Contextualizing Language Understanding with Graph-based Knowledge Representations

---

A

Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Sanxing Chen

December 2020

APPROVAL SHEET

This

Thesis

is submitted in partial fulfillment of the requirements
for the degree of

Master of Science

Author: Sanxing Chen

This Thesis has been read and approved by the examing committee:

Advisor: Yangfeng Ji

Advisor:

Committee Member: Vicente Ordóñez Román

Committee Member: Yanjun Qi

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

December 2020

Abstract

Language understanding requires not only linguistic knowledge but also relies on knowledge that is external to textual symbols. A vast amount of knowledge is stored in the form of graph-structured data in many application domains. Despite a growing interest in developing knowledge-driven approaches in the community, how to build powerful representations of graph-structured knowledge and e□ectively incorporate them into language understanding models remains a challenging problem in natural language processing research.

This thesis explores the direction of contextualizing language understanding with graph-based knowledge representations. I first demonstrate the challenges of building meaningful interactions between language representations and domain-specific knowledge representations in the task of cross-domain Text-to-SQL semantic parsing. By citing this example, I point out the idea of fostering multiple connections between the two representations in their di□erent levels of abstraction and utilize the idea to substantially improve two graph neural network-based semantic parsers. To implement this idea in a more general form to benefits more language understanding tasks, I propose a new knowledge graph representation model that shares a similar Transformer architecture design with prevalent language models. In the task of factoid question answering, I show that the proposed knowledge representations can be e□ectively integrated into state-of-the-art pre-trained language models via a simple cross-modality attention mechanism.

Acknowledgments

The past one and half year has been a wonderful journey for me. I will never forget the excitement of receiving a graduate program admission for the first time from the University of Virginia. Here, I had many adventures and accumulated most of my research experience. All of these would not have been possible without the help and guidance from many people I was fortunate to meet.

First of all, I would like to thank my advisor, Yangfeng Ji, from whom I have learned so many invaluable things. He taught me how to perform research, how to convince people with an idea, and how to pursue an academic career. From academic work to personal matters, he generously supported me to the best of his abilities. Without him, it would be uncertain if I can thrive in the master's program.

I would also like to thank other members of my thesis committee, Professors Yanjun Qi and Vicente Ordonez, for devoting their time to provide me with valuable feedback and take care of the procedures to ensure my graduation.

I want to thank my collaborators at Microsoft, Xiaodong Liu and Jianfeng Gao from Microsoft Research, Jian Jiao and Bruce Zhang from Bing Ads. I am grateful to have the opportunity to intern at Microsoft and learn many things from them.

I also want to thank the entire UVA NLP group, which I have the honor to be part of. We are fellow travelers with a similar destination, and we are friends who care about each other. I want to specifically thank Aidan San and Stephanie Schoch for helping me with many of my writing. And I want to thank Hanjie Chen and Wanyu Du for companionship in a foreign country.

Finally, thanks to my dearest family—my father who spares no e□ort to support my study, my mother who cares about my well-being every day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding natural languages, which are structured systems humans developed to communicate between individuals, is considered a grand challenge to artificial intelligence research. Language embodies humans' reasoning process with various knowledge. It trade full clarity for e□ciency by keeping only essential content but ignoring shared knowledge about the context. Although some of the contextual information is still expressed in the forms of language in a larger space, lots of the valuable contexts are from other modalities. Therefore, trying to understand language by only learning from textual symbols could raise several issues.

- Some ambiguities are impossible to be resolved due to the absence of necessary contexts, e.g., in a sentence "the man saw a jaguar cross the jungle",[1] we do not know whether the "jaguar" here is a jaguar car or an animal, while this ambiguity can be resolved if a picture of the scene is presented.

- Linguistic knowledge learned from one domain is hard to generalize to other domains because of context switching, e.g., semantic parsers trained on geography domain cannot generalize to flight booking domain without explicitly capturing the domain di□erences.

- Learning general knowledge requires ine□cient training on large amounts of data, e.g., huge language models trained on billions of words can recall some

[1] This example is adapted from Raiman and Raiman (2018).

factual knowledge (Petroni et al., 2019).

As we will see through the rest of this thesis, these problems are widespread across many critical applications of natural language processing, hindering the current research progress. Moreover, from a linguistic perspective, researchers have even argued that it is theoretically impossible to learn the meaning of language by only training models on linguistic forms (Bender and Koller, 2020). We also see a growing interest in grounded language learning in recent reflections (Tamari et al., 2020).

Contextualizing language understanding with data from other modalities is not a new idea. Harnad (1990) identifies the famous "symbol grounding problem", suggesting that language understanding systems should have the innate ability to ground the meaning of language to a underlying symbol system, with the help of sensory data of human perception. It further proposes a potential solution of using neural networks to learn the "categorical representations" for sensory data which contains invariant features that are grounded to the symbol system. Early work uses statistical models to exploit the correlations between multimodal data within the same context. For example, Roy and Pentland (2002) explores lexicon acquisition by building an information-theoretic model which is proposed to find consistent structure between paired speech and image data. Mintz et al. (2009) perform distant-supervised relation extraction by aligning text to existing knowledge graphs, e.g., the Freebase (Bollacker et al., 2008). After the successful applications of neural networks in many domains, many works have enjoyed the benefits of having a consistent framework for multimodal data. Convolution neural networks have been applied to build multimodal neural language models which can be used to retrieve phrase descriptions of a given image or the other way around (Kiros et al., 2014). The Transformer architecture (Vaswani et al., 2017) has also been used to learn high quality cross-modality representations in large amounts of image and sentence pairs (Tan and Bansal, 2019), after its success in learning representations of languages (Devlin et al., 2019).

This thesis investigates ways that language can be supplied with external contexts in the form of graph-structured data. Among various forms of external context that are useful for language understanding, I choose graphs because of their wide

availability in many domains and full expressiveness in representing the relationship between entities. For many natural language processing tasks, there are external contexts that are represented by graphs. For example, the senses of words and the relations between them (e.g., "is a" relation between "cat" and "mammal") can constitute a WordNet (Miller, 1995), which is beneficial for tasks like computing word similarity and word sense disambiguation. Visual information can be essentialized to scene graphs to contextualize language understanding with visual knowledge (Yu et al., 2020). Besides natural language processing, much early work has been put into storing general-purpose world knowledge by constructing large-scale knowledge bases represented in the form of graphs. These knowledge graphs are naturally-occurred semantics and have wide implications for a variety of applications in natural language processing such as answering factoid questions.

Specifically, this work makes a step forward in bridging language and graph-structured data. I approach this problem from both sides, formulated as the following research questions.

- Starting from languages: how to ground language understanding to external graph-structured data via the interactions between neural representations of language and graph.
- Starting from graphs: how to represent graph-structured data in a way that it can be better integrated into language understanding models, without losing its expressiveness.

Starting with the first direction, in the task of text-to-SQL semantic parsing, I attempt to map natural language questions to SQL queries that are executable against a database. I study this task in a cross-domain setting where the databases I used for training the semantic parser are not the same as those I used for testing. This is achieved by encoding the domain-specific context — the schema graph of a database to a graph representation that can be used by the parser. Focusing on the both crucial and challenging problem of selecting the correct entities (tables and columns) for the generated SQL query, I formulate two linking processes that I can

use to ground two different types of semantic units (shallow and compositional) to entities in the database schema graph via two different linking paths, at different levels of graph representation. One of the key insights here is that the linguistic units bearing shallow semantics can mostly be grounded to shallow graph representations, while those bearing compositional semantics will require structural representations of the graph. Interestingly, I find that in a widely-used cross-domain text-to-SQL parsing dataset, these two types of semantic units are well-separable with the help of partially generated SQL code segments. Also, this case study demonstrates the needs and benefits of external context in a fundamental task of natural language understanding.

Motivated by the benefits brought by proper multilevel interactions between language and knowledge representations that I previously observed, I explore the possibility of creating a new graph representation model which can foster such interactions naturally. The current graph representations are usually learnt by various graph neural networks. Such network variant is known to suffer from depth limitation, result in commonly shallow graph representations. After the success of large pre-trained language models (Devlin et al., 2019), the Transformer (Vaswani et al., 2017) has become the de facto model for many language tasks. To explain its unprecedented power, researchers have performed probing analysis and found that meaningful levels of abstraction of language occur in representations from pre-trained Transformer's different layers. For example, representations from the lower layers often encode more syntactic information while those from higher layers capture complex semantics (Tenney et al., 2019; Lin et al., 2019). Therefore, it is challenging to combine a deep hierarchical language representations with a shallow representations of graph while keeping meaningful interaction favourably. As such, taking knowledge graphs as test beds, I adapt the Transformer to learn knowledge graph representation so that the model architectures for language and knowledge graphs are homogeneous, enabling the possibility of an elegant merger using a simple cross-attention mechanism. I perform evaluations in long-standing knowledge graph completion benchmarks to show that the proposed Transformer-based graph representations (called HittER) are

more expressive in the sense of generalizing to unseen relational facts during inference, compared to a long line of work in knowledge graph representations learning.

Lastly, to evaluate whether the proposed knowledge graph representations HittER can enable meaningful interactions with deep language representations via their levels of abstraction, I combine a HittER model pre-trained on a knowledge graph with a state-of-the-art pre-trained Transformer-based language model BERT. Experiments on a factoid question answering task, pertaining to questions of facts in the knowledge graph I used for pre-training, demonstrate the validity of my hypothesis.

Chapter 2

Text-to-SQL Parsing with Schema Graphs

In this chapter, I explore how external graph-structured data can benefit language understanding, using cross-domain Text-to-SQL semantic parsing as an example.[1] Semantic parsing aims at extracting the precise meaning of language and representing it with a machine-understandable meaning representation. As a key task in natural language processing, semantic parsing has attracted lots of research attention because it focuses on the fundamental problems of extracting meaning from language and bridging the gap between human and computer. Although a variety of logic forms have been studied as targets of meaning representation over the decades, Text-to-SQL semantic parsing has nevertheless attracted a large amount of attention due to the desire of natural language interfaces to database (NLIDB) in both the academia and industry. As a Text-to-SQL parser translates user query utterances to SQL code that is executable against a database, lots of existing work focuses on learning a specific database (e.g., geographical database), treating it as a latent world state. This learning paradigm ignores the valuable information stored in the database and

[1] Material in this chapter is adapted from:

- Sanxing Chen, Aidan San, Xiaodong Liu and Yangfeng Ji. A Tale of Two Linkings: Dynamically Gating between Schema Linking and Structural Linking for Text-to-SQL Parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

leads to semantic parsers that are unable to generalize to other databases. Recent work on cross-domain Text-to-SQL semantic parsing tries to fix this issue by proposing a new learning paradigm where a semantic parser is given the database as input and will be evaluated in databases that are not seen during training. Under this setting, the parser receives inputs of two modalities, i.e., text of user query and the database depicted by its schema graphs. Although recent work has explored ways to build representations for both inputs, it is unclear how to capture meaningful interactions between these two representations.

Given our task definition, the output meaning representation (i.e., SQL code) is meant to capture the precise semantics of question utterances and also compliant with the rules defined in the database. This unique role makes it an ideal intermediate representation for us to bridge the gap between the two representations of text and graph. Moreover, previous work has shown that current Text-to-SQL parsers have no trouble in generating SQL skeletons that are matched with the general semantics of question utterances. Instead, they struggle with selecting the correct entities (tables and columns in a database) to fill in the slots in SQL skeletons which requires deeper understanding.

