). \tag{2.5}$$

Similarly a convolutional layer with residual connection is applied to the output of abbreviation-attention layer:

$$H_{Bk} = \widehat{B_k} + Conv(\widehat{B_k}). \tag{2.6}$$

With abbreviation-attention layer, all the abbreviations in the sentences are attending to the algorithm candidates. The final output $H_{Bk,e1}$ and $H_{Bk,e2}$ of the algorithm candidates in H_{Bk} are selected as the entity representation, which fuses all abbreviation information in the paragraph.

Character Embedding Some of the abbreviations are rarely mentioned in the text, which may result in an insufficiently trained word embedding. Since the abbreviation is often created by summarized text, similar abbreviations probably imply overlapped word sequences. To leverage this intuition, we use character embedding that describes character-level information of abbreviations and we apply a character-level convolutional layer followed by a max-pooling layer to get a character-level abbreviation representation.

For an abbreviation with corresponding character embedding sequences $C =< c_1, c_2, ..., c_n >$, we apply a convolution kernel followed by a max-pooling layer.

$$H_c = max(Conv(c_i)) \quad 1 \leq i \leq n \tag{2.7}$$

Fusion Layer Finally, We use a single layer on top of the encoded paragraph representation $H_{Ak,<CLS>}$ and the algorithm candidate representation E to model their interaction. The candidate representation E is constructed by concatenating original word embedding X_{e1}, X_{e2}, character embedding $H_{c,e1}, H_{c,e2}$, attended abbreviation embedding $H_{Bk,e1}, H_{Bk,e2}$. The final fusion layer predicts a final relational score for one instance.

$$E = [X_{e1}, X_{e2}, H_{c,e1}, H_{c,e2}, H_{Bk,e1}, H_{Bk,e2}] \tag{2.8}$$

$$o_2 = W_2([H_{Ak,<CLS>}, E]) + b_2 \tag{2.9}$$

Combined Relation Extraction

Predicting whether an algorithm candidate pair is compared forms a multi-instance learning problem [43, 53]. For each pair, a bag of instances may contain two candidates. The entity-level prediction is an aggregation over multiple mention pair instances. Based on different assumptions, different weighting strategies have been proposed such as max-pooling [53] and selective attention [6].

We follow at-least-one assumption, where a positive example has at least one instance implies the comparative relation, and use max-pooling to select the instance with the maximum score for an entity pair in both single-sentence and cross-sentence module.

The final score of an algorithm candidate pair (e_1, e_2) is a interpolation of the aggregated prediction score $O_1(z|S)$ of the single-sentence module and $O_2(z|S)$ of the cross-sentence module. The trainable weights λ_1 and λ_2 are jointly learned from the data to reflect the importance of single-sentence and cross-sentence part. The weights are limited to be positive and have a total sum of 1.

$$\begin{gathered} O(z|S) = \lambda_1 O_1(z|S) + \lambda_2 O_2(z|S) \\ \lambda_1, \lambda_2 > 0, \lambda_1 + \lambda_2 = 1 \end{gathered} \tag{2.10}$$

Finally, we use softmax to normalize the scores to get a probability distribution $p_z = softmax(O(z|S))$, and relation prediction loss is defined as a cross-entropy loss: $L_{RE} = -\sum_{i=1}^{2} y_i \cdot log(p_{z,i})$, where each $y_i \in \{0, 1\}$ indicates algorithm candidate pair relation is true for which class (without/with relation).

Entity Typing

Previous relation extraction modules do not distinguish the types of the abbreviations. Few types other than algorithm exists, even though using abbreviations as algorithm candidates addresses the problem of candidate recognition. In addition, introducing abbreviations may increase chance of ambiguity. For example, "GAN" could be an algorithm (Generative Adversarial Network) but also a gene in biology. "CNN" could be an algorithm Convolutional Neural Network but also a television channel (Cable News Network).

Inspired by word sense disambiguation methods that label super sense types for word clusters [32], we jointly predict the types for abbreviation candidates with relation extraction task to distinguish abbreviations for downstream roadmap construction. Considering the limited types of abbreviation, we pre-define a fixed type inventory instead of using clustering and labeling word clusters.

We use a projection matrix W_3 on top of the attended algorithm candidate representation after abbreviation attention to predict the type of the candidate abbreviation, and the scores are normalized with softmax function: $p_t = softmax(W_t H_{Bke})$.

The type prediction loss also applies the cross-entropy loss: $L_{TP} = -\sum_{i=1}^{T} y_{t,i} \cdot log(p_{t,i})$, where there are total T types and each $y_{ti} \in \{0, 1\}$ indicates the correctness for ith type.

We add a type constraint to the loss function, considering that a comparative relation only holds for candidates with the same type. For a compared algorithm candidate pair $e1, e2$ in the ground truth, a type constraint loss is defined as a kl-divergence of two predicted types, where $L_{TC} = D_{KL}(p_{t,e1}, p_{t,e2})$.

The final score is a weighted sum of all the loss functions, with weights as hyper parameters.

$$L = \gamma_1 \cdot L_{RE} + \gamma_2 \cdot L_{TP} + \gamma_3 \cdot L_{TC} \tag{2.11}$$

2.4.3 Weakly Supervised Training Data

The labels for comparative relation is hardly available from existing datasets and curated knowledge bases. We propose a weakly supervised approach based on our observation that in the same row or same column of the table, mentioned abbreviations are often comparative, including compared algorithms, datasets, or metrics etc. This gives us an opportunity to create positive training examples from the table without human effort.

We first used a table parsing tool [54] to extract tables from raw pdf files of papers. Then we processed the parsed results to identify abbreviations in the same row or column. We enumerated and labeled aligned abbreviation pairs as positive examples with comparative relation. The supervision from the table gives compared abbreviations of various types.

We randomly sample other non-positive candidate pairs as negative examples in training. To reduce the huge number of unrelated and non-informative negative examples, we follow the minimum-span strategy in [47], and limit sampled negative candidate pairs to the co-occurred pairs shown in a limited length of continuous sentences. Intuitively, most compared algorithms are kept since authors tend to describe compared algorithms coherently in a short paragraph.

2.5 Generating Algorithm Roadmap

Previous comparative relation extraction step produces a large set of compared abbreviation pairs, and each pair corresponds to an undirected edge in *algorithm roadmap*. Our goal is to derive direction for the edge and connecting individual pairs.

The evolutionary relation has a strong association with the comparative relation. The publication time is a strong indicator of evolution direction for compared algorithms. We use first occurrence time in the corpus of an abbreviation as an approximation. For those pairs identified with the same occurrence time, we expect usually low frequent algorithm is evolved from high frequent one.

Algorithm 1 Algorithm Roadmap Construction

```
Input: Comparative algorithm list $P$, time dictionary $T$, frequency dictionary $F$
Output: Algorithm roadmap $\mathcal{G}$
1: for all $(e1, e2) \in P$ do
2:     if $T[e1] < T[e2]$ or ($T[e1] = T[e2]$ and $F[e1] > F[e2]$) then
3:         $\mathcal{G}.add(e1 \rightarrow e2)$
4:     else
5:         $\mathcal{G}.add(e2 \rightarrow e1)$
6: return $\mathcal{G}$
```

Example Pairs “GAN” and “DCGAN” are mined compared algorithms. We locate their first appearance time, and find that “GAN” was published first in 2014, and “DCGAN” was first published in 2015. $GAN(2014) \rightarrow DCGAN(2015)$ is predicted as the direction where “DCGAN” is a successor.

When connecting individual pairs, only candidates above certain probability threshold are kept. In addition, candidates with different types in different pairs except for type *Other* are considered as separated nodes for roadmap construction.

2.6 Experiments

The following section is organized in this way, first we describe datasets and implementation details, second, we show held-out and manual evaluation results of different methods in comparative relation extraction task, third, we perform case studies to visualize the constructed *algorithm roadmaps*.

2.6.1 Dataset

We crawled papers from scientific conferences in domains including machine learning, natural language processing, and database. The corpora include papers in NeurIPS/NIPS (Annual Conference on Neural Information Processing Systems) from 1987 to 2017 [1], ACL (Annual Meeting of the Association for Computational Linguistics) from 1974 to 2017 [2], and VLDB (The Proceedings of the VLDB Endowment) from 2008 to 2017 [3]. The statistics of each of datasets is shown in Table 2.3.

Dataset	documents	sentences	positive pairs	abbreviations
NeurIPS	7k	3144k	9k	66k
ACL	5k	2277k	22k	71k
VLDB	2k	1289k	5k	40k

Table 2.3: Dataset statistics of NeurIPS, ACL, VLDB dataset.

From these datasets, we extract algorithm candidate mentions, apply weak supervision to extract types from texts and comparative relation labels from tables as described in section 2.4. We split train and test data with a ratio of 80% and 20%. Among training data, 10% is separated as validation data.

2.6.2 Implementation Details

The paper pdf files are converted into plain text files by using Linux pdftotext tool, and non-ascii letters are removed. For each dataset, we keep a word vocabulary with all abbreviations and other words with minimum frequency threshold 5. The max

[1]https://nips.cc/
[2]http://aclweb.org/anthology/
[3]https://www.vldb.org/pvldb/

paragraph length is set to 160 words, and the max number of continuous sentences considered is set to 20. Paragraphs longer than the threshold are cut off.

The model is implemented in pytorch [55] and trained on a single GeForce GTX 1080 GPU. The dimensions of word embedding, character embedding, and positional embedding are set to 100, 50 and 10 respectively. The word embedding is pre-trained in each scientific publication corpus with Skip-Gram model implemented in Gensim. The kernel size of the convolutional layer is set to 7.

We use 200 filters for the convolutional layer in the single-sentence module, and the same number of filters as input dimension for the convolutional layer in each Transformer block. We apply layer normalization [56] to each component of transformer block, and adopt dropout [57] to the input layer, piece-wise max-pooling and Transformer block with a dropout rate 0.3. The number of Transfomer block layer is set to 1, since we did not observe performance gain in increasing layers.

We use Adam optimizer [58] with a learning rate 0.001. In training, the batch size is set to be 32, and for each positive example, we sample 5 different negative examples. In validation and test, we use all examples. The maximum number of epochs is set to be 16, where the result with best positive-class validation F1 is kept.

2.6.3 Results

We conduct both held-out evaluation and manual evaluation on our method and several baseline methods in the task of comparative relation extraction, where models predict whether given candidate pairs are comparative or not. Evaluated methods can be divided to unsupervised methods including co-occurrence based methods [59], word-similarity based methods [60], and supervised relation extraction methods [7]. The pattern-based method [4] is not compared due to its low recall in our task.

Test set from weakly supervised table data is used for held-out evaluation. The evaluation is harsh due to the limited number of positive examples, and noisy because of few table parsing errors. In the manual evaluation, for each supervised method, we randomly sample 100 examples from their positive predictions and ground truth positive set in test data, and combine those into a unified manual test set. We let human annotators to label the pairs, where we do not distinguish compared algorithms, datasets or metrics following the criteria of our weak supervision. In the following we introduce evaluated methods in detail.

PCNN_single: Piecewise CNN model [7], which is one of the state of art single-sentence relation extraction methods. PCNN_single only uses single-sentence instances for candidate pairs.

PCNN_cross: The same PCNN model as PCNN_single where cross-sentence instances are also used.

Sent_cooccur: A method similar to co-occurrence method used in hypernym detection [59]. Sent_cooccur calculates the co-occurrence frequency of candidate pairs in one sentence. A threshold that decides a positive-negative ratio closest to the ground-truth test data is used.

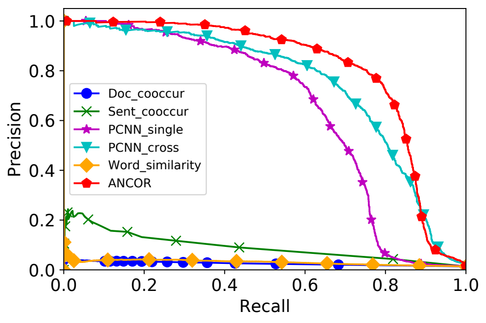
Doc_cooccur: Similar to Sent_cooccur, where the co-occurrence frequency in one document is used instead.

Word_similarity: A method predicts comparative relation score based on word embedding similarity, where the embedding is pre-trained with the Skip-Gram model [60] implemented in Gensim [4] for each corpus. The threshold is decided similarly to Sent_cooccur.

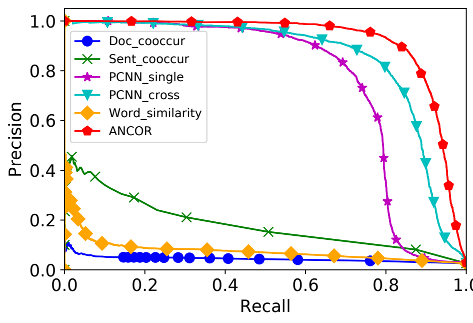
CANTOR: Our proposed cross-sentence relation extraction method, which con-

[4]https://radimrehurek.com/gensim/

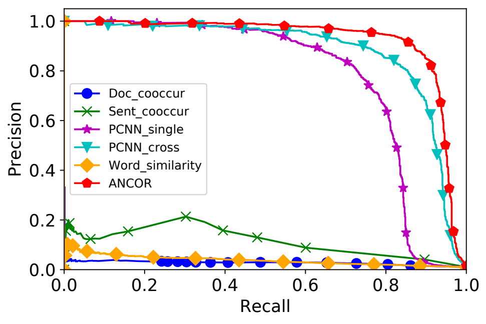
siders both single-sentence and cross-sentence instances, all abbreviations in the context, and jointly typing the candidates.



(a) NeurIPS



(b) ACL



(c) VLDB

Figure 2.3: Precision-Recall curve of different relation extraction methods for finding comparative relation on NeurIPS, ACL and VLDB dataset with held-out evaluation.

Figure 2.3 shows the held-out evaluation and Table 2.4 shows the manual evaluation for all different methods. For held-out evaluation, we draw the precision-recall curve of all methods, and for manual evaluation we calculate the weighted macro F1 and AUC(Area under the ROC Curve). AUC depicts the ranking correctness, where F1 does not take the rank into account.

