ABSTRACT

Title of dissertation: TOWARDS PRACTICAL COMPLEX QUESTION ANSWERING

Chen Zhao, 2022

Dissertation directed by: Professor Jordan Boyd-Graber
Department of Computer Science
College of Information Studies
Department of Linguistics
Institute for Advanced Computer Studies

Professor Hal Daumé III
Department of Computer Science
Department of Linguistics
Institute for Advanced Computer Studies

Question answering (QA) is one of the most important and challenging tasks for understanding human language. With the help of large scale benchmarks, there is tremendous success on building neural QA systems, and such progress has been deployed into commercial systems like search engines. However, most QA systems target at rather simple questions that can be answered within a single evidence piece (e.g., a sentence). In many real scenarios, users also ask complex questions that require multiple evidence pieces, and search engines fail to answer them. The goal of this dissertation work is to tackle complex QA problem from different angles.

We first study complex QA using text collections as a knowledge source. We build two QA systems that rely on a free-text knowledge graph from Wikipedia. Through extracting a question grounded sub-graph and using graph neural network

to reason over this graph, the proposed QA systems are state-of-the-art on multiple complex QA benchmarks. Then we present two solutions to address some key assumptions that make state-of-the-art QA systems difficult to generalize beyond specific benchmarks. We first address the assumption that the given text collections is semi-structured by hyperlinks. We propose a multi-step dense retrieval method to model the implicit relationships between evidence pieces. The retriever is competitive to state-of-the-arts on complex QA benchmarks, without using any semi-structured information. To further address the assumption that annotated evidence labels are given during training, we focus on the weakly-supervised setting, with only question-answer pairs available. We propose an iterative approach that improves over a weak retriever by alternately finding evidence from the up-to-date model, and encouraging the model to learn the most likely evidence. Without using any evidence labels, our approach is on par with fully-supervised counterparts.

We also study complex QA using tables as a knowledge source. We focus on a practical problem that is dismissed by benchmarks: domain generalization on mathematical operation over columns. We first construct benchmarks to quantify this problem, then we address this problem by incorporating the necessary domain knowledge through table schema preprocessing. Our approach significantly outperforms baselines on this problem, and as a result, boosts the overall performance.

TOWARDS PRACTICAL COMPLEX QUESTION ANSWERING

by

Chen Zhao

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Jordan Boyd-Graber, Chair/Advisor
Professor Hal Daumé III, Co-Advisor
Professor Philip Resnik, Dean's Representative
Professor Abhinav Srivastava, Member
Professor Hannaneh Hajishirzi, Member

Acknowledgments

I am very fortunate to have got the opportunity to pursue my PhD at UMD. Throughout this wonderful journey, many people have helped in both research and life. This dissertation would not have been possible without them.

First I would like to thank my advisors, Jordan Boyd-Graber and Hal Daumé III, for their guidance and advice. They offered me great freedom in selecting research topics, building my research agenda, as well as insightful feedbacks to motivate a research problem, propose solutions, and present the results. Jordan taught me to always think about the reasons beyond numbers; Hal always motivated me to explore new things and encouraged me to talk to experts that are not in my area. Most importantly, through conversations over the years, I have been benefited tremendously to enjoy doing research, and do it in the correct way. I feel I am super fortunate to be advised by both of you, thank you.

Throughtout my PhD, I have been working closely with my collaborators, Chenyan Xiong and Tianze Shi. Chenyan and I worked together on most projects in the dissertation, I learned a lot from his high research standard. Tianze and I collaborated for more than three years, his academic perfectionism improved my research taste; and he is a best listener, encouraged me to be positive about life, especially during the Covid time. Thank you.

I am also thankful to all my committee members for both thesis proposal and defense: David Jacobs, Philip Resnik, Abhinav Srivastava and Hannaneh Hajishirzi, for their insightful comments and questions that helped me improve this dissertation.

Being part of the CLIP community is among the best parts of my PhD. Through CLIP, I met many friends and colleagues. Among many wonderful individuals, I want to special thank: Hua He guided me get into NLP research and PhD life; Weiwei Yang and Xing Niu for all the Wednesday lunchs over two years. Finally, thank you to the CLIP faculty and staff for creating and nurturing an amazing community.

My graduate journey could not be complete without wonderful summer internships. In my early days of research, I was fortunate to be mentored by Yeye He at Microsoft Research, who was outstandingly cooperative and patient with me. I was also fortunate to work with amazing people at Microsoft Semantic Machines, including Anthony Platanios, Yu Su, Adam Pauls and Jason Eisner. Brainstorming new ideas with you is always a pleasure.

Bridge is part of my life. I'm grateful to play many national events during my PhD, and learn from world-class players. I want to first thank my partner and mentor, Jing Liu, who has taught me tremendous skills and is always patient with my silly mistakes. I want to thank my teammates Hua Poon, Choon Chou Loo, Linlin Hu and Dunga Liu. Finally I want to thank Simon de Wijs and Bauke Muller for practicing and playing online tournaments with me during Covid time.

Last but not the least, I would like to thank my family. My parents and grandparents have been supporting me wholeheartedly for my pursuit of dreams, words are not enough to express my gratitude for them. Finally, I would like to thank my wife Xin Qian who supported me and continues to support me to accomplish my dreams and goals. Thank you all.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Question Answering (QA) is an important task in natural language processing (NLP) and artificial intelligence (AI). The ultimate goal of QA is to build a computer system that automatically comprehends the question and responds to the answer. On one hand, QA evaluates humans language understanding capabilities of AI systems such as Turing test for a long time [Lehnert, 1977b]; On the other hand, in many commercial systems like search engines and chatbots, QA is an essential functionality component. Therefore, improving QA systems contributes to both academia and industry.

With the rapid development of neural network based models for sequence modeling [Graves, 2012, Kim, 2014, Vaswani et al., 2017], tremendous success have been made on building QA systems, along with large banchmarks such as SQuAD [Rajpurkar et al., 2016] to evaluate them. When pre-training on large corpora such as Wikipedia and then fine-tuning on the benchmarks, large transformer based language models (e.g., BERT [Devlin et al., 2019]) outperform the estimated crowdworker accuracy on benchmarks. And such progress has been deployed into commercial search engines (e.g., Google, Bing), as they directly output the answers to the user questions with high confidence.

However, QA systems are still far from perfect. The questions in the benchmark datasets are rather simple, which can be answered within a single evidence piece [Min et al., 2018]. In many real-world scenarios, people also ask complex questions that require combining multiple evidence pieces to find the answers. Existing search engines fail to answer them. Can we build QA systems to answer these questions in the benchmarks, and further deploy our model into real commercial systems? In this dissertation, we present several approaches to tackle this problem from different angles.

1.1 Simple Quetsions v.s. Complex Questions

Question answering can take different forms. We focus on factoid questions, in contrast to open-ended questions that require complex answers (e.g., questions start with "how" and "why"), factoid questions often ask about stated facts, with only a few words as the answers. To answer factoid questions, QA systems need to search the evidence from knowledge source, which can be a structured knowledge graph like FREEBASE [Bollacker et al., 2008] and DBPEDIA [Mendes et al., 2011], a semi-structured table, or unstructured textual documents. We further define *Simple Questons* and *Complex Questions* by its reasoning complexity:

– **Simple questions** require single step reasoning, where the goal of QA systems is to find and extract the evidence directly available in the knowledge source, which can be a node in a knowledge graph, a cell in a table, or a span in a passage. As shown at the top of Figure 1.1, with the help of large trans-

former based language models, search engines can directly output the answer Menlo Park to such questions "In which city is Facebook located?"

– **Complex questions**, by definition, require multiple reasoning steps. These questions often include multi-sentence questions from trivia games, multi-hop questions, highly compositional questions, etc. Compared with simple questions, these questions are more challenging, and existing search engines often fail to find the answers. As an example, at the bottom of Figure 1.1, the question "In which city was Facebook launched?" requires *multi-hop* information to answer. QA systems need to first find *Facebook is launched in Harvard*, then *Harvard university is in Cambridge*, and select the answer as Cambridge.

In this dissertation, we tackle complex question answering problem under two types of knowledge sources, text and tables.

1.2 Open-domain Complex Question Answering over Text

Compared with a structured knowledge graph, using large text collections as a knowledge source has significantly higher coverage, since a known issue of knowledge graphs is incompleteness (i.e., it's impossible to store all world knowledge into a structured form). Therefore QA systems from the text can potentially answer any human posted factoid questions as long as the evidence is available in the textual documents.

Question answering from text (also referred as the open-domain question answering) has a long history in the research community. Most systems follow a

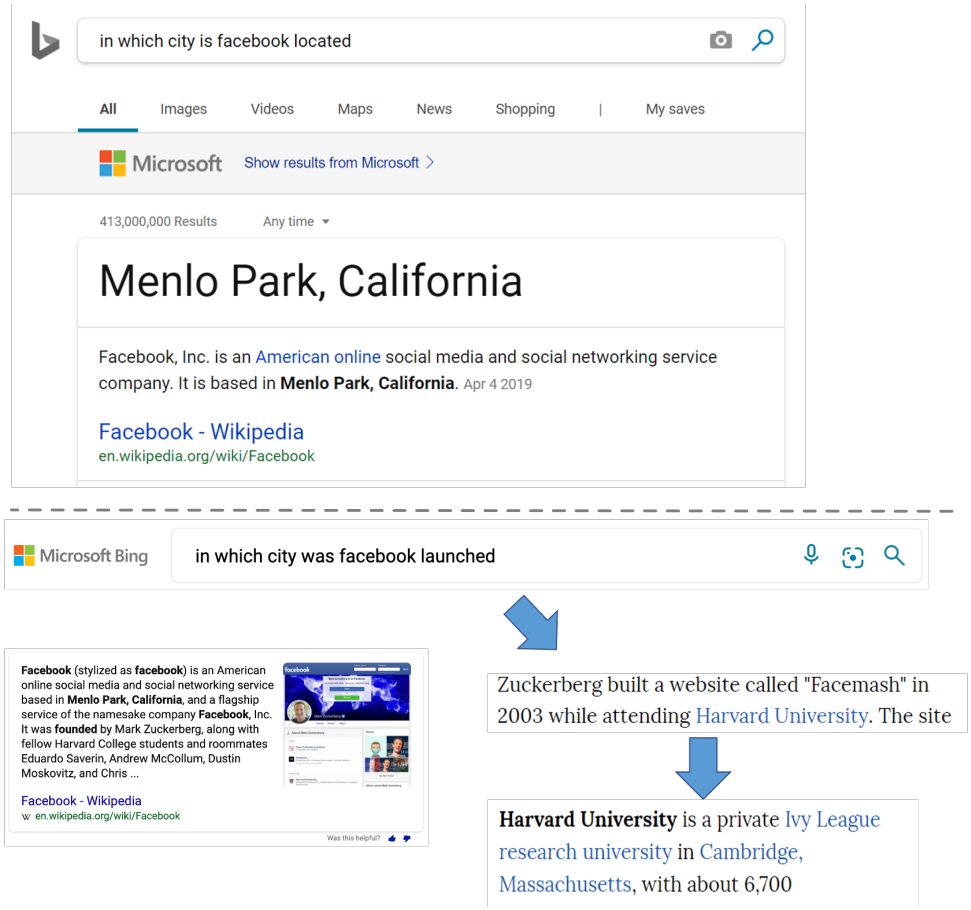


Figure 1.1: Top: An example simple question that search engine answers it correctly. Bottom: An example complex question that requires multiple reasoning steps, search engines fail to answer it.

two-step approach [Voorhees et al., 1999]: An information retrieval model finds a few passages that are most likely to have the answer, and another statistical model selects an answer from these passages, which is also referred as the reading comprehension (RC) task. Following these early systems, Chen et al. [2017] proposed the first open-domain question answering system focused on simple questions, using state-of-the-art neural network as the reading comprehension task.

This thesis follows the retriever-reader framework to answer complex questions. Unlike simple questions, most complex questions are entity-centric, linguistically rich, and include multiple or multi-hop clues to the answer. These complex questions create several challenges: First, QA systems need to extract multiple evidence pieces from different places to satisfy complex information needs. Then, QA systems should synthesize the extracted information for answer prediction, which can be aggregating (multi-evidence) or chaining (multi-hop) multiple evidence pieces.

We propose two QA systems to tackle complex questions. They rely on a free-text knowledge graph from Wikipedia, with entities (i.e., Wikipedia page titles) as nodes and hyperlinks between entities as edges. Then we map each question into a dense and high coverage evidence graph. A graph neural network finally combines evidence pieces to select an answer. The proposed QA systems significantly outperforms previous approaches in multiple complex QA benchmarks.

However, the proposed strong QA systems make several key assumptions that only exist in academic testbeds:

- The free-text knowledge graph is based on Wikipedia, which contains semi-structured information (i.e., the entities are connected by hyperlinks), albeit many real-world corpora (e.g., medical records) often lack this structure.
- In addition to question-answer pairs, evidence labels are also available for training, which are expensive to annotate, and create dataset artifacts.

We further propose two approaches to address these limitations. First, we propose a multi-step dense retrieval method to model the implicit relationships between

evidence pieces. The retriever is competitive to state-of-the-arts on complex QA benchmarks, without using any semi-structured information. To further address the second limitation, we focus on weakly-supervised setting, with only question-answer pairs available. Our proposed approach iteratively improves over a weak retriever, and finally is on par with fully-supervised methods. This means that we do not need evidence labels to accurately find evidence to answer complex questions, a surprising and important result that shows large potential to extend to any type of complex questions that do not come with evidence annotations.

1.3 Complex Question Answering over Tabular Data

Next we switch our attention to answering highly compositional questions using tables. Tables provide useful semi-structured information, and question answering over tables is often framed as a semantic parsing problem: generate a program that represents the meaning of the question and can execute on the table to output an answer. We find that state-of-the-art approaches struggle to answer complex questions that require complex operations over table cells. As shown in Figure 1.2, to generate correct programs, we need to map domain specific phrases (e.g., `how long`) to program segments that operate over multiple table elements(e.g., `Term.2 - Term.1`). The problem is even more challenging under the domain generalization setting, where these mappings are not seen during training. We first construct benchmarks to quantify this problem, then we propose to incorporate the necessary domain knowledge through preprocessing table schemas. Our proposed approach

Name	Term
Pier Piccio	1926-1927
Armando Armani	1927-1928
Giuseppe Valle	1930-1933

Question: **When** did Armando Armani's **term start**?

Program: `SELECT "Term"._1 FROM t WHERE "Name" = 'Armando Armani'`

Question: **How long** was Armando Armani's **term**?

Program: `SELECT "Term"._2 - "Term"._1 FROM t WHERE "Name" = 'Armando Armani'`

Figure 1.2: Two complex questions that require mapping domain specific phrases to operations over table columns.

significantly outperforms baselines on this problem.

1.4 Contributions

We make several contributions on complex question answering:

– **Complex QA with a Free-Text Knowledge Graph.** We propose both models and data to study open-domain complex question answering over text. We construct a human-authored, entity centric dataset QBLink from trivia games. Then we propose a QA system to tackle such trivia-style complex questions by first constructing an free-text knowledge graph over Wikipedia, and then find evidence from the graph to select the answer. We further extend this idea by proposing a neural model to model this semi-structured knowledge graph, with strong results on multi-hop QA benchmark and is generalizable to

other complex reasoning tasks.

– **Complex QA Systems In More Real Scenarios.** We first remove the assumption that the given text collections is semi-structured by hyperlinks. We propose a new multi-step retrieval approach that is competitive to state-of-the-art systems on multi-hop QA benchmarks, without using the hyperlinks. We next focus on training a multi-step retrieval model without using gold evidence labels (i.e., weakly-supervised setting). Our method is on-par with fully-supervised counterparts on multiple benchmarks. By removing these two assumptions, our proposed QA system can be further adapted to potentially answer any complex questions from users.