Thus our goal in this chapter is to explore how meaningful interactions between text and graph representations can help us identify the correct entities for a SQL output. To accomplish this goal, I first formulate two linking processes: 1) schema linking, which identifies entities that are explicitly mentioned in question utterances by matching with entity names in the schema graph; 2) structural linking, which identifies entities that are not directly mentioned in question utterances but related to other entities in the SQL code, according to their structural relationships in the schema graph. Next, I design a model that can distinguish between these two linking paths for each entity given its role in the SQL query.

2.1 Related Work

2.1.1 Semantic Parsing

Semantic parsing research focus on mapping natural language to formal languages like lambda calculus (Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010; Liang et al., 2011; Dong and Lapata, 2016), Prolog-style queries (Zelle and Mooney, 1996; Tang and Mooney, 2000), and more recently to SQL (Warren and Pereira, 1982; Popescu et al., 2003; Giordani and Moschitti, 2009; Zhong et al., 2017; Iyer et al., 2017). It can also tackle the problem of parsing natural language descriptions to complicated general-purpose programming language such as Python (Ling et al., 2016; Rabinovich et al., 2017; Yin and Neubig, 2017).

Text-to-SQL parsing requires strict *structured prediction* due to its application scenario where the output SQL will be sent to an executor program directly. To enhance the capacity of an auto-regressive model to capture structural information, current state-of-the-art semantic parsers usually adopt a grammar-based decoder (Xiao et al., 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017). Rather than directly generating the tokens in a traditional sequence-to-sequence manner, grammar-based decoders produce a sequence of production rules to construct an abstract syntax tree (AST) of the corresponding SQL. As the grammar constraints narrows down the search space to only grammatically valid ASTs, those parsers can usually generate well-formed SQL skeletons (Guo et al., 2019; Bogin et al., 2019a).

However, it is still difficult for current state-of-the-art models to fill in the skeletons with semantically correct entities, especially when they are required to generalize to unseen DB schemas (Yu et al., 2018; Suhr et al., 2020). To predict the correct entity, the model should have a database (DB) schema grounded understanding of the NL question, which means that the model should be able to jointly learn the semantics in the NL question and the structured knowledge in a given database. Our proposed method is tested for Text-to-SQL parsing and can be adapted to other semantic parsing applications.

2.1.2 Structural Mismatch

Programming languages like SQL express the same intent in a completely different way from natural language by design (Kate, 2008). The phenomenon called structural mismatch widely exists between language and various programming language and is a major challenge in semantic parsing (Dong, 2019). To alleviate the structural mismatch problem, early approaches rely on linguistic formalisms like parsing results from flexible CCGs (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2011, 2013). Chen et al. (2016) proposed to use sentence rewriting to revise the natural language question to a new question which has the same structure with the targeted logical form. Recently, Guo et al. (2019) proposed to first translate the natural language question to an intermediate representation (IR) designed to bridge natural language and SQL, then use a deterministic algorithm to convert the IR to SQL. In addition to taking a considerable amount of engineering effort, their designed IR is still unable to cover some SQL grammars like the self-join in the `ON` clause, and is more challenging to apply to other programming languages. We deal with this problem by explicitly modeling the prediction structure with external predefined structure (*i.e.*, DB schema) by structural linking.

2.1.3 Memory Pointer Network

Memory networks were first introduced in the context of the question answering task, where they served as a differentiable long-term knowledge base to enhance auto-regressive model's poor memory (Weston et al., 2015; Sukhbaatar et al., 2015). Copy mechanisms use attention as a pointer to select and copy items from source text, thus addressing the problem of a variable output vocabulary size (Vinyals et al., 2015; See et al., 2017). Recent research has applied memory-augmented pointer networks to various NLP tasks, including task-oriented dialogue (Wu et al., 2019) and also semantic parsing (Liang et al., 2017; Guo et al., 2019). Our dynamic gating mechanism can also be seen a memory controller, except our memory is read-only and acts as both the query and key in a pointer network. Different from most current techniques,

our pointer network does not only perform copying but can also point to a start point of structural linking.

2.2 Model

In this section, we first formulate the Text-to-SQL semantic parsing task in section 2.2.1. We will then describe the details of our proposed method in section 2.2.2.

2.2.1 Text-to-SQL Semantic Parsing

The task of Text-to-SQL semantic parsing is to predict a SQL query S based on input $(Q; G)$ where $Q = \{q_1, \ldots, q_{|Q|}\}$ is the NL question and $G = (V, E)$ is the DB schema being queried. In the schema, $V = \{(e_1, t_1), \ldots, (e_{|V|}, t_{|V|})\}$ is a set which usually contains two types of entities (*i.e.*, tables and columns[2]) and their textual descriptions (*i.e.*, table names and column names), while $E = \{(e_1^{(s)}, e_1^{(t)}, l_1), \ldots, (e_{|E|}^{(s)}, e_{|E|}^{(t)}, l_{|E|})\}$ contains the relations l between source entity $e^{(s)}$ and target entity $e^{(t)}$, *e.g.*, table-column relationships, foreign-primary key relationships,[3] etc. The output $S = \{a_1, \ldots, a_{|S|}\}$ is a sequence of decoder actions which further compose an AST of SQL.

Typical state-of-the-art Text-to-SQL parsers, consist of three components: a NL encoder, a schema encoder and a grammar decoder (Guo et al., 2019; Bogin et al., 2019a).

The NL encoder takes the NL question tokens Q as input, maps them to word embeddings $\boldsymbol{E}_Q$, then feeds them to a Bi-LSTM (Hochreiter and Schmidhuber, 1997). The hidden states of the Bi-LSTM serve as the contextual word representation of each token.

The schema encoder takes G as input and builds a relation-aware entity representation for every entity in the schema. The initial representation of an entity is a combination of its words embeddings and type information. Then self-

[2]Note that columns may have more fine-grained types like binary, numeric, string and date/time, primary/foreign etc.

[3]In SQL, a foreign key in one table is used to refer to a primary key in another table to link these two tables together for joint queries.

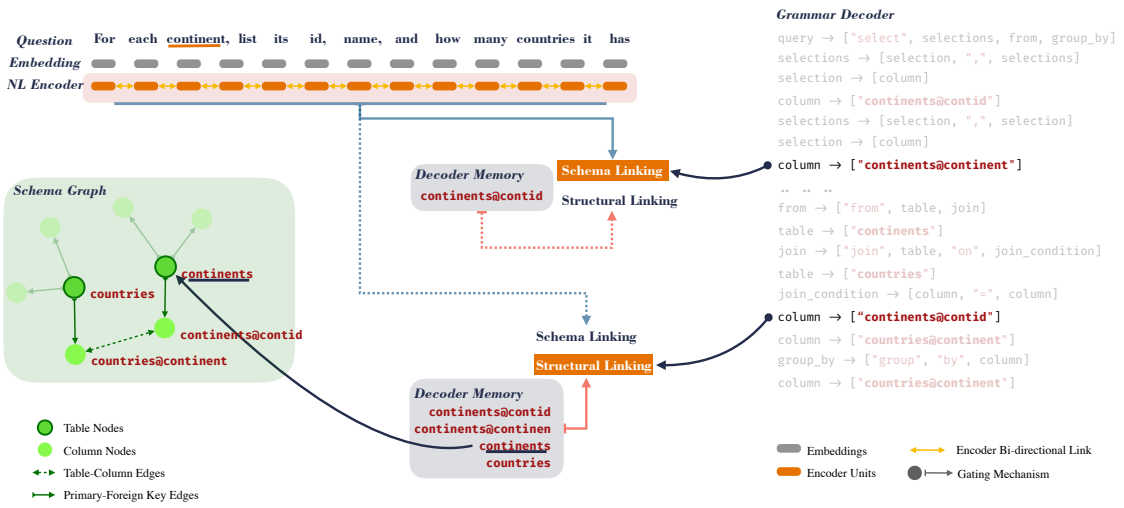

Figure 2-1: An illustration of our proposed method when running the first example shown in Table 2.1. This figure shows two independent entity generation procedures, the top one favors schema linking while the bottom one favors structural linking. Some details are omitted for the sake of simplicity. Grammars are simplified to fit in the limited space, readers are encouraged to refer to (Krishnamurthy et al., 2017) for details.

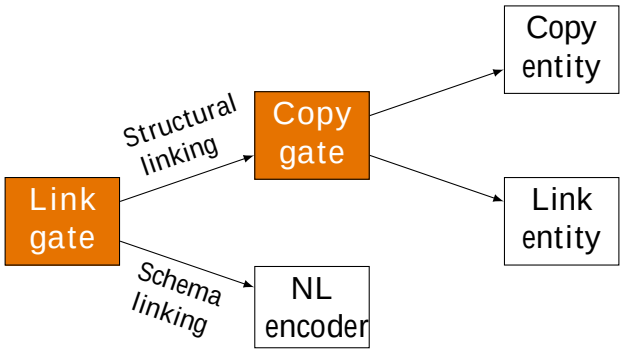

Figure 2-2: An illustration of our gating mechanism.

1	Q: *For each continent, list its id, name, and how many countries it has?* `select t1.contid, t1.continent, count(*) from continents as t1 join countries as t2 on t1.contid = t2.continent group by t1.contid;`
2	Q1: *What is the average, minimum, and maximum age of all singers from France?* Q2: *What is the average, minimum, and maximum age for all French singers?* `select avg(age), min(age), max(age) from singer where country = 'France';`
3	Q: *What is the first name and gender of the all the students who have more than one pet?* `select t1.fname, t1.sex from student as t1 join has_pet as t2 on t1.stuid = t2.stuid group by t1.stuid having count(*) > 1`

Table 2.1: Several examples that are taken from the Spider dataset. Entity mentions are underlined in NL questions. Q1 and Q2 are paraphrases of each other which should lead to the same SQL result. Wavy underline indicates the mention can only be resolved by linking to a cell value or common sense reasoning.

attention (Zhang et al., 2019a; Shaw et al., 2019) or graph-based models (Bogin et al., 2019a; Wang et al., 2020a) are utilized to exploit the relational information between each pair of entities from the DB schema, thus produce the final representation of all entities $H_V \in \mathbb{R}^{|V| \times \dim}$. We will detail this in section 2.2.3.

Finally, a grammar decoder (Xiao et al., 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017) generates an AST of output SQL in a depth-first order. The decoder is typically an auto-regressive model (*e.g.*, LSTM) which estimates the probability of generating an action sequence.

There are *two cases* of using actions. Depending a specific case, an action is either (i) producing a new production rule to unfold the leftmost non-terminal node in the AST, or (ii) generating an entity (*e.g.*, a table or a column) from the DB schema if it is required by last output production rule. In the former case, at step t, the decoder normally uses its hidden states h_t to retrieve a context vector c_t from the NL encoder. Then an action embedding a_t is produced based on the concatenation of h_t and c_t. This action embedding will directly predict a production rule from the target

vocabulary which is a subset of a fixed number of production rules. For the latter case, the decoder need to estimate a probability distribution over a schema-specific vocabulary under grammatical constraints which come from the structure of both the output SQL and the DB schema, as well as semantic constraints implied in the NL question.

2.2.2 Dynamic Gating

In this paper, we focus on the decision made when the decoder is looking for an entity to fill in a slot (*i.e.*, case (ii) in the last paragraph). Our decoder predicts the entity based on a mixed probability model consisting of two processes:

- Schema linking. The decoder attends to the output of the NL encoder (which can be seen as selecting a most relevant NL mention), then finds the corresponding entity based on string-matching or embedding-matching results.
- Structural linking. The decoder self-attends to the output states from those previous decoding steps which have generated entities, then finds another entity which is structurally linked to the attended entity.

The choice between them is controlled by a gating mechanism called the *link gate*.