Overall, due to a limited number of positive examples from weak supervision, the unsupervised methods show a low precision on the held-out evaluation. The manual

Method	NeurIPS		ACL		VLDB	
	AUC	F1	AUC	F1	AUC	F1
sent_cooccur	0.68	0.71	0.66	0.71	0.70	0.67
doc_cooccur	0.57	0.69	0.46	0.68	0.62	0.68
word_similarity	0.62	0.70	0.67	0.72	0.66	0.72
PCNN_single	0.73	0.71	0.72	0.68	0.78	0.67
PCNN_cross	0.75	0.71	0.76	0.74	0.82	0.76
CANTOR (ours)	**0.82**	**0.74**	**0.79**	**0.78**	**0.85**	**0.78**

Table 2.4: Manual evaluation of different relation extraction methods for finding comparative relation on NeurIPS, ACL and VLDB datasets.

test set looses the strict condition of positive examples, moreover, its construction filters most negative examples from the unsupervised methods. These two result in increased performance of unsupervised methods. However, neither co-occurrence based model or similarity is good at modeling the ranking of comparative pairs, thus result in a low AUC.

Co-occurrence is one indicator for comparative relation of abbreviations with good recall while suffering from low precision. This is because counting co-occurrence introduces non-comparative abbreviations into the results. Sentence-level co-occurrence model has a better performance than the document-level model since compared candidates are more likely to appear in a short context. Word similarity model performs between two co-occurrence methods. The word embedding captures the context of type rather than comparative relation. On the other hand, a large number of candidates are rarely mentioned, which leads to insufficient training of word embedding.

The supervised relation extraction methods generally outperform the unsupervised methods. The relation extraction model PCNN_single that uses single-sentence works well, but its precision drops rapidly when recall increases. PCNN_cross considering cross-sentence instances further improves the performance of the model, which shows the importance of cross-sentence instances in finding comparative relation. Our CANTOR method outperforms all these methods, which implies better modeling of cross-sentence comparative relation.

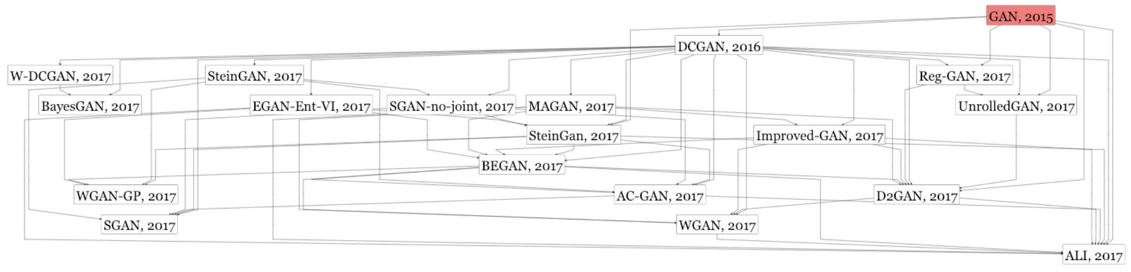
Ablation Study We do an ablation study to show the performance of different components. We use the manual test data in NeurIPS dataset collected by random samples from positive predictions and held-out positive examples from supervised methods to evaluate the components. As shown in Table 2.5, stacking self-attention, abbreviation-attention, typing and combined modeling improves the model performance.

Method	AUC	F1
Self-Attention	0.77	0.72
+Abbreviation-Attention	0.78	0.72
+Typing	0.78	0.74
CANTOR (ours)	**0.82**	**0.74**

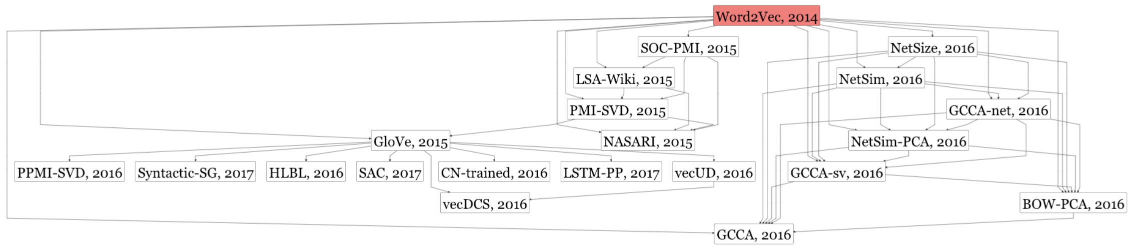
Table 2.5: Ablation study on different components in NeurIPS dataset.

2.6.4 Case Study

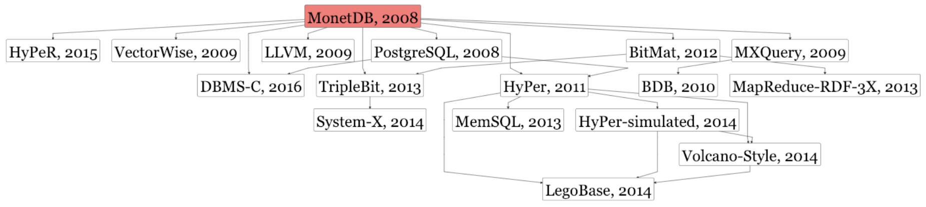
For each dataset, we mine compared algorithms from the entire corpus with our trained CANTOR model and connect the individual pairs with the approach described in section 2.5. In Figure 2.4, we show parts of the *algorithm roadmaps* constructed



(a) Partial roadmap for "GAN" in NeurIPS dataset.



(b) Partial roadmap for "Word2Vec" in ACL dataset.



(c) Partial roadmap for "MonetDB" in VLDB dataset.

Figure 2.4: Examples of partial *algorithm roadmaps* for query "GAN" in NeurIPS dataset, query "Word2Vec" in ACL dataset, and query "MonetDB" in VLDB dataset.

from different datasets. In each figure, each node contains its abbreviation name and its first occurrence time described in section 2.5 in its dataset. To be noticed, this time is not necessarily equal to the first publication time, as the algorithm is non-necessarily published in this conference.

"GAN" (Generative Adversarial Networks) is a deep generative model [26], which has been extensively cited since proposed. Researchers even maintain a "GAN zoo" [5] to keep track of various kinds of "GAN" successors.

In NeurIPS dataset, our method mines its direct successors such as "DCGAN," "SteinGAN," "UnrolledGAN," "Reg-GAN" and "ALI." Then we keep identifying the successors of each successor. For example, "DCGAN" has successors including "W-DCGAN," "SteinGAN," and "Improved-GAN" etc. The comparison of our mined algorithms with algorithms in "GAN zoo" reveals a good precision in found successors. Our current method does not distinguish different forms of an abbreviation, thus "SteinGAN" and "SteinGan" are viewed as separated candidates. A minimum confidence score threshold can be used to control each level of the roadmap to trade off the precision and recall.

Similarly in ACL dataset, query "Word2Vec" usually stands for a word embedding method. Our method identifies its direct successor such as "Glove," "GCCA," "NetSize," and "NetSime." And Glove has successors including "HLBL," "SAC" and "vecDCS" etc. In VLDB dataset, query "MonteDB" is a database management system, our method finds its direct successor such as "VectorWise," "HyperR," "PostgreSQL." And "MXQuery" has successors such as "BDB" and "MapReduce-RDF-3X." Among the results, "LLVM" is a compiler backend used by some database management system. This error comes from incorrect table parsing, and the pair is

[5] https://github.com/hindupuravinash/the-gan-zoo

treated as a positive example in training data.

Overall our method mines compared algorithms with good quality, though it has potential drawbacks. Some errors come from the direction derivation, mostly because of the incorrect time information and the lack of entity linking. For example, in ACL dataset, “LSA-Wiki” is actually a baseline method compared with “Word2Vec” that uses Latent Semantic Analysis used on Wikipedia. However, this abbreviation as a whole first appears in 2015, resulting in an error in direction. On the other hand, the first appearance time of an algorithm in the dataset is non-necessarily the first time this algorithm was proposed since algorithms could firstly show up in other conferences/journals or even in other domains. Some of these conferences/journals are not non-open access, which means data sources for mining *algorithm roadmap* are naturally incomplete. Fortunately, the rise of open access repositories such as arXiv [6], alleviates the data source incompleteness problem.

2.7 Conclusion

We proposed a new task of *mining algorithm roadmap* in scientific literature, and present a weakly supervised method towards solving this problem. Our method automatically identifies candidate mentions and relation labels, extracts comparative relation, and connects individual pairs into a roadmap. We showed that leveraging weak supervision from the table and modeling both single-sentence and cross-sentence narrations from text achieve effective results.

Overall, this work draws the attention of providing easy-to-digest knowledge and a global picture for users. It reduces the human intelligence needed for digesting, by describing area development to users with almost zero supervision.

[6]https://arxiv.org/

Chapter 3

Interactive Knowledge Mining and Corpus Exploration

3.1 Introduction

The number of scientific publications is ever increasing. According to the prominent STM report, the number of journal articles published in 2014 alone approached 2.5 million. It takes much longer time to digest a scientific paper than a webpage, posting a hard constraint on the number of papers a researcher can read. The problem becomes much severe for intelligence analysts who need to browse papers and quickly grasp the major activities in new research areas. They either rely on manually constructed taxonomies (e.g. ACM Computing Classification System) or figure it out through their own reading. Manually constructing taxonomies requires a large amount of human effort, which is not only expensive, but also could be quickly outdated for fast developing areas. Researchers or analysts thus have to read papers and blogs by themselves for finding right keywords to keep track of emerging topics and have a comprehensive view of a research area.

In order to address the aforementioned issues, we develop **FTS** to automatically mine concept terms from a technical corpus selected by the user and construct faceted taxonomy for fast exploration of scientific publications. FTS supports multiple functionalities, including taxonomy construction, unsupervised document categorization, query suggestion and trend analysis, to save the literature search and analysis time.

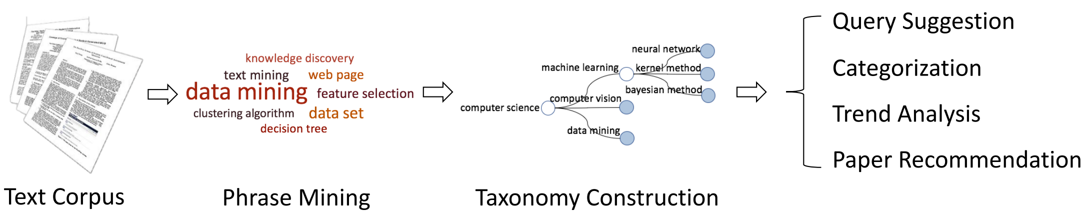


Figure 3.1: The Workflow of FTS

In FTS, we re-examined the automatic taxonomy construction problem [61] by adopting the newest embedding and deep learning techniques. FTS was built upon our latest research results in text mining [62, 63, 64, 65, 66, 67], many of which were published in the recent SIGKDD conferences. The FTS project was developed by researchers from the Army Research Lab, MITRE, UCSB and UIUC.

3.2 Main Functions

The workflow of FTS system is depicted in Figure 3.1. It has the following modules.

3.2.1 Phrase Mining

FTS adopts advanced phrase mining techniques such as SegPhrase [62] and AutoPhrase [63] developed by our team to mine high-quality phrases from massive text data.

The main idea behind these data-driven approaches is to find frequent n-grams from text and rectify the raw frequency with some quality estimation criteria. A classifier is trained to classify phrases based on quality estimation features and labeled phrase examples. Furthermore, with distant supervision from general knowledge bases such as Wikipedia, training labels are automatically generated without human effort. FTS analyzed the title and abstract of 1.2 million computer science papers downloaded from *DBLP*[1] and *Semantic Scholar*[2] and extracted around 180k concept terms.

There are very frequent concepts such as "machine learning" and "data mining," as well as rare, but interesting ones such as "quantum learning" and "quantum neural networks." The mined concepts are building blocks for downstream modules such as taxonomy construction and trend analysis. The text and meta information, as well as the mined concept terms, are indexed by Elasticsearch for fast document retrieval.

3.2.2 Taxonomy Construction

Among the mined concept terms, the important ones are selected and connected together to generate a taxonomy from the underlying text corpus. We have two recent works, TaxonGen [66] and HiExpan [64], to be published in SIGKDD 2018. Both of them developed sophisticated taxonomy construction algorithms.

[1]https://dblp.uni-trier.de/
[2]https://www.semanticscholar.org/

TaxonGen [66] generates taxonomy with hierarchical clustering and term embeddings. Each node of the resulting taxonomy is a cluster of concept terms and its representative term is automatically selected. The spherical clustering and locally-trained term embeddings were proposed to boost the taxonomy quality. HiExpan [64] focuses on interactive taxonomy generation under user guidance. With weakly-supervised set expansion and depth expansion, the taxonomy is constructed to fit the user's desire. The generated results could be further adapted to multi-faceted taxonomies with different facets. With AutoPhrase, TaxonGen and HiExpan, FTS can dynamically generate concept terms and taxonomies based on a subset of documents identified by users. Users can also interact with the system to further refine the taxonomy.

Suppose an analyst wants to investigate what is going on in an emerging research area, e.g., "quantum learning." She could either upload a paper collection related to "quantum learning" or use Elasticsearch to retrieve a set of related papers in our system. Next, she could generate a taxonomy for this specific collection. Based on the result, she may modify the paper collection or query (e.g. adding an additional term "quantum neural networks"), repeat the above process and refine the taxonomy. These two steps can iterate.

3.2.3 Intelligent Categorization and Search

Once a taxonomy is built, the next step is to put publications in different taxonomy nodes. An unsupervised document categorization technique, UNEC [67], was developed in FTS. UNEC is a cascade embedding approach: Based on a concept similarity graph built from concept embedding, the concepts are embedded into a hidden category space given only category names. UNEC can quickly help analysts identify research hot spots in a taxonomy.

In addition to categorization, FTS also provides query suggestion capability which can suggest terms related to a user query. This function can help a user to adjust the targeted documents for further analysis. For example, given a query "quantum learning," FTS could suggest "quantum Turing machine," "RL algorithm" etc. FTS developed an embedding based query suggestion approach, which leverages the mined concepts and discovers the relation among these concepts through word embedding. In addition, SetExpan [65] is adopted in FTS in order to expand the query set and rank related concepts. SetExpan expands the query set by using skip-gram features and ensembles the ranks of expanded terms.