– **Complex QA Over Tables.** Lastly, we study complex question answering over tables. We focus on a problem that is important in real scenarios, but dismissed by benchmarks: domain generalization on math operation over columns. We first propose benchmarks to quantify this problem, then we propose to incorporate the necessary domain knowledge through table schema preprocessing. Our proposed method significantly outperforms baselines on this problem, and as a result boosting the overall performance.

1.5 Roadmap

In Chapter 2, we first review deep learning and NLP preliminaries used in this dissertation. Then we introduce history of question answering, and question answering over different knowledge sources. Finally we discuss question answering

over two knowledge sources in the thesis, text collections and tables.

In Chapter 3, we study one type of complex questions—questions from trivia games. These questions have multiple sentences, including diverse clues. We first build a human-authored dataset QBLINK from real Quizbowl competitions. The question is a multi-sentence description of the answer entity, in which each sentence includes diverse relations pointing to the answer. Then we introduce a QA system DELFT, which combines the advantages of knowledge graph question answering (KGQA) and RC approaches. DELFT inherits the structure reasoning ability from KGQA with high coverage free-text. It first extracts multiple evidence from Wikipedia on different sup-parts of the question and builds an evidence graph, then a novel graph neural network models on the constructed graph and selects the answer by combining multiple evidence pieces. DELFT outperforms reading comprehension based models, BERT-based answer ranking and BERT memory networks with big margins.

In Chapter 4, we focus on a more general setting for complex QA: the answer is not limited to entities as in previous chapter, but any text span; the evidence retrieval is not only single-hop, but extends to multi-hops. We introduce TRANSFORMER-XH, which upgrades transformer for modeling semi-structured text, in which different text sequence connects with each other. It includes an extra hop attention in its layers that aggregates representations from different textual pieces following their structure while maintaining the powerful pre-trained transformer abilities over each text sequence individually. On multi-hop QA task, TRANSFORMER-XH outperforms previous state-of-the-art systems that conduct sev-

eral single-hop reading comprehension steps to simulate the multi-hop reasoning in order to adopt the pre-train Transformers. We further apply TRANSFORMER-XH to multi-evidence fact verification task, the results are competitive to the state-of-the-art, showing TRANSFORMER-XH is generalizable to multiple tasks.

In Chapter 5, we focus on removing the first assumptions for state-of-the-art complex QA systems: corpus is semi-structured by hyperlinks. Building on dense retrieval methods, we propose a new multi-step retrieval approach (BeamDR) that iteratively forms an evidence chain through beam search in dense representations. When evaluated on multi-hop question answering, BeamDR is competitive to state-of-the-art systems, without using any semi-structured information. Through query composition in dense space, BeamDR captures the implicit relationships between evidence in the reasoning chain.

In Chapter 6, we focus on removing another assumption: evidence labels during training. State-of-the-art neural approaches require intermediate evidence annotations for training. Such intermediate annotations are expensive, and methods that rely on them cannot transfer to the more common setting, where only question–answer pairs are available. We investigate whether models can learn to find evidence from a large corpus, with only distant supervision from answer labels for model training, thereby generating no additional annotation cost. We introduce a novel approach (DistDR) that iteratively improves over a weak retriever by alternately finding evidence from the up-to-date model and encouraging the model to learn the most likely evidence. Without using any evidence labels, DistDR is on par with fully-supervised state-of-the-art methods on both multi-hop and single-hop QA

benchmarks. Our analysis confirms that DistDR finds more accurate evidence over iterations, which leads to model improvements.

In Chapter 7, we study answering complex questions using tables as a knowledge source. We focus on a specific out-of-domain generalization problem that is of significant practical importance: matching domain specific phrases to composite operation over columns. To study this problem, we first propose a synthetic dataset along with a re-purposed train/test split of the SQUALL dataset as new benchmarks to quantify domain generalization over column operations, and find existing state-of-the-art parsers struggle in these benchmarks. We then propose to address this problem by incorporating prior domain knowledge through preprocessing table schemas, and design a method that consists of two components: schema expansion and schema pruning, which can be easily applied to different base parsers. We show that on this domain generalization over column operations problem, our proposed method significantly outperforms baseline parsers, and as a result boosting the underlying parsers' overall performance.

In Chapter 8, we summarize the results presented in this thesis and propose future directions.

Chapter 2: Background

In this chapter, we first review deep learning and NLP techniques used in this thesis (Section 2.1). Then we introduce the history of question answering (Section 2.2), and question answering over knowledge graph (Section 2.3). Next we discuss question answering over two knowledge sources in the thesis, text collections (Section 2.4) and tables (Section 2.5).

2.1 Deep Learning and NLP Preliminaries

In this section, we review some deep learning and NLP techniques used in this thesis.

2.1.1 Transformer

The Transformer model proposed by Vaswani et al. [2017] has become the new standard solution in many NLP tasks. Instead of using recurrent neural network (RNN) variants to model text sequence, it involves multiple stacked self-attention layers that converts the input X into $\{H^0, H^1, ..., H^l, ...H^L\}$, starting from H^0, the embeddings (i.e., low-dimensional vectors that represent the meaning of the input), to the final layer of depth L. The key idea of Transformer is its attention mechanism,

which calculates the l-th layer output H^l using the input H^{l-1} from the previous layer:

$$H^l = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V^T, \tag{2.1}$$

$$Q^T; K^T; V^T = W^q \cdot H^{l-1}; W^k \cdot H^{l-1}; W^v \cdot H^{l-1}. \tag{2.2}$$

It includes three projections on the input H^{l-1}: Query (Q), Key (K), and Value (V) (Figure 2.1). Specifically, the slices of token h_i^l is:

$$h_i^l = \sum_j \text{softmax}_j\left(\frac{q_i^T \cdot k_j}{\sqrt{d_k}}\right) \cdot v_j, \tag{2.3}$$

which first calculates its attention to all other tokens j in the sequence and then combines the token values v_j into a new representation h_i^l, using the normalized attention weights. Multiple attentions can be used in one Transformer layer and concatenated as multi-head attention.

For sentence modeling (encoding), the query, key and value are the same (embeddings of input x), which is noted as self-attention. For sequence generation, the query is from the previous output, and the key and values are the outputs of the input representations.

2.1.2 Transformer based Language Model Pre-training

The Transformer based models pre-trained on large scale text collections like Wikipedia have been the new standard solution for multiple NLP tasks. BERT [Devlin et al., 2019, Bidirectional Encoder Representations from Transformers] is the representative. Unlike fixed word [Pennington et al., 2014] or sentence [Conneau

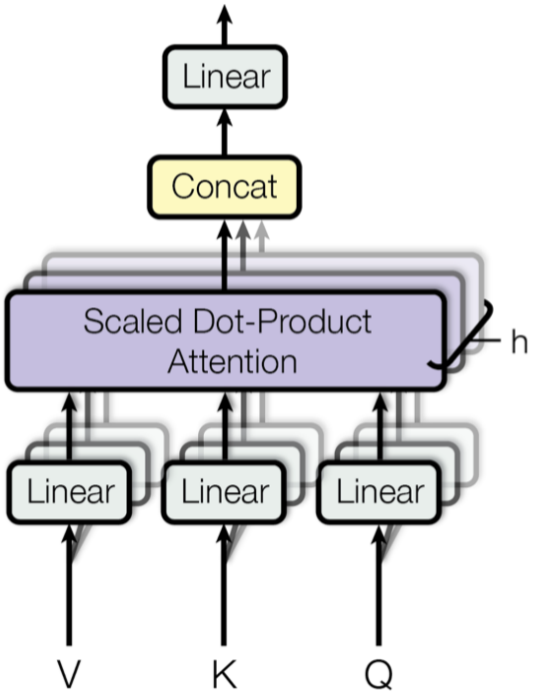


Figure 2.1: Multi-head attention in Transformer.

et al., 2017] embeddings, the BERT embedding is contextualized, i.e., it is *context-sensitive* that the embeddings are extracted dynamtically from a left-to-right and a right-to-left language model. Figure 2.2 shows the model architecture of BERT. BERT is based on Transformer, the original paper provides two model settings: BERT base has 12 layers, with hidden state (H) size of 768 and 12 heads; BERT large has 24 layers, with hidden state size of 1024 and 16 heads.

Input/Output. The input to BERT is a text sequence, which can be a single sentence (not just a linguistic sentence but arbitrary of span text) or a pair of sentences (e.g., question–passage pair). The first token of the text sequence is a special [CLS] token. For pair of sentences, another special token [SEP] is used

to separate the two sentences. Then, a learned embedding to every input token is included to indicate if it belongs to sentence A or B. The BERT model outputs a contextualized representation for each token.

Pre-training. The goal of language model pre-training is to learn the knowledge in large corpora, and later can be used to initialize the model for new tasks. Pre-training BERT is on two corpus: BooksCorpus and English Wikipedia. And two tasks are used for pre-training:

1. Masked language model: To train a bidirectional representation, some of the input tokens (about 15%) are masked at random, and then the model's job is to predict those tokens. Similar to the standard language modeling, the final hidden vectors corresponding to the mask tokens are fed into a softmax function over the vocabulary. Through this task, the model automatically looks at both directions of the input.

2. Next Sentence Prediction: To train a model that understands the sentence relationships, a binary classification task over representations that predicts whether one sentence follows the other sentence given a pair of sentence as inputs. During training, 50% of the time two sentences connect to each other, and the other 50% not.

Fine-tuning The goal is fine-tuning is to use the pre-trained model parameters as the initialization, and adapt into the new tasks that we care about. Since BERT has pre-trained on large corpora, we hope most of the knowledge useful for the down-

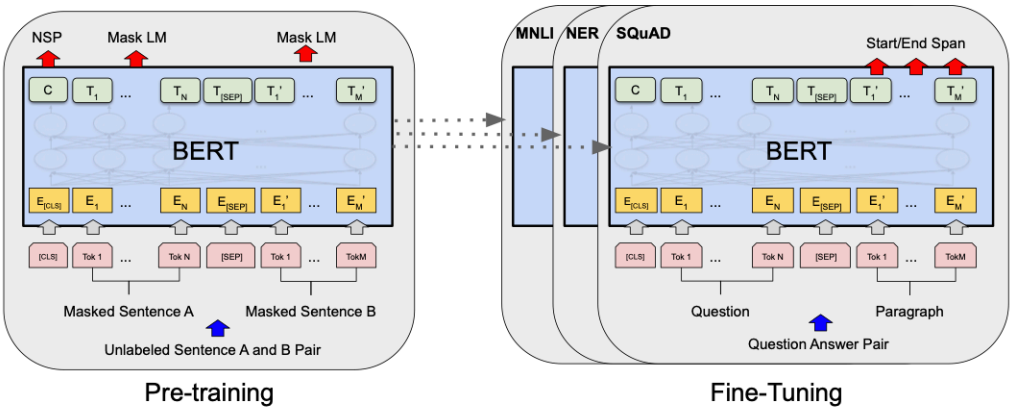


Figure 2.2: Model architecture for BERT [Devlin et al., 2019].

stream tasks has been stored in its parameters. BERT can be used in multiple NLP tasks including single sequence input such as sentiment classification, and sequence pair input such as natural language inference and reading comprehension. During fine-tuning, after K pre-trained Transformer layers, BERT outputs the contextualized representations for every input token. Only the last task specific layer depends on the actual task.

Other Pre-training Models. The propose of BERT model creates a new era for deep learning in NLP due to its strong performance on multiple benchmarks. After BERT, many other pre-training approaches are proposed including XLNET [Yang et al., 2019], MT-DNN [Liu et al., 2019a], ROBERTA [Liu et al., 2019b] etc. These models use more advanced training strategies, and with much larger pre-training corpus and model size, they further improve upon BERT on benchmarks.

2.1.3 Graph Neural Network for NLP

Graph neural networks [Zhou et al., 2018, GNN] have been widely used in multiple NLP tasks such as relation extraction [Zhang et al., 2018], semantic role labeling [Marcheggiani and Titov, 2017] , question answering [De Cao et al., 2019]. GNN follows the standard gather-apply-scatter mechanism [Scarselli et al., 2008] to learn the graph node representations with homogeneous updates. In this section, we discuss two main variants of GNN, graph convolutional networks [Kipf and Welling, 2017, GCN] and graph attention networks [Veličković et al., 2017, GAT].

Graph Convolutional Networks Figure 2.3 shows the GCN model architecure.[1] Similar to convolutional neural networks, which aggregates the neighbor pixels (or words) in the convolutional layer to get the local representations, for each node in the graph, GCN averages the neighbor nodes' representation to aggregate the local information.

GCN takes a graph as input, it first (layer $l = 0$) initializes the node representation $\mathbf{h}_i^{(0)}$. Then each next layer $l + 1$ updates the node representations as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \frac{1}{c_{ij}} \mathbf{W}\mathbf{h}_j^{(l)}\right), \tag{2.4}$$

Where $N(i)$ is a set of node i neigibors, c_{ij} is a normalization constant, and $\mathbf{W}$ is a learnable shared parameter. The last layer's node representation $\mathbf{h}^{(L)}$ is used for downstream tasks.

[1] https://tkipf.github.io/graph-convolutional-networks

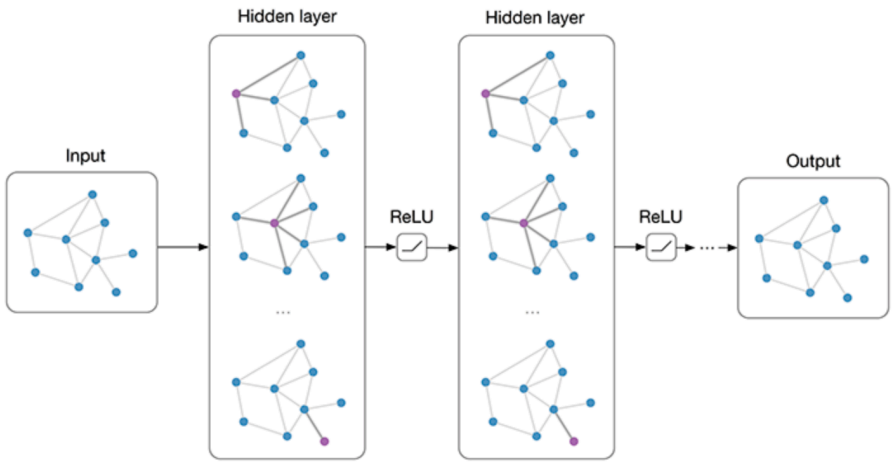


Figure 2.3: Multi-layer Graph Convolution network (GCN). At each layer, the node's hidden representation is conducted by the convolution operation over all neighbors' representations.