Formally, the marginal probability of generating an entity e is defined as follows:

$$\Pr(a_t = e) = \Pr(e|\text{Schm})\Pr(\text{Schm}) + \Pr(e|\text{Strct})\Pr(\text{Strct}) \tag{2.1}$$

where $\Pr(\text{Schm})$ and $\Pr(\text{Strct})$ are the probability of choosing schema linking and structural linking respectively. They are further computed as:

$$_{\text{link}} = \text{Sigmoid}(\text{FF}(a_t)) \tag{2.2}$$

$$\Pr(\text{Schm}) = _{\text{link}} \tag{2.3}$$

$$\Pr(\text{Strct}) = 1 - _{\text{link}} \tag{2.4}$$

eq. (2.2) stands for our proposed link gate which is computed by a_t. The reason

for purely basing the gate value on a_t is that intuitively the choice between the two processes is about the role of the current entity we want to generate. The role of an entity is determined by the SQL clause that contains it. Since a_t is directly used to predict a production rule in case (i), it should be able to capture this information. The link gate allows the decoder to dynamically choose between information from our two linking processes, and prevents them from interfering each other.

In practice, we model the probability of the schema linking process generating an entity mentioned in the NL question, namely Pr(e_j|Schm), as a multiplication of the attention weights $\quad 2\ \mathrm{R}^{|Q|}$ over the NL encoder outputs and a schema linking matrix $M\ 2\ \mathrm{R}^{|Q|\times|V|}$. The probability of the structural linking process, namely Pr(e_j|Struct), is similarly computed by multiplying decoder self-attention weights and a structural linking matrix $T\ 2\ \mathrm{R}^{|V|\times|V|}$.

The structural linking matrix T captures the relationship between every pair of entities given the relational DB schema. Common structural links include relations between a table and its columns, a table and its primary/foreign key, a primary key and one of its linked foreign keys in other tables, etc. There are also multi-steps links which are the combinations of the one-step links listed above. Note that there may not be a unique link between every pair of entities and some entities may not have a link between them at all. Meanwhile, an entity can link to itself which can be considered to be a special zero-step structural link. It is the only structural link modeled in most current Text-to-SQL semantic parsers.

We compute the structural linking probability between an entity e_i and e_j by an additive attention mechanism (Bahdanau et al., 2015), as follows:

$$T_{i;j} = \mathrm{v}^{>} \tanh(\mathrm{W}\ \ [e_i; e_j]) \tag{2.5}$$

where e_i and e_j are the corresponding entity representations retrieved from H_V, while v and W are both trainable parameters. Compared to the dot-product attention used in Bogin et al. (2019a) and Bogin et al. (2019b), the additive attention we used here is expected to capture more of the structural relationship between entity pairs

rather than only the similarity of entity representations.

T is expected to capture all types of relationships between entities, but it can be overwhelmed by the large workload. So, we single out the zero-step relationship (*i.e.*, copying) and address it by another structural linking matrix $T_{\text{copy}} = I$, which is trivially an identity matrix in this case. To choose between copying and other types of links, a *copy gate* ($_{\text{copy}}$) is obtained in the same manner as to how we compute the link gate in eq. (2.2).

We use decoder self-attention to find the past generated entity which could have structural constraints on the entity that we currently want to generate.

$$= \text{Softmax}(\text{Attention}(a_t; H_m)) \tag{2.6}$$

$$H_m = f a_i j i < t; a_i \; 2 \; e g \tag{2.7}$$

H_m is a memory matrix consisting of the action embeddings from every past decoding step which has generated an entity. In this way, we compute two sets of attention weights $_{\text{copy}}$ and $_{\text{link}}$ for copying and linking separately using the additive attention (Bahdanau et al., 2015) again. The motivation for separate attention weights is that these two linking patterns might need to attend to di erent generated entities.

Overall the probability of generating an entity via structural linking is modeled as:

$$\Pr(a_t = e_i j \text{S t r c t}) = \; _{\text{copy}}(\; _{\text{copy}} T_{\text{copy}})_i + (1 - \; _{\text{copy}})(\; _{\text{link}} T)_i \tag{2.8}$$

Finally, this probability is mixed with the probability of schema linking controlled by the link gate.

2.2.3 Implementation

In this section, we describe how we integrate our proposed method into a grammar decoder and leverage the entity representation from a GNN module.

We use the type constrained grammar decoder from Krishnamurthy et al. (2017).

To predict a_t at time step t, the decoder will first obtain the context vector c_t from the NL encoder by performing dot-product attention (Luong et al., 2015). Then the action embedding is generated by a feed-forward network taking the concatenation of decoder hidden state and context vector as input.

$$a_t = \text{FF}([h_t; c_t]) \tag{2.9}$$

a_t is used to predict the production rule or estimate the gate values in the entity generation process.

We adopt the idea from (Bogin et al., 2019a,b) to learn a schema relation-aware entity representation H_V by a GNN module.[4] The initial embedding of each entity $h_e^{(0)}$ is defined as a non-linear transformation of the combination of its type embedding and the average over the word embeddings of its neighbors in the schema graph. In later time steps, the hidden state is updated by a gated recurrent unit (Cho et al., 2014; Li et al., 2016) as $h_e^{(l)} = \text{GRU}\left(h_e^{(l-1)}; x_e^{(l)}\right)$, where the input $x_e^{(l)}$ is defined as a weighted summation over the hidden states of its neighbor entities:

$$x_e^{(l)} = \sum_{t \in \{\$, !, \$g\}} \sum_{(s,e,l) \in E, l=t} W_t h_s^{(l-1)} + b_t$$

They consider three edge types, *i.e.*, bidirectional edges between a table and its contained columns $\$$, unidirectional edges between a foreign key and a connected primary key and its reverse version $!$. Given a fixed GNN recurrence step L, we have the final hidden states of all the entities in the graph as the entity representation $H_V = \{h_e^{(L)} | (e,t) \in V\}$. We also adopt their schema linking module to create a schema linking matrix M based on word embedding similarity and some simple manually design features (*e.g.*, editing distance and lemma). In their GlobalGNN (Bogin et al., 2019b), an additional GNN and an auxiliary training loss are added to filter out irrelevant nodes in the graph, thus producing a better entity representation.

[4] We choose these models because of the ability of GNNs to model various types of structural links, and they are among a few state-of-the-art models that are publicly available at the time of writing.

Hardness	# Example
Easy	250
Medium	440
Hard	174
Extra	170
All	1034

Table 2.2: Number of examples in the development set of Spider with different hardness levels associated with the SQL need to be generated.

To augment our model with pretrained BERT embeddings, we follow Hwang et al. (2019) and Zhang et al. (2019a) to feed the concatenation of NL question and the textual descriptions of DB entities to BERT and use the top layer hidden states of BERT as the input embeddings.

2.3 Experiments

We evaluate the effectiveness of our proposed method by integrating it into two state-of-the-art semantic parsers on the Spider dataset and further ablate out some components to understand their contributions.

2.3.1 Experiment Setup

We implement our model using PyTorch (Paszke et al., 2019a) and AllenNLP (Gardner et al., 2018). For the GNN and GlobalGNN models we revise and build upon the code released in (Bogin et al., 2019a,b). We re-ran the experiment and report the results on our re-implementation and found our results slightly improves upon their reported results. In BERT experiments, we use the base uncased BERT model with 768 hidden size provided by HuggingFace's Transformers library (Wolf et al., 2019). We follow the database split setting of Spider, where any databases that appear at testing time are ensured to be unseen at training time.

Model	Acc.	Easy	Medium	Hard	Extra
GNN	47.7%	68.8%	51.8%	31.2%	22.9%
+ Ours	50.7%	66.4%	54.8%	42.8%	25.3%
GlobalGNN	49.3%	69.2%	53.0%	32.8%	27.6%
+ Ours	52.8%	70.4%	55.7%	46.6%	25.9%
+ BERT	53.5%	76.0%	57.3%	36.2%	28.3%
+ BERT + Ours	57.6%	73.6%	61.6%	48.9%	32.9%

Table 2.3: Exact Set Matching Accuracy on SQL queries with different hardness levels in the development set of Spider. Greatest improvements in the Hard level; small fluctuation in Easy level due to gate bias.

2.3.2 Experimental Results

The experimental results in table 2.3 show that our proposed gating mechanism leads a substantial improvement on all the GNN, GlobalGNN, and BERT baselines. Spider questions are divided into different levels of difficulty (hardness). Most of the improvements come from gains in complicated (*i.e.*, Medium, Hard and Extra Hard) SQL generation. Specially, we observe up to 13.8% gains in the Hard set when applying our method on the GlobalGNN baseline. One major contribution comes from the partial matching F1 score of `IUEN` (*i.e.*, SQL clauses `INTERSECT`, `UNION`, `EXCEPT`, `NESTED` which only appear in Hard and Extra Hard levels) increasing from 25.4% to 39.7%. We also notice that the SQL output well-formedness is improved. For instance, before applying our method the decoder would occasionally select the same columns twice to perform the `ON` clause.[5] After applying our dynamic gating, this issue is virtually eliminated (error rate from 2% to 0.2%).[6]

As shown in fig. 2-3, the values of both gates are polarized to 0 or 1, thus making the gating mechanism act as a binary gate. These statistics coincide with our hypothesis that *most entity generation decisions in Text-to-SQL can be solely made by evidence from either schema linking or structural linking*. In addition, among all the cases where structural linking is chosen and the copy gate takes control, fewer than 20% of cases favor copying. This suggests that there are lots of circumstances where

[5] This is different from the case of intentionally self join.

[6] Although this issue can be resolved by engineering more grammar rules, we leave it as an indicator of the improvement of the well-formedness of the output SQL.

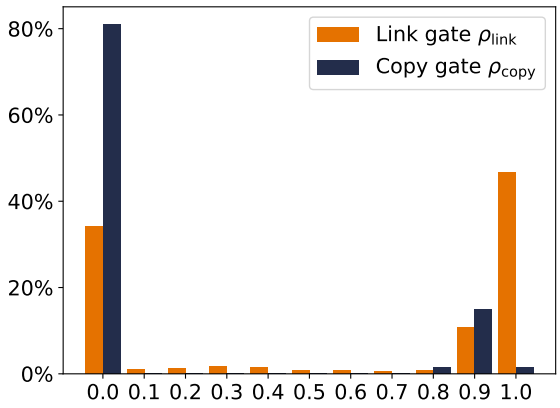

Figure 2-3: Value distributions of the link gate and copy gate measured in dev set of Spider using the GlobalGNN model. Values of copy gate are considered when the corresponding link gate value is small ($_{\text{link}}$ < 0:1).

di□erent kinds of structural linking are adopted.

2.3.3 Alternative Approaches and Ablation

We also conduct several experiments to examine several design choices in our proposed method.

Sharing Action Embedding. In the design of our gating mechanism, one critical decision is to use the action embedding a_t to perform decoder self-attention and produce the gating values. This is based on our intuition that the action embedding captures the structural information of the output SQL at the current position. To verify this decision, we conduct an ablation experiment by using a *dedicated embedding* to produce the gating value. This dedicated embedding is produced in exactly same way as we generated a_t in eq. (2.9), but uses a di□erent set of parameters for the feed-forward network. As we can see from table 2.4 (“dedicated embed"), sharing the parameters with the action embedding is important.