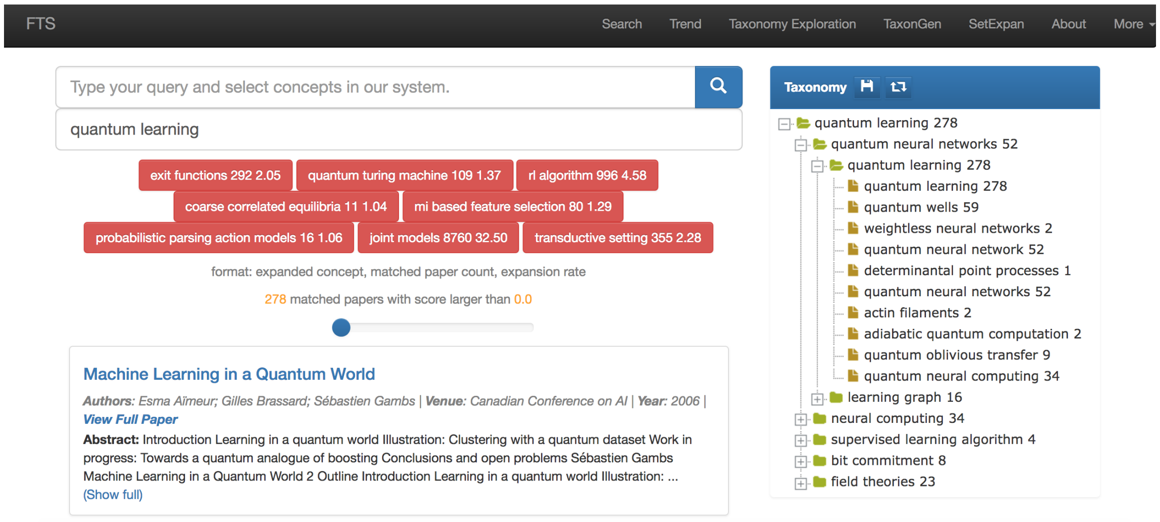
3.2.4 Trend Analysis

When the time information of publications is plugged in, FTS can naturally support trend analysis. It could show the research focus change over time, and compare the research strength of different countries or topics, e.g. showing the strength and the developing trend of "quantum learning" in countries like Russia, China, and the United States. In that way, analysts can be aware of subareas that are booming as well as who are actively involved in those areas.

3.3 Demonstration

We developed a user-friendly web site for analysts to easily interact with FTS. A prototype system is available at `http://fts.cs.ucsb.edu/`. Figure 3.2 shows a screen shot of FTS.

In this demo, users can search papers with their own query, interactively refine queries, select targeted documents and generate taxonomies. Users can navigate


FTS Search Trend Taxonomy Exploration TaxonGen SetExpan About More

Type your query and select concepts in our system.

quantum learning

exit functions 292 2.05 | quantum turing machine 109 1.37 | rl algorithm 996 4.58 | coarse correlated equilibria 11 1.04 | mi based feature selection 80 1.29 | probabilistic parsing action models 16 1.06 | joint models 8760 32.50 | transductive setting 355 2.28

format: expanded concept, matched paper count, expansion rate

278 matched papers with score larger than 0.0

Machine Learning in a Quantum World

Authors: *Esma Aïmeur; Gilles Brassard; Sébastien Gambs* | *Venue:* *Canadian Conference on AI* | *Year:* *2006* |
View Full Paper
Abstract: Introduction Learning in a quantum world Illustration: Clustering with a quantum dataset Work in progress: Towards a quantum analogue of boosting Conclusions and open problems Sébastien Gambs Machine Learning in a Quantum World 2 Outline Introduction Learning in a quantum world Illustration: ...
(Show full)

Taxonomy

- quantum learning 278
  - quantum neural networks 52
    - quantum learning 278
      - quantum learning 278
      - quantum wells 59
      - weightless neural networks 2
      - quantum neural network 52
      - determinantal point processes 1
      - quantum neural networks 52
      - actin filaments 2
      - adiabatic quantum computation 2
      - quantum oblivious transfer 9
      - quantum neural computing 34
    - learning graph 16
  - neural computing 34
  - supervised learning algorithm 4
  - bit commitment 8
  - field theories 23

Figure 3.2: A Snapshot for Query "Quantum Learning": Suggested query terms, query result and generated taxonomy.

the taxonomy and pick up a few interesting sub-areas and their papers for detailed examination. This will liberate them from coming up with concept terms they are not familiar with. Users can also visualize trends of different topics through time.

A strong motivation for this demonstration is to show the power of text data mining for analyzing scientific publications. Without FTS, one has to read and manually label many research papers before a comprehensive view can be derived. FTS simplifies this process. We hope FTS will benefit many researchers and promote research towards advanced mining of scientific literature.

Chapter 4

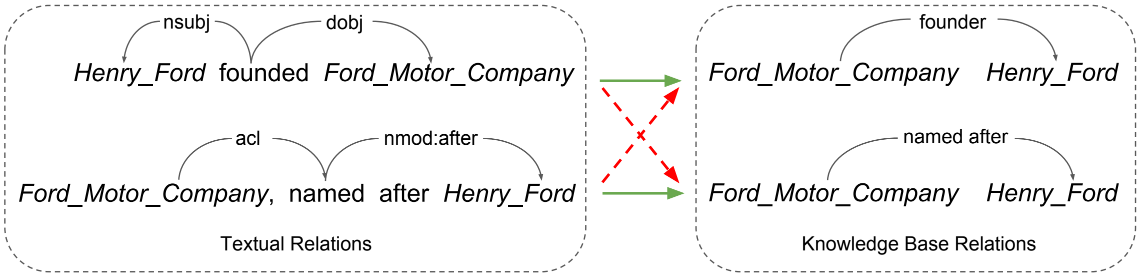
Transferable Global Textual Relation Embedding

4.1 Introduction

Pre-trained embeddings such as word embeddings [60, 68, 69, 50] and sentence embeddings [70, 71] have become fundamental NLP tools. Learned with large-scale (e.g., up to 800 billion tokens [68]) open-domain corpora, such embeddings serve as a good prior for a wide range of downstream tasks by endowing task-specific models with general lexical, syntactic, and semantic knowledge.

Inspecting the spectrum of granularity, a representation between lexical (and phrasal) level and sentence level is missing. Many tasks require relational understanding of the entities mentioned in the text, e.g., relation extraction and knowledge base completion. Textual relation [72], defined as the shortest path between two entities in the dependency parse tree of a sentence, has been widely shown to be the main bearer of relational information in text and proved effective in relation extraction tasks [73, 74]. If we can learn a *general-purpose embedding for textual relations*, it may facilitate many downstream relational understanding tasks by providing general relational knowledge.

Similar to language modeling for learning general-purpose word embeddings, distant supervision [15] is a promising way to acquire supervision, at no cost, for training general-purpose embedding of textual relations. Recently Su et al. [74] propose to leverage global co-occurrence statistics of textual and KB relations to learn embeddings of textual relations, and show that it can effectively combat the wrong labeling problem of distant supervision (see Figure 4.1 for example). While their method, named GloRE, achieves the state-of-the-art performance on the popular New York Times (NYT) dataset [43], the scope of their study is limited to relation extraction with small-scale in-domain training data.


nsubj
dobj
Henry_Ford founded Ford_Motor_Company
acl
nmod:after
Ford_Motor_Company, named after Henry_Ford
Textual Relations
founder
Ford_Motor_Company Henry_Ford
named after
Ford_Motor_Company Henry_Ford
Knowledge Base Relations

	$\xleftarrow{dobj} founded \xrightarrow{nsubj}$	$\xrightarrow{acl} named \xrightarrow{nmod:after}$
founder	2468.0	24.0
named_after	305.0	347.0
...	...	...

Figure 4.1: *Up:* The wrong labeling problem of distant supervision. The Ford Motor Company is both founded by and named after Henry Ford. The KB relation *founder* and *named_after* are thus both mapped to all of the sentences containing the entity pair, resulting in many wrong labels (red dashed arrows). *Down:* Global co-occurrence statistics from our distant supervision dataset, which clearly distinguishes the two textual relations.

In this work, we take the GloRE approach further and apply it to large-scale, domain-independent data labeled with distant supervision, with the goal of learning general-purpose textual relation embeddings. Specifically, we create the largest ever distant supervision dataset by linking the entire English ClueWeb09 corpus (half a billion of web documents) to the latest version of Freebase [35], which contains 45 million entities and 3 billion relational facts. After filtering, we get a dataset with over 5 million unique textual relations and around 9 million co-occurring textual and KB relation pairs. We then train textual relation embedding on the collected dataset in a way similar to [74], but using Transformer [48] instead of vanilla RNN as the encoder for better training efficiency.

To demonstrate the usefulness of the learned textual relation embedding, we experiment on two relational understanding tasks, relation extraction and knowledge base completion. For relation extraction, we use the embedding to augment PCNN+ATT [6] and improve the precision for top 1000 predictions from 83.9% to 89.8%. For knowledge base completion, we replace the neural network in [75] with our pre-trained embedding followed by a simple projection layer, and gain improvements on both MRR and HITS@10 measures. Our major contributions are summarized as following:

- We propose the novel task of learning general-purpose embedding of textual relations, which has the potential to facilitate a wide range of relational understanding tasks.
- To learn such an embedding, we create the largest distant supervision dataset by linking the entire English ClueWeb09 corpus to Freebase. The dataset is publicly available[1].

[1]https://github.com/czyssrs/GloREPlus

- Based on the global co-occurrence statistics of textual and KB relations, we learn a textual relation embedding on the collected dataset and demonstrate its usefulness on relational understanding tasks.

4.2 Related Work

Distant supervision methods [15] for relation extraction have been studied by a number of works [43, 76, 53, 7, 6, 77, 78]. [74] use global co-occurrence statistics of textual and KB relations to effectively combat the wrong labeling problem. But the global statistics in their work is limited to NYT dataset, capturing domain-specific distributions.

Another line of research that relates to ours is the universal schema [79] for relation extraction, KB completion, as well as its extensions [75, 80]. Wrong labeling problem still exists since their embedding is learned based on individual relation facts. In contrast, we use the global co-occurrence statistics as explicit supervision signal.

4.3 Textual Relation Embedding

In this section, we describe how to collect large-scale data via distant supervision (§4.3.1) and train the textual relation embedding (§4.3.2).

4.3.1 Global Co-Occurrence Statistics from Distant Supervision

To construct a large-scale distant supervision dataset, we first get the English ClueWeb09 corpus [81], which contains 500 million web documents. We employ the

FACC1 dataset [82] to map ClueWeb09 to Freebase. We identify over 5 billion entity mentions in ClueWeb09 and link them to Freebase entities. From the linked documents, we extract 155 million sentences containing at least two entity mentions. We then use the Stanford Parser [83] with universal dependencies to extract textual relations (shortest dependency paths) between each pair of entity mentions[2], leading to 788 million relational triples (subject, textual relation, object), of which 451 million are unique.

Following [74], we then collect the global co-occurrence statistics of textual and KB relations. More specifically, for a relational triple (e_1, t, e_2) with textual relation t, if (e_1, r, e_2) with KB relation r exists in the KB, then we count it as a co-occurrence of t and r. We count the total number of co-occurrences of each pair of textual and KB relation across the entire corpus. We then normalize the global co-occurrence statistics such that each textual relation has a valid probability distribution over all the KB relations, which presumably captures the semantics of the textual relation. In the end, a bipartite relation graph is constructed, with one node set being the textual relations, the other node set being the KB relations, and the weighted edges representing the normalized global co-occurrence statistics.

Filtering. When aligning the text corpus with the KB, we apply a number of filters to ensure data quality and training efficiency: (1) We only use the KB relations in Freebase Commons, 70 domains that are manually verified to be of release quality. (2) Only textual relations with the number of tokens (including both lexical tokens and dependency relations) less than or equal to 10 are kept. (3) Only non-symmetric textual relations are kept, because symmetric ones are typically from conjunctions like "and" or "or", which are less of interest. (4) Only textual relations with at least two

[2]To be more precise, only shortest dependency paths without any other entity on the path are extracted.

occurrences are kept. After filtering, we end up with a relation graph with *5,559,176* unique textual relations, *1,925* knowledge base (KB) relations, and *8,825,731* edges with non-zero weight. It is worth noting that these filters are very conservative, and we can easily increase the scale of data by relaxing some of the filters.

4.3.2 Embedding Training

Considering both effectiveness and efficiency, we employ the Transformer encoder [48] to learn the textual relation embedding. It has been shown to excel at learning general-purpose representations [50].

The embedded textual relation token sequence is fed as input. For example, for the textual relation $\xleftarrow{dobj}$ $founded$ $\xrightarrow{nsubj}$, the input is the embedded sequence of $\{< -dobj >, founded, < nsubj >\}$. We project the output of the encoder to a vector z as the result embedding. Given a textual relation t_i and its embedding z_i, denote $\{r_1, r_2, ..., r_n\}$ as all KB relations, and $\tilde{p}(r_j|t_i)$ as the global co-occurrence distribution, the weight of the edge between textual relation t_i and KB relation r_j in the relation graph. The training objective is to minimize the cross-entropy loss:

$$L = -\sum_{i,j} \tilde{p}(r_j|t_i) log(p(r_j|t_i)), \tag{4.1}$$

Where

$$p(r_j|t_i) = (softmax(Wz_i + b))_j. \tag{4.2}$$

W and b are trainable parameters.

We use the filtered relation graph in §4.3.1 as our training data. To guarantee that the model generalizes to unseen textual relations, we take 5% of the training data as validation set. Word embeddings are initialized with the GloVe [68] vectors[3]. Dependency relation embeddings are initialized randomly. For the Transformer model, we

[3]https://nlp.stanford.edu/projects/glove/

use 6 layers and 6 attention heads for each layer. We use the Adam optimizer [58] with parameter settings suggested by the original Transformer paper [48]. We train a maximum number of 200 epochs and take the checkpoint with minimum validation loss for the result.

We also compare with using vanilla RNN in GloRE [74]. Denote the embedding trained with Tranformer as **GloRE++**, standing for both new data and different model, and with RNN as **GloRE+**, standing for new data. We observe that, in the early stage of training, the validation loss of RNN decreases faster than Transformer. However, it starts to overfit soon.

4.4 Experiments

In this section, we evaluate the usefulness of the learned textual relation embedding on two popular relational understanding tasks, relation extraction and knowledge base completion. *We do not fine-tune the embedding*, and only use in-domain data to train a single feedforward layer to project the embedding to the target relations of the domain. We compare this with models that are specifically designed for those tasks and trained using in-domain data. If we can achieve comparable or better results, it demonstrates that the general-purpose embedding captures useful information for downstream tasks.