Graph Attention Network. Graph Attention Networks (GAT) introduces an attention mechanism as a substitute for the convolution function. Similar as the attention mechanism, the weighted representations used by GAT let the models able to focus on the most useful neighbors. Figure 2.4 shows the GAT attention mechanism. [2] At layer l, we first conduct a linear transofmeration from $h_i^{(l)}$ to $z_i^{(l)}$,

$$\mathbf{z}_i^{(l)} = \mathbf{W}\mathbf{h}_i^{(l)}, \tag{2.5}$$

where $\mathbf{W}$ is a model parameter. Then we compute the pariwise attention score between two neigibor nodes as

$$e_{ij}^{(l)} = LeakyReLU(\mathbf{W}_a([\mathbf{z}_i^{(l)}; \mathbf{z}_j^{(l)})]). \tag{2.6}$$

[2] `https://docs.dgl.ai/tutorials/models/1_gnn/9_gat.html`

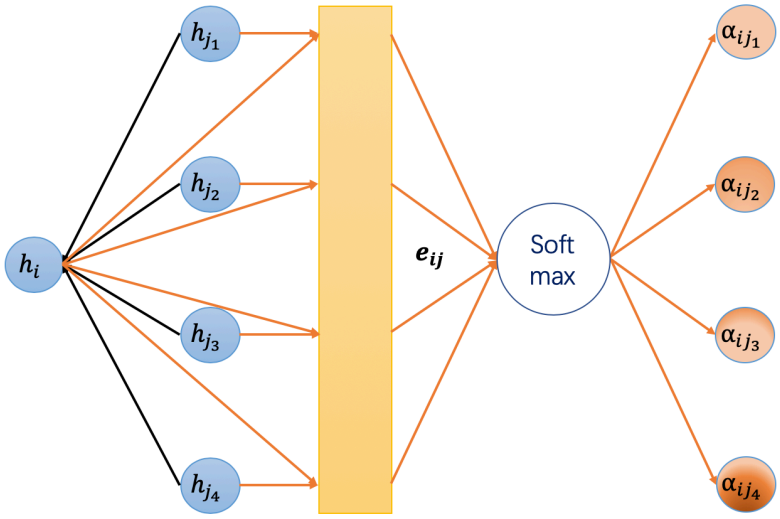


Figure 2.4: GAT attention mechanism.

It first concatenates representations for node i and j, then passes through a linear transformation layer with parameter $\mathbf{W}_a$, and applies a LeakyReLU [He et al., 2015] in the end. Then we apply a softmax function to normalize the attention scores as

$$\alpha_{ij}^{(l)} = \frac{exp(e_{ij}^{(l)})}{\sum_{k \in N(i)} exp(e_{ik}^{(l)})}. \tag{2.7}$$

Finally, the representations of neighbors are combined, weighted by the attention scores,

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in N(i)} \alpha_{ij}^{(l)} \mathbf{z}_j^{(l)} \right). \tag{2.8}$$

2.2 Question Answering Definition

The ultimate goal of question answering is to have a computer system that takes the posted question as input and produces an output answer. Rather than just a specific task, QA is usually viewed as a format for humans to satisfy information

needs [Gardner et al., 2019]. Question answering can take different forms. From the types of answers, QA is divided into multiple choice QA, extractive QA, and free-from QA. Multiple choice QA provides a few candidate answers; the answers in extractive QA are spans in the context; and the answers in free-form QA can be any text sequence. In terms of question types, QA is categorized into factoid QA, in which the questions ask about stated facts, and non-factoid QA, which usually start with "how" and "why", and require complex answers. Depending on the number of reasoning types, QA is divided into simple QA that requires only a single reasoning step, and complex QA that needs multiple reasoning steps.

2.3 Question Answering over Knowledge Graph

To answer questions, QA systems need knowledge sources. Knowledge Graph (KG) is a structured knowledge source with well-defined schema, where the entities are the nodes, and the relations between entities are the edges. Many large knowledge graph has been proposed like FREEBASE [Bollacker et al., 2008], DBPEDIA [Mendes et al., 2011], WIKIDATA [Vrandečić and Krötzsch, 2014], YAGO [Suchanek et al., 2007]. Knowledge-graph question answering (KGQA) is to use the knowledge graph to answer questions about facts. Many datasets like WEBQUESTIONS [Berant et al., 2013] and SIMPLEQUESTIONS [Bordes et al., 2015] are constructued for KGQA along with new QA systems which conduct structual reasoning on the facts in the KG to find the answer. A typical KGQA task is to search for entities or entity attributes from a KG. For example, given the question "who was the last emperor of China",

the model identifies emperors of China and then reasons to determine which emperor is the last and find the answer, Puyi. KGQA approaches solve the problem by either parsing questions into executable logical form [Cai and Yates, 2013, Kwiatkowski et al., 2013] or aligning questions to sub-structures of a knowledge graph [Yao and Van Durme, 2014, Yih et al., 2015]. However, a known issue for knowledge graphs is that they are incomplete, as it is impossible to extract every world knowledge into the fixed schema. This problem is more nuanced for complex questions, as the clues from questions are usually more abstract, making knowledge graph question answering even more brittle.

Compared with highly structured KGs, unstructured text collections such as Wikipedia, newswire, or Web scrapes are cheaper but noiser for QA, we detail QA over text in Section 2.4. Another commonly used knowledge source is tables, which includes semi-structured information, specifically for highly compositional questions. We introduce QA over tables in Section 2.5.

2.4 Question Answering over Text

We start with introducing early systems, then we discuss the open-domain QA framework, finally we present QA benchmarks.

2.4.1 Early Systems

Question answering over text has a long history. The first QA system [Simmons et al., 1964] is among the initial development of AI systems. There are several

early QA and knowledge representation systems: The LUNAR system [Woods and WA, 1977] answers questions about moon rocks and soil based on a database; Lehnert [1977a] built a QA system based on unstructured text data; KL-ONE [Woods and Schmolze, 1992] is a knowledge representation system in the tradition of semantic networks and frames. But these systems are limited to small hand-created scripts, therefore unable to generalize to new domains. The TREC QA competition [Voorhees et al., 1999] is the first to focus on open-domain question answering. Most QA systems follow a two-step approach, which become the de facto standard for open-domain QA systems such as Chen et al. [2017].

2.4.2 Open-domain (Simple) Question Answering

In open-domain QA, given an input question, QA systems aim to predict an answer from large text corpora. Following Chen et al. [2017], modern QA systems take two steps: The first is the retrieval step, which aims to find relevant documents (passages) that contain the answer; The second is the reader step—often multi-tasked with reranking step—that extracts the answer from those retrieval outputs.

2.4.3 Retrieval Models

Term Matching based Retrieval. A straightforward retrieval approach is to use unsupervised sparse vector space models, including TFIDF or BM25. These methods use sparse vectors [Salton, 1968] to compare textual overlaps between questions and documents (passages), and are widely used in multiple information retrieval sys-

tems including search engines. These retrieval systems are good at (exact) match question tokens to passage tokens, but by design, fail to find matchings with synonyms or paraphrases.

Dense Retrieval. Unlike classic retrieval techniques, dense retrieval methods [Karpukhin et al., 2020] match distributed text representations [Bengio et al., 2013] rather than sparse vectors. With encoders (e.g., BERT) to embed query q and passage p into dense vectors $E_Q(q)$ and $E_P(p)$, the relevance score f is computed by a similarity function $sim(\cdot)$ (e.g., dot product) over two vector representations:

$$f(q, p) = sim(E_Q(q), E_P(p)). \tag{2.9}$$

After encoding passage vectors offline, we can efficiently retrieve passage through approximate nearest neighbor search over the maximum inner product with the query, i.e., MIPS [Shrivastava and Li, 2014, Johnson et al., 2017].

Training Dense Retrieval models. The goal of training is to learn embedding functions that differentiate positive (relevant) and negative passages that do not include the answer string. The input is the query q, a positive passage p^+ and m sampled negative passages p_j^-. We update the negative log likelihood (NLL) loss:

$$L(q, p^+, p_1^-, ..., p_m^-) = \frac{e^{f(q,p^+)}}{e^{f(q,p^+)} + \sum_{j=1}^{m} e^{f(q,p_j^-)}}. \tag{2.10}$$

In addition to use local in-batch or term matching negative samples, advanced training strategies select negatives from the whole corpus, which makes training more robust and effective [Guu et al., 2020, Xiong et al., 2021a].

2.4.4 Reader with Pre-trained Language Models

After the retriever, QA systems need a reader, which combines passage selection (reranking) and span extraction. Unlike retrieval, the reader encodes pairwise information between question and passage, thus giving a more accurate (but slower) prediction.

As shown in Figure 2.2, state-of-the-art readers fine-tune large language models such as BERT. With strong capacities, these language model based approaches outperform other neural models such as BiDAF [Seo et al., 2016], and even exceeds the estimated crowdworker accuracy on closed-domain setting (i.e., reading comprehension).

Specifically, the BERT encoder with input formatted as `[CLS] question [SEP]` $\texttt{title}_1$ `[SEP]` $\texttt{passage}_1$ `[SEP] ...` $\texttt{title}_n$ `[SEP]` $\texttt{passage}_n$`[SEP]`, where `[CLS]` and `[SEP]` are special tokens, and each $\texttt{passage}_t$ is an passage p_t. First, we use `[CLS]` token's representation $\mathbf{U}_{\texttt{[CLS]}}$ to estimate the probability that the passage p contains the answer:

$$P(p \mid q) = \text{softmax}(\mathbf{U}_{\texttt{[CLS]}}^{\intercal} \mathbf{w}_{\text{rank}}), \tag{2.11}$$

where $\mathbf{w}_{\text{rank}}$ is a weight vector. Then answers are predicted with a span start and end classifier:

$$P(\text{start} \mid q, p) = \text{softmax}(\mathbf{U}\mathbf{w}_{\text{start}}), \tag{2.12}$$

$$P(\text{end} \mid q, p) = \text{softmax}(\mathbf{U}\mathbf{w}_{\text{end}}), \tag{2.13}$$

where $\mathbf{w}_{\text{start}}$ and $\mathbf{w}_{\text{end}}$ are weights. From the highest-scored passage $\hat{p}$, we select the

answer with the highest span probability $P(\text{start} \mid \hat{p}, q) \times P(\text{end} \mid \hat{p}, q)$. The training objective is the log-likelihood of the positive passage for reranking and maximum marginal likelihood over all spans in the positive passage for span extraction.

2.4.5 Large Scale QA Datasets

A major contributor to the success of recent QA systems is the vast improvement in the availability of large-scale datasets, including both simple QA and complex QA.

Simple QA datasets. Since 2016, multiple large scale datasets are proposed to evaluate reading comprehension tasks (i.e., closed domain QA). Using Wikipedia as the knowledge source, these datasets can evaluate open-domain QA as well. Following Rodriguez and Boyd-Graber [2021], we categorize these datasets into the Manchester Paradigm, to test the knowledge of the evaluated QA system, and the Cranfield Paradigm, for the information-seeking purpose.

For the Manchester Paradigm, the annotators are asked to write questions given passages. Stanford question answering dataset [Rajpurkar et al., 2016, SQuAD] is the first proposed dataset. SQuAD contains more than $100,000$ question-answer pairs written by crowdworkers on 500+ Wikipedia articles. In contrast to previous RC datasets [Richardson et al., 2013, Onishi et al., 2016], which provides a list of answer candidates, SQuAD requires systems to select a *span* in the given passage as the answer.

However, SQuAD is vulnerable to adversarial attacks. Jia and Liang [2017]

shows that neural reading comprehension models with high numbers on SQuAD datasets perform poorly under simple attacks like adding ungrammatical sequences of words, while the human performance does not get affected, and proposes SQuAD 2.0, to include the *no answer* option to the question to further evaluate the model robustness.

For the Cranfield Paradigm, usually the questions already exist, and annotators find passages and answers to these questions. These datasets include TriviaQA [Joshi et al., 2017b], which the questions are from trivia games; MS MARCO [Nguyen et al., 2016], the questions are from bing queries; and NaturalQuestions [Kwiatkowski et al., 2019], the questions are from Google queries.

Complex QA datasets. In addition to large scale simple QA benchmarks, several large scale datasets are proposed for complex QA, these datasets are categoried into the Manchester Paradigm.

HotpotQA [Yang et al., 2018] is a multi-hop QA dataset. It includes 112k crowd-sourced questions designed to require multiple evidence pieces, which are the first paragraphs of Wikipedia pages. It has two types of questions: bridge questions require hopping via an outside entity, and comparison questions compare a property of two entities. There are two settings in HotpotQA. The Distractor setting provides golden evidence paragraphs together with TF-IDF retrieved negatives. This thesis focuses on the FullWiki setting (i.e., open-domain setting), which requires systems to retrieve evidence paragraphs from the entire Wikipedia.

However, there's a gap between HotpotQA setting and real scenarios, mainly

for the following reasons:

- To ensure the questions are multi-hop (since it's non-trivial to write multi-hop questions from scratch), annotators are pre-given two hyperlink connected passages and write a complex question that uses both passages (therefore these two passages are evidence labels). Later work [Min et al., 2019c] mentions that there are shortcuts that some questions only need single evidence piece to find the answers.

- The knowledge source is the first passages of the entire Wikipedia. And Wikipedia contains semi-structured hyperlinks, an important metadata for answering complex questions. However, not all text corpora has such semi-structured information. For example, the Web corpus, or some in-domain corpus like medical records are unstructured.

This thesis presents soluations to mitivate these limitations, to close the gap between benchmarks and real scenarios.

QANTA [Iyyer et al., 2014] is another complex QA dataset collected from Quizbowl competitions. Each question is a sequence of sentences providing increased information about the answer entity. Often the first few sentences require multiple reasoning steps to find the answer (and therefore are complex questions). Since there's no evidence annotation, state-of-the-art QA systems often struggle to answer them.

Other datasets focus on different complex reasonings, but mostly on closed-domain setting. DROP [Dua et al., 2019] focuses on numberical reasoning, and

QUOREF [Dasigi et al., 2019] focuses on conferences.

2.5 Question Answering over Tables

Tables also include essential information to answer questions. Question answering over tables is often framed as a semantic parsing problem: to translate pairs of natural language questions and tables to executable SQL queries, also known as text-to-SQL parsing [Androutsopoulos et al., 1995, Minock et al., 2008]. Formally, our goal is to map a pair (q, T), where q is a natural language question and T is a table, to an executable program π that, when executed against table T, will produce the answer α to question q. We focus on the fully-supervised setting where the target executable program π^* is provided as supervision for training our parser.

Evaluation Benchmarks. Text-to-SQL parsing became popular after the introduction of large-scale datasets and evaluation benchmarks. Zhong et al. [2017] first introduced WIKISQL, which contains Wikipedia tables paired with questions and annotated with SQL queries, albeit the queries are generated from a limited set of templates. SPIDER was introduced by Yu et al. [2018] the following year. It contains more complex questions and SQL queries and focuses on generalizing to previously unseen database schemas, but the dataset has the artifact from its annotation design that the references columns are often mentioned verbatim in the natural language questions. Deng et al. [2021] attempt to address this limitation by repartitioning SPIDER to produce a more realistic benchmark, and Lee et al. [2021] propose a challenging test set from Kaggle for evaluating parsers trained on SPIDER

dataset. Recently, Shi et al. [2020] introduced SQUALL, a dataset that annotates WIKITABLEQUESTIONS [Pasupat and Liang, 2015] with SQL queries and refined column types like `Date`, `Score`, `(T1, T2)`, and `List[T]`.

Models. Neural encoder-decoder models have recently gained popularity for text-to-SQL parsing [e.g., Xu et al., 2017]. We focus on two models that represent the current state-of-the-art for SQUALL and SPIDER, respectively: SEQ2SEQ of Shi et al. [2020] and SMBOP of Rubin and Berant [2021].

Both models concatenate the question with a textual representation of the table schema, separated by a special `[SEP]` token, and feed the combined sequence to a pre-trained instance of the BERT [Devlin et al., 2019] language model. The activations of the last layer represent the encoded representations of the question and the table schema. SEQ2SEQ then uses a *autoregressive decoder*, which represents programs as token sequences and at each decoding step it: (1) predicts the next token type (i.e., whether the next token is a SQL keyword, a column name, or a literal value), and (2) predicts the token conditioned on its type. SMBOP, on the other hand, uses *bottom-up decoding*, which represents programs as abstract syntax trees and constructs these trees in a bottom-up fashion (i.e., it starts by predicting the leave nodes and then recursively composes generated sub-trees into new trees and ranks them, in a way that resembles beam search), until it reaches the tree root.