Keeping entities. Guo et al. (2019) also uses a memory-augmented pointer network to perform a copy mechanism which assists column selection. In contrast with our memory matrix consisting of action embeddings (eq. (2.7)), their memory matrix consists of the entity embeddings of columns that have been selected previously. They further remove the columns from the candidates in the schema linking process

Model	Dev Acc. (%)	
	GlobalGNN	GlobalBERT
Base	49.3	53.5
Ours	52.8	57.6
dedicated embed	50.0	55.4
removing entity	49.4	57.1
without copy	50.9	55.8

Table 2.4: Alternative approaches and ablation results.

once they are generated to prevent the decoder from repeatedly generating the same columns. To determine if our dynamic gating and structural linking module can add enough structural constraints to the decoding process to resolve this kind of problem, we conduct an experiment where we also remove the entities in the schema linking process after they are generated (*i.e.*, "removing entity" in table 2.4) to see if it further improves the model. Our results shows that this change can actually hurts the model. Specifically, we observe a drop in accuracy of the `WHERE` and `IUEN` clauses, which suggests that in our context, the information about a specific entity in the schema linking process is still useful even after the entity has been generated once.

Copy Gate. In addition, removing the copy gate and copy mechanism also harms the performance of our model (*i.e.*, "without copy" in table 2.4). This result confirms that it is beneficial to handle different types of structural links separately. We hypothesize that different types of structural links conflict with each other, so they are hard to fit in one structural linking matrix. Overall, these two results further supports our claim that the copy gate can determine when to copy or link to an entity by itself.

2.3.4 Error Analysis

Gate bias. Error analysis reveals that our gating mechanism is biased, *e.g.*, for the first few entities being selected in a SQL the link gate is trained to favor schema linking in most cases. But, such bias could sometimes be wrong. In such cases where structural linking is needed but absent, the model may select duplicated columns or the wrong table during decoding. Similar problem happens to the copy gate. Out of

NL	*Show the stadium name and the number of concerts in each stadium.*
SQL	SELECT stadium.name $_{l = N/A;\ c = N/A}$, COUNT(□) FROM concert $_{l = 0:15;\ c = 0:00}$ JOIN stadium $_{l = 0:00;\ c = 0:00}$ ON concert.stadium_id = stadium.stadium_id $_{l = 0:09;\ c = 0:00}$ $_{l = 0:00;\ c = 0:00}$ GROUP BY concert.stadium_id; $_{l = 0:00;\ c = 0:96}$
NL	*Which city has most number of departing flights?*
SQL	SELECT airports.city $_{l = N/A;\ c = N/A}$ FROM airports $_{l = 0:82;\ c = 0:01}$ JOIN flights $_{l = 0:00;\ c = 0:00}$ ON flights.airportcode = flights.sourceairport $_{l = 0:00;\ c = 0:00}$ $_{l = 0:00;\ c = 0:00}$ GROUP BY airports.city $_{l = 0:50;\ c = 1:00}$ ORDER BY COUNT (□) DESC LIMIT 1

Table 2.5: Sample predictions of our model. $_{link}$ and $_{copy}$ are abbreviated as $_{l}$ and $_{c}$ respectively. Gate values are not applicable (denoted by N/A) for the first entity since it has no previously generated entity. Some of the results reflect gate bias, see text for details.

all the SQL clause components, only the GROUP BY clause's partial matching F1 score drops (about 3%). This is caused by the copy gate biasing toward copying an entity from memory in this case. It is true that the entity needed in the GROUP BY clause is usually selected, but the information from schema linking can still be beneficial,[7] *e.g.*, in the second example of table 2.5, the model wants to copy the wrong entity but the link gate rectifies it with schema linking information. Our gating mechanism only relies on the current action embedding (which can be seen as short-term structural information) to determine the gate values. We believe that introducing more global structural constraints is a promising direction to find a more flexible and accurate gating mechanism.

2.4 Discussion

Short attention spans. In our experiments, we notice that the action history the model usually attends to is very short, *i.e.*, the model only utilizes the the output memory of the most recent three entities in 99% of cases. This coincides with similar

[7] The semantics of some pronouns (*e.g.*, "each" and "which" in the examples of table 2.5) in NL question match with the GROUP BY clause, but this could be a dataset bias.

findings in language modeling (Daniluk et al., 2017) where the augmented-memory was expected to facilitate the modeling of long-range dependencies but failed to do so. Although long-term context is important for language modeling, it is not as important in our Text-to-SQL scenario since most dependencies in programming languages like SQL lie within a short span. We are interested in exploring semantic parsing tasks which requires long-term structural constraints using our method in the future.

Structural linking patterns. We have shown the effectiveness of our method in dealing with different types of structural links separately using different components of the model in the previous section. So far, the only special type of structural links we can explicitly model is the copy mechanism, and we treat all other types of links uniformly using additive attention in Equation 2.5. This might limit the model's ability to take advantage of the complicated relationships between entities. Currently, the entity representation provided by the GNN model is still difficult to explain, because the node representations contain a mix of information from different message-passing steps. One could imagine training GNNs with different message-passing steps each modeling a different level of structural linking, could lead to a more clear and expressive linking pattern.

Chapter 3

Graph Representation Learning with Hierarchical Transformers

The process of formulating two types linkings showed that, the levels of abstraction in both the representations of language and graph-structured data can be utilized to form more meaningful interactions between them. However, manually finding such interactions requires a deep understanding of the application domain. In this chapter, I investigate a more expressive model that works for generic graph-structured data while having a structure that foster the development of the levels of abstraction[1]. Additionally, in text-to-SQL, we have seen how domain-specific graph-structured data can benefit natural language understanding applications to generalize across domains. In this chapter, I study another type of graph-structured data, which is associated with more general world knowledge, in order to benefit more language understanding tasks.

Knowledge graphs have been long used for storing factual information about real-world entities (e.g., places and people) and the relationships between them. Building representations for knowledge graphs to be further used in downstream tasks

[1]Material in this chapter is adapted from:

- Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang and Yangfeng Ji. HittER: Hierarchical Transformers for Knowledge Graph Embeddings. *arXiv preprint arXiv:2008.12813.*, 2020.

has attracted consistent research attention for decades. The major challenges of learning representations for knowledge graphs derive from their complicated multi-relational structures and severe incompleteness. Besides their broad application scenarios, knowledge representations can be intrinsically evaluated based on the ability of inferring facts that are missing in training data.

This chapter presents HittER, a Hierarchical Transformer model to jointly learn Entity-relation composition and Relational contextualization based on a source entity's neighborhood. My proposed model consists of two different Transformer blocks: the bottom block extracts features of each entity-relation pair in the local neighborhood of the source entity and the top block aggregates the relational information from the outputs of the bottom block. I further design a masked entity prediction task to balance information from the relational context and the source entity itself. Evaluated on the task of link prediction, my approach achieves new state-of-the-art results on two standard benchmark datasets FB15K-237 and WN18RR.

Note that although HittER is evaluated intrinsically in this chapter, it is motivated by the language understanding application scenario and can be used to by exporting its computed representations of a subgraph centered around an entity. The following chapter will showcase HittER's flexibility by integrating it into state-of-the-art language models.

3.1 Related Work

Learning representations of knowledge graph has been extensively studied in several diverse directions.

3.1.1 Triple-based Methods

Most of the previous work focuses on exploiting explicit geometric properties in the embedding space to capture different relations between entities. Early work uses translational distance-based scoring functions defined on top of entity and relation embeddings (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Ji et al., 2015).

Another line of work uses tensor factorization methods to match entities semantically. Starting from simple bi-linear transformations in the euclidean space (Nickel et al., 2011; Yang et al., 2015), numerous complicated transformations in various spaces have been hence proposed (Trouillon et al., 2016; Ebisu and Ichise, 2018; Sun et al., 2018; Zhang et al., 2019b; Chami et al., 2020; Tang et al., 2020). Such methods effectively capture the intuition from observation of data but suffer from unobserved geometric properties and are generally limited in expressiveness.

In light of recent advances in deep learning, more powerful neural network modules such as Convolutional Neural Networks (Dettmers et al., 2018), Capsule Networks (Nguyen et al., 2019) are also introduced to capture the interaction between entity embeddings and relation embeddings. Similar to our *entity Transformer* block, Wang et al. (2019) use the Transformer to contextualize an entity embedding with the corresponding relation embedding. These methods produce richer representations and better performance on predicting missing links in knowledge graphs. However, they only learn from pairwise local connectivity patterns in the graph but ignore the structured information stored in the graph context.

3.1.2 Context-aware Methods

Various forms of graph contexts have been proven effective in recent work on neural networks operating in graphs under the message passing framework (Bruna et al., 2014; Defferrard et al., 2016; Kipf and Welling, 2017). Schlichtkrull et al. (2018, R-GCN) adapt the Graph Convolutional Networks to realistic knowledge graphs which are characterized by their highly multi-relational nature. Teru et al. (2020) incorporate an edge attention mechanism to R-GCN, showing that the relational path between two entities in a knowledge graph contains valuable information about their relations in an inductive learning setting. Vashishth et al. (2020) explore the idea of using existing knowledge graph embedding methods to improve the entity-relation composition in various Graph Convolutional Network-based methods. Bansal et al. (2019) borrow the idea from Graph Attention Networks (Veličković et al., 2018), using a bi-linear attention mechanism to selectively gather useful information from

neighbor entities. Di□erent from their simple single-layer attention formulation, we use the advanced Transformer to capture both the entity-relation and entity-context interactions. Nathani et al. (2019) also propose an attention-based feature embedding to capture multi-hop neighbor information, but unfortunately, their reported results have been proven to be unreliable in a recent re-evaluation (Sun et al., 2020b).

3.2 Model

We introduce our proposed hierarchical Transformer model (Figure 3-1) in this section. In Section 3.2.1, we formally define the link prediction task in a knowledge graph, and demonstrate how to solve it by a simple Transformer scoring function. We then cover the detailed architecture of our proposed model in Section 3.2.2. Finally, we discuss our strategies to learn balanced contextual representations of an entity in Section 3.2.3.

3.2.1 Transformers for Link Prediction

Formally, a knowledge graph can be viewed as a set of triplets ($G = f(e_s; r_p; e_o)g$) and each has three items including the subject e_s *2* E, the predicate r_p *2* R, and the object e_o *2* E to describe a single fact (link) in the knowledge graph. Our model approximates a pointwise scoring function : $E \times R \times E$ *7!* R which takes a triplet as input and produces a score reflecting the plausibility of the fact represented by the triplet. In the task of link prediction, given a triplet with either the subject or the object missing, the goal is to find it from the set of all entities E. Without loss of generality, we describe the case where an incomplete triplet ($e_s; r_p$) is given and we want to predict the object e_o. And vice versa, the subject e_s can be predicted in a similar process, except that the reciprocal predicate will be used to distinguish these two cases (Lacroix et al., 2018). We call the entity in the incomplete triplet the source entity e_{src} and call the entity we want to predict the target entity e_{tgt}.