4.4.1 Relation Extraction

We experiment on the popular New York Times (NYT) relation extraction dataset [43]. Following GloRE [74], we aim at augmenting existing relation extractors with the textual relation embeddings. We first average the textual relation embeddings of all

Precision@N	100	300	500	700	900	1000
PCNN+ATT	97.0	93.7	92.8	89.1	85.2	83.9
PCNN+ATT+GloRE	97.0	97.3	94.6	93.3	90.1	89.3
PCNN+ATT+GloRE+	**98.0**	**98.7**	**96.6**	93.1	89.9	88.8
PCNN+ATT+GloRE++	**98.0**	97.3	96.0	**93.6**	**91.0**	**89.8**

Table 4.1: Relation extraction manual evaluation results: Precision of top 1000 predictions.

Model	Overall		With mentions		Without mentions	
	MRR	HITS@10	MRR	HITS@10	MRR	HITS@10
DistMult (KB only)	35.8	51.8	27.3	39.5	39	56.3
Conv-DistMult	36.5	52.5	28.5	41.4	39.4	56.5
Emb-DistMult (GloRE+)	36.4	52.6	**28.8**	**41.8**	39.3	56.7
Emb-DistMult (GloRE++)	**36.6**	**53.0**	28.0	40.8	**39.8**	**57.1**
E+DistMult (KB only)	37.8	53.5	29.5	43	40.9	57.3
Conv-E+Conv-DistMult	38.7	**54.4**	**30.0**	**43.8**	41.9	58.2
Emb-E+Emb-DistMult (GloRE+)	38.8	54.2	**30.0**	43.3	42.0	58.2
Emb-E+Emb-DistMult (GloRE++)	**38.9**	**54.4**	**30.0**	43.5	**42.1**	**58.3**

Table 4.2: Results of KB completion on FB15k-237 dataset

contextual sentences of an entity pair, and project the average embedding to the target KB relations. We then construct an ensemble model by a weighted combination of predictions from the base model and the textual relation embedding.

Same as [74], we use PCNN+ATT [6] as our base model. GloRE++ improves its best F_1-score from 42.7% to 45.2%, slightly outperforming the previous state-of-the-art (GloRE, 44.7%). As shown in previous work [74], on NYT dataset, due to a significant amount of false negatives, the PR curve on the held-out set may not be an

Subject and object	Francis Clark Howell, Kansas City	
KB relation	people.person.place_of_birth	
Textual relation in NYT train set	$\xleftarrow{nsubjpass} born \xrightarrow{nmod:on} nov. \xrightarrow{nmod:in}$	
Corresponding sentence in NYT train set	...Francis Clark Howell was born on nov. 27, 1925, in Kansas City, ...	
Top-5 nearest neighbors in ClueWeb train set		
Textual relation	Cosine similarity	A corresponding sentence in ClueWeb raw data
$\xleftarrow{nsubjpass} born \xrightarrow{nmod:in} 1295 \xrightarrow{nmod:in}$	0.61	...According to the Lonely Planet Guide to Venice, St. Roch was born in 1295 in Montpellier, France, and at the age of 20 began wandering...
$\xleftarrow{nsubjpass} born \xrightarrow{nmod:in} 1222 \xrightarrow{nmod:in}$	0.61	...Isabel BIGOD was born in 1222 in Thetford Abbey, Norfolk, England...
$\xleftarrow{nsubjpass} born \xrightarrow{dobj} Lannerback \xrightarrow{nmod:in}$	0.60	...Yngwie (pronounced "ING-vay") Malmsteen was born Lars Johann Yngwie Lannerback in Stockholm, Sweden, in 1963, ...
$\xleftarrow{nsubjpass} born \xrightarrow{nmod:in} Leigha \xrightarrow{appos} Muzaffargarh \xrightarrow{nmod:in}$	0.57	...Satya Paul - Indian Designer Satya Paul was born in Leigha, Muzaffargarh in Pakistan, and came to India during the partition times...
$\xleftarrow{nsubjpass} born \xrightarrow{nmod:on} raised \xrightarrow{nmod:in}$	0.55	...Governor Gilmore was born on October 6, 1949 and raised in Richmond, Virginia...

Table 4.3: Case study: Textual relation embedding model can well generalize to unseen textual relations via capturing common shared sub-structures.

accurate measure of performance. Therefore, we mainly employ manual evaluation. We invite graduate students to check top 1000 predictions of each method. They are present with the entity pair, the prediction, and all the contextual sentences of the entity pair. Each prediction is examined by two students until reaching an agreement after discussion. Besides, the students are not aware of the source of the predictions. Table 4.1 shows the manual evaluation results. Both GloRE+ and GloRE++ get improvements over GloRE. GloRE++ obtains the best results for top 700, 900 and 1000 predictions.

4.4.2 Knowledge Base Completion

We experiment on another relational understanding task, knowledge base (KB) completion, on the popular FB15k-237 dataset [75]. The goal is to predict missing relation facts based on a set of known entities, KB relations, and textual mentions. [75] use a convolutional neural network (CNN) to model textual relations. We replace their CNN with our pre-trained embedding followed by one simple feed-forward projection layer.

As in [75], we use the best performing DistMult and E+DistMult as the base models. DistMult [84] learns latent vectors for the entities and each relation type, while model E [79] learns two latent vectors for each relation type, associated with its subject and object entities respectively. E+DistMult is a combination model that ensembles the predictions from individual models, and is trained jointly. We conduct experiments using only KB relations (*KB only*), using their CNN to model textual relations (*Conv*), and using our embedding to model textual relations (*Emb*).

The models are tested on predicting the object entities of a set of KB triples disjoint from the training set, given the subject entity and the relation type. Table

4.2 shows the performances of all models measured by mean reciprocal rank (MRR) of the correct entity, and HITS@10 (the percentage of test instances for which the correct entity is ranked within the top 10 predictions). We also show the performances on the two subsets of the test set, with and without textual mentions. The pre-trained embedding achieves comparable or better results to the CNN model trained with in-domain data.

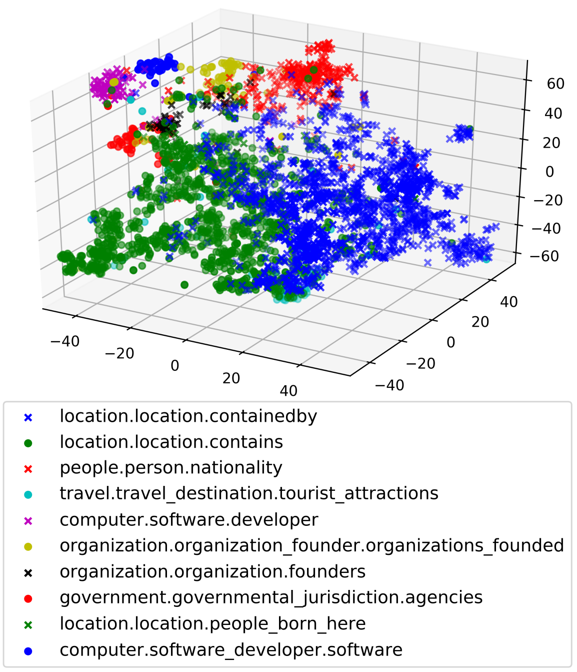


Figure 4.2: t-SNE visualization of our textual relation embeddings on ClueWeb validation data

[3]The result of our implementation is slightly different from the original paper. We have communicated with the authors and agreed on the plausibility of the result.

4.5 Analysis

t-SNE visualization. To measure the intrinsic property of the learned textual relation embedding, we apply t-SNE visualization [85] on the learned embedding of ClueWeb validation set.

We filter out infrequent textual relations and assign labels to the textual relations when they co-occur more than half of the times with a KB relation. The visualization result of GloRE++ embedding associating with the top-10 frequent KB relations is shown in Figure 4.2. As we can see, similar textual relations are grouped together while dissimilar ones are separated. This implies that the embedding model can well generate textual relation representation for unseen textual relations, and can potentially serve as relational features to help tasks in unsupervised setting.

Case Study. To show that the embedding model generalizes to unseen textual relations via capturing crucial textual sub-patterns, we randomly pick some textual relations in NYT train set but not in ClueWeb train set, and compare with its top-5 nearest neighbors in ClueWeb train set, based on the similarity of the learned embedding. A case study is shown in Table 4.3. We can see that the KB relation *place_of_birth* often collocates with a preposition *in* indicating the object fits into a location type, and some key words like *born*. Together, the sub-structure *born in* serves as a strong indicator for *place_of_birth* relation. There is almost always some redundant information in the textual relations, for example in the textual relation $\xleftarrow{nsubjpass} born \xrightarrow{nmod:on} nov. \xrightarrow{nmod:in}$, the sub-structure $\xrightarrow{nmod:on} nov.$ does not carry crucial information indicating the target relation. A good textual relation embedding model should be capable of learning to attend to the crucial semantic patterns.

4.6 Conclusion

We introduce a new type of pre-trained embedding to represent textual relations. To solicit global co-occurrence frequencies as supervision signal from large open-domain corpus, we create the largest distant supervision dataset by linking the entire English ClueWeb09 to Freebase. Experiments on relation extraction and KB completion demonstrate that the learned embedding provides good prior knowledge to boost model performance. It reduces the human effort needed for designing task-specific textual relation models or collecting more labeled data.

Chapter 5

Modeling Structure and Text of KB with Pre-trained LM

5.1 Introduction

Knowledge graphs (KGs) are essential in a wide range of tasks such as question answering and recommendation systems [86]. As many knowledge graphs are substantially incomplete in practice, knowledge graph completion (KGC) becomes a must in many applications [87].

Embedding-based methods such as TransE [9], Complex [10], ConvE [88], RotatE [12] and TuckER [13], achieve the state-of-the-art performance on a few KGC benchmarks. However, the drawbacks of these approaches are obvious as they are limited to the *transductive* setting where entities and relations need to be seen at training time. In reality, new entities and relations emerge over time (*inductive* setting). The cost of retraining may be too high for dynamically populated knowledge graphs. In addition to the inductive setting, explainability, few-shot learning and transfer learning cannot be easily solved by these specialized embedding methods.

Logical induction methods partially meet the aforementioned need by seeking probabilistic subgraph patterns (GraIL [89]), logical rules (AMIE [16], RuleN [17]) or their differentiable counterparts (NEURAL-LP [18], DRUM [19]). The following shows a logical rule which is explainable, can be generalized, and can handle unseen entities,

$$(x, \textit{president\_of}, y) \wedge (z, \textit{capital\_of}, y) \rightarrow (x, \textit{work\_at}, z). \quad (5.1)$$

These logical rules introduce inductive ability for predicting missing links in KG. For example, once the rule in (5.1) is learned, the model can generalize to other *president*, *capital* and *country*.

Despite the compelling advantage of the existing logical induction methods, their inductive learning power is limited as it only exploits the structural information while ignoring the textual information associated with entities and relations, and furthermore, prior knowledge carried in these texts. This weakens the model's usability when only small knowledge graphs are available – a typical few-shot setting. Moreover, none of them can handle unseen but relevant relations in KG completion.

In this work, we propose an all-in-one solution, called BertRL (BERT-based Relational Learning), a model that combines rule-based reasoning with textual information and prior knowledge by leveraging pre-trained language model, BERT [90]. In BertRL, we linearize the local subgraph around entities in a target relation (h, r, t) into paths $p : (h, r_0, e_1), (e_1, r_1, e_2), \ldots, (e_n, r_n, t)$, input $(h, r, t) : p$ to BERT, and then fine-tune. BertRL is different from KG-BERT [91] where only relation instance (h, r, t) is fed to BERT. While this difference looks small, it actually lets BertRL reason explicitly via paths connecting two entities. KG-BERT's prediction is mainly based on the representation of entities and relations: Knowledge graph is memorized

Method	**Transductive Setting**	**Inductive Setting**			**Prior Knowledge**	**Explainable**
		Unseen Entities	**Unseen Relations**	**Reasoning with context**		
TuckER	✓	×	×	×	×	×
RuleN	✓	✓	×	✓	×	✓
GraIL	✓	✓	×	✓	×	×
KG-BERT	✓	✓	✓	×	✓	×
BERTRL (ours)	✓	✓	✓	✓	✓	✓

Table 5.1: Comparison of BertRL with other relation prediction algorithms on their capability of handling the transductive setting, unseen entities in the inductive setting, their potential of dealing with unseen relations, usage of prior knowledge, the explainability of their inference process, and whether they can reason with the context of entities in the knowledge graph explicitly. BertRL and KG-BERT are provided with knowledge graph, entity names and relation names. We take TuckER as a representative of embedding-based methods.

inside BERT and reasoning is implicit. In BertRL, knowledge is dynamically retrieved from the knowledge graph during inference: Reasoning is conducted explicitly, which enables BertRL to achieve explainability and much higher accuracy. Table 5.1 illustrates the difference among these approaches.

Our approach naturally generalizes to unseen entities. It also has the potential to handle some unseen relations. Empirical experiments on inductive knowledge graph completion benchmarks demonstrate the superior performance of BertRL in comparison with state-of-the-art baselines: It achieves an absolute increase of 6.3% and 6.5% in Hits@1 and MRR on average. In a few-shot learning scenario, it can even achieve a maximum of 32.7% and 27.8% absolute Hits@1 and MRR improvement.

In the transductive setting, BertRL performs competitively with the state-of-the-art embedding methods and surpasses the inductive learning counterparts. In few-shot learning (partially transductive), BertRL again introduces a larger margin

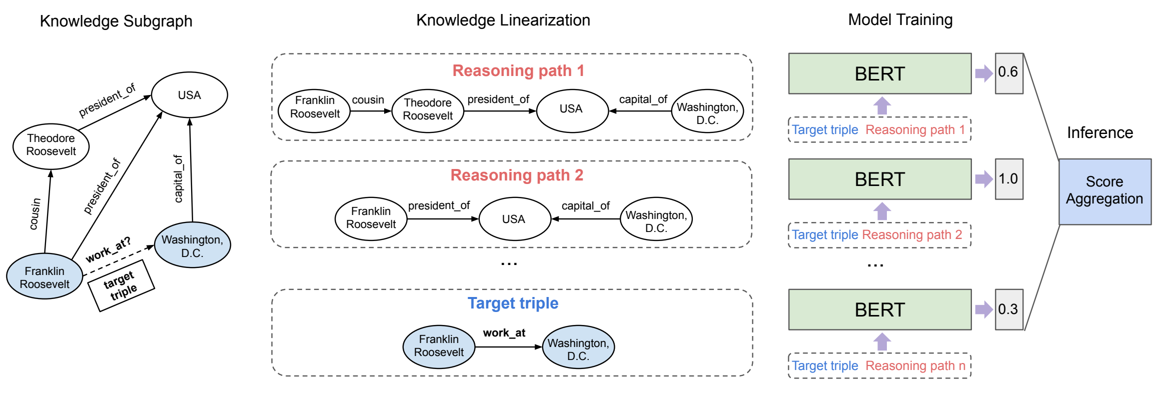


Figure 5.1: The BertRL pipeline.

over the baselines.