Chapter 3: Multi-Evidence QA with a Free-Text Knowledge Graph

In this chapter, we mainly study questions from trivia games. As categorized in the Manchester paradigm [Rodriguez and Boyd-Graber, 2021], these competitions test human knowledge and evaluate which answerers (it's either QA systems or humans) are smarter (i.e., winner of the competitions). To serve this goal, the questions used in such competitions are complex, linguistically rich, and include multiple clues to the answer. Therefore, these questions are naturally occurring testbeds for complex QA systems. We first construct a new dataset QBLINK from Quizbowl competitions with about 56000 human authored questions. Then we build a QA system DELFT which builds a free-text knowledge graph from Wikipedia as the knowledge source, with entities as nodes and sentences in which entities co-occur as edges. DELFT grounds each question into a subgraph linking question entity nodes to candidates using text sentences as edges, creating a dense and high coverage semantic graph. A novel graph neural network reasons over the free-text graph—combining evidence on the nodes via information along with edge sentences—to select a final answer. DELFT significantly outperforms previous approaches in trivia style QA tasks.[1]

[1]This Chapter describes work published in EMNLP 2018 [Elgohary et al., 2018] and WWW 2020 [Zhao et al., 2020a].

3.1 Introduction

Factoid question answering [Wang, 2006, Iyyer et al., 2014, Gardner et al., 2019, QA], which asks precise facts about entities, is a long-standing problem in natural language processing (NLP) and information retrieval (IR). A preponderance of knowledge graphs (KG) have been fruitfully applied to factoid QA over knowledge graphs (Section 2.3). Whether these approaches work depends on the underlying KG: they are effective when the knowledge graphs have high coverage on the relations targeted by the questions, which is often not the case: Obtaining relation data is costly and requires expert knowledge to design the relation schema and type systems [Paulheim, 2018]. For example, Google's Knowledge Vault has 570M entities, fourteen times more than FREEBASE, but only has 35K relation types: similar to FREEBASE [Dong et al., 2014, Bollacker et al., 2008].

In the real world, human knowledge is often tested through competitions like Quizbowl [Boyd-Graber et al., 2012] or *Jeopardy!* [Ferrucci et al., 2010]. Most of the questions in these settings contain complex and diversed relations, which are unlikely to be covered by the well-formatted closed form relations in knowledge graphs, and are often phrased obliquely, making existing KGQA methods brittle. To study these questions, we introduce QBLINK, a new dataset of about 56,000 human authored questions. Figure 3.1 shows an example from the QBLINK dataset. The question includes multiple relations that can point to the answer, the relations in the question are complex and defy categorization into KG relationships. For example, the relation "depict"—much less "paint a cityscape of"—from the first

question sentence "Vermeer painted a series of cityscapes of this Dutch city" is not a frequent KG relation.

One possible solution to sparse coverage is reading comprehension (RC), which finds an answer directly from unstructured text [Chen, 2018]. Because it extracts answers from paragraphs and documents—pre-given or retrieved [Rajpurkar et al., 2016, Clark and Gardner, 2018, Zhong et al., 2019]—RC has much broader coverage [Nguyen et al., 2016]. However, current RC datasets mainly evaluate on reasoning within a single paragraph [Min et al., 2018]; existing RC models focus on extracting from single evidence. Synthesizing multiple pieces of evidence to answer complex questions remains an on-going research topic [Yang et al., 2018, Min et al., 2019b].

To build a QA system that can answer real-world, complex factoid questions using unstructured text, we propose DELFT: **D**eciphering **E**ntity **L**inks from **F**ree **T**ext. DELFT constructs a free-text knowledge graph from Wikipedia, with entities (Wikipedia page titles) as nodes, and—instead of depending on pre-defined relations—DELFT uses free-text sentences as edges. This subgraph are the nodes relevant to the question, grounded to text by connecting question entities to candidate answer entities with extracted free-text sentences as evidence edges. The constructed graph supports complex modeling over its graph structures and also benefits from the high coverage of free-text evidence.

Unlike existing knowledge graphs, in which the relations between entities are well-defined, DELFT leverages informative, diverse, but noisy relations using a novel graph neural network (GNN) (Chapter 2.1.3). Each edge in our graph is associated with multiple sentences that give explain how entities interact with each other.

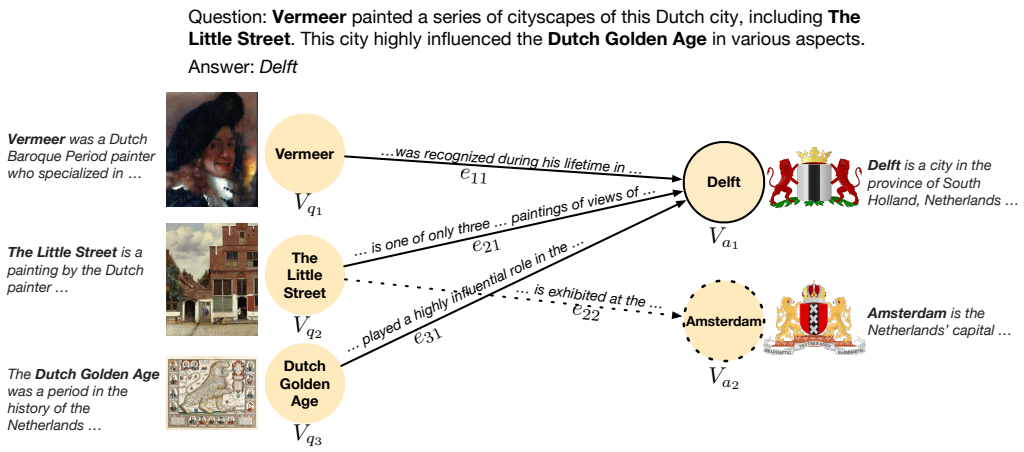


Figure 3.1: An example question grounded free-text knowledge graph in DELFT. The graph has both question entity nodes (left side) and candidate entity nodes (right side), and use natural language sentences from Wikipedia for both nodes and edges.

DELFT uses the GNN to distinguish the useful evidence information from unrelated text and aggregates multiple evidence from both text and the graph structure to answer the question.

We evaluate DELFT on three question answering datasets: QBLINK, QANTA, and TRIVIAQA. DELFT outperforms reading comprehension based model [Chen et al., 2017], BERT-based answer ranking [Devlin et al., 2019], and a BERT-based memory network approach [Weston et al., 2014] that uses the same evidence but no graph structure, with significant margins. Its accuracy improves on more complex questions and when dense evidence is available, while MR cannot take advantage of additional evidence.

Ablation reveals the importance of each model component: node representa-

tion, edge representation, and evidence combination. Our graph visualization and case studies further illustrate that our model could aggregate multiple pieces of evidence to make a more reliable prediction.

3.2 Dataset Construction

This section describes QBLINK's construction. QBLINK is based on the *bonus questions* of **Q**uiz **B**owl tournaments. Unlike previous work that only uses the starter (or tossup) questions [Boyd-Graber et al., 2012], bonus questions are not interruptable (players always hear the complete question) and have greater variability in difficulty. Bonus questions start with a lead-in, which sets the stage for the rest of the question, followed by a sequence of related questions (Figure 3.2).

Specifically, we collect bonus questions from `http://quizdb.org` for the tournaments in 2008–2018. Each question is categorized by topic as history, literature, science, geography, fine arts, philosophy, religion, mythology, social sciences, current events or current events. We filter out too short questions (fewer than ten tokens), and only keep questions with exactly three sub-questions. One advantage of working with QB data is that the community emphasizes sharing and redistribution of old questions: new students can practice and improve without paying for or licensing questions.

We map the answers to unambiguous Wikipedia pages using combination of rule based matching and fuzzy string matching, then filter out the questions whose answers are not mapped to any Wikipedia page (12.5% of the questions).

Lead-in: Only twenty-one million units in this system will ever be created. For 10 points each:

Question 1: Name this digital payment system whose transactions are recorded on a "block chain".

Answer: Bitcoin

Question 2: Bitcoin was invented by this person, who, according to a dubious Newsweek cover story, is a 64-year-old Japanese-American man who lives in California.

Answer: Satoshi Nakamoto

Question 3: This online drugs marketplace, Chris Borglum's one-time favorite, used bitcoins to conduct all of its transactions. It was started in 2011 by Ross Ulbricht using the pseudonym Dread Pirate Roberts.

Answer: Silk Road

Figure 3.2: An example sequence of questions from QBLink. The lead-in and question 1 are asking about the same object/answer. The subject of question 2 is the same as the object of question 1. All questions are about a narrow topic, Bitcoin.

To keep our development and test set intact and and of a reasonable percentage of questions, we use the questions in 2014 tournament (the year with the largest number of questions) for development and testing, and the rest of the questions are used for training (Table 3.1). We use TagMe [Ferragina and Scaiella, 2010] for mention detection and linking question text to Wikipedia.

Num. Questions (Num. Sequences) × 3	
Training	45,747 (15,249)
Develop	3,630 (1,210)
Testing	6,555 (2,185)
Num. Questions Tokens	
Training	32.6 ± 9.6
Developme	33.5 ± 9.9
Testing	32.1 ± 10.24
Num. TagMe Entities	
Training	2.46 ± 1.68
Developme	2.49 ± 1.68
Testing	2.48 ± 1.77
Num. Unique Answers	43,597
Num. Unique Answer Pages	18,529

Table 3.1: Statistics about QBLINK. Most questions are fairly long and contain 2.5 entity mentions, making the questions relatively complex.

3.3 Task Definition

The problem we consider is *factoid question answering* (Section 2.2), in which the system answers natural language questions using facts. Factoid QA is both widely studied in academia and a practical tool used by commercial search engines and conversational assistants. However, there is a discrepancy between the widely

studied factoid QA academic benchmarks and real applications: the benchmarks are often designed for the existing relations in knowledge graphs [Berant et al., 2013, Bordes et al., 2015], while in reality, minimal KG coverage precludes broader deployment.

For example, WebQuestions [Berant et al., 2013] are written by crowdworkers targeting a triple in Freebase; LC-QUAD [Dubey et al., 2019] targets Wikidata and DBPEDIA: the questions are guaranteed to be covered by the KG relations. However, although the entities in your favorite KG are rich, the recall of closed-form relations is often limited. For example, in Figure 3.1, the answer Delft is an entity in Wikipedia and FREEBASE, but the relation "painting the view of" appears in neither DBPEDIA nor FREEBASE and is unlikely to be covered in other knowledge graphs. The relationships between complicated, multi-faceted entities defy trite categorization in closed-form tuples; depending on a knowledge graph to provide all the possible relationships limits the potential of KGQA systems.

We focus on a more realistic setting: open-domain factoid question answering, whose questions are designed to test *human* knowledge and include ineffable links between concepts [Jennings, 2006]. Figure 3.1 shows an example question. The question has multiple Question Entity Nodes (left side of graph), and links the question entities to Candidate Entity Nodes (right). Intuitively, to find the answer, the system needs to first extract multiple clues from different pieces of question. Our proposed DELFT constructs a high coverage Free-Text Knowledge Graph by harvesting natural language sentences in the corpus as graph edges and grounds each question into its related subgraph (Section 3.4). Then, to model over the

fruitful but noisy graph, it uses a GNN to distinguish useful evidence from noise and then aggregates them to make the prediction (Section 5.2).

3.4 Free-Text Graph Construction

Answering real world factoid questions with current knowledge graphs falters when KG relations lack coverage; we instead use a free-text KG to resolve the coverage issue. One attempt to incorporate free-text corpus is to use Open Information Extraction [Lu et al., 2019, OPENIE] to extract relations from natural language sentences as graph edges. However OPENIE approaches favor precision over recall, and heavily rely on well defined semantics, falling prey to the same narrow scope that makes traditional KG approaches powerful. Instead, we build the knowledge graph directly from free-text corpus, represent sentences which contain the two entities (endpoints of the edge) as indirect relations. We leave the QA model (which only needs to output an answer entity) to figure out *which* sentences contain the information to answer the question, eliminating the intermediate information extraction step.

This section first discusses the *construction* of a *free-text* knowledge graph and then *grounding* questions to it.

3.4.1 Graph Construction

The free-text knowledge graph uses the same nodes V as existing knowledge graphs: entities and their attributes, which can be directly inherited from existing

knowledge graphs. The Evidence Edges E, instead of closed-form relations, are harvested natural language sentences from a corpus.

Entity Nodes The graph inherits Wikipedia entities as the nodes V (of course, entities from other corpora could be used). To represent the node, we use the first sentence of the corresponding document as its *node gloss*.

Free-Text Edges The Evidence Edges between nodes are sentences that pairs of entities in Wikipedia co-occur (again, other corpora such as ClueWeb [Callan et al., 2009] could be used). For specificity, let us find the edges that could link entities a and b. First, we need to know where entities appear. Both a and b have their own Wikipedia pages—but that does not give us enough information to find where the entities appear *together*. TagMe [Ferragina and Scaiella, 2010] finds entities mentioned in free text; we apply it to Wikipedia pages. Given the entity linker's output, we collect the following sentences as potential edges: (1) sentences in a's Wikipedia page that mention b, (2) sentences in b's page that mention a, and (3) sentences (anywhere) that mention both a and b.

3.4.2 Question Grounding

Now that we have described the general components of our graph, we next *ground* a natural language question to a subgraph of the KG. We then find an answer candidate in this subgraph.

Specifically, for each question, we ground the full free-text knowledge graph

into a question-related subgraph (Figure 3.1): a bipartite graph with *Question Entity Nodes* (e.g., `Dutch Golden Age`) on the left joined to *Candidate Entity Nodes* on the right (e.g., Delft) via *Evidence Edge* (e.g., "Delft played a highly influential role in the `Dutch Golden Age`").

Question Entity Nodes DELFT starts with identifying the entities in question $\mathcal{X}_q$ as the Question Entity Nodes $V_q = \{v_i \,|\, v_i \in \mathcal{X}_q\}$. In Figure 3.1, for instance, `Vermeer`, `The Little Street` and `Dutch Golden Age` appear in the question; these Question Entity Nodes populate the left side of DELFT's grounded graph. We use TagMe to identify question entities.

Candidate Entity Nodes Next, our goal is to find Candidate Entity Nodes. In our free-text KG, Candidate Entity Nodes are the entities with connections to the entities related to the question. For example, Delft occurs in the Wikipedia page associated with Question Entity Node `Vermeer` and thus becomes a Candidate Entity Node. The goal of this step is to build a set likely to contain—has high coverage of—the answer entity.

We populate the Candidate Entity Node V_a based on the following two approaches. First, we *link* entities contained in the Wikipedia *pages* of the Question Entity Nodes to generate Candidate Entity Nodes. To improve Candidate Entity Node recall, we next *retrieve* entities: after presenting the question text as a query to an IR engine (ElasticSearch [Gormley and Tong, 2015]), the entities in the top retrieved Wikipedia pages also become Candidate Entity Nodes.

These two approaches reflect different ways entities are mentioned in free text. Direct mentions discovered by entity linking tools are straightforward and often correspond to the clear *semantic* information encoded in Wikipedia ("In 1607, Khurram became engaged to Arjumand Banu Begum, who is also known as Mumtaz Mahal"). However, sometimes relationships are more *thematic* and are not mentioned directly; these are captured by IR systems ("She was a recognizable figure in academia, usually wearing a distinctive cape and carrying a walking-stick" describes Margaret Mead without named entities). Together, these two approaches have good recall of the Candidate Entity Node set V_a (Table 3.4).