Link prediction can be done in a straightforward manner with a Transformer encoder (Vaswani et al., 2017) as the scoring function, depicted inside the dashed

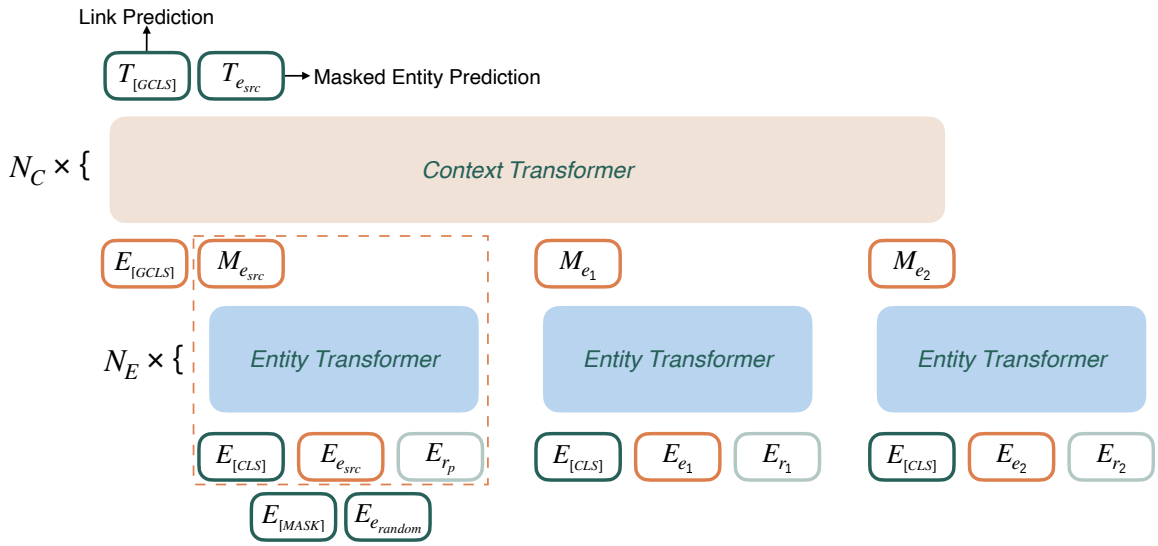

Figure 3-1: Our model consists of two Transformer blocks organized in a hierarchical fashion. The bottom Transformer block captures the interactions between a entity-relation pair while the top one gathers information from an entity's graph neighborhood. Taking the entity embeddings E_e and the relation embeddings E_r as input, the output embedding $T_{[\text{GCLS}]}$ is used for predicting the target entity. We sometimes mask or replace $E_{e_{src}}$ with $E_{[MASK]}$ or $E_{e_{random}}$. In which case, an additional output embedding $T_{e_{src}}$ can be used to recover the perturbed entity. The dashed box indicates a simple context-independent baseline where $M_{e_{src}}$ is directly used for link prediction.

box in Figure 3-1. Our inputs to the Transformer encoder are randomly initialized embeddings of the source entity e_{src}, the predicate r_p, and a special [CLS] token which serving as an additional bias term. Three di□erent learned type embeddings are directly added to the three token embeddings similar to the input representations of BERT (Devlin et al., 2019). Then we use the output embedding corresponding to the [CLS] token ($M_{e_{src}}$) to predict the target entity, which is implemented as follows. We first compute the plausibility score of the true triplet as a dot-product between $M_{e_{src}}$ and the token embedding of the target entity. In the same way, we also compute the plausibility scores for all other candidate entities and normalize them using the softmax function. Lastly, we use the normalized distribution to get the cross-entropy loss $L_{\text{LP}} = -\log p(e_{tgt} \; j \; M_{e_{src}})$ for training. We will use this model as a simple *context-independent* baseline later in experiments. A similar approach has been explored in Wang et al. (2019).

Learning knowledge graph embeddings from one triplet at a time ignores the

abundant structural information in the graph context. Our model, as described in the following section, also considers the relational neighborhood of the source vertex (entity), which includes all of its adjacent vertices in the graph, denoted as $N_G(e_{src}) = f(e_{src}; r_i; e_i)g$.[2]

3.2.2 Hierarchical Transformers

We propose a hierarchical Transformer model for knowledge graph embeddings (Figure 3-1). The proposed model consists of two blocks of multi-layer bidirectional Transformer encoders.

We employ the Transformer described in Section 3.2.1 as our bottom Transformer block, called the *entity Transformer*, to learn interactions between an entity and its associated relation type. Different from the previous described context-independent scenario, this entity Transformer is now generalized to also encode information from a relational context. In specific, there are two cases in our context-dependent scenario:

1. We consider the source entity with the predicate in the incomplete triplet as the first pair;

2. We consider an entity from the graph neighborhood of the source entity with the relation type of the edge that connects them.

The bottom block is responsible of packing all useful features from the entity-relation composition into vector representations to be further used by the top block.

The top Transformer block is called the *context Transformer*. Given the output of the previous entity Transformer and a special [GCLS] embedding, it contextualizes the source entity with relational information from its graph neighborhood. Similarly, three type embeddings are assigned to the special [GCLS] token embedding, the intermediate source entity embedding, and the other intermediate neighbor entity embeddings. The cross-entropy loss for link prediction is now changed as follows.

[2]Our referred neighborhood is slightly different from the formal definition since we only consider edges connecting to the source vertex.

$$L_{\text{LP}} = -\log p(e_{tgt} \mid T_{[GCLS]}) \tag{3.1}$$

The top block does most of the heavy lifting to aggregate contextual information together with the information from the source entity and the predicate, by using structural features extracted from the output vector representations of the bottom block.

3.2.3 Balanced Contextualization

Trivially supplying contextual information to the model during learning might cause problems. On one hand, since a source entity often contains particular information for link prediction, the model may learn to ignore the additional contextual information, which could also be noisy. On the other hand, the introduction of rich contextual information could in turn downgrade information from the source entity and cause potential over-fitting problems. Inspired by the successful Masked Language Modeling pre-training task in BERT, we propose a two-step *Masked Entity Prediction* task (MEP) to balance the process of contextualization during learning.

To avoid the first problem, we apply a masking strategy to the source entity of each training example as follows. During training, we randomly select a proportion of training examples in a batch. With certain probabilities, we replace the input source entity with a special mask token `[MASK]`, a random chosen entity, or just leave it unchanged. The purpose of these perturbations is to introduce extra noise to the information from the source entity, thus forcing the model to learn contextual representations. The probability of each category is dataset-specific hyper-parameter: for example, we can mask out the source entity more frequently if its graph neighborhood is denser (in which case, the source entity can be easily replaced by the additional contextual information).

In terms of the second problem, we want to promotes the model's awareness of the masked entity. Thus we train the model to recover the perturbed source entity based on the additional contextual information. To do this, we use the output embedding

corresponding to the source entity $T_{e_{src}}$ to predict the correct source entity via a classification layer.[3] We can add the cross-entropy classification loss to the previous mentioned link prediction loss as an auxiliary loss, as follows.

$$L_{\text{MEP}} = -\log p(e_{src} \mid T_{e_{src}}) \tag{3.2}$$

$$L = L_{\text{LP}} + L_{\text{MEP}} \tag{3.3}$$

This step is important when solely relying on the contextual clues is insufficient to do link prediction, which means the information from the source entity needs to be emphasized. And it is otherwise unnecessary when there is ample contextual information. Thus we use dataset-specific configurations to strike a balance between these two sides. However, the first step of entity masking is always beneficial to the utilization of contextual information according to our observations.

In addition to the MEP task, we implement a uniform neighborhood sampling strategy where only a fraction of the entities in the graph neighborhood will appear in a training example. This sampling strategy acts like a data augmenter and similar to the edge dropout regularization in graph neural network methods (Rong et al., 2020). We also have to remove the ground truth target entity from the source entity's neighborhood during training. It will otherwise create a dramatic train-test mismatch because the ground truth target entity can always be found from the source entity's neighborhood during training while it can rarely be found during testing. The model will thus learn to naively select an entity from the neighborhood.

3.3 Experiments

We describe our experiments in this section. Section 3.3.1 introduces two popular benchmarks for link prediction. We then describe our evaluation protocol in Section 3.3.2, and the detailed experimental setup in Section 3.3.3. At last, our proposed method are assessed both quantitatively and qualitatively in Section 3.3.4, and several

[3] We share the same weight matrix in the input embeddings layer and the linear transformation of this classification layer.

Model	FB15K-237					WN18RR				
	#Params	MRR"	Hits"			#Params	MRR"	Hits"		
			@1	@3	@10			@1	@3	@10
RESCAL (Nickel et al., 2011)	6M	.356	.266	.390	.535	6M	.467	.439	.478	.516
TransE (Bordes et al., 2013)	2M	.310	.218	.345	.495	21M	.232	.061	.366	.522
DistMult (Yang et al., 2015)	4M	.342	.249	.378	.531	21M	.451	.414	.466	.523
ComplEx (Trouillon et al., 2016)	4M	.343	.250	.377	.532	5M	.479	.441	.495	.552
ConvE (Dettmers et al., 2018)	9M	.338	.247	.372	.521	36M	.439	.409	.452	.499
RotatE (Sun et al., 2018)	15M	.338	.241	.375	.533	20M	.476	.428	.492	.571
TuckER (Balazevic et al., 2019)	-	.358	.266	.394	.544	-	.470	.443	.482	.526
CompGCN (Vashishth et al., 2020)	-	.355	.264	.390	.535	-	.479	.443	.494	.546
RotH (Chami et al., 2020)	8M	.344	.246	.380	.535	21M	.496	.449	.514	.586
HittER	16M	.373	.279	.409	.558	24M	.503	.462	.516	.584

Table 3.1: Comparison between the proposed method and a set of baselines evaluated on two standard datasets. Results of RotatE, TuckER, CompGCN, and RotH are taken from the original papers. Numbers in bold represent the best results.

ablation studies are conducted in Section 3.3.5.

3.3.1 Datasets

We evaluate our proposed method on two standard benchmark datasets FB15K-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018) for link prediction.[4] FB15K-237 is a subset sampled from the Freebase (Bollacker et al., 2008) with trivial inverse links removed. It stored facts about topics in movies, actors, awards, etc. WN18RR is a subset of the WordNet (Miller, 1995) which contains structured knowledge of English lexicons. Statistics of these two datasets are shown in Table 3.2. Notably, WN18RR is much sparser than FB15k-237 which implies it has less structural information in the local neighborhood of an entity. This will affect our configurations of the masked entity prediction task consequently.

3.3.2 Evaluation Protocol

The task of link prediction in a knowledge graph is defined as an entity ranking task. Essentially, for each test triplet, we remove the subject or the object from it and

[4]We intentionally omit the original FB15K and WN18 datasets because of their known flaw in test-leakage (Toutanova and Chen, 2015).

Dataset	FB15K-237	WN18RR
# Entities	14,541	40,943
# Relations	237	11
# Triples	310,116	93,003
# Avg. degree	42.7	4.5

Table 3.2: Dataset statistics. The WN18RR dataset is significantly sparser than the FB15K-237 dataset.

let the model predict which is the most plausible answer among all possible entities. After scoring all entity candidates and sorting them by the computed scores, the rank of the ground truth target entity is used to further compute various ranking metrics such as mean reciprocal rank (MRR) and hits@k, $k \in \{1, 3, 10\}$. We report all of these ranking metrics under the filtered setting proposed in Bordes et al. (2013) where valid entities except the ground truth target entity are filtered out from the rank list.

3.3.3 Experimental Setup

We implement our proposed method in PyTorch (Paszke et al., 2019b) under the LibKGE framework (Ru□nelli et al., 2020). To perform a fair comparison with some early baseline methods, we reproduce the baseline results by using the best hyper-parameter configurations for them from LibKGE.[5]

Our model consists of a three-layers entity Transformer and a six-layers context Transformer. Each Transformer layer has eight heads. The dimension size of hidden states is 320 across all layers except that we use 1280 dimensions for the position-wise feed-forward networks inside Transformer layers suggested by Vaswani et al. (2017). We set the maximum numbers of uniformly sampled neighbor entities for every example in the FB15K-237 and WN18RR dataset to be 50 and 12 respectively. Such configurations ensure most examples (more than 85% of the cases in both datasets) can have access to its entire local neighborhood during inference. During training, we further uniformly sample 70% and 50% of entities from these fixed-size sets in the

[5] These configurations consider many recent training techniques and are found by extensive quasi-random search. Thus the results are generally much higher then the initially reported ones.