Finally, we analyze how BertRL performs in unseen relation prediction, its explainability, its training and inference time, and conduct an ablation study on a few design choices.

5.2 Inductive Relation Prediction by BERT

Problem Formulation. Knowledge graph consists of a set of triples $\{(h_i, r_i, t_i)\}$ with head, tail entities $h_i, t_i \in \mathcal{E}$ (the set of entities) and relation $r_i \in \mathcal{R}$ (the set of relations). Given an incomplete knowledge graph G, the relation prediction task is to score the probability that an unseen relational triple (h, r, t) is true, where h and t denote head and tail entities and r refers to a relation. (h, r, t) is also called target relational triple.

Our model scores a relational triple in two steps: (Step 1) Extracting and linearizing the knowledge $G(h, t)$ surrounding entities h and t in G; (Step 2) Scoring the triple with $G(h, t)$ by fine-tuning the pre-trained language model BERT.

5.2.1 Model Details

Step 1: Knowledge Linearization. The knowledge $G(h,t)$ surrounding entities h and t in a knowledge graph G provides important clues for predicting missing links between h and t. $G(h,t)$ could be exploited in various ways: It could be any subgraph around h and t and even not necessarily be connected. However, the different choices of $G(h,t)$ will affect the model complexity and its explainability. RuleN [17] uses all the paths connecting h and t up to k length. GraIL [89] uses a subgraph that merges all of these paths, aiming to leverage structural information. In order to use pre-trained language models like BERT, we need to linearize $G(h,t)$ as $\ell(G(h,t))$ and concatenate it with (h,r,t) as valid input to BERT,

$$(h,r,t) : \ell(G(h,t)). \tag{5.2}$$

Our intuition is that BERT shall have the capability of learning signals in $G(h,t)$ that could be correlated with (h,r,t), and BERT shall be able to handle noisy and erroneous inputs.

Subgraph. One straightforward linearization of a subgraph would be concatenating text of its edges one by one separated by a delimiter such as a semicolon. This formalism has two major issues. First, local subgraphs could be very large: The size grows exponentially with respect to their diameters. Hence concatenated edges may not fit into the available BERT models. Second, the subgraph edges are unordered, which might incur additional cost for BERT to learn orders and produce correct scoring. We will show experiment of subgraph-based linearization design in Section 5.3.7.

Paths. Another linearization method is collecting all of the paths up to length k connecting h and t. We call them *reasoning paths*. Each reasoning path between h

and t consists of a sequence of triples $h \to t : (h, r_0, e_1), (e_1, r_1, e_2), ..., (e_n, r_n, t)$.

There are two ways of leveraging reasoning paths: One called *combined paths*, puts all the paths together as one input to BERT, thus allowing the interaction across different path units. The other called *individual paths*, takes each path as a separate input to BERT. Each reasoning path induces the target triple individually with a certain confidence score, and the final result is an aggregation of individual scores. In practice, the first method generates one sample concatenating all paths, while the second one separates each path into individual training samples.

Intuitively, the *combined paths* representation is more expressive as it could consider all the paths together and should perform better. The *individual paths* representation might generate many false associations as most of the paths are irrelevant to the target triple. Surprisingly, we found BERT is robust to those false associations taken in the training stage and is able to pick up true ones. We suspect that the individual paths representation has simpler training samples and likely most relation predictions can be achieved by one path in the existing KGC benchmarks.

Our final design takes the individual paths representation. The performance of different designs is presented in Section 5.3.7.

In order to better leverage the knowledge learned in a pre-trained language model, we adopt natural question patterns [92]. Take Figure 5.1 as an example. It could be "[CLS] Question: Franklin Roosevelt work at what ? Is the correct answer Washington D.C. ? [SEP] Context: Franklin Roosevelt president of USA; Washington D.C. capital of USA;" Each individual path will form a training/inference instance.

Step 2: BERT Scoring. In BertRL, since we take individual paths as a linearization approach, each pair of triple and reasoning path is scored individually. For each target triple, one or a few reasoning paths would indicate the truth of the triple. This forms a multi-instance learning problem [93], where predictions need to be aggregated for a bag of instances. We take a simplified realization - training individually and applying maximum aggregation of bag scoring at inference time.

BertRL uses a linear layer on top of [CLS] to score the triple's correctness, which can be regarded as a binary classification problem. It models the probability of label y ($y \in \{0, 1\}$) given the text of triple (h, r, t) and the text of reasoning path $h \to t$,

$$p(y|h, r, t, h \to t). \tag{5.3}$$

At inference time, the final score of a target triple (h, r, t) is the maximum of the positive class scores over all of its reasoning paths:

$$score(h, r, t) = \max_{\rho} p(y = 1|h, r, t, h \xrightarrow{\rho} t). \tag{5.4}$$

The path corresponding to the maximum score can be used to explain how the prediction is derived. We leave a more sophisticated aggregation function for future study.

5.2.2 Training Regime

In order to train BertRL, both positive and negative examples are needed. We follow the standard practice to view existing triples in KG as positive. Then, for each positive triple, we do *negative sampling* to sample m triples corrupting its head or tail. Specifically, we randomly sample entities from common k-hop neighbors of head and tail entities, and make sure negative triples are not in KG. We do not include empty reasoning path examples in training, and always give a minimum confidence score for empty path in inference.

When constructing reasoning paths for a triple, we hide the triple in KG and find other paths to simulate missing link prediction. As the maximum length of the reasoning paths increases, the number of paths may grow exponentially. Many paths are spurious and not truly useful for inducing the triple. We do *path sampling* at training time to get at most n paths between target entities and take shorter paths first.

Finally we use cross entropy loss to train our model:

$$\mathcal{L} = -\sum_{\tau} (y_\tau \log p_\tau + (1 - y_\tau) \log(1 - p_\tau)), \tag{5.5}$$

where $y_\tau \in \{0, 1\}$ indicates negative or positive label, and $\tau \in \mathbb{D}^+ \cup \mathbb{D}^-$. The negative triple set $\mathbb{D}^-$ is generated by previously mentioned method that corrupts head h or tail entity t in a positive triple $(h, r, t) \in \mathbb{D}^+$ with a sampled entity h' or t', i.e.,

$$\mathbb{D}^- = \{(h', r, t) \notin \mathbb{D}^+ \cup (h, r, t') \notin \mathbb{D}^+\}. \tag{5.6}$$

5.3 Experiments

We evaluate our method on three benchmark datasets: WN18RR [88], FB15k-237 [75], and NELL-995 [20], using their inductive and transductive subsets introduced by [89] [1]. WN18RR is a subset of WordNet, a KG contains lexical relations between words. FB15k-237 is a subset of Freebase, a large KG of real-world facts. NELL-995 is a dataset constructed from high-confidence facts of NELL, a system constantly extracting facts from the web. The statistics of these datasets are given in Table 5.3; the details of the variants will be given later.

Through experiments, we would like to answer the following questions about BertRL: (1) How does it generalize to relation prediction with unseen entities in the inductive setting? (2) How does it perform in the traditional transductive setting? (3) Does it work well in few-shot learning? (4) Does it have the potential

[1]https://github.com/kkteru/grail

	split	#relations	#nodes	#links
WN18RR	train	9	2,746	6,670
	ind-test	8	922	1,991
	train-1000	9	1,362	1,001
	train-2000	9	1,970	2,002
FB15k-237	train	180	1,594	5,223
	ind-test	142	1,093	2,404
	train-1000	180	923	1,027
	train-2000	180	1,280	2,008
	train-rel50	50	1,310	3,283
	train-rel100	100	1,499	3,895
NELL-995	train	88	2,564	10,063
	ind-test	79	2,086	5,521
	train-1000	88	893	1,020
	train-2000	88	1,346	2,011

Table 5.2: Statistics of the three datasets and their variants.

to generalize to unseen relations? (5) How its reasoning path explains prediction? (6) What is the training and inference time? (7) How important is the knowledge linearization design?

	WN18RR			FB15k-237			NELL-995		
	1,000	2,000	6,678 (full)	1,000	2,000	5,223 (full)	1,000	2,000	10,063 (full)
RuleN	0.649	0.737	0.745	0.207	0.344	0.415	0.282	0.418	0.638
GraIL	0.516	**0.769**	**0.769**	0.273	0.351	0.390	0.295	0.298	0.554
KG-BERT	0.364	0.404	0.436	0.288	0.317	0.341	0.236	0.236	0.244
BERTRL	**0.713**	0.731	0.755	**0.441**	**0.493**	**0.541**	**0.622**	**0.628**	**0.715**

Table 5.3: Inductive results (Hits@1)

	WN18RR			FB15k-237			NELL-995		
	1,000	2,000	6,678 (full)	1,000	2,000	5,223 (full)	1,000	2,000	10,063 (full)
RuleN	0.681	0.773	0.780	0.236	0.383	0.462	0.334	0.495	0.710
GraIL	0.652	**0.799**	**0.799**	0.380	0.432	0.469	0.458	0.462	0.675
KG-BERT	0.471	0.525	0.547	0.431	0.460	0.500	0.406	0.406	0.419
BERTRL	**0.765**	0.777	0.792	**0.526**	**0.565**	**0.605**	**0.736**	**0.744**	**0.808**

Table 5.4: Inductive results (MRR)

Baselines and Implementation Details. We compare BERTRL with the state-of-the-art inductive relation prediction methods GraIL [89] and RuleN [17]. GraIL uses graph neural network to reason over local subgraph structures. RuleN explicitly derives path-based rules and shows high precision. We use the public implementation provided by the authors and adopt the best hyper-parameter settings in their work. Differentiable logical rule learning methods like NeurLP [18] and DRUM [19] are not included, as their performance is not as good as GraIL and RuleN [89]. For the transductive setting, we pick one of the state-of-the-art embedding methods, TuckER [13] and path-based method MINERVA [94], as representatives for evaluation. For TuckER, we use implementation in LibKGE [95] with the provided best configuration in the library. For MINERVA, we use the official implementation and best configuration provided by authors.

We also compare against a BERT-based KGC method KG-BERT [91], where only relation triple (h, r, t) is fed to BERT. This is a special case of BertRL with an empty reasoning path. In our experiments, we do not feed additional description other than entity and relation names as KG-BERT [91] did. The descriptions such as Wikipedia and WordNet synsets definitions often directly contain the missing entities, making knowledge graph completion leans to relation extraction rather than graph reasoning. Therefore, in this work, we keep a relatively pure knowledge graph setting as many graph embedding algorithms do. The extra requirement of names are readily available in most scenarios. In practice, both BertRL and KG-BERT can be extended to accept additional information as this is what BERT is designed for.

Both BertRL and KG-BERT were implemented in PyTorch using Huggingface Transformers library. We employ BERT base model (cased) with 12 layers and 110M parameters and run experiments with a GTX 1080 Ti GPU with 11GB RAM. We use a batch size of 32 and fine-tune models BertRLand KG-BERT using Adam optimizer. Based on validation set performance, the best learning rate is selected from 1e-5 to 1e-4, and training epoch is gradually increased until performance is not improved. Learning rate 5e-5 is set for BertRL and 2e-5 for KG-BERT, and training epoch is 2 and 5 respectively. We sample 10 negative triples in negative sampling, and 3 reasoning paths in path sampling, and keep increasing the size does not improve performance. We did not observe improvement on negative sampling relation as well. We set other hyperparameters to package default values.

Evaluation Task. Following GraIL [89], our default evaluation task is to predict $(h, r, ?)$ and $(?, r, t)$: specifically, ranking each test triple among 50 other negative candidates. The negative triples are not in KG and generated by randomly replacing head (or tail) entity of each test triple. The sampling is going to speed up the evaluation process. The performance will be lower if the ranking is done among the full entity set.

Metrics. We evaluate models on Hits@1 and Mean Reciprocal Rank (MRR). Hits@1 measures the percentage of cases in which positive triple appears as the top 1 ranked triple, while MRR takes the average of the reciprocal rank for positive triples.

5.3.1 Inductive Relation Prediction

We first evaluate the model's ability to generalize to unseen entities. In a fully inductive setting, the entities seen in training and testing are completely disjoint. For all the methods, we extract paths from the target head entity to the tail entity with length up to 3 or the subgraph containing these paths.

Datasets. We conduct our experiment using the inductive subsets of WN18RR, FB15k-237, and NELL-995 introduced by [89]. Each subset consists of a pair of graphs *train-graph* and *ind-test-graph*. The former is used for training, and the latter provides an incomplete graph for relation prediction. *train-graph* contains all the relations present in *ind-test-graph*. However, their entity sets do not overlap. In GraIL, WN18RR, FB15k-237, and NELL-995 each induces four random inductive subsets (v1, v2, v3 and v4). We pick one subset for each (WN18RR v1, FB15k-237 v1 and NELL-995 v2). For each inductive dataset, we did stratified sampling on *train-graph* to create few-shot variants. The links are down-sampled to a number around 1,000 and 2,000, while keeping an unchanged proportion of triples for each relation. The few-shot training graph *train-1000* and *train-2000* contain all relations in its full setting, thus covering the relations in *test-graph* as well. The statistics of these variants are shown in Table 5.3.

	Transductive			**Transductive (Few-shot)**					
	WN18RR	FB15k-237	NELL-995	WN18RR		FB15k-237		NELL-995	
	6,670	5,223	10,063	1,000	2,000	1,000	2,000	1,000	2,000
RuleN	0.646	0.603	0.636	0.548	0.605	0.374	0.508	0.365	0.501
GraIL	0.644	0.494	0.615	0.489	0.633	0.267	0.352	0.198	0.342
MINERVA	0.632	0.534	0.553	0.106	0.248	0.170	0.324	0.152	0.284
TuckER	0.600	0.615	**0.729**	0.230	0.415	0.407	0.529	0.392	0.520
BERTRL	**0.655**	**0.620**	0.686	**0.621**	**0.637**	**0.517**	**0.583**	**0.526**	**0.582**

Table 5.5: Transductive results (Hits@1)

	Transductive			**Transductive (Few-shot)**					
	WN18RR	FB15k-237	NELL-995	WN18RR		FB15k-237		NELL-995	
	6,670	5,223	10,063	1,000	2,000	1,000	2,000	1,000	2,000
RuleN	0.669	0.674	0.736	0.567	0.625	0.434	0.577	0.453	0.609
GraIL	0.676	0.597	0.727	0.588	0.673	0.375	0.453	0.292	0.436
MINERVA	0.656	0.572	0.592	0.125	0.268	0.198	0.364	0.182	0.322
TuckER	0.646	0.682	**0.800**	0.258	0.448	0.457	0.601	0.436	0.577
BERTRL	**0.683**	**0.695**	0.781	**0.662**	**0.673**	**0.618**	**0.667**	**0.648**	**0.693**

Table 5.6: Transductive results (MRR)

Results. BertRL significantly outperforms the baselines in most settings as shown in Tables 5.3 and 5.3, particularly by around 10 absolute Hits@1 and MRR points in FB15k-237 and NELL-995. These two KGs have more relations and are associated with open-world knowledge (learned by BERT) compared with WN18RR. Methods like GraIL and RuleN are not able to incorporate such prior knowledge.