Evidence Edges DELFT needs edges as evidence signals to know which Candidate Entity Node to select. The edges connect Question Entity Nodes V_q to Candidate Entity Nodes V_a with natural language. In Figure 3.1, the Wikipedia sentence "Vermeer was recognized during his lifetime in Delft" connects the Question Entity Node `Vermeer` to the Candidate Entity Node Delft. Given two entities, we directly find the Evidence Edges connecting them from the free-text knowledge graph.

Final Graph Formally, for each question, our final graph of DELFT $G = (V, E)$ includes the follows: Nodes V include Question Entity Nodes V_q and Candidate Entity Nodes V_a; Evidence Edges E connect the nodes: each edge e contains Wikipedia sentence(s) $\mathcal{S}(k), k = 1, \ldots, K$ linking the nodes.

3.4.3 Question Graph Pruning

The graph grounding ensures high coverage of nodes and edges. However, if the subgraph is *too* big, it slows training and inference. DELFT prunes the graph with a simple filter to remove weak, spurious clues and to improve computational efficiency.

Candidate Entity Node Filter We treat the Candidate Entity Nodes filtering as a ranking problem: given input Candidate Entity Nodes and question, score each connected Evidence Edge with a relevance score. During inference, the nodes with top-K highest scores are kept (we choose the highest Evidence Edge score as the node relevance score), while the rest are pruned. DELFT fine-tunes BERT as the filter model.[2] To be specific, for each connected Evidence Edge, we concatenate the question and sentence, along with the Candidate Entity Node gloss as the node context into BERT (Section 2.4.4):

```
[CLS] Question [SEP] Edge Sent [SEP] Node Gloss [SEP],
```

We apply an affine layer and sigmoid activation on the last layer's [CLS] representation to produce scalar value.

During training, for each question, we use Evidence Edge connected to the answer node as positive training example, and random sample 10 negative Evidence Edges as negative examples. To keep the DELFT's GNN training efficient, we keep the top twenty nodes for training set. For more comprehensive recall, development

[2]For efficiency, We use TFIDF filtering (top 1000 Evidence Edges are kept) before BERT.

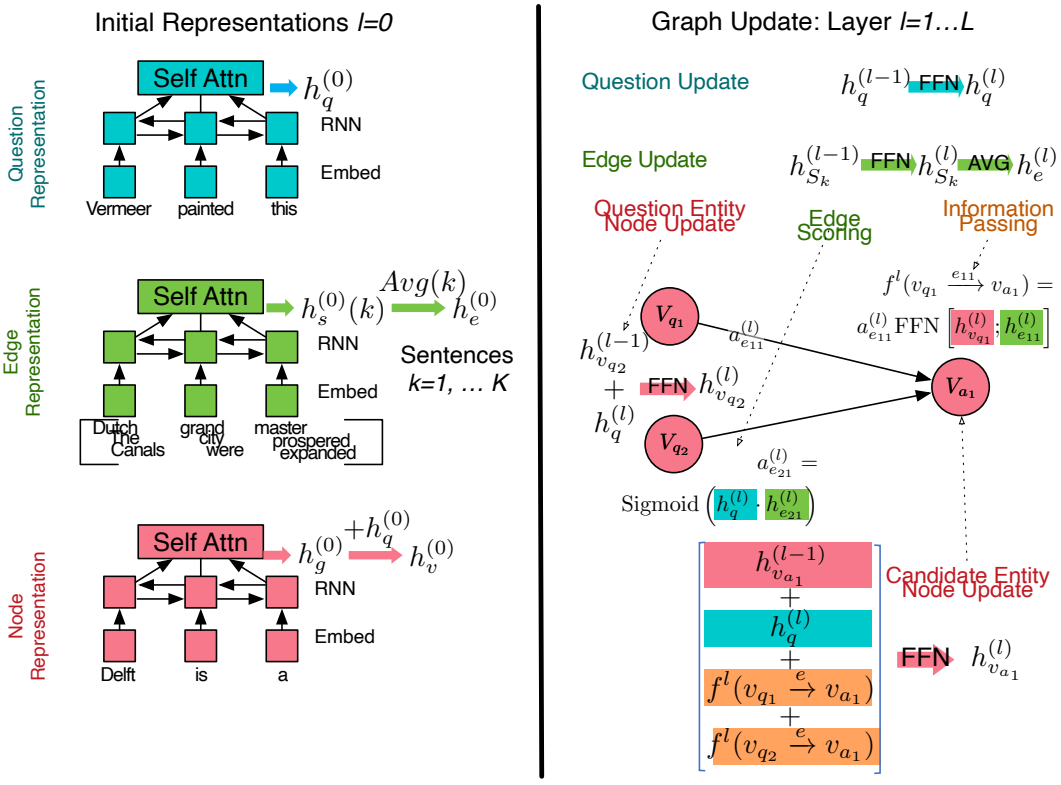


Figure 3.3: Model architecture of DELFT's GNN. The left side shows the initial representation of the network. The right side illustrates the graph update.

and test keep the top fifty nodes. If the answer node is not kept, DELFT cannot answer the question.

Edge Sentence Filter Since there is no direct supervision signal for which edge sentences are useful, we use TFIDF [Salton and McGill, 1983] to filter sentences. For all sentences $\mathcal{S}$ in Evidence Edge $e \in E$, we compute each sentence's TFIDF cosine similarity to the question, and choose the top five sentences for each Evidence Edge.

3.5 Free-text Graph Modeling

Given the question $\mathcal{X}_q$ and the grounded free-text graph G from Section 3.4, the goal is to find the correct answer node from the Candidate Entity Nodes. Recently several GNN based approaches (Section 2.1.3) answer questions by modeling the knowledge graph. Unlike previous approaches that represent graph nodes and edges as fixed embeddings, DELFT adopts a GNN to find the answer using a free-text knowledge graph. We make the following motivating observations:

Graph Connectivity The correct Candidate Entity Node usually has many connections to Question Entity Nodes. For example, in Figure 3.1, the correct Candidate Entity Node Delft connects to all three question Question Entity Nodes. Thus, a correct Candidate Entity Node should *aggregate* information from multiple Question Entity Nodes.

Edge Relevance The Evidence Edges with sentences "closer" to the question are likely to be more helpful to answering. For example, in Figure 3.1, the Question Entity Node `The Little Street` has two edges to Candidate Entity Nodes Delft and Amsterdam, but the Evidence Edge with sentence "`The Little Street` is one of only three paintings of views of Delft" is more similar to the question. The model needs to prioritize Evidence Edges that are similar to the question.

Node Relevance In addition to relevant edges, we also use node information to focus attention on the right Candidate Entity Node. One aspect is ensuring we get

the entity type correct (e.g., not answering a place like Uluru to a "who" question like "who took managing control of the Burra Burra mine in 1850?"). For both Question Entity Node and Candidate Entity Node, we use entity gloss sentences as node features. For example, the gloss of the candidate entity node says "Delft is a Dutch city", which aligns what the question asks ("this dutch city"). Similarly, a Question Entity Node whose gloss sentence better matches a question are more likely to contain the clues for the answer.

Iterative Refinement Figure 3.3 illustrates the architecture of DELFT's GNN. A key component of the GNN is that multiple layers ($^{(l)}$) refine the representations of Candidate Entity Nodes, hopefully focusing on the correct answer.

The first layer learns the initial representation of the nodes and edges, using their textual features and the relevance to the question (Section 3.5.1). Each successive layer combines this information, focusing on the Evidence Edges and Candidate Entity Nodes that can best answer the question (Section 3.5.2). The final layer scores the Candidate Entity Nodes to produces an answer (Section 3.5.3).

3.5.1 Initial Representations

We start with representations for questions, Question Entity Nodes, Candidate Entity Nodes, and Evidence Edges (GNN layer 0). These representations are combined in subsequent layers.

Question and Node Representation Each question $\mathcal{X}_q$ and node gloss (both Candidate Entity Node and Question Entity Node) $\mathcal{X}_g$ is a sequence of word tokens $\mathcal{X} = (x_1, \ldots, x_n)$. Individual word embeddings $(\mathbf{E}_{x_1}, \ldots, \mathbf{E}_{x_n})$ become a sequence representation $\mathbf{h}_q^{(0)}$ ($\mathbf{h}_g^{(0)}$) using a Recurrent neural network (RNN) with a self-attention (SELF-ATTN) layer:

$$\mathbf{h}_{x_u} = \text{RNN}\left(\mathbf{E}_{x_1}, \ldots, \mathbf{E}_{x_n}\right) \tag{3.1}$$

where u is token position of sequence $\mathcal{X}$. A self-attention layer over the RNN hidden layer first weights the hidden states by a learnable weight vector $\mathbf{w}_x$

$$a_{x_u} = \text{softmax}\left(\mathbf{w}_x \cdot \mathbf{h}_{x_u}\right), \tag{3.2}$$

then a final representation $h_x^{(0)}$ is a weighted average of all hidden states

$$\mathbf{h}_x^{(0)} = \sum_u a_{x_u} \mathbf{h}_{x_u}. \tag{3.3}$$

The node representation $\mathbf{h}_v^{(0)}$ of each node $v \in V$ is a sum of gloss representation $\mathbf{h}_g^{(0)}$ and question representation $\mathbf{h}_q^{(0)}$.

$$\mathbf{h}_v^{(0)} = \mathbf{h}_g^{(0)} + \mathbf{h}_q^{(0)}. \tag{3.4}$$

The representation $\mathbf{h}_v^{(0)}$ is applied to both question entity nodes $\mathbf{h}_{v_q}^{(0)}$ and candidate answer nodes $\mathbf{h}_{v_a}^{(0)}$.

Edge Representation Effective Evidence Edges in DELFT should point from entities mentioned in the question to the correct Candidate Entity Node. We get the representation $\mathbf{h}_e^{(0)}$ of edge $e \in E$ by first embedding each edge sentence $S(k)$'s

tokens $\mathcal{X}_s = (s_1, ..., s_n)$ into $(\mathbf{E}_{s_1}, ..., \mathbf{E}_{s_n})$, then encoding with a RNN layer:

$$\mathbf{h}_{s_u} = \text{RNN}\left(\mathbf{E}_{s_1}, ..., \mathbf{E}_{s_n}\right). \tag{3.5}$$

Then we fuse the question information into each edge sentence. An inter attention layer [Seo et al., 2016] based on the question representation $\mathbf{h}_q^{(0)}$ first weights each sentence token position u

$$a_{s_u} = \text{softmax}\left(\mathbf{h}_{s_u} \cdot \mathbf{h}_q^{(0)}\right); \tag{3.6}$$

the weight is then combined into the question-aware edge sentence k's representation $\mathbf{h}_s(k)$:

$$\mathbf{h}_s^{(0)}(k) = \sum_u a_{s_u} \mathbf{h}_{s_u}. \tag{3.7}$$

Now that the edges have focused on the evidence that is useful to the question, we average all edge sentences' representations $\mathbf{h}_s^k$ into a single edge representation

$$\mathbf{h}_e^{(0)} = \text{Avg}_k\left(\mathbf{h}_s^{(0)}(k)\right). \tag{3.8}$$

3.5.2 Graph Update

Given the initial representation of the question $\mathbf{h}_q^{(0)}$, nodes $\mathbf{h}_v^{(0)}$, and edges $\mathbf{h}_e^{(0)}$, DELFT's GNN updates the representations through stacking multiple layers. It passes the representations from the question nodes to the candidate nodes by combining multiple evidence edges' information. After this, the representations of the candidate nodes in the final layer accumulates information from the question nodes (v_q), the question text (q), the evidence edges (e_{ij}), and previous candidate representations. These updated node representations are then used to calculate the answer scores.

For each layer l, DELFT's GNN first updates the question and edge representation with a feed forward network (FFN) layer (*Representation Forwarding*), then it scores each edge by its relevance to the question (*Edge Scoring*), finally passing the information (*information passing*) from the question entity nodes (*Question Entity Nodes Update*) to candidate entity nodes (*Answer Entity Nodes Update*). We discuss updates going from top to bottom in Figure 3.3.

Question Entity Nodes Update Question Entity Nodes' representations $\mathbf{h}_{v_q}^{(l)}$ combine the question representation $\mathbf{h}_q^{(l)}$ and the previous layer's node representation $\mathbf{h}_{v_q}^{(l-1)}$,

$$\mathbf{h}_{v_q}^{(l)} = \text{FFN}\left(\mathbf{h}_{v_q}^{(l-1)} + \mathbf{h}_q^{(l)}\right). \tag{3.9}$$

Questions and Edges The representations of questions (q) and edges (e)—initially represented through their constituent text in their initial representation—are updated between layer $l-1$ and l through a feedforward network. A question's single vector is straightforward,

$$\mathbf{h}_q^{(l)} = \text{FFN}\left(\mathbf{h}_q^{(l-1)}\right), \tag{3.10}$$

but edges are slightly more complicated because there may be multiple sentences linking two entities together. Thus, each individual sentence k has its representation updated for layer l,

$$\mathbf{h}_s^{(l)}(k) = \text{FFN}\left(\mathbf{h}_s^{(l-1)}(k)\right), \tag{3.11}$$

and those representations are averaged to update the overall edge representation,

$$\mathbf{h}_e^{(l)} = \text{Avg}_k \left(\mathbf{h}_s^{(l)}(k) \right). \tag{3.12}$$

Edge Scoring Each edge has an edge score a_e; higher edge scores indicate the edge is more useful to answering the question. The GNN computes an edge score a_e for each edge representation $\mathbf{h}_e^{(l)}$ based on its similarity to this layer's question representation:

$$a_e^{(l)} = \text{Sigmoid} \left(\mathbf{h}_q^{(l)} \cdot \mathbf{h}_e^{(l)} \right). \tag{3.13}$$

Information Passing The representation of the edge is not directly used to score Candidate Entity Nodes. Instead, a representation is created from both the source Question Entity Node $\mathbf{h}_{v_q}^{(l)}$ and the combined edges $\mathbf{h}_e^{(l)}$ concatenated togther, feeds that through a feed-forward network (to make the dimension consistent with previous layer and question representation), and weights by the edge score (a_e, Equation 3.13),

$$f^{(l)} \left(v_q \xrightarrow{e} v_a \right) = a_e^{(l)} \text{FFN} \left(\left[\mathbf{h}_{v_q}^{(l)}; \mathbf{h}_e^{(l)} \right] \right). \tag{3.14}$$

Candidate Entity Nodes Update The updated representation for each Candidate Entity Node combines the previous layer's Candidate Entity Node representation, the question representation, and the passed information.

$$\mathbf{h}_{v_a}^{(l)} = \text{FFN} \Big(\underbrace{\mathbf{h}_{v_a}^{(l-1)}}_{\text{previous}} + \underbrace{\mathbf{h}_q^{(l)}}_{\text{question}} + \underbrace{\sum_{v_q \in V_q} f^{(l)}(v_q \xrightarrow{e} v_a)}_{\text{Information Passing}} \Big). \tag{3.15}$$

After iterating through several GNN layers, the candidate node representations aggregate information from the question nodes, edges, and candidate nodes themselves.

3.5.3 Answer Scoring

Finally, DELFT uses a multi-layer perception (MLP) to score the Candidate Entity Node using the final layer L's representations,

$$p(v_a; G) = \text{Sigmoid}\left(\text{MLP}(\mathbf{h}_{v_a}^{(L)})\right). \tag{3.16}$$

The GNN is trained using binary cross entropy loss over all Candidate Entity Nodes v_a. At test time, it chooses the answer with the highest $p(v_a, G)$.