Entity	Top 5 Neighbors
Dominican Republic	Costa Rica, Ecuador, Puerto Rico, Colombia, El Salvador
Republic	Presidential system, Unitary state, Democracy, Parliamentary system, Constitutional monarchy
MMPR	Power Rangers, Sonic X, Ben 10, Star Trek: Enterprise, Code Geass
Wendee Lee	Liam O'Brien, Michelle Ru□, Hilary Haag, Chris Patton, Kari Wahlgren
Drama	Thriller, Romance Film, Mystery, Adventure Film, LGBT
Land reform	Pronunciamento, Premium, Protest march, Reform, Birth-control reformer
Reform	Reform, Land reform, Optimization, Self-reformation, Enrichment
Cover	Surface, Spread over, Bind, Supply, Strengthen
Covering	Sheet, Consumer goods, Flap, Floor covering, Coating
Phytology	Paleobiology, Zoology, Kingdom fungi, Plant life, Paleozoology

Table 3.3: Nearest neighbors of first five entities on FB15K-237 and WN18RR based on the cosine similarity between learned entity embeddings from our proposed method.

FB15K-237 and WN18RR dataset.

We train our models using Adam (Kingma and Ba, 2015) with a learning rate of 0:01 and an L2 weight decay rate of 0:1. The learning rate linearly increases from 0 over the first tenth training steps, and linearly decreases through the rest of the steps. We apply dropout (Srivastava et al., 2014) with a probability $p = 0:1$ for all layers, except that $p = 0:6$ for the embedding layers. We apply label smoothing with a rate 0:1 to prevent the model from being over-confident during training. We train our models using a batch size of 512 for at most 500 epochs and employ early stopping based on MRR in the validation set.

When training our model with the masked entity prediction task, our dataset-specific configurations are listed as follows:

- WN18RR: In 80% of examples in a batch, 60% of examples are masked out. The rest of the examples are divided in a 3:7 ratio for replaced and unchanged ones.

- FB15K-237: 50% of examples in a batch are masked out. No replaced or unchanged ones. We do not include the auxiliary loss.

All experimental code and model checkpoints will be publicly available after the anonymity period.

3.3.4 Experimental Results

Table 3.1 shows that the results of HittER compared with baseline methods including some early methods and previous SOTA methods.[6] We outperform all previous work by a substantial margin across nearly all the metrics. Comparing to some previous methods which target some observed patterns of specific datasets, our proposed method is more general and is able to give more consistent improvements over the two standard datasets. For instance, the previous SOTA in WN18RR, RotH explicitly captures the hierarchical and logical patterns by hyperbolic embeddings. Comparing to it, our model performs better especially in the FB15K-237 dataset which has a set of diverse relation types. On the other hand, our models have comparable numbers of parameters to the baseline methods, since entity embeddings contribute to the majority of the parameters.

Table 3.3 lists the entity clustering results of first few entities in each dataset, based on our learned entity representations. Clusters in FB15K-237 usually are entities of the same type, such as South/Central American countries, government systems, and American voice actresses. While clusters in WN18RR are generally looser but still relevant to the topic of the central word.

3.3.5 Ablation Studies

To figure out the contributions from each aspect of our proposed method, we perform several ablation studies in this section.

Table 3.4 shows the results of removing the masked entity prediction task described in Section 3.2.3 (*i.e.*, "No MEP") and entirely removing the context Transformer from

[6] Lacroix et al. (2018) use 2000 dimension vectors for entity embeddings which leads to an incomparable parameters size. So we do not include it in our comparisons.

Model	FB15K-237		WN18RR	
	MRR	H@10	MRR	H@10
Full HittER	.379	.563	.501	.586
No MEP	.373	.560	.489	.564
No context	.371	.554	.473	.539

Table 3.4: Results of ablation studies on the development sets.

Relation Name	Count	No ctx	Full	Gain
hypernym	1174	.144	.181	26%
derivationally related form	1078	.947	.947	0%
member meronym	273	.237	.316	33%
has part	154	.200	.235	18%
instance hypernym	107	.302	.330	9%
synset domain topic of	105	.350	.413	18%
verb group	43	.930	.931	0%
also see	41	.585	.595	2%
member of domain region	34	.201	.259	29%
member of domain usage	22	.373	.441	18%
similar to	3	1	1	0%

Table 3.5: Development set MRR and relative improvement percentage of our proposed method with or without the *context Transformer* respect to each relation in the WN18RR dataset.

the full model (the context-independent Transformer for link prediction described in Section 3.2.1, *i.e.*, "No context"). In FB15K-237, we find that the "No context" model is already very strong which demonstrates our entity Transformer's capability of capturing interactions between entities and their associated relations. Adding contextual information can further improve our model while our proposed entity masking strategy plays a very important role in both datasets.

Breaking down the model's performance by type of relations in WN18RR, Table 3.5 shows that incorporating contextual information brings us substantial improvements on two major relation types, namely the *hypernym* and the *member meronym* relations, which both include many examples belong to the challenging one-to-many relation categories defined in Bordes et al. (2013).

Inferring the relationship between two entities can be viewed as a process of aggre-

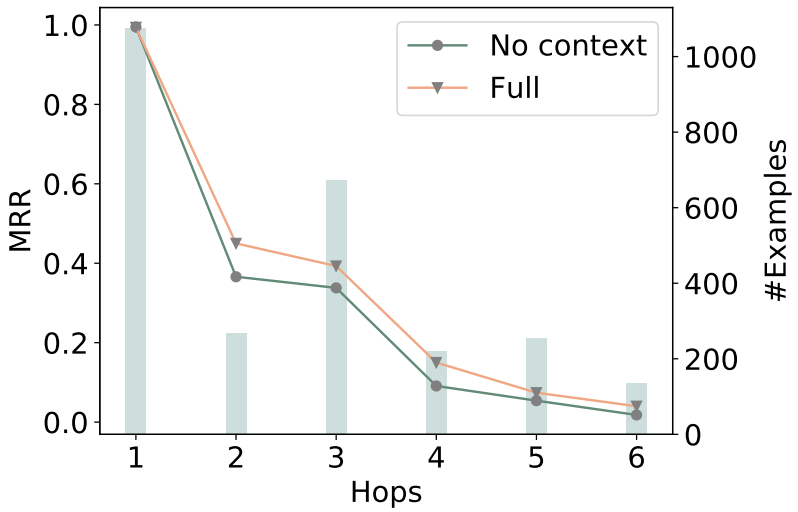

Figure 3-2: Development set mean reciprocal rank (MRR) in the WN18RR dataset grouped by the number of hops. The bar chart shows the number of examples in each group.

gating information from the graph paths between them (Teru et al., 2020). To gain further understanding of what the role does contextual information play from this perspective, we group examples in the development set of WN18RR by the number of hops (*i.e.*, the shortest path length in the undirected training graph) between the subject and the object in each example (Figure 3-2). From the results, we can see that the MRR metric of each group decreases by the number of hops of the examples. This matches our intuition that aggregating information from longer graph paths is generally harder and such information is more unlikely to be meaningful. Comparing models with and without the contextual information, the contextual model performs much better in groups of multiple hops ranging from two to four. The improvement also shrinks as the number of hops increasing.

3.4 Discussion

3.4.1 Right Context for Link Prediction

Structural information of knowledge graphs can come from multiple forms, such as graph paths, sub-graphs, and the local neighborhood that we used in this work. In

addition, these context forms can be represented in terms of the relation type, the entity, or both of them.

In this work, we show that a simple local neighborhood is sufficient to greatly improve a link prediction model. In early experiments in the FB15K-237 dataset, we actually observe that masking out the source entity all the time does not harm the model performance much. This shows that the contextual information in a dense knowledge graph dataset like FB15K-237 is meaningful enough to replace the source entity itself in the link prediction task.

Recently, Wang et al. (2020c) argue that graph paths and local neighborhood should be jointly considered when only the relation types is used (throwing out entities). Although some recent work has made a first step towards utilizing graph paths for knowledge graph embeddings (Wang et al., 2019, 2020b), there are still no clear evidence showing its effectiveness.

3.4.2 Limitations of the *1vsAll* Scoring

Recall that HittER learns a representation for an incomplete triplet $(e_s; r_p)$ and then computes the dot-product between it and all the candidate target entity embeddings. This two-way scoring paradigm, which is often termed *1vsAll* scoring, supports fast training and inference when the interactions between the source entity and the predicate are captured by some computation-intensive operations (*i.e.*, the computations of Transformers in our case), but unfortunately loses three-way interactions. We intentionally choose 1vsAll scoring for two reasons. On one hand, 1vsAll together with cross-entropy training has shown a consistent improvement over other alternative training configurations empirically (Ruffinelli et al., 2020). On the other hand, it ensures a reasonable speed for the inference stage where the 1vsAll scoring is necessary.

Admittedly, early interactions between the source entity and the target entity can provide valuable information to inform the representation learning of the incomplete triplet $(e_s; r_p)$. For instance, we find that a simple bilinear formulation of the source entity embeddings and the target entity embeddings can be trained to reflect the

distance (measured by the number of hops) between the source entity and the target entity in the graph. We leave the question of how to e□ectively and e□ciently incorporate such early fusion for future work.

Chapter 4

Factoid Question Answering with Knowledge Augmented Language Models

The last chapter demonstrated the effective use of HittER by performing an intrinsic evaluation task link prediction. In this chapter, I mainly focus on providing a proof-of-concept of how to infuse the hierarchical knowledge graph representations produced by HittER into Transformer-based pre-trained language models like BERT. My proposed model combines language representations and knowledge graph representations by a simple word-entity cross-attention mechanism, forming lots of meaningful layer-wise interactions between BERT and HittER.

Previous research has shown that after pretraining on huge corpora containing billions of words, BERT-like language models can recall some factual knowledge such as a celebrity's birthday or birthplace. I show that such ability can be substantially improved by utilizing representations provided by HittER which is directly trained on knowledge bases that contain factual information. Preliminary experiments on two Freebase question answering datasets show that the model improves the language model's ability to answer questions related to relational factual knowledge. In addition, I show that a proper connection strategy between the layer hierarchies of BERT and HittER is important for downstream task performance.

4.1 Related Work

After the successes of language models in many language tasks, people have been interested in probing what kinds of information are stored inside the parameters of these large language models after training on billion of words (Kovaleva et al., 2019; Rogers et al., 2020). Research has discovered that, in addition to a vast amount of linguistic knowledge (Tenney et al., 2019), these language models also demonstrate an ability to recall factual knowledge, serving as a kind of unsupervised knowledge representation (Petroni et al., 2019). Nevertheless, language models can be further improved by injecting knowledge from external sources, especially for knowledge-intensive downstream tasks. Peters et al. (2019) proposes KnowBert to inject knowledge about entities into BERT by integrating a word-to-entity attention module to a few layers of BERT. It uses knowledge representations learned by a relatively simple linear knowledge embedding model TuckER (Balazevic et al., 2019), while the HittER model I used can generate multiple layers of output representations.

A recently-proposed Entities as Experts model (Févry et al., 2020) employs a similar entity embedding retrieval and recontextualization process like the KnowBert. But the entity embeddings it used are trained with the language model using Wikipedia text with entities identified by hyperlinks. Improving on the Entities as Experts model, Verga et al. (2020) proposes the Facts as Experts model to directly incorporate information of factual assertions, rather than hoping the entity embeddings store enough factual information. Di□erent from the two previous models, my proposed method does not require training from scratch on a huge amount of data but uses both pre-trained language and knowledge representation. Moreover, the HittER is pre-trained on the entire knowledge graph with contextual information from the graph neighborhood of an entity, so it can capture more structural information from the graph context.