In the few-shot setting, BertRL stays robust and outperforms the baselines by an even larger margin. When more links are dropped in training graph, BertRL achieves more performance gain over the baselines. BertRL enjoys all sources of knowledge: structural (reasoning paths), textual (embedding), and prior knowledge (pre-trained language model). They all play an important role in knowledge graph completion.

In both settings, BertRL performs better than KG-BERT, the version without reasoning paths inputted. It shows that incorporating paths allows pre-trained language models to gain explicit reasoning capability. On the other hand, with the triple information alone, KG-BERT is able to make a certain amount of correct inferences, suggesting that prior knowledge stored in pre-trained language models can be leveraged to do knowledge graph completion as manifested in [91]. BertRL combines explicit reasoning capability, prior knowledge, and language understanding all together in one model and has significant advantages.

5.3.2 Transductive Relation Prediction

BertRL can also be applied in the transductive setting and be compared with the baselines.

Datasets. To evaluate the transductive performance, we train these models on *train-graph* introduced in the inductive setting and test on links with the same set of entities. We use a list of test triples with 10% size of *train-graph*. In a few-shot setting, we reuse the few-shot *train-graph* used in the inductive setting and tested on the aforementioned test links. At testing time, full *train-graph* is used to collect knowledge around target entities (otherwise, the setting will be close to the inductive one). The few-shot setting makes datasets partially transductive, as some entities become unseen when links are dropped randomly. For TuckER and MINERVA, we assign a minimum score for both positive and negative triples containing unseen entities.

Results. Tables 5.3.1 and 5.3.1 show that BertRL outperforms the baselines in most of full and few-shot settings. It performs competitively with TuckER in the full setting and surpasses RuleN and GraIL. It implies that BertRL's strong performance is not limited to inductive learning. In the few-shot setting, *train-graph* becomes sparse and unseen entities appear in testing. BertRL again largely outperforms all the methods, which once more demonstrates the advantage of simultaneously exploiting all knowledge sources.

Unseen relation	Hits@1	Similar seen relation
/film/film_format	1.000	/film/genre, /film/language
/person/spouse_s./marriage/spouse	1.000	/person/spouse_s./marriage/type_of_union
/pro_athlete/teams./sports_team_roster/team	1.000	/football_player/current_team./sports_team_roster/team
/artist/origin	0.000	-
/record_label/artist	0.100	-
/ethnicity/languages_spoken	0.250	/person/languages

Table 5.7: Examples of the best and worst performing unseen relation prediction of BertRL, trained on a 50 relations subset of FB15k-237.

5.3.3 Unseen Relation Prediction

As BertRL leverages a pre-trained language model, it has the potential to predict unseen relations in a zero-shot setting, which is not possible for traditional inductive learning methods like RuleN and GraIL. In this section, we examine how BertRL can generalize for unseen relations.

Datasets. We create down-sampled training datasets from full FB15k-237 *train-graph*, and test on *ind-test-graph*. Specifically, we sample 50 and 100 relations weighted by their proportion in *train-graph*. Triples with sampled relations are collected to create graphs *train-rel50* and *train-rel100*. Each relation in FB15k-237 has a multi-level hierarchy, e.g., */film/language*. Words are shared in different relations, which helps the generalization to unseen relations.

Results. Table 5.3.3 shows Hits@1 results. Both KG-BERT and BertRL make some correct predictions even without seeing the relations in training. Furthermore, Table 5.7 shows the best and worst performed unseen relation prediction on *train-rel50*. A detailed explanation of each relation is shown in Appendix B.

	50 relations	100 relations
KG-BERT	0.266	0.450
BERTRL	0.485	0.500

Table 5.8: Unseen relation prediction results (Hits@1)

For each unseen relation, we manually identified a relevant relation in training set. The best performing relations often have close meaning counterparts in training, while the worst performing relations do not. This phenomenon indicates that in zero-shot setting, BertRL generalizes to unseen relations through similar text. We suspect that knowledge captured by pre-trained language models also helps zero-shot learning.

5.3.4 Training and Inference Time

We investigate training and inference time, using the transductive setting of FB15k-237 as an example. Figure 5.2 shows the running time of BertRL compared with other methods using their default packages without further optimization. The running time is highly implementation and device dependent, however, the curves still show a trend and gives a rough scale of it. The training time of BertRL gradually increases as the number of training triples grows. The inference time of BertRL does not depend on the training data size and is slower than RuleN. Running time is one important factor in practice, and we leave how to speed up BertRL to future work.

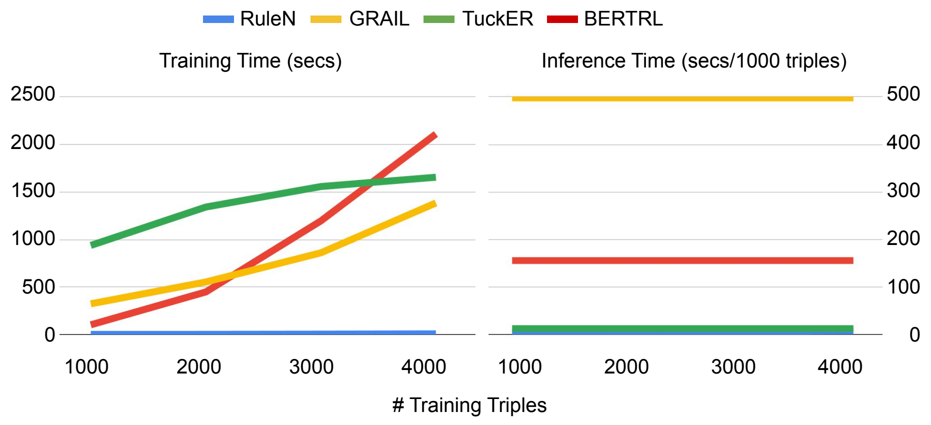


Figure 5.2: Training and inference time with respect to number of training triples. Inference time of TuckER is slightly higher than RuleN and the curves are overlapped.

5.3.5 Explainability

As stated in Section 5.1, rules like (5.1) are explainable to humans. BertRL achieves certain explainability by leveraging reasoning paths and implicitly memorizes these rules through training. For a prediction task $(h, r, ?)$, BertRL is going to generate many instances for different tail entity t by concatenating triple (h, r, t) with each path $h \to t$. Those with the highest scores are chosen as the answer. We can regard the path chain as the explanation of deriving (h, r, t). We conduct manual case study using FB15k-237 dataset as an example. The relations are in the form of multi-level hierarchy. In case studies, we remove the first level of each relation or rephrase them as their simplified version. Table 5.3.5 shows used relations and their corresponding explanations.

The following KG completion query *(Chris, acts_in_film, ?)* is to find what film the actor Chris acts in. The instance ranked highest by BertRL consists of target triple *(Chris, acts_in_film, Jackie Brown)*, reasoning path *(Chris, nominated_for_same_award_with, Robert); (Robert, acts_in_film, Jackie Brown);* and an assigned score 0.95. It could be naturally explained as follows: Chris likely acts in film Jackie Brown, since Robert shares the same award nomination with Chris and also acts in Jackie Brown.

We then examined the percentage of the explanations that are meaningful to humans. We randomly sampled 100 test triples from FB15k-237 and ask human annotators to check their top-1 path chains highly scored by BertRL. Human judges found that 84% of the path chains make sense, indicating strong explainability.

Relation	Explanation
/actor/film./performance/film	acts in film
/award_nominee/award_nominations-./award_nomination/award_nominee	nominated for same award
/person/spouse_s./marriage/spouse	spouse
/person/spouse_s./marriage/type_of_union	marriage type
/film/film_format	film format
/film/genre	film genre
/film/language	film language
/pro_athlete/teams./sports_team_roster/team	team
/football_player/current_team-./sports_team_roster/team	football player team
/artist/origin	music artist origin
/record_label/artist	music record company artist
/person/languages	person language
/ethnicity/languages_spoken	ethnicity language

Table 5.9: Relation Explanation in Used Examples.

5.3.6 Training and Inference Time

The training time of BertRL gradually increases as the number of training triples grows. The inference time of BertRL relates to *test-graph* rather than training data size, but it is slower than RuleN. We visualizes effect of size factor for BertRL and baselines in Appendix A. In practice, the running time also relates to implementation and device. We leave how to speed up BertRL to future work.

5.3.7 Ablation study

Table 5.3.7 shows the effect of different design choices in BertRL, mainly knowledge linearization and path sampling. We use the FB15k-237 inductive dataset and its few-shot subset for evaluation.

Combined Paths. As discussed in Section 5.2.1, *combined paths* is one way linearizing structural knowledge. Although it includes more information in one input, it does not outperform *individual paths*. This indicates that BERT struggles to learn from complex input when training data is limited, which might be explained by Occam's razor.

Subgraph (Edge List). Edge list is the worst performing linearization option. Linking entities in the input and then recognizing patterns could be more challenging for BERT than reasoning along paths where edges are ordered by their connection.

Path Sampling. We evaluate the performance of *path sampling* by randomly selecting n paths between entities. Path sampling could speed up training as the training data becomes small. The performance is still good even when the number of sampled paths is very small, indicating BertRL is robust to the size of the training set.

	1,000	2,000	full
Subgraph (edge list)	0.361	0.398	0.463
Combined paths	0.351	0.461	0.505
5 sampled individual paths	0.466	0.490	0.532
10 sampled individual paths	0.449	0.505	0.500
BERTRL (individual paths)	0.441	0.493	0.541

Table 5.10: Ablation study of BERTRL variants (Hits@1)

5.4 Related Work

Transductive Models. Most existing knowledge graph completion methods are embedding based, such as TransE [9], Complex [10], ConvE [88], RotatE [12] and TuckER [13]. These methods learn low-dimensional embedding of entities and relations to capture relational information of the graph. They are naturally transductive and need expensive retraining for new nodes in inductive setting.

Some methods, e.g., R-GCN [96], DeepPath [20], MINERVA [94] and DIVA [97], learn to aggregate information from local subgraph and paths. However, they cannot be directly applied to the inductive setting as entity/node specific embeddings are needed.

Inductive Models. In contrast to the transductive setting, probabilistic rule learning AMIE [16] and RuleN [17] could apply learned rules to unseen entities. NeuralLP [18] and DRUM [19] learns differentiable rules in an end-to-end manner. GRAIL [89] extracts subgraph connecting target entities and learns a general graph neural network to score a prediction. These methods are in nature inductive as they learn entity irrelevant rules or models and conduct reasoning with knowledge graph information only.

Besides these studies, there are methods learning to generate inductive embedding for unseen nodes. [98] and [99] rely on the node features which may not be easily acquired in many KGs. [100] and [101] generate embedding for unseen nodes by learning to aggregate embedding from neighbors using GNNs. [102] proposes to estimate embedding under translational assumption. However, those paradigms require a certain number of known entities and cannot be applied to entirely new graphs.

Pre-trained Language Models. Pre-trained language models, such as BERT [90], GPT-2 [103] and GPT-3 [104] revolutionize recent natural language processing studies. [105] introduces LAMA benchmark and shows that pre-trained language models themselves already capture some factual knowledge even without fine-tuning.

KG-BERT [91] aims to leverage the power of pre-trained language model in knowledge graph completion, where it represents triples as text sequences and uses BERT to learn scoring function for relation prediction. Though it can be applied in the inductive setting, its prediction is mainly based on the pre-trained representation of entities and relations; it does not learn a general reasoning mechanism like GRAIL and BertRL.

5.5 Conclusion

We proposed BertRL, a pre-trained language model based approach for knowledge graph completion. By taking reasoning path and triple as input to a pre-trained language model, BertRL naturally handles unseen entities and gains the capability of relational reasoning. In few-shot learning, it outperforms competitive baselines by an even larger margin. It has the potential to generalize to unseen relations in a zero-shot setting. It not only achieves the state-of-the-art results in inductive learning, but is also shown to be effective in transductive learning.

Overall, this work opens a new direction of combining the power of pre-trained language models and logic reasoning. This could simultaneously assist in explaining the completion results via reasoning paths and reduces the need for human supervision via a pre-trained language model.

Chapter 6

Commonsense KB Completion with Pre-trained LM

6.1 Introduction

Previous chapter 5 and other studies [9, 10, 88, 12, 13, 17], target completing general knowledge bases by inferring missing facts from existing ones. However, most conventional knowledge bases (e.g. FB15k-237, WN18NN) explored by those approaches are normalized and dense. Abundant information is expressed in the graph structure, while textual information of graph entities and relations is secondary. A model without the capability of understanding the text will not have a significantly decreased performance.

However, there are often real-world knowledge bases with sparse and unnormalized entities. Traditional embedding-based methods face more direct challenges, as many related entities are not neighboring in the graph. It is often the case that one entity is expressed in two different terms without normalization. Pure graph-based embedding approaches struggle to exploit such information. For these scenarios, practical knowledge base completion approaches should have an internal mechanism of connecting the dots in the knowledge graph with not just structural but textual information.