3.6 Experiments

We evaluate on three datasets with expert-authored (as opposed to crowd-worker) questions. QBLINK(Section 3.2) is an entity-centric dataset with human-authored questions. The task is to answer the entity the question describes. We use the released dataset for evaluation. QANTA [Iyyer et al., 2014] is a QA dataset collected from Quizbowl competitions. Each question is a sequence of sentences providing increased information about the answer entity. TRIVIAQA [Joshi et al., 2017b] includes questions from trivia games and is a benchmark dataset for RC. We use its unfiltered version, evaluate on its validation set, and split 10% from its unfiltered training set for model selection. Unlike the other datasets, TRIVIAQA is relatively simpler; it mentions fewer entities per question; as a result DELFT has lower accuracy.

	QBLink	qanta	triviaqa
Training	42219	31489	41448
Dev	3276	2211	4620
Test	5984	4089	5970
# Tokens	31.7 ± 9.4	129.2 ± 32.0	16.5 ± 8.6
# Entities	6.8 ± 2.4	21.2 ± 7.3	2.2 ± 1.3
% 1-3 Entities	9.6%	0	86.9%
% 4-6 Entities	36.7%	0	13.1%
% 7-9 Entities	36.5%	0	0
% 10+ Entities	17.1%	100%	0

Table 3.2: The three expert-authored datasets used for experiments. All are rich in entities, but the qanta dataset especially frames questions via an answer's relationship with entities mentioned in the question.

We focus on questions that are answerable by Wikipedia entities. To adapt triviaqa into this factoid setting, we filter out all questions that do not have Wikipedia title as answer. We keep 70% of the questions, showing good coverage of Wikipedia Entities in questions. All QBLink and qanta questions have entities tagged by TagMe. TagMe finds no entities in 11% of triviaqa questions; we further exclude these. Table 3.2 shows the statistics of these three datasets sand the fraction of questions with entities.

Layer	Description
All RNN	1 layer Bi-GRU, 300 hidden dimension
All FFN	600 dimension, ReLU activation
MLP	2 layers with 600, 300 dimenstions, ReLU activation
Attention	600 dimension Bilinear
Self-Attention	600 dimension Linear
Layers	$L = 3$

Table 3.3: Parameters in DELFT's GNN.

3.6.1 Question Answering Methods

We compare the following methods:

- QUEST [Lu et al., 2019] is an unsupervised factoid QA system over text. For fairness, instead of Google results we apply QUEST on IR-retrieved Wikipedia documents.
- DRQA [Chen et al., 2017] is a reading comprehension model for open-domain QA that retrieves documents and extracts answers.
- DOCQA [Clark and Gardner, 2018] improves multi-paragraph reading comprehension, and is among the strongest on TRIVIAQA. Their suggested settings and pre-trained model on TRIVIAQA are used.
- BERT-Entity fine-tunes BERT [Devlin et al., 2019] on the question-entity name pair to rank candidate entities in DELFT's graph.
- BERT-Sent fine-tunes BERT on the question-entity gloss sequence pair to rank

	QBLink	qanta	triviaqa
# Candidate Answer Entities per Question	1607 ± 504	1857 ± 489	1533 ± 934
Answer Recall in All Candidates	92.4%	92.6%	91.5%
Answer Recall after Filtering	87.6%	83.9%	86.4%
Answer Recall within Two Hops along DBpedia*	38%	–	–
# Edges to Correct Answer Node (+)	5.07 ± 2.17	12.33 ± 5.59	1.87 ± 1.12
# Edges to Candidate Entity Node (-)	2.35 ± 0.99	4.41 ± 2.02	1.21 ± 0.35
# Evidence Sentences per Edge (+)	12.3 ± 11.1	8.83 ± 6.17	15.53 ± 17.52
# Evidence Sentences per Edge (-)	4.67 ± 3.14	4.48 ± 1.88	3.96 ± 3.33

Table 3.4: Coverage and density of generated free-text entity graph. (+) and (-) mark the statistics on correct answer nodes and incorrect nodes, respectively. (*) is the result from our manual labeling on 50 QBLink questions.

candidate entities in DELFT's graph.

- BERT-MemNN is a memory network [Weston et al., 2014] using fine-tuned BERT. It uses the same evidence as DELFT but collapses the graph structure (i.e., edge evidence sentences) by concatenating all evidence sentences into a memory cell.
- We evaluate our method, DELFT, with GLoVE embedding [Pennington et al., 2014] and BERT embeddings.

	QBLink					QANTA		TRIVIAQA		
	ALL	1-3	4-6	7-9	10+	ALL	10+	ALL	1-3	4-6
QUEST	0.07	0	0.09	0.09	0	-	-	-	-	-
DRQA	38.3	35.4	37.5	39.2	39.6	47.4	47.4	40.3	40.1	41.2
DOCQA	-	-	-	-	-	-	-	49.4	49.3	49.8
BERT-ENTITY	16.2	16.1	16.3	16.2	16.4	34.2	34.2	25.1	24.5	29.0
BERT-SENT	34.8	34.7	34.8	34.7	34.6	54.2	54.2	44.5	44.4	45.1
BERT-MEMNN	35.8	32.7	36.1	36.5	34.3	56.1	56.1	51.3	50.9	54.0
DELFT-GLOVE	54.2	45.5	55.0	56.4	53.2	65.8	65.8	51.3	50.1	59.5
DELFT-BERT	**55.1**	**46.8**	**55.5**	**57.1**	**55.5**	**66.2**	**66.2**	**52.0**	**50.5**	**61.1**

Table 3.5: Answer Accuracy (Exact Match) on ALL questions as well as question groups with different numbers of entities: e.g., 0–3 are questions with fewer than four entities. We omit empty ranges (such as very long, entity-rich Quizbowl questions). DELFT has higher accuracy than baselines, particularly on questions with more entities.

3.6.2 Implementation

Our implementation uses PyTorch [Paszke et al., 2017] and its DGL GNN library.[3] We keep top twenty candidate entity nodes in the training and fifty for testing; the top five sentences for each edge is kept. The parameters of DELFT's GNN layers are listed in Table 3.3. For DELFT-BERT, we use BERT output as contex-

[3] https://github.com/dmlc/dgl

tualized embeddings.

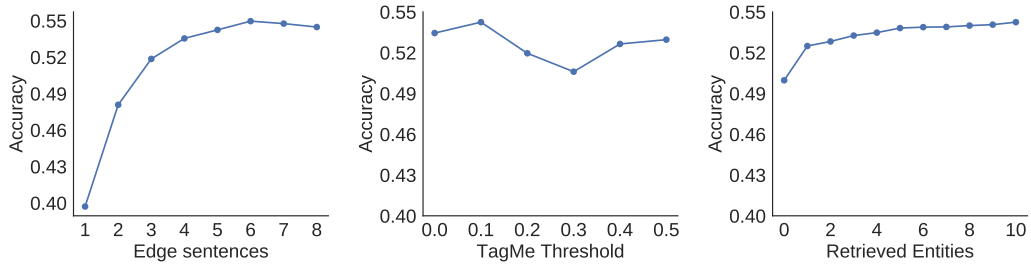
For BERT-Entity and BERT-Sent, we concatenate the question and entity name (entity gloss for BERT-Sent) as the BERT input and apply an affine layer and sigmoid activation to the last BERT layer of the [CLS] token; the model outputs a scalar relevance score.

BERT-MemNN concatenates all evidence sentences and the node gloss, and combines with the question as the input of BERT. Like BERT-Entity and BERT-Sent, an affine layer and sigmoid activation is applied on BERT output to produces the answer score.

DrQA retrieves 10 documents and then 10 paragraphs from them; we use the default setting for training. During inference, we apply TagMe to each retrieved paragraph and limit the candidate spans as tagged entities. DocQA uses the pre-trained model on TRIVIAQA-unfiltered dataset with default configuration applied to our subset.

3.7 Evaluation Results

Three experiments evaluate DELFT's graph coverage, answer accuracy, and source of effectiveness. Then we visualize the GNN attention and examine individual examples.



(a) (b) (c)

Figure 3.4: Accuracy of DELFT on different variations of Free-Text Knowledge Graphs. We find that more sentences per edge, reasonable threshold and number retrieved entities help the accuracy.

3.7.1 Graph Coverage

DELFT's graph has high coverage (Table 3.4). Each question is connected to an average of 1500+ candidate nodes; 90% of them can be answered by the connected nodes. After filtering to 50 candidates, more than 80% questions are answerable. In comparison, we manually examined 50 randomly sampled QBLINK questions: only 38% of them are reachable within two hops in the DBpedia graph.

DELFT's graph is dense. On average there are five (QBLINK) and twelve (QANTA) edges connecting the correct answer nodes to the question entity nodes. TRIVIAQA questions have two entities on average and more than one is connected to the correct answer. Each edge has eight to fifteen evidence sentences.

The free-text knowledge graph naturally separates the correct answer by its structure. Compared to incorrect answers (-), the correct ones (+) are connected by significantly more evidence edges. The edges also have more evidence sentences.

Ablation	Accuracy	
No Node Pruning	42.6	-21.4%
No Gloss Representation	49.2	-9.3%
No Edge Evidence Sentence	48.8	-10.0%
No Edge Importance	52.6	-3.0%
No Self Attention Layer	53.0	-2.2%
DELFT-GLOVE	54.2	–

Table 3.6: Ablation Study of DELFT-Glove on QBLINK (ALL questions). Each DELFT variant removes one component and keeps everything else fixed.

Aided by free-text evidence, the coverage of the structured graph is no longer the bottleneck. The free-text knowledge graph provides enough evidence and frees the potential of structured QA. At the same time, the rich evidence also inevitably introduces noise. The next experiment examines whether DELFT—given the answer somewhere in the graph—can find the single correct answer.

3.7.2 Answer Accuracy

DELFT outperforms[4] all baselines on both full datasets and dataset subsets based on the number of entities in a question (Table 7.3).

QUEST falters on these questions, suggesting that some level of supervision is required. On more complicated factoid QA datasets QBLINK and QANTA, DELFT

[4]Recall, however, that we exclude questions that with no entities or whose answer is not an entity.

improves over DrQA, the reading comprehension baseline. These datasets require reasoning over multiple sentences (either within a long question's sentence or across multiple questions); however, DrQA is tuned for single sentence questions. DELFT—by design—focuses on matching questions' text with disparate evidence. In the RC benchmark dataset TRIVIAQA, DELFT still beats DrQA (albeit on an entity-focused subset). It is also better than DocQA, one of the strongest models on TRIVIAQA. With our Free-Text Knowledge Graph, DELFT better locates necessary evidence *sentences* via graph structure, while RC only uses retrieved paragraphs.

BERT-Entity fares poorly because it only has entity name information; even with the help strong pre-trained model, this is too limited answer complex questions. BERT-Sent incorporates the gloss information but lags other methods. DELFT outperforms both baselines, since it combines useful text evidence and KG connections to answer the question.

Compared to BERT-MemNN, which uses the same evidence and BERT but without structure, DELFT's structured reasoning thrives on complex questions in QBLink and QANTA. On TRIVIAQA, which has fewer than two edges per candidate entity node, DELFT's accuracy is close to BERT-MemNN, as there is not much structure.

As questions have more entities, DELFT's relative accuracy increases. In comparison, almost all other methods' effectiveness stays flat, even with more evidence from additional question entities.

3.7.3 Ablation Study

We ablate both DELFT's graph construction and GNN components to see which components are most useful. We use QBLINK dataset and DELFT-GLOVE embeddings for these experiments. For each ablation, one component is removed while keeping the other settings constant.

Graph Ablation The accuracy grows with more sentences per edge until reaching diminishing returns at six entities (Figure 3.4(a)). Fewer than three sentences significantly decreases accuracy, prematurely removing useful information. It's more effective to leave the GNN model to distinguish the signal from the noise. We choose five sentences per edge.

Because we retrieve entities automatically (rather than relying on gold annotations), the threshold of the entity linking process can also be tuned: are more (but noisier) entities better than fewer (but more confident) entities? Using all tagged entities slightly hurts the result (Figure 3.4(b)), since it brings in uninformative entities (e.g., linking "name the first woman in space" to "Name"). Filtering too aggressively, however, is also not a good idea, as the accuracy drops with aggressive thresholds (> 0.2), removing useful connections. We choose 0.1 as threshold.

To see how sensitive DELFT is to automatic entity linking, we manually annotate twenty questions to see the accuracy with perfect linking. The accuracy is on par with Tagme linked questions: both get fifteen right. We would need a larger set to more thoroughly examine the role of linker accuracy.

id	Example	Explanation
1(+)	**Q**: This `New England Patriots` quarterback was named `Super Bowl MVP`. He had three touchdown passes during the game: one each to `Deion Branch`, `David Givens`, and `Mike Vrabel`. **A**: Tom Brady **P**: Tom Brady	Substantial evidence points to Tom Brady: "he plays for `New England Patriots`", and "he had touchdown passes with `Deion Branch`". DELFT aggregates evidence and makes the correct prediction.
2(-)	**Q**: Name this `European` nation which was divided into Eastern and Western regions after `World War II`. **A**: Germany **P**: Yumen Pass	No informative question entities that would lead to the key evidence sentence "Germany divides into East Germany and West Germany".
3(-)	**Q**: `Telemachus` is the son of this hero, who makes a really long journey back home after the `Trojan War` in an epic poem by `Homer`. **A**: Odysseus **P**: Penelope	DELFT can't make right prediction since the wrong candidate Penelope (Odysseus's wife) shares most of the extracted evidence sentences with the correct answer (e.g., their son `Telemachus`).

Table 3.7: Three examples from QBLINK dataset with DELFT output. Each example has a question (Q), answer (A) and DELFT predcition (P), along with an explanation of what happened. The first is correct (+), while the last two are wrong (-).

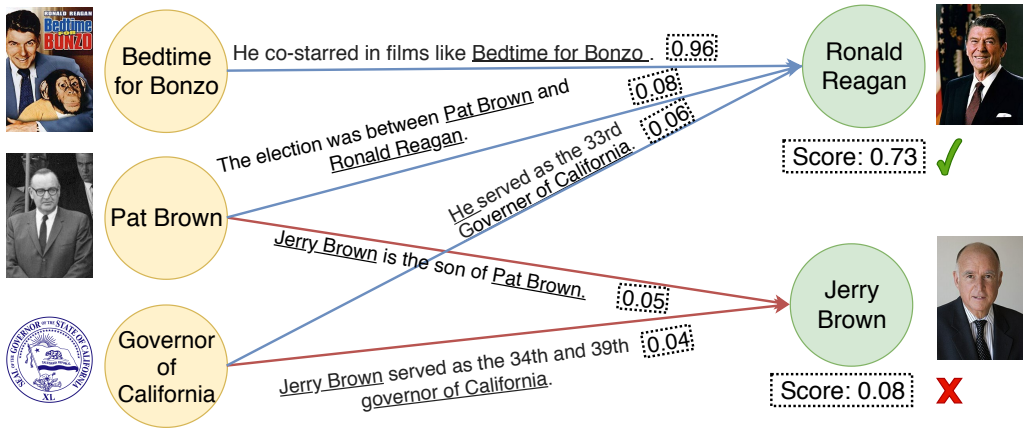


Figure 3.5: An example DELFT subgraph. The learned edge weights in GNN are in the brackets.

Recall that DELFT uses not just the entities in the question but also searches for edges similar to question text. Figure 3.4(c) shows the accuracy when retrieving n additional entities. The benefits plataeu after three entities.

Model Ablation In addition to different data sources and preprocessing, DELFT also has several model components. We ablate these in Table 4.5. As expected, both node (gloss) and edge evidence help; each contributes $\sim 10\%$ accuracy. Edge importance scoring, which controls the weights of the information flow from Question Entity Node to Candidate Entity Nodes, provides $\sim 3\%$ accuracy. The input representation is important as well; the self-attention layer contributes $\sim 2.2\%$ accuracy.