Recent work has also tried using the Transformer architecture to jointly model language and knowledge. CoLAKE (Sun et al., 2020a) flattens the fact assertion triples into a sequence of entities interleaved with their relations to the context entity

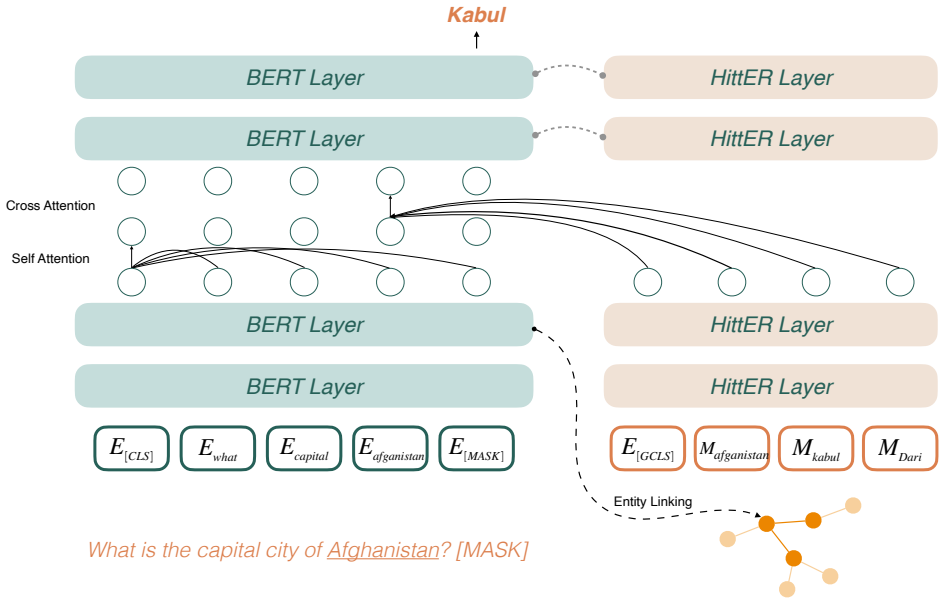

Figure 4-1: The architecture of my proposed HitBERT model for factoid questions answering task. It consists of two Transformer-based models, i.e., a language model BERT on the left and a knowledge graph representation model HittER on the right. Every higher layer of BERT is connected to a HittER layer by a word-entity cross-attention mechanism.

in a natural language sentence. LUKE (Yamada et al., 2020) keeps only the entities and throws away the relations between them from the inputs of the model. In the model, the relations between entities are introduced via a relation classification task. My proposed model has relations between entities modeled in a hierarchical fashion to fully exploit the structural information stored in pre-constructed knowledge bases. Furthermore, my strategy of combining BERT and HittER after pre-training allows them to be trained on large corpora separately to get the best from both worlds. Beyond entity-based factual knowledge, the cross-attention mechanism in Transformer has been extended for modeling cross-modality data such as images and languages (Tan and Bansal, 2019).

4.2 Model

In this section, I first formulate our factoid questions answering task and then describe the details of my proposed HitBERT model.

Figure 4-1 shows the overall architecture of the proposed HitBERT model. Inside in, both the BERT and the HittER model can generate multiple layers of language or knowledge representations, given two sequences of input tokens consist of words and entities respectively. Here I assume the entity token sequence is retrieved by an oracle entity linker for simplicity. In practice, such entity linker can be learned by using similarity scores defined as the dot-product of word representations from lower layer BERT representations and the lookup entity embeddings.

Formally, BERT takes the word tokens sequence of the input factoid question to generate the language representation h_i^n for every i-th token in its n-th layer. On the other side, a entity token sequence of the input subgraph centered in an entity is passed to the HittER to generate knowledge graph representations e_i^n for every for every i-th entity in its n-th layer.

Both the BERT and the HittER model use stacked Transformer building blocks consist of a multi-head self-attention layer and fully connected layers. When work in one of the both models independently, the self-attention layer takes outputs from last layer, transforming them into three vectors: a query q, a key k, and a value v, via linear transformations with different set of parameters. The attention mechanism is essentially operated as follows:

$$\text{Attention}(q; k; v) = \text{softmax}\left(\frac{qk^T}{\sqrt{d_k}}\right) v \tag{4.1}$$

where d_k is the dimensionality of the key used to scale the attention scores.[1]

To enhancing BERT with HittER, after the self-attention layer in a BERT layer, I introduce a word-entity cross-attention layer where the query is still from the outputs of last layer while the key and value are from the outputs of a corresponding HittER

[1] This formulation is simplified by intentional omitting the multi-head decomposition and concatenation. In practice, the query, key, and value can be batched as matrices to speed up computations.

layer. Specifically, if the i-th layer of BERT and the j-th layer of HittER is connected, it is operated as follows.

$$\hat{h}^i = \text{Attention}(h^{i-1}; h^{i-1}; h^{i-1}) \tag{4.2}$$

$$h^i = \text{Attention}(\hat{h}^i; e^j; e^j) \tag{4.3}$$

The linear transformations and residual connections between layers are omitted here for simplicity. In this manner, the factual knowledge stored in the representations of HittER is infused to BERT in a layer-wise fashion, so that I can further apply different strategies in connecting layers of BERT and HittER. As we will see in the experiments, different strategies have different impact on the performance of the proposed HitBERT model in the factoid question answering task.

HitBERT answers a factoid question by predicting the word tokens of the answer. A factoid question is converted to a Cloze question via appending the special `[MASK]` tokens to the end. For simplicity, I assume that the number answer tokens is known to HitBERT and append the exact number of `[MASK]` tokens to the end of a question sentence.[2] For example, in Figure 4-1, the Cloze question is represented as the token sequence *{'What'; 'is'; 'the'; 'capital'; 'city'; 'of'; 'Afghanistan'; '?';* `[MASK]`*}*. Similar to the masked language modeling objective in BERT, HitBERT is trained to recover the answer tokens replaced by `[MASK]`. A standard classification loss is computed to supervise the training.

4.3 Experiments

In the last section, we are left with the question of how can we effectively connect layers of BERT and HittER to construct HitBERT. In this section, I perform two line of experiments on two open domain question answering datasets to demonstrate that (1) the connection strategy does matter, (2) HitBERT can enhance BERT's ability to recall factual information.

[2] In practice, this assumption can be removed by adding a sequence generation model on top of BERT.

	FreebaseQA		WebQuestionSP	
	Full dataset	In FB15K-237	Full dataset	In FB15K-237
Train	20358	3713	3098	850
Dev	3994	743	-	-
Test	3996	755	1639	484

Table 4.1: Number of examples in two Freebase question answering datasets.

4.3.1 Datasets

I perform a series of experiments to evaluate the proposed HitBERT on two factoid question answering datasets: FreebaseQA and WebQuestionSP, which are both created from Freebase. The statistics of the two datasets are listed in Table 4.1.

4.3.2 Experimental Setup

Due to a formidable computation cost of training a HittER that can cover all entities used in the two factoid question answering datasets, I use the HittER model pre-trained on the FB15K-237 dataset described in the last chapter to perform experiments. To gain a better sense of the benefits HitBERT brought to us, in addition of the results on the full datasets, I also report results under a filtered settings, i.e., retaining only the questions that are related to entities in the FB15K-237 dataset. Table 4.1 show the number of examples on both datasets under the two settings. I use the accuracy measurement where a questions is answered correctly if the model predictions exactly match with the ground truth answer.

For the BERT model, I use the BERT base model released in the original paper. The BERT base model has 12 Transformer layers while the HittER model has 6 Transformer layers. The HitBERT is trained for 20 epochs. I use the Adam optimizer (Kingma and Ba, 2015) with the weight decay fix introduced in Loshchilov and Hutter (2019) and a linear-decayed learning rate schedule. The learning rate linearly increases from 0 over the first tenth training steps, and linearly decreases through the rest of the training steps. The peak learning rate is $5e-6$ for the parameters in the pre-trained BERT and HittER, and it is increased to $5e-5$ for parameters in the

Strategy	a	b	c	d	e	f
Dev Acc. (%)	28.5	28.2	26.8	23.7	22.9	22.7

Table 4.2: Performance of each connection strategy in Figure 4-2 measured by the development set accuracy under the filtered setting on the FreebaseQA dataset.

	FreebaseQA		WebQuestionSP	
	Full dataset	In FB15K-237	Full dataset	In FB15K-237
BERT	18.7%	23.7%	23.0%	44.2%
HitBERT	19.7%	28.5%	26.9%	50.2%

Table 4.3: Accuracy scores of HitBERT (using best performing strategy) in two Freebase question answering datasets, compared to the results of vanilla BERT.

newly added word-entity cross-attention modules.

4.3.3 Connection Strategy Testing

I design several reasonable connection strategies and test the HitBERT model composed using each strategy under the FB15k-237 filtered setting on the FreebaseQA dataset. Among six alternatives that I have experimented with, I find that connecting BERT and HittER in a pattern illustrated in Figure 4-2a gives us the best performance. A most straightforward strategy in Figure 4-2b performs similarly, suggesting that the first layer representation of HittER is not essential when representations of other layers are presented. The rest of the strategies, e.g., the strategies in Figure 4-2c and in Figure 4-2d which connect a higher or lower part of BERT, or the strategies in Figure 4-2e and Figure 4-2f which use only the top/bottom layer of HittER, are all nonideal strategies and could hurt the performance dramatically. This result reinforces my previous argument of meaningful interactions between language representations and knowledge representations are important in Chapter 2.

4.3.4 Experimental Results

According to table 4.3, BERT exhibits a fairly good understanding of factual knowledge in both datasets. However, HitBERT consistently improves on BERT in all

four comparisons, showing that incorporating relatively small amount of external knowledge can substantially benefit a language model trained on billions of words. In consistent with our intuition, the gains brought by HitBERT in filtered sets are much higher than those of the full dataset setting because filtered sets have significant larger proportions of answerable questions in terms of the factual knowledge stored in FB15K-237. Using a HittER model pre-trained on a larger knowledge graph such as the entire Freebase is expected to largely improve the performance of HitBERT in answering factoid question. Moreover, the proposed HitBERT can be easily adapted to many other knowledge-intensive language understanding task without modifications.