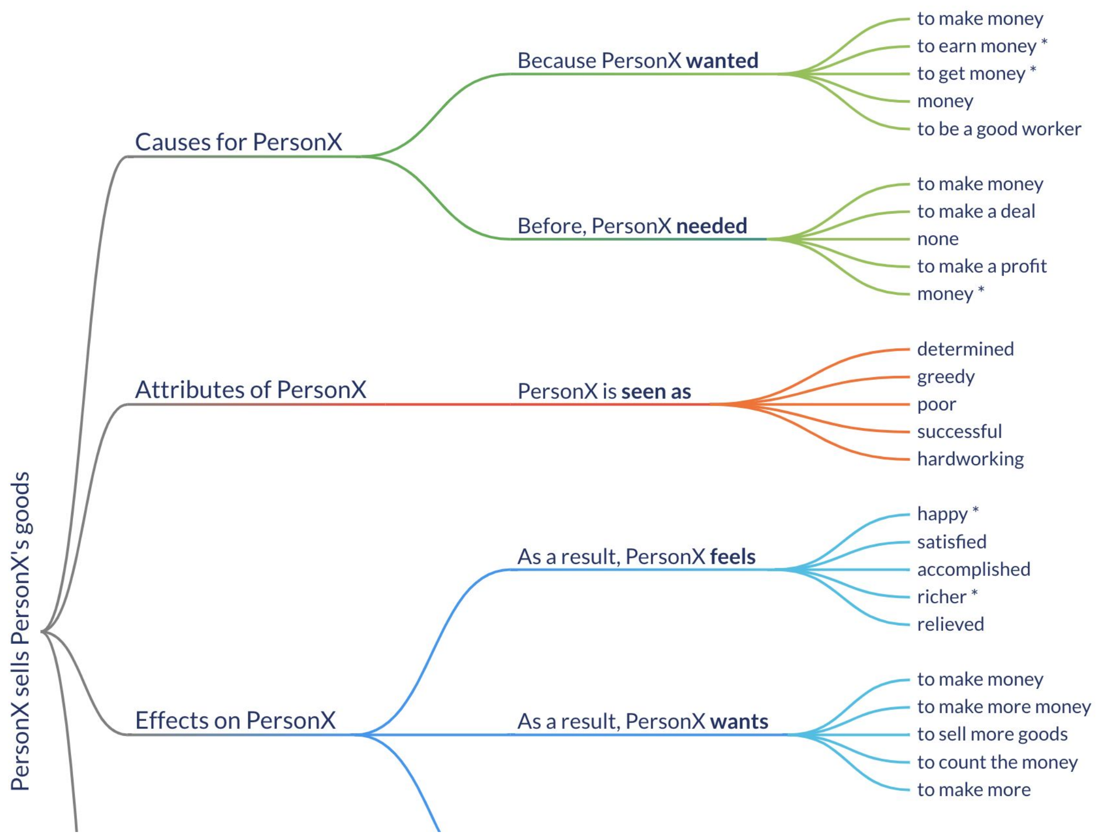


Figure 6.1: A partial example from commonsense knowledge base dataset ATOMIC [1].

Commonsense knowledge bases like ATOMIC [106] and ConceptNet [107] are typical representative of unnormalized and sparse datasets. They are crowd-sourced datasets either store if-then rules or semantic relationship between words and concepts. In contrast to commonly used knowledge base datasets FB15k-237 and WN18RR, those two datasets are sparser and unnormalized.

[1]https://mosaickg.apps.allenai.org/

To tackle the textual variation in the commonsense KB, recent approaches introduce pre-trained language models. There are two paradigms: The first paradigm directly applies pre-trained language for knowledge base completion. For example, COMET [108] directly applies GPT-2 [103] to input head and relation of the knowledge triple and predict the tail entities, with few additional specific tokens. It generates "novel" entities that usually not existing in the knowledge base. It is argued that such kinds of completion violate the traditional knowledge base completion setting where completed entities must exist in KB. Other knowledge base completion methods include KG-BERT [91] and our proposed BertRL in the previous chapter. These approaches face the efficiency problem. For a large knowledge graph, they need an additional retrieval model to rank full entities. This is due to their reliance on feeding complete target triple to BERT, where every candidate triples need to be feed into BERT for scoring. The other paradigm leverage the pre-trained language model as initialization for embedding-based methods. Then they reuse the embedding-based methods for the knowledge base completion. For example, [14] takes ConvTransE [14], one of the mainstream embedding-based approaches for knowledge base completion and initialize it with BERT [50] embedding. Furthermore, [109] and [110] build Graph Neural Networks (GNN) on top of the initialized embedding and densify the graph through similarity in the pre-trained language models.

However, the approaches of initializing using BERT embedding and fine-tunes embedding-based or Graph Neural Network models naturally break the learning into separate steps. In this chapter, we argue that a simpler method should be considered before diverge into more complicated embedding-based or GNN approaches. We propose to use a simple decoding strategy in a pre-trained generative model – simply applying a tire (prefix) tree to limit the decoding to existing entities. This largely outperforms the existing sophisticated approaches and reveals problems of the current evaluation of commonsense knowledge base completion datasets. More careful evaluation should be designed to evaluate the reasoning performance of different models.

6.2 Generative KB Completion with Constrained Decoding

6.2.1 Problem Formulation

Knowledge graph $\mathcal{G}$ is represented as a set of triples $\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are entity and relation set. In this chapter, given an incomplete knowledge graph G, we target on knowledge graph completion task to predict missing tail entities given the query relation $(h, r, ?)$ or predict missing head entities given the query relation $(?, r, t)$. The goal is to rank the true entity higher than other candidate entities.

6.2.2 Pre-trained Generative Model

Knowledge base completion can be realized through the sequence to sequence models by feed the query head entity and relation to the input and generate the missing tail entities. We use BART [111], a denoising autoencoder for pre-training sequence-to-sequence models and fine-tunes it on the commonsense knowledge base completion datasets. Specially, we use special markers to wrap the head and tail entity and mark the beginning of the generation. For example, we feed "[cat] Has A Tail " as input and "GEN eye" as ground truth generation. Then the model generates the tail entity.

6.2.3 Prefix Tree Decoding

Recent work [112] successfully applies sequence to sequence model in the setting of entity retrieval and entity linking. The decoding can be limited to a small scope of candidates rather than the whole space of natural language.

As shown in Figure 6.2, the trie tree is a type of tree structure for locating specific keys in a set. It deterministically stores the key prefixes, where searching processing walks along the tree path through prefix matching. By building a trie tree of candidate entities in the KB, the beam search can only choose the next tokens limited in the tree. Invalid beams are discarded whenever no valid branches can be selected from the current trie tree in decoding.

This trie tree decoding is naturally applicable to knowledge base completion. At inference time, only allowed entities can be generated. It addresses the "novel" entities problem of pre-trained language models and keeps the pre-trained information without separately using multiple steps of initialization and graph embedding training. In training, an unaltered negative log-likelihood objective is applied in BART by feeding the input and label.

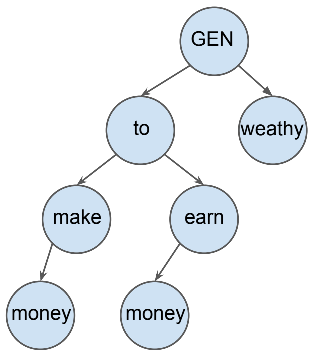


Figure 6.2: An toy trie tree example of vocabulary set {"to make money", "wealthy", "to earn money"}. A special staring token is used for every term in the dictionary.

6.3 Experiments

6.3.1 Datasets

Most existing datasets, e.g., FB15k-237 and WN18RR, for knowledge base completion are dense and normalized. Structural information plays a crucial role, while textual information is secondary. Commonsense knowledge bases such as ConceptNet and ATOMIC are sparse and unnormalized, which leads to challenges in handling textual variations and passing information in the graph through text. In our experiment, we use ConceptNet commonsense knowledge base subsets CN-100k, introduced by [113]. It samples 100k Open Mind Common Sense (OMCS) entries in the ConceptNet 5 dataset as training data. Most 1200 and the next 1200 confident triples are selected for test and validation. We also use ATOMIC, a dataset includes everyday commonsense knowledge entities in form of *if-then* relations [106].

6.3.2 Baselines

We compare our approach with state-of-the-art baselines in various categories.

Direct Pre-trained LM: COMET [108] directly applies GPT-2 to input head and relation of the knowledge triple and predict the tail entities.

Embedding: embedding-based methods including DistMult, ComplEx, ConvE, and RotatE are trained from scratch where no textual information from pre-trained language models is provided.

Embedding + LM + GNN: Graph embeddings are initialized by pre-trained LM embedding. And graph neural networks are built on top of the initial embedding layer. Certain ways of graph densification are applied to reduce the sparsity of the graph. They rely on entity similarity from pre-trained language models, either using heuristics [109] or further learning [110].

6.3.3 Results

We evaluate models on Hits@k (k=3,10) and Mean Reciprocal Rank (MRR). Hits@k measures the percentage of cases in which positive triple appears as the top k ranked triple, while MRR takes the average of the reciprocal rank for positive triples.

Table 6.3.3 shows the comparison of different methods. It is obvious that textual information is necessary for this task, embedding without language model initialization is hard to achieve high performance. Due to the free-form generation nature of COMET, it hardly generates exact entities in the given entity set. GNN-based models achieve significantly higher performance relying on both initialized BERT embedding. Graph neural networks and graph densifier contribute to the performance.

However, our approach which is extremely simple still outperforms the SOTA methods by a large margin. No structure at all is provided to the language model, only the query itself (head entity, relation) is provided. This may suggest that what contributes to the improvement of previous approaches needs to be examined. It is questionable whether the previous GNN-based method learns a better "reasoning" ability or a better way to exploiting information from a pre-trained language model.

Dataset	Entities	Relations	Train Edges	Valid Edges	Test Edges	Avg. In-Degree	Unseen Entity %
CN-100K	78,334	34	100,000	1,200	1,200	1.31	6.7%
ATOMIC	304,388	9	610,536	87,700	87,701	2.58	37.6%
FB15K-237	14,541	237	272,115	17,535	20,466	18.76	0%

Table 6.1: Statistics of CKG datasets. Unseen Entity % indicates the percentages of unseen entities in all test entities.

Model	CN-100K			ATOMIC		
	MRR	Hits@3	Hits@10	MRR	Hits@3	Hits@10
DistMult	10.62	10.94	22.54	12.39	<u>15.18</u>	18.30
ComplEx	11.52	12.40	20.31	<u>14.24</u>	14.13	15.96
ConvE	20.88	22.91	34.02	10.07	10.29	13.37
RotatE	24.72	28.20	45.41	11.16	11.54	15.60
COMET*	6.07	2.92	21.17	4.91	2.40	21.60
Malaviya et al.	52.25	58.46	73.50	13.88	14.44	18.38
InductivE	<u>57.35</u>	<u>64.50</u>	<u>78.00</u>	14.21	14.82	<u>20.57</u>
Ours	**62.84**	**70.71**	**82.91**	**15.20**	**16.25**	**21.98**

Table 6.2: Comparison of CKG completion results on CN-100K and ATOMIC datasets. Baseline results are taken from [110]. COMET is evaluated for only predicting missing tails, the two direction results are estimated to be slightly greater than half of its values. Best performed results are bolded, and second-best results are underlined.

6.3.4 Case Study

Figure 6.3 shows a case study of our seq2seq model using ATOMIC dataset as an example. It shows rank lists of our model predictions given head entity and relation in the text form.

We can see that many of them look correct from human eyes, some of them are almost the same e.g. to exercise and exercise. Though ConceptNet has a lighter issues, but it is still there. On the other hand, those completion often have similar cases in the training sets.

This suggests that the benchmark shall be re-investigated: commonsense KB is largely unnormalized, those paraphrased entities need to be filtered in order to evaluate the true "reasoning" ability of different models. This needs to be established before designing more and more complex models.

PersonX gets candy, PersonX then ___

eats candy

smiles*

eats the candy

eats

eats it

gains weight

chews

gains candy

PersonX gains muscle, Before, PersonX needed ___

to exercise

to work out

exercise*

to eat healthy

to train

to workout

work out

eat healthy

Figure 6.3: A case study of KB completion results of our trained model on ATOMIC dataset. Each query is followed by a rank list of predicted entities. All existing facts and None entities are filtered. * means the ground truth, while others are non-oracle predictions.

6.4 Conclusion

We studies commonsense knowledge base completion, where knowledge graphs are unnormalized and sparse. By leveraging a pre-trained seq2seq model and a constrained decoding strategy using built prefix tree of KB entities, we could largely outperforms complex SOTA methods. We suggest that to evaluate the reasoning ability of commonsense KBC methods, standard evaluation should be re-examined carefully and redesigned.

Chapter 7

Conclusion

In this dissertation, we discussed the explosion of data impedes democratizing knowledge among the wider part of the population. We then argued that knowledge mining and reasoning, which extract and organize knowledge from unstructured and structured knowledge from the web, are promising ways towards realizing this goal. The main contribution of this dissertation is we propose several complementary approaches towards effort-saving knowledge mining and reasoning. It is one main obstacle that implements these approaches in practice. Our approaches contribute to saving human effort in three orthogonal dimensions: reducing the need for human supervision, easing human intelligence for digesting, and accepting answers, enhancing human-machine interaction in the loop of extraction.

The ultimate goal of research in this area is to build a knowledge mining and system that every human can easily interact with. It can continuously extract knowledge from the web and existing knowledge and grow smarter and smarter. With such a system, users can have a knowledgeable assistant, doing knowledge-intensive tasks, and users themselves can focus on high-level intelligent decision making and are free from tedious jobs. Several future research directions are as follows:

Unsupervised or Self-supervised Knowledge Extraction. Textual information nowadays is almost unlimited, where abundant knowledge is buried in the text. An unsupervised/self-supervised way of automatically extract, organize and present that knowledge to humans would efficiently exploit that information.

Knowledge Reasoning. Though large pre-trained language models are shown to be powerful in various tasks, it is still doubtful that they are capable of reasoning. Learning a model aware of the text variations and ability to learn is still far from solved. A model with this ability would access a much broader scope of knowledge and achieve higher intelligence.

Human-in-the-loop knowledge mining and reasoning. There may be a gap between the extracted knowledge and human-desired knowledge. Humans are naturally needed to be in the loop of extract to guide the effective extraction. On the other hand, users need to check the quality of the extraction, reasoning to accept/reject the results. How to efficiently and effectively doing these would pose challenges to existing researches.

Bibliography

[1] "Size of wikipedia."
`https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`.
Accessed: 2021-08-31.

[2] "Size of wikidata."
`https://www.wikidata.org/wiki/Wikidata:Statistics`. Accessed:
2021-08-31.

[3] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang, *An overview of microsoft academic service (mas) and applications*, in *Proceedings of the 24th international conference on world wide web*, pp. 243–246, 2015.

[4] M. A. Hearst, *Automatic acquisition of hyponyms from large text corpora*, in *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pp. 539–545, Association for Computational Linguistics, 1992.

[5] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, *et. al.*, *Relation classification via convolutional deep neural network.*, in *Proceedings of the International Conference on Computational Linguistics*, pp. 2335–2344, 2014.