3.7.4 Graph Visualization

Figure 3.5 shows a question with GNN output: the correct Candidate Entity Node Ronald Reagan connects to all three Question Entity Nodes. The edge from `Bedtime for Benzo` to Reagan is informative—other Candidate Entity Nodes (e.g., politicians like Jerry Brown) lack ties to this cinema masterpiece. The GNN model correctly (weight 0.96) favors this edge. The other edges are less distinguishable. For example, the edge from `Governor of California` to Ronald Reagan and Jerry Brown are both relevant (both were governors) but unhelpful. Thus, the GNN has similar weights (0.06 and 0.04) for both edges, far less than `Bedtime for Bonzo`. By aggregating edges, DELFT's GNN selects the correct answer.

3.7.5 Case Study

To gain more insights into DELFT model's behavior, we further sample some examples from QBLINK. Table 3.7 shows two positive examples (1) and two negative examples (2 and 3). With multiple pieces of evidence, DELFT could aggregate different pieces together, and make more accurate predictions (Example 1). However some common sources of error include: too few informative entities in the question (Example 2) or evidence that overlaps too much between two Candidate Entity Nodes (Example 3).

3.8 The View Beyond DELFT

Real-world factoid QA requires answering diverse questions across domains. Relying on existing knowledge graph relations to answer these questions often leads to highly accurate but brittle systems: they suffer from low coverage. To overcome the bottleneck of structure sparsity in existing knowledge graphs, DELFT inherits KGQA-style reasoning with the widely available free-text evidence. DELFT builds a high coverage and dense free-text knowledge graph, using natural language sentences as edges. To answer questions, DELFT grounds the question into the related subgraph connecting entities with free-text graph edges and then uses a graph neural network to represent, reason, and select the answer using evidence from both free-text and the graph structure.

Combining natural language and knowledge-rich graphs is a common problem: e-mail and contact lists, semantic ontologies and sense disambiguation, and semantic parsing. While this chapter focuses mainly on approaches targeted at questions from trivia games, where the (multi-sentence) questions often contain multiple clues, future work should explore whether these approaches are also useful for dialog, language modeling, or *ad hoc* search.

More directly for question answering, more fine-grained reasoning could help solve the example of Table 3.7: while both Odysseus and Penelope have Telemachus as a son, only Odysseus made a long journey and should thus be the answer. In the next chapter, we tackle such problems by recognizing contents of nodes in free-text, through modeling semi-structured text graph.

Chapter 4: Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention

Previous chapter introduces DELFT, which achieves high answer accuracy on trivia questions. However, the setting we consider has some limitations: the answer is an entity; the evidence is reached within single-hop retrieval. In this chapter, we focus on a more general setting, in which the answer is any text span, and the evidence requires multiple hops to retrieve. We introduce TRANSFORMER-XH, which uses eXtra Hop attention to enable intrinsic modeling of semi-structured texts in a fully data-driven way. Its new attention mechanism "hops" across the connected text sequences in addition to attending over tokens within each sequence. Thus, TRANSFORMER-XH better conducts joint multi-evidence reasoning by propagating information between documents and constructing global contextualized representations. On multi-hop question answering, TRANSFORMER-XH leads to a simpler multi-hop QA system which outperforms previous state-of-the-art. Furthermore, simply applying TRANSFORMER-XH to multi-evidence fact verification task achieves competitive results to the state-of-the-art, showing its strong generalizability. [1]

[1] This Chapter describes work published in ICLR 2020 [Zhao et al., 2020b].

4.1 Introduction

Transformers effectively model natural language in *sequential* form (Section 2.1.1). Nevertheless, in many NLP tasks, text does not simply appear as a linear sequence of tokens (i.e., unstructured) but rather carries meaningful semi-structure in the form of sections, headings, and hyperlinks. Such semi-structure can be represented abstractly as trees or graphs with nodes and edges; and the tasks can be performed as joint reasoning on these more general structures as input. Multi-hop question answering [Yang et al., 2018] is one such task in which semi-structure plays an important role since the evidence required to formulate the answer is scattered across multiple documents, requiring systems to jointly reason across links between them.

Recent approaches leverage pre-trained Transformers (e.g., BERT) for multi-hop question answering (QA) by converting the structural reasoning task into sub-tasks that model flat sequences. For example, Min et al. [2019c] decompose a multi-hop question into a series of single-hop questions; Ding et al. [2019] conduct several steps of single-hop reading comprehension to simulate the multi-hop reasoning. The hope is that additional processing to fuse the outputs of the sub-models can recover all the necessary information from the original semi-structure. While pre-trained Transformer language models have shown improvements on multi-hop QA, manipulating the inherent semi-structure of the problem to fit the rigid requirements of out-of-the-box models can introduce problematic assumptions or information loss.

This chapter presents TRANSFORMER-XH (eXtra Hop), which upgrades Transformers with the ability to natively represent semi-structured texts. TRANSFORMER-

XH introduces extra hop attention in its layers that connect different text pieces following their inherent semi-structure while also maintaining the powerful pre-trained Transformer abilities over each textual piece individually. Our extra hop attention enables 1) a more global representation of the evidence contributed by each piece of text as it relates to the other evidence, and 2) a more natural way to jointly reason over an evidence graph by propagating information along edges necessary to complete the task at hand.

We apply TRANSFORMER-XH to two tasks: HOTPOTQA, the multi-hop question answering task, and FEVER, the fact verification benchmark whose claims often require multiple pieces of evidence to support [Thorne et al., 2018]. Rather than decomposing the task into a series of sub-tasks to fit the constraints of pre-trained Transformers, TRANSFORMER-XH is a solution that fits the problem as it naturally occurs. It is a single model that represents and combines evidence from multiple documents to conduct the reasoning process. On HOTPOTQA's FullWiki setting, which requires strong multi-hop reasoning capability [Min et al., 2019c, Jiang and Bansal, 2019], TRANSFORMER-XH outperforms COGQA [Ding et al., 2019], the previous start-of-the-art, by 12 points on answer F1. On FEVER 1.0 shared task, TRANSFORMER-XH outperforms GEAR, the Graph Neural Network based approach significantly. On both applications, TRANSFORMER-XH beats the contemporary BERT based pipeline SR-MRS [Nie et al., 2019a], by 2-3 points.

The results follow from our simple yet effective design, with one unified model operating over the inherent semi-structure of the task, rather than melding the outputs from disparate sub-tasks adapted to the sequential constraints of pre-trained

Transformers. Our ablation studies demonstrate TRANSFORMER-XH's efficacy on questions that are known to require multi-hop reasoning [Min et al., 2019c] and on verifying multi-evidence claims [Liu et al., 2019c]. Our analyses confirm that the source of TRANSFORMER-XH's effectiveness success is due to the eXtra Hop attention's ability to fuse and propagate information across multiple documents.

4.2 Model

This section first discusses preliminaries on sequential Transformers, then we show how we incorporate eXtra hop attention to create TRANSFORMER-XH.

4.2.1 Preliminaries

Transformers represent a sequence of input text tokens $X = \{x_1, ..., x_i, ..., x_n\}$ as contextualized distributed representations. We review Trasformer architecture in Section 2.1.1.

A challenge of Transformer is that its attention is calculated over all token pairs, which is hard to scale to long text sequences. TRANSFORMER-XL (eXtra Long) addresses this challenge by breaking down longer texts, e.g., a multi-paragraph document, into a sequence of text segments: $\{X_1, ..., X_\tau, ..., X_\zeta\}$, and propagates the information between adjacent text segments using the following attention:

$$\tilde{H}_\tau^{l-1} = [\text{Freeze}(H_{\tau-1}^{l-1}) \circ H_\tau^{l-1}]. \tag{4.1}$$

It concatenates ($\circ$) the representation of the previous segment $H_{\tau-1}^{l-1}$ to the current segment as segment level recurrences. The new representation $\tilde{H}_\tau^{l-1}$ includes

the information from the previous segment and is integrated in the new attention mechanism:

$$\tilde{Q}^T; \tilde{K}^T; \tilde{V}^T = W^q \cdot H_\tau^{l-1}; W^k \cdot \tilde{H}_\tau^{l-1}; W^v \cdot \tilde{H}_\tau^{l-1}. \tag{4.2}$$

The attention over the previous segment allows TRANSFORMER-XL to effectively model long form text data recurrently as a sequence of text chunks [Dai et al., 2019].

Nevertheless, in many scenarios, the text segments are organized in nontrivial structures beyond a linear sequence. For example, documents are connected by hyperlinks in a graphical structure that does not readily simplify to form a linear sequence, prohibiting TRANSFORMER-XL's recurrent approach.

4.2.2 TRANSFORMER-XH with eXtra Hop Attention

TRANSFORMER-XH models semi-structured text sequence by linking them with eXtra Hop attention following their original structure. As illustrated in Figure 4.1, to model three connected documents $d_2 \rightarrow d_1 \rightarrow d_3$, TRANSFORMER-XH uses eXtra Hop attention to propagate information along the graph edges, enabling information sharing between connected text sequence.

Formally, the semi-structured text includes a set of nodes, $\mathcal{X} = \{X_1, ..., X_\tau, ...X_\zeta\}$, each corresponding to a text sequence, and an edge matrix E, which includes the connections (e.g., links) between them. The goal is to learn representations $\mathcal{H} = \{\tilde{H}_1, ..., \tilde{H}_\tau, ...\tilde{H}_\zeta\}$, that incorporate not only the local information in each sequence X, but also the global contexts on the entire semi-structured text $\{\mathcal{X}, E\}$.

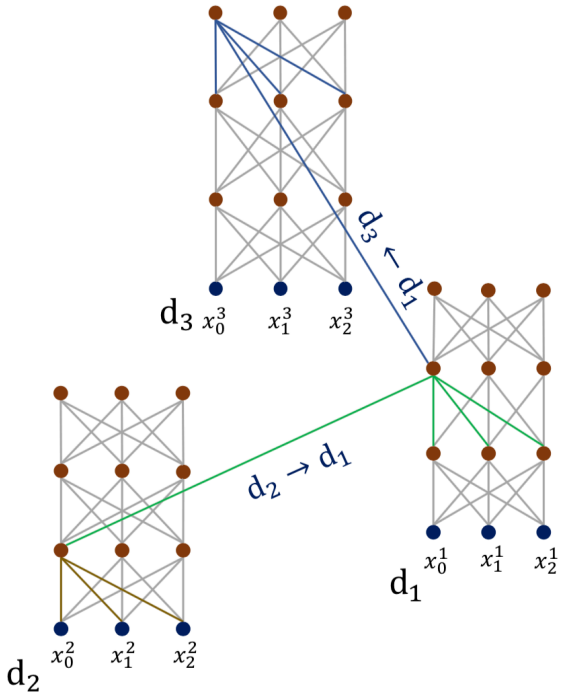


Figure 4.1: The eXtra Hop attention in TRANSFORMER-XH. The hop attention on the path $d_2 \rightarrow d_1 \rightarrow d_3$.

TRANSFORMER-XH achieves this by two attention mechanisms: in-sequence attention and eXtra Hop attention. The *in-sequence attention* is the same as vanilla Transformer: in layer l, token i gathers information from other tokens inside the same text piece τ:

$$h^l_{\tau,i} = \sum_j \text{softmax}_j \left(\frac{q^T_{\tau,i} \cdot k_{\tau,j}}{d_k} \right) \cdot v_{\tau,j}. \tag{4.3}$$

The *eXtra Hop attention* uses the first token in each sequence – the added special token "[CLS]" – as an "attention hub", which attends on all other connected nodes' hub token. In layer l, the τ-th text sequence attends over another text

sequence η if there is an edge between them ($e_{\tau\eta} = 1$):

$$\hat{h}^l_{\tau,0} = \sum_{\eta;e_{\tau\eta}=1} \text{softmax}_\eta \left(\frac{\hat{q}^T_{\tau,0} \cdot \hat{k}_{\eta,0}}{\sqrt{d_k}} \right) \cdot \hat{v}_{\eta,0}. \tag{4.4}$$

Node τ calculates the attention weight on its neighbor η using hop query $\hat{q}_{\tau,0}$ and key $\hat{k}_{\eta,0}$. Then it uses the weights to combine its neighbors' value $\hat{v}_{\eta,0}$ and forms a globalized representation $\hat{h}^l_{\tau,0}$.

The two attention mechanism are combined to form the new representation of layer l:

$$\tilde{h}^l_{\tau,0} = \text{Linear}([h^l_{\tau,0} \circ \hat{h}^l_{\tau,0}]), \tag{4.5}$$

$$\tilde{h}^l_{\tau,i} = h^l_{\tau,i}; \forall i \neq 0. \tag{4.6}$$

Note that the non-hub tokens ($i \neq 0$) still have access to the hop attention in the previous layer through Eqn. (4.3).

One layer of eXtra Hop attention can be viewed as a single-step of information propagation along edges E. For example, in Figure 4.1, the document node d_3 updates its representation by gathering information from its neighbor d_1 using the hop attention $d_1 \rightarrow d_3$. When multiple TRANSFORMER-XH layers are stacked, this information in d_1 includes both d_1's local contexts from its in-sequence attention, and cross-sequence information from the hop attention $d_2 \rightarrow d_1$ of the $l-1$ layer. Hence, an L-layer TRANSFORMER-XH can attend over information from up to L hops away.

Together, three main properties equip TRANSFORMER-XH to effectively model raw semi-structured text data: the propagation of information (values) along edges,

the importance of that information (hop attention weights), and the balance of in-sequence and cross-sequence information (attention combination). The representations learned in $\mathcal{H}$ can innately express nuances in semi-structured text that are required for complex reasoning tasks such as multi-hop QA and natural language inference.

4.3 Application to Multi-Hop Question Answering

This section describes how TRANSFORMER-XH applies to the multi-hop QA, in the benchmark dataset HOTPOTQA [Yang et al., 2018]. Given a question q, the task is to find an answer span a in a large open-domain document corpus(we consider the first paragraph of all Wikipedia pages here). By design, the questions are complex and often require information from multiple documents to answer. For example, in the case shown in Figure 4.2, the correct answer Cambridge requires combining the information from both the Wikipedia pages `Facebook` and `Harvard University`. To apply TRANSFORMER-XH in the open domain multi-hop QA task, we first construct an evidence graph and then apply TRANSFORMER-XH on the graph to find the answer.

Evidence Graph Construction. The first step is to find the relevant documents D for the question q and connect them with edges E to form the graph G. Our set D consists of three sources. The first two sources are from canonical information retrieval and entity linking techniques:

- D_{ir}: the top 100 documents retrieved by DRQA's retriever [Chen et al., 2017],

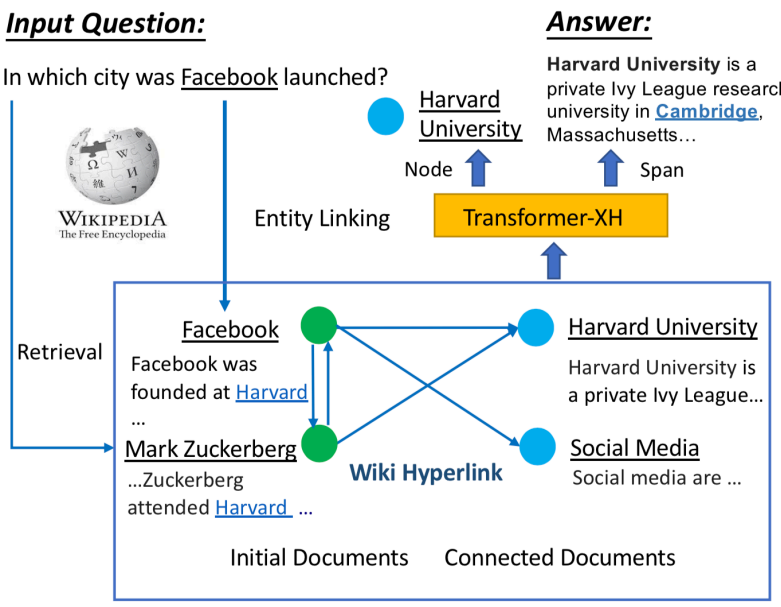


Figure 4.2: One example of TRANSFORMER-XH applying on multi-hop QA. We first construct the question grounded evidence graph, then apply TRANSFORMER-XH to get representations for each node, and finally predict the most relevant node and answer.

which is TF-IDF scoring to the question.