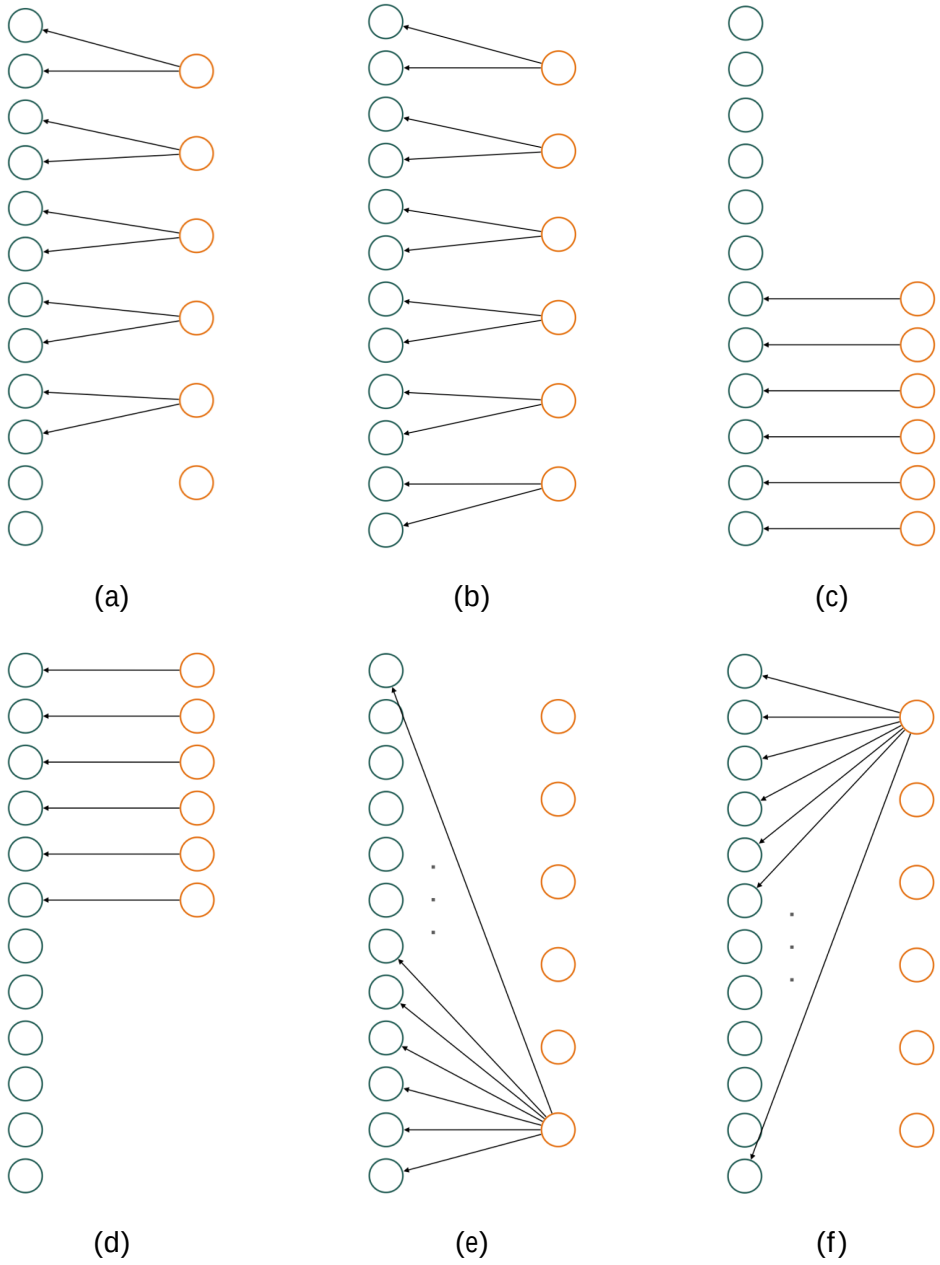

Figure 4-2: Several alternatives of HitBERT's connection strategy between 12 BERT layers (left green) and 6 HittER layers (right orange). Among them, strategy (a) achieves best performance in experiments.

Chapter 5

Conclusion

In this thesis, my main goal is to present to the audience the promising idea of using graph-based knowledge representations to contextualize machine understanding of natural language. In Chapter 2, the cross-domain Text-to-SQL semantic parsing task serves as an explanatory example of why we need to utilize graph-based knowledge representations in a very important application. By describing the challenges I faced and the application-specific insights I utilized in manually designing the two linking processes, I further point out the influential and challenging problem of building meaningful interactions between language representations and knowledge representations for general-purpose models. To make a step toward this problem, in Chapter 3, I design a more powerful representation model for learning graph-structured data. The ultimate goal behind the design is to unify the modeling architecture between language and graph-based knowledge to foster meaningful interactions between them, via the attention-like mechanism that has been proved to be e□ective. As a by-product and an intrinsic evaluation of my proposed model, it achieves state-of-the-art results in long-standing knowledge graph completion benchmarks. Finally, in Chapter 4, I demonstrate how exactly the meaningful interactions between language and knowledge representations can be built by proposing HitBERT. Use Freebase factoid question answering tasks as examples, I show clear evidence that HitBERT can substantially benefit large language representations trained on billions of words by adding only a little amount of parameters and computational overheads. Future

research can easily apply the proposed knowledge graph representations-augmented language model to knowledge-intensive language understanding tasks.

As the artificial intelligence community increasing its awareness of the importance of developing knowledge-driven approaches, I hope the contributions made by my thesis can provide inspiration for future research in incorporating graph-structured knowledge and building more advanced language understanding systems that are as knowledgeable as human beings.

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Balazevic, I., Allen, C., and Hospedales, T. (2019). TuckER: Tensor factorization for knowledge graph completion. 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)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Bansal, T., Juan, D.-C., Ravi, S., and McCallum, A. (2019). A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4387–4392, Florence, Italy. Association for Computational Linguistics.

Bender, E. M. and Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Bogin, B., Berant, J., and Gardner, M. (2019a). Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy. Association for Computational Linguistics.

Bogin, B., Gardner, M., and Berant, J. (2019b). Global reasoning over database structures for text-to-SQL parsing. 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)*, pages 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*.

Chami, I., Wolf, A., Juan, D.-C., Sala, F., Ravi, S., and Ré, C. (2020). Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.

Chen, B., Sun, L., Han, X., and An, B. (2016). Sentence rewriting for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777, Berlin, Germany. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Daniluk, M., Rocktäschel, T., Welbl, J., and Riedel, S. (2017). Frustratingly short attention spans in neural language modeling. In *International Conference on Learning Representations*.

De□errard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dong, L. (2019). *Learning Natural Language Interfaces with Neural Models*. phdthesis, the University of Edinburgh.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Ebisu, T. and Ichise, R. (2018). Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Févry, T., Soares, L. B., FitzGerald, N., Choi, E., and Kwiatkowski, T. (2020). Entities as experts: Sparse memory access with entity supervision. *arXiv preprint arXiv:2004.07202*.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Giordani, A. and Moschitti, A. (2009). Semantic mapping between natural language questions and sql queries via syntactic pairing. In *International Conference on Application of Natural Language to Information Systems*, pages 207–221. Springer.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., and Zhang, D. (2019). Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hwang, W., Yim, J., Park, S., and Seo, M. (2019). A comprehensive exploration on wikisql with table-aware word contextualization. In *Proceedings of the Second KR2ML workshop at NeurIPS 2019*.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., and Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, Beijing, China. Association for Computational Linguistics.

Kate, R. (2008). Transforming meaning representation grammars to improve semantic parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 33–40, Manchester, England. Coling 2008 Organizing Committee.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014). Multimodal neural language models. In *International conference on machine learning*, pages 595–603.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the dark secrets of BERT. 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)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Krishnamurthy, J., Dasigi, P., and Gardner, M. (2017). Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Kwiatkowksi, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA. Association for Computational Linguistics.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA. Association for Computational Linguistics.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Lacroix, T., Usunier, N., and Obozinski, G. (2018). Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, pages 2863–2872.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2016). Gated graph sequence neural networks. In *International Conference on Learning Representations*.

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. (2017). Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA. Association for Computational Linguistics.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI conference on artificial intelligence*.

Lin, Y., Tan, Y. C., and Frank, R. (2019). Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Ling, W., Blunsom, P., Grefenstette, E., Hermann, K. M., Kočiský, T., Wang, F., and Senior, A. (2016). Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609, Berlin, Germany. Association for Computational Linguistics.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Luong, T., Pham, H., and Manning, C. D. (2015). E□ective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Nathani, D., Chauhan, J., Sharma, C., and Kaul, M. (2019). Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, Florence, Italy. Association for Computational Linguistics.

Nguyen, D. Q., Vu, T., Nguyen, T. D., Nguyen, D. Q., and Phung, D. (2019). A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2180–2189, Minneapolis, Minnesota. Association for Computational Linguistics.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019a). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019b). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., and Smith, N. A. (2019). Knowledge enhanced contextual word representations. 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)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., and Miller, A. (2019). 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)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Popescu, A.-M., Etzioni, O., and Kautz, H. (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM.

Rabinovich, M., Stern, M., and Klein, D. (2017). Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Raiman, J. and Raiman, O. (2018). Deeptype: Multilingual entity linking by neural type system evolution. In *AAAI*.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*.

Rong, Y., Huang, W., Xu, T., and Huang, J. (2020). Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

Roy, D. K. and Pentland, A. P. (2002). Learning words from sights and sounds: A computational model. *Cognitive science*, 26(1):113–146.

Ru□nelli, D., Broscheit, S., and Gemulla, R. (2020). You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Shaw, P., Massey, P., Chen, A., Piccinno, F., and Altun, Y. (2019). Generating logical forms from graph representations of text and entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 95–106, Florence, Italy. Association for Computational Linguistics.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Suhr, A., Chang, M.-W., Shaw, P., and Lee, K. (2020). Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388, Online. Association for Computational Linguistics.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Sun, T., Shao, Y., Qiu, X., Guo, Q., Hu, Y., Huang, X., and Zhang, Z. (2020a). Colake: Contextualized language and knowledge embedding. *arXiv preprint arXiv:2010.00309*.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. (2018). Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Sun, Z., Vashishth, S., Sanyal, S., Talukdar, P., and Yang, Y. (2020b). A re-evaluation of knowledge graph completion methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5516–5522, Online. Association for Computational Linguistics.

Tamari, R., Shani, C., Hope, T., Petruck, M. R. L., Abend, O., and Shahaf, D. (2020). Language (re)modelling: Towards embodied language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6268–6281, Online. Association for Computational Linguistics.

Tan, H. and Bansal, M. (2019). LXMERT: Learning cross-modality encoder representations from transformers. 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)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Tang, L. R. and Mooney, R. J. (2000). Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, Hong Kong, China. Association for Computational Linguistics.

Tang, Y., Huang, J., Wang, G., He, X., and Zhou, B. (2020). Orthogonal relation transforms with graph context modeling for knowledge graph embedding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2713–2722, Online. Association for Computational Linguistics.

Tenney, I., Das, D., and Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Teru, K. K., Denis, E., and Hamilton, W. L. (2020). Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning*.

Toutanova, K. and Chen, D. (2015). Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016). Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

Vashishth, S., Sanyal, S., Nitin, V., and Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Verga, P., Sun, H., Soares, L. B., and Cohen, W. W. (2020). Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *arXiv preprint arXiv:2007.00849*.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Wang, B., Shin, R., Liu, X., Polozov, O., and Richardson, M. (2020a). RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Wang, H., Kulkarni, V., and Wang, W. Y. (2020b). Dolores: Deep contextualized knowledge graph embeddings. In *Automated Knowledge Base Construction*.

Wang, H., Ren, H., and Leskovec, J. (2020c). Entity context and relational paths for knowledge graph completion. *arXiv preprint arXiv:2002.06757*.

Wang, Q., Huang, P., Wang, H., Dai, S., Jiang, W., Liu, J., Lyu, Y., Zhu, Y., and Wu, H. (2019). Coke: Contextualized knowledge graph embedding. *arXiv preprint arXiv:1911.02168*.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Warren, D. H. and Pereira, F. C. (1982). An e□cient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *International Conference on Learning Representations*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Wu, C.-S., Socher, R., and Xiong, C. (2019). Global-to-local memory pointer networks for task-oriented dialogue. In *International Conference on Learning Representations*.

Xiao, C., Dymetman, M., and Gardent, C. (2016). Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Yamada, I., Asai, A., Shindo, H., Takeda, H., and Matsumoto, Y. (2020). Luke: Deep contextualized entity representations with entity-aware self-attention. *arXiv preprint arXiv:2010.01057*.

Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Yin, P. and Neubig, G. (2017). A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Yu, F., Tang, J., Yin, W., Sun, Y., Tian, H., Wu, H., and Wang, H. (2020). Ernie-vil: Knowledge enhanced vision-language representations through scene graph. *arXiv preprint arXiv:2006.16934*.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic. Association for Computational Linguistics.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666, Arlington, Virginia, United States. AUAI Press.

Zhang, R., Yu, T., Er, H., Shim, S., Xue, E., Lin, X. V., Shi, T., Xiong, C., Socher, R., and Radev, D. (2019a). Editing-based SQL query generation for cross-domain context-dependent questions. 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)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Zhang, S., Tay, Y., Yao, L., and Liu, Q. (2019b). Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2735–2745.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.