[6] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, *Neural relation extraction with selective attention over instances*, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 2124–2133, 2016.

[7] D. Zeng, K. Liu, Y. Chen, and J. Zhao, *Distant supervision for relation extraction via piecewise convolutional neural networks.*, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 1753–1762, Association for Computational Linguistics, 2015.

[8] T. M. Mitchell, W. W. Cohen, E. R. Hruschka, P. P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, *Never-ending learning*, *Communications of the ACM* (2015).

[9] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, *Translating embeddings for modeling multi-relational data*, in *NIPS*, 2013.

[10] T. Trouillon, C. R. Dance, Éric Gaussier, J. Welbl, S. Riedel, and G. Bouchard, *Knowledge graph completion via complex tensor factorization*, *JMLR* (2017).

[11] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng, *Embedding entities and relations for learning and inference in knowledge bases*, *CoRR* (2014).

[12] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, *Rotate: Knowledge graph embedding by relational rotation in complex space*, in *ICLR*, 2019.

[13] I. Balažević, C. Allen, and T. M. Hospedales, *Tucker: Tensor factorization for knowledge graph completion*, in *Empirical Methods in Natural Language Processing*, 2019.

[14] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, *End-to-end structure-aware convolutional networks for knowledge base completion*, in *AAAI*, 2019.

[15] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, *Distant supervision for relation extraction without labeled data*, in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 1003–1011, Association for Computational Linguistics, 2009.

[16] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, *Amie: Association rule mining under incomplete evidence in ontological knowledge bases*, in *WWW '13*, 2013.

[17] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, and H. Stuckenschmidt, *Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion*, in *ISWC*, 2018.

[18] F. Yang, Z. Yang, and W. W. Cohen, *Differentiable learning of logical rules for knowledge base reasoning*, in *NIPS*, 2017.

[19] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang, *Drum: End-to-end differentiable rule mining on knowledge graphs*, in *NeurIPS*. 2019.

[20] W. Xiong, T. Hoang, and W. Y. Wang, *Deeppath: A reinforcement learning method for knowledge graph reasoning*, in *EMNLP*, 2017.

[21] N. Lao, T. Mitchell, and W. W. Cohen, *Random walk inference and learning in a large scale knowledge base*, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 529–539, Association for Computational Linguistics, 2011.

[22] M. Ware and M. Mabe, *The stm report: An overview of scientific and scholarly journal publishing*, .

[23] R. Snow, D. Jurafsky, and A. Y. Ng, *Learning syntactic patterns for automatic hypernym discovery*, in *Advances in neural information processing systems*, pp. 1297–1304, 2005.

[24] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han, *Taxogen: Constructing topical concept taxonomy by adaptive term embedding and clustering*, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[25] T. L. Griffiths, M. I. Jordan, J. B. Tenenbaum, and D. M. Blei, *Hierarchical topic models and the nested chinese restaurant process*, in *Advances in neural information processing systems*, pp. 17–24, 2004.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, 2014.

[27] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, *Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications*, in *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2017.

[28] G. Bordea, E. Lefever, and P. Buitelaar, *Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2)*, in *Proceedings of the 10th International Workshop on Semantic Evaluation*, pp. 1081–1091, 2016.

[29] X. Ma and E. Hovy, *End-to-end sequence labeling via bi-directional lstm-cnns-crf*, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 1064–1074, 2016.

[30] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, *Dbpedia spotlight: shedding light on the web of documents*, in *Proceedings of the 7th international conference on semantic systems*, pp. 1–8, ACM, 2011.

[31] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, *Automated phrase mining from massive text corpora*, *IEEE Transactions on Knowledge and Data Engineering* **30** (2018), no. 10 1825–1837.

[32] A. Panchenko, E. Ruppert, S. Faralli, S. P. Ponzetto, and C. Biemann, *Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation*, in *Proceedings of the 15th Conference of the*

European Chapter of the Association for Computational Linguistics, vol. 1, pp. 86–98, 2017.

[33] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, *Cross-sentence n-ary relation extraction with graph lstms*, *TACL* (2017).

[34] P. Verga, E. Strubell, and A. McCallum, *Simultaneously self-attending to all mentions for full-abstract biological relation extraction*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, 2018.

[35] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, *Freebase: a collaboratively created graph database for structuring human knowledge*, in *Proceedings of the ACM SIGMOD International conference on Management of data*, pp. 1247–1250, ACM, 2008.

[36] F. M. Suchanek, G. Kasneci, and G. Weikum, *Yago: a core of semantic knowledge*, in *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, ACM, 2007.

[37] W. Wu, H. Li, H. Wang, and K. Q. Zhu, *Probase: A probabilistic taxonomy for text understanding*, in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 481–492, ACM, 2012.

[38] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, *Toward an architecture for never-ending language learning.*, in *AAAI*, vol. 5, p. 3, Atlanta, 2010.

[39] Q. He, B. Chen, J. Pei, B. Qiu, P. Mitra, and L. Giles, *Detecting topic evolution in scientific literature: how can citations help?*, in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 957–966, ACM, 2009.

[40] D. Mimno, H. Wallach, and A. McCallum, *Gibbs sampling for logistic normal topic models with graph-based priors*, in *NIPS Workshop on Analyzing Graphs*, vol. 61, 2008.

[41] *we analyzed 16625 papers to figure-out where ai is headed next*, 2019. [Online; accessed Aug-2019].

[42] *Ai index*, 2019. [Online; accessed Aug-2019].

[43] S. Riedel, L. Yao, and A. McCallum, *Modeling relations and their mentions without labeled text*, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 148–163, Springer, 2010.

[44] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, *The automatic content extraction (ace) program-tasks, data, and evaluation.*, in *LREC*, vol. 2, p. 1, 2004.

[45] V. Shwartz, Y. Goldberg, and I. Dagan, *Improving hypernymy detection with an integrated path-based and distributional method*, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016.

[46] X. Ren, Z. Wu, W. He, M. Qu, C. R. Voss, H. Ji, T. F. Abdelzaher, and J. Han, *Cotype: Joint extraction of typed entities and relations with knowledge bases*, in *Proceedings of the 26th International Conference on World Wide Web*, pp. 1015–1024, International World Wide Web Conferences Steering Committee, 2017.

[47] C. Quirk and H. Poon, *Distant supervision for relation extraction beyond the sentence boundary*, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1171–1182, 2017.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[49] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, in *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[50] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, *arXiv preprint arXiv:1810.04805* (2018).

[51] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

[52] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[53] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning, *Multi-instance multi-label learning for relation extraction*, in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pp. 455–465, 2012.

[54] C. A. Clark and S. Divvala, *Looking beyond text: Extracting figures, tables and captions from computer science papers*, in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[55] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch*, .

[56] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, *arXiv preprint arXiv:1607.06450* (2016).

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, *The Journal of Machine Learning Research* **15** (2014), no. 1 1929–1958.

[58] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[59] G. Grefenstette, *Inriasac: Simple hypernym extraction methods*, in *Proceedings of the 9th International Workshop on Semantic Evaluation*, 2015.

[60] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[61] X. Liu, Y. Song, S. Liu, and H. Wang, *Automatic taxonomy construction from keywords*, KDD'12, 2012.

[62] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han, *Mining quality phrases from massive text corpora*, in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD'15)*, (Melbourne, Victoria, Australia), 2015.

[63] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, *Automated phrase mining from massive text corpora*, in *IEEE Transactions on Knowledge and Data Engineering (TKDE'18)*, 2018.

[64] J. Shen, Z. Wu, D. Lei, C. Zhang, X. Ren, M. T. Vanni, B. M. Sadler, and J. Han, *Hiexpan: Task-guided taxonomy construction by hierarchical tree expansion*, in *ACM SIGKDD Conference on Knowledge Discovery and Pattern Mining*, KDD '18, (London, UK), 2018.

[65] J. Shen, Z. Wu, D. Lei, J. Shang, X. Ren, and J. Han, *Setexpan: Corpus-based set expansion via context feature selection and rank ensemble*, in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 17*, (Skopje, Macedonia), 2017.

[66] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han, *Taxogen: Constructing topical concept taxonomy by adaptive term embedding and clustering*, in *ACM SIGKDD Conference on Knowledge Discovery and Pattern Mining*, KDD '18, (London, UK), 2018.

[67] K. Li, H. Zha, Y. Su, and X. Yan, *Unsupervised neural categorization for scientific publications*, in *Proceedings of the 2018 SIAM International Conference on Data Mining*, SDM '18, (San Diego, CA, USA), 2018.

[68] J. Pennington, R. Socher, and C. Manning, *Glove: Global vectors for word representation*, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

[69] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, *Deep contextualized word representations*, in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[70] Q. Le and T. Mikolov, *Distributed representations of sentences and documents*, in *Proceedings of the International Conference on Machine Learning*, pp. 1188–1196, 2014.

[71] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, *Skip-thought vectors*, in *Advances in neural information processing systems*, pp. 3294–3302, 2015.

[72] R. C. Bunescu and R. J. Mooney, *A shortest path dependency kernel for relation extraction*, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 724–731, Association for Computational Linguistics, 2005.

[73] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, *Classifying relations via long short term memory networks along shortest dependency paths.*, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 1785–1794, Association for Computational Linguistics, 2015.

[74] Y. Su, H. Liu, S. Yavuz, I. Gur, H. Sun, and X. Yan, *Global relation embedding for relation extraction*, in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[75] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, *Representing text for joint embedding of text and knowledge bases*, in *EMNLP*, 2015.

[76] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, *Knowledge-based weak supervision for information extraction of overlapping relations*, in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 541–550, Association for Computational Linguistics, 2011.

[77] G. Ji, K. Liu, S. He, J. Zhao, *et. al.*, *Distant supervision for relation extraction with sentence-level attention and entity descriptions.*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[78] Y. Wu, D. Bamman, and S. Russell, *Adversarial training for relation extraction*, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2017.

[79] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, *Relation extraction with matrix factorization and universal schemas*, in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2013.

[80] P. Verga, D. Belanger, E. Strubell, B. Roth, and A. McCallum, *Multilingual relation extraction using compositional universal schema*, in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2016.

[81] J. Callan, M. Hoy, C. Yoo, and L. Zhao, *Clueweb09 data set*, 2009.

[82] E. Gabrilovich, M. Ringgaard, and A. Subramanya, "FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0)." `http://lemurproject.org/clueweb09/`, 2013.

[83] D. Chen and C. D. Manning, *A fast and accurate dependency parser using neural networks*, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 740–750, Association for Computational Linguistics, 2014.

[84] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, *Embedding entities and relations for learning and inference in knowledge bases*, in *Proceedings of the International Conference on Learning Representations*, 2015.

[85] L. v. d. Maaten and G. Hinton, *Visualizing data using t-sne*, *Journal of machine learning research* **9** (2008), no. Nov 2579–2605.

[86] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, *A survey on knowledge graphs: Representation, acquisition and applications*, *arXiv preprint arXiv:2002.00388* (2020).

[87] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, *A review of relational machine learning for knowledge graphs*, *IEEE* (2016).

[88] T. Dettmers, M. Pasquale, S. Pontus, and S. Riedel, *Convolutional 2d knowledge graph embeddings*, in *AAAI*, 2018.

[89] K. K. Teru, E. Denis, and W. L. Hamilton, *Inductive relation prediction by subgraph reasoning.*, *arXiv: Learning* (2020).

[90] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, in *NAACL-HLT (1)*, 2019.

[91] L. Yao, C. Mao, and Y. Luo, *Kg-bert: Bert for knowledge graph completion*, *arXiv preprint arXiv:1909.03193* (2019).

[92] T. Schick and H. Schütze, *It's not just size that matters: Small language models are also few-shot learners*, *arXiv preprint arXiv:2009.07118* (2020).

[93] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, *Multiple instance learning: A survey of problem characteristics and applications*, *Pattern Recognition* **77** (2018) 329–353.

[94] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, *Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning*, in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[95] S. Broscheit, D. Ruffinelli, A. Kochsiek, P. Betz, and R. Gemulla, *Libkge-a knowledge graph embedding library for reproducible research*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 165–174, 2020.

[96] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, *Modeling relational data with graph convolutional networks*, in *European Semantic Web Conference*, pp. 593–607, Springer, 2018.

[97] W. Chen, W. Xiong, X. Yan, and W. Y. Wang, *Variational knowledge graph reasoning*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1823–1832, 2018.

[98] W. L. Hamilton, R. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, in *NIPS*, 2017.

[99] A. Bojchevski and S. Günnemann, *Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking*, in *ICLR*, 2018.

[100] P. Wang, J. Han, C. Li, and R. Pan, *Logic attention based neighborhood aggregation for inductive knowledge graph embedding*, in *AAAI*, 2019.

[101] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, *Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach*, in *IJCAI*, 2017.

[102] D. Dai, H. Zheng, F. Luo, P. Yang, B. Chang, and Z. Sui, *Inductively representing out-of-knowledge-graph entities by optimal estimation under translational assumptions*, *arXiv preprint arXiv:2009.12765* (2020).

[103] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, *OpenAI blog* **1** (2019), no. 8 9.

[104] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et. al.*, *Language models are few-shot learners*, *arXiv preprint arXiv:2005.14165* (2020).

[105] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, *Language models as knowledge bases?*, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, 2019.

[106] M. Sap, R. LeBras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, *Atomic: An atlas of machine commonsense for if-then reasoning*, in *AAAI*, 2019.

[107] R. Speer and C. Havasi, *Conceptnet 5: A large semantic network for relational knowledge*, in *The People's Web Meets NLP*, pp. 161–176. Springer, 2013.

[108] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, *Comet: Commonsense transformers for automatic knowledge graph construction*, in *The 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, (Florence, Italy), July, 2019.

[109] C. Malaviya, C. Bhagavatula, A. Bosselut, and Y. Choi, *Commonsense knowledge base completion with structural and semantic context*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 2925–2933, 2020.

[110] B. Wang, G. Wang, J. Huang, J. You, J. Leskovec, and C.-C. J. Kuo, *Inductive learning on commonsense knowledge graph completion*, .

[111] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, *arXiv preprint arXiv:1910.13461* (2019).

[112] N. De Cao, G. Izacard, S. Riedel, and F. Petroni, *Autoregressive entity retrieval*, *arXiv preprint arXiv:2010.00904* (2020).

[113] X. Li, A. Taheri, L. Tu, and K. Gimpel, *Commonsense knowledge base completion*, in *ACL*, pp. 1445–1455, Aug., 2016.