- D_{el}: the Wikipedia documents associated with the entities that appear in the question, annotated by entity linking systems: TAGME [Ferragina and Scaiella, 2010] and CMNS [Hasibi et al., 2017]. We keep the TAGME output entity and three highest scored entities per surface form (a phrase in the question linked with entities) from CMNS, and use its corresponding Wikipedia document as D_{el}.

For better retrieval quality, we use a BERT ranker [Nogueira and Cho, 2019] to re-rank on the set $D_{ir} \cup D_{el}$.The input to the BERT is the concatenation of question

and first paragraph of document:

```
[CLS] Question [SEP] First Paragraph of Document.
```

Then a linear layer is added on the last layer's [CLS] representation to score the relevance of the document. We use BERT base and fine-tune it using the relevance label (from supporting facts labels on the benchmark dataset) with cross-entropy loss. The top two highest scored documents from D_{ir} and the top one document per entity position (surface form) in D_{el} are kept as the first stage BERT IR documents.

Then the third source D_{exp} includes all documents connected to or from any top ranked documents via Wikipedia hyperlinks (e.g., "Facebook" $\rightarrow$ "Harvard University"). A document is included if it is linked to or links to a document in the first two sources. We use the same BERT ranker to rank D_{exp} and keep the top 15 documents in D_{exp}.

The final graph comprises all documents from the three sources as nodes $\mathcal{X}$. The edge matrix E is flexible. We experiment with various edge matrix settings, including directed edges along Wikipedia links, i.e., $e_{ij} = 1$ if there is a hyperlink from document i to j, bidirectional edges along Wiki links, and fully-connected graphs, which rely on TRANSFORMER-XH to learns the edge importance.

Similar to previous work [Ding et al., 2019], the textual representation for each node in the graph is the [SEP]-delimited concatenation of the question, anchor text (the text in the hyperlink in parent nodes pointing to the child node), and the paragraph itself. For example, in Figure 4.2, each node input is: `[CLS] In which city was Facebook launched ? [SEP] Facebook was founded at Harvard University`

`[SEP] Harvard University is a private Ivy League ... .`

Transformer-XH on Evidence Graph. TRANSFORMER-XH takes the input nodes $\mathcal{X}$ and edges E, and produces the global representation of all text sequences:

$$\mathcal{H}^L = \text{TRANSFORMER-XH}(\mathcal{X}, E). \tag{4.7}$$

Then we add two task-specific layers upon the last layer's representation $\mathcal{H}^L$: one auxiliary layer to predict the relevance score of the evidence node, and one layer to extract the answer span within it:

$$p(\text{relevance}|\tau) = \text{softmax}\left(\text{Linear}(\tilde{h}^L_{\tau,0})\right); \tag{4.8}$$

$$p(\text{start}|\tau, i), p(\text{end}|\tau, j) = \text{softmax}\left((\text{Linear}(\tilde{h}^L_{\tau,i})\right), \text{softmax}\left((\text{Linear}(\tilde{h}^L_{\tau,j})\right). \tag{4.9}$$

The final model is trained end-to-end with cross-entropy loss for both tasks in a multi-task setting. During inference, we first select the document with the highest relevance score, and then the start and end positions of the answer within that document.

4.4 Application to Fact Verification

This section describes how TRANSFORMER-XH applies to the fact verification task in FEVER [Thorne et al., 2018]. Given a claim and a trustworthy background corpus, i.e., Wikipedia, the task is to verify whether the evidence in the corpus `SUPPORTS`, `REFUTES`, or there is `NOT ENOUGH INFO` to verify the claim. Similar to multi-hop QA, the first step is to construct an evidence graph using the text pieces

in the background corpus and then TRANSFORMER-XH can be easily applied to conduct reasoning on these evidence pieces.

Evidence Graph Construction. Many previous FEVER systems first retrieve the evidence sentences for the claim and then verify it [Nie et al., 2019a, Zhou et al., 2019]. This first step is similar as the retrieval stage in HOTPOTQA. And the second step is a multi-evidence reasoning task, where TRANSFORMER-XH is applied.

We keep the evidence sentence retrieval step consistent with previous methods. The sentence retrieval results of SR-MRS is not yet released at the time of our experiments, thus we instead use the BERT-based retrieval results from another contemporary work [Liu et al., 2019c].

We construct the evidence graph using the top five sentences from Liu et al. [2019c] as the nodes $\mathcal{X}$ and fully connected edges E. Following Liu et al. [2019c], the representation of each node is the concatenation of the claim, the Wikipedia title (entity name) of the document that includes the sentence, and the evidence sentence.

Transformer-XH on Evidence Graph. TRANSFORMER-XH takes the evidence graph $\{X, E\}$ and learns to verify the claim to three categories: $y \in \{$`SUPPORT`, `REFUSE`, `NOT ENOUGH EVIDENCE`$\}$. Similar to the application in HOTPOTQA, it first produces the global representation of the graph:

$$\mathcal{H}^L = \text{TRANSFORMER-XH}(\mathcal{X}, E). \tag{4.10}$$

Then two task-specific layers are added upon the last layer. The first layer

conducts the fact prediction per node using the "[CLS]" token:

$$p(y|\tau) = \text{softmax}\left(\text{Linear}(\tilde{h}^L_{\tau,0})\right). \tag{4.11}$$

The second layer learns to measure the importance of each node in the graph:

$$p(s|\tau) = \text{softmax}\left(\text{Linear}(\tilde{h}^L_{\tau,0})\right), \tag{4.12}$$

The node level predictions and node importance are combined to the final prediction for the claim:

$$p(y|\mathcal{X}, E) = \sum_{\tau} p(s \mid \tau) \cdot p(y \mid \tau). \tag{4.13}$$

Like the HOTPOTQA scenario, we use multi-task learning that combines the node prediction task and the claim verification task. The first task uses the evidence sentence label provided by FEVER and cross-entropy loss on Eqn. 4.12. The second task uses the final verification label (i.e., `SUPPORT`, `REFUTE`, `NOT ENOUGH INFO`) with cross-entropy loss on Eqn. 4.13.

4.5 Experimental Methodologies

We conduct experiments on HOTPOTQA, the multi-hop question answering benchmark [Yang et al., 2018], and FEVER, the fact verfication benchmark [Thorne et al., 2018].

4.5.1 Multi-Hop Question Answering on HOTPOTQA

Dataset. We review HOTPOTQA in Section 2.4.5.

Metrics. We use official evaluation metrics of HotpotQA: exact match (EM) and F1 on answers (Ans), supporting facts (Supp), and the combination (Joint). The supporting facts prediction is an auxiliary task that evaluates finding the evidence sentences. Joint EM is the product of the two EMs. Joint F1 first multiplies the precision and recall from Ans and Supp, then combines the Joint precision and recall to F1.

Baseline. The main baselines include Cognitive QA (CogQA, [Ding et al., 2019]) and Semantic Retrieval MRS (sr-mrs, [Nie et al., 2019a]). CogQA uses several fine-tuned bert reading comprehension models to find hop entities and candidate spans, and then uses a bert based Graph Convolution Network to rank the candidate spans. sr-mrs is a contemporary work and was the previous leaderboard rank one. It is a bert based pipeline and uses fine-tuned bert models to first rank the documents (twice), then to rank sentences to find supporting facts, and finally conducts bert rc on the concatenated evidence sentences.

We also re-implement CogQA and upgrade its ir with our bert ir model (bert on $D_{ir} \cup D_{el}$, same as Transformer-XH), for fair comparisons. We include other approaches on the FullWiki setting: Official Baseline [Yang et al., 2018], mupet [Feldman and El-Yaniv, 2019], qfe [Nishida et al., 2019], and DecompRC [Min et al., 2019a],

Implementation Details. We first describe the other components for HotpotQA dataset. The whole qa system starts with question type classification. We train

TRANSFORMER-XH separately on each question type over their evidence graph. Besides answer prediction, we also adopt BERT based model for predicting supporting sentences.

- **Question classification**: The first component of our system is to classify the question into bridge and comparison types. We adopt BERT classification fine-tuning setting on HOTPOTQA questions using the question type labels provided in HOTPOTQA. The classifier achieves 99.1% accuracy on the dev set. We use the classifier to split the questions into Comparison and Bridge.

- **Supporting facts classification**: The supporting facts prediction task is to extract all sentences that help get the answer. For bridge questions, these sentences usually cover different pieces of questions. And for comparison questions, the supporting facts are the properties of two question entities. We design one model architecture for this task, but we train two models on each type to reflect the inherent difference. We use BERT as our base model, and on top of BERT, we conduct a multi-task learning scheme. The first task is document relevance prediction, similar as TRANSFORMER-XH, we add a linear layer on the [CLS] token of BERT to predict the relevance score. The other task is the sentence binary classification, we concatenate the first and last token representation of each sentence in the document through a linear layer, the binary output decides whether this sentence is the supporting sentence.

  For bridge questions, we predict supporting facts after answer prediction from TRANSFORMER-XH to resume the inference chain. We start by predicting

supporting facts in the answer document. The other document is chosen from the parents of the answer document in the evidence graph. [2]. Compared with the contemporary model [Nie et al., 2019a], which does not limit the search space along the inference chain (i.e., the answer document may not be relevant to the other supporting page), our method more naturally fits the task purpose.

For comparison questions, after extracting the first step documents D, we simply run this supporting facts prediction model to select the top-2 documents, and predict the corresponding supporting facts.

We use DGL [Wang et al., 2019] for implementing TRANSFORMER-XH and COGQA(W. BERT IR) with batch size 1 (i.e., one graph for each batch), and keep the other parameters same as default BERT setting. We train TRANSFORMER-XH separately on two different types of questions, following previous research [Ding et al., 2019]. We train TRANSFORMER-XH and the GNN of COGQA(W. BERT IR) for 2 epochs. All other BERT based models use the default BERT parameters and train the model for one epoch. The in-sequence attention and other standard Transformer components in TRANSFORMER-XH are initialized by the pre-trained BERT base model [Devlin et al., 2019]. The extra hop attention parameters are initialized randomly and trained from scratch. The final model uses three hop steps. For bridge questions, we build the evidence graph described in Section 4.3. And for comparison questions, we build the fully-connected graph on the set $D_{ir} \cup D_{el}$ and train TRANSFORMER-XH separately.

[2] If the answer document does not have parent node, we choose from all documents

4.5.2 Fact Verification on FEVER

Dataset. The FEVER task provides a claim sentence and requires the system to classify it into three categories: `SUPPORTS`, `REFUTES`, and `NOT ENOUGH INFO`, using the Wikipedia corpus as the evidence source. It provides 185,455 claims with manual labels and uses the Wikipedia dump in June 2017 which includes 5.4 million documents.

Metrics. There are two official evaluation metrics in FEVER: Label Accuracy (LA), which evaluates the classification accuracy of the verification labels, and FEVER Score, which evaluates both the correctness of the evidence sentences used in verification and the LA. The latter is close to Joint EM in HOTPOTQA and is the main metric. We use the official evaluation scripts from FEVER task and we refer to Thorne et al. [2018] for more details.

Experimental Setups. We follow the experiment settings used by previous research in FEVER 1.0 shared task, i.e. Nie et al. [2019a], Zhou et al. [2019], and Liu et al. [2019c]. Similar as Liu et al. [2019c], we also split the data into single and multi evidence categories and evaluate TRANSFORMER-XH on the two splits.

Baselines. The baselines include GEAR [Zhou et al., 2019] and two contemporary work, SR-MRS [Nie et al., 2019a] and KGAT [Liu et al., 2019c]. SR-MRS uses similar adaptations as TRANSFORMER-XH from HOTPOTQA to FEVER. GEAR is a graph attention network based approach specially designed for fact verification. KGAT

further improves GEAR's GAT by adding the kernel information, and is the previous state-of-the-art with BERT base. We also include the BERT Concat baseline Liu et al. [2019c] which concatenates the evidence sentences to a text sequence and applies BERT on it.

Implementation Details. We use the retrieval result from Liu et al. [2019c] and connect all sentences as a fully connected graph. We follow similar parameter settings as HOTPOTQA. We use pre-trained BERT base model to initialize the Transformer components. The extra hop attention parameters are initialized randomly and trained from scratch, and three hop steps are used. We train TRANSFORMER-XH for two epochs.

4.6 Evaluation Results

This section first presents the evaluation results on HOTPOTQA and FEVER. Then it conducts ablation studies, analyses, and case studies on HOTPOTQA to understand the effectiveness of TRANSFORMER-XH.

4.6.1 Overall Result

HotpotQA FullWiki results are presented in Table 4.1. TRANSFORMER-XH outperforms previous methods by significant margins. Besides, TRANSFORMER-XH leads to a much simpler QA system. Previously, in order to utilize pre-trained BERT, HOTPOTQA approaches adapted the multi-hop reasoning task to comprise multiple sub-tasks. For example, given the retrieved documents, COGQA (W. BERT IR) first

	Dev						Test					
	Ans		Supp		Joint		Ans		Supp		Joint	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Official Baseline	23.9	32.9	5.1	40.9	47.2	40.8	24.0	32.9	3.9	37.7	1.9	16.2
DecompRC	-	43.3	-	-	-	-	30.0	40.7	-	-	-	-
QFE	-	-	-	-	-	-	28.7	38.1	14.2	44.4	8.7	23.1
MUPPET	31.1	40.4	17.0	47.7	11.8	27.6	30.6	40.3	16.7	47.3	10.9	27.0
CogQA	37.6	49.4	23.1	58.5	12.2	35.3	37.1	48.9	22.8	57.7	12.4	34.9
SR-MRS*	46.5	58.8	39.9	71.5	26.6	49.2	45.3	57.3	38.7	70.8	25.1	47.6
CogQA (w. BERT IR)	44.8	57.7	29.2	62.8	18.5	43.4	-	-	-	-	-	-
Transformer-XH	**54.0**	**66.2**	**41.7**	**72.1**	**27.7**	**52.9**	**51.6**	**64.1**	**40.9**	**71.4**	**26.1**	**51.3**

Table 4.1: Results (%) on HotpotQA FullWiki Setting. Dev results of previous methods are reported in their papers. Test results are from the leaderboard. Contemporary work is marked by *.

	Question Type				Reasoning Type			
	Comparison		Bridge		Single-Hop		Multi-Hop	
	EM	F1	EM	F1	EM	F1	EM	F1
CogQA	43.3	51.1	36.1	49.0	45.1	61.1	31.1	39.4
SR-MRS*	62.0	68.9	42.4	56.1	52.3	68.4	41.3	50.3
CogQA (w. BERT IR)	54.1	60.9	42.4	56.9	52.0	69.3	38.6	47.8
Transformer-XH (w. BERT IR)	59.9	65.8	**52.4**	**66.3**	**62.2**	**78.3**	**46.8**	**55.7**
Transformer-XH (w. SR-MRS)	**64.3**	**70.7**	47.9	62.3	58.1	74.3	45.3	55.2

Table 4.2: Dev Ans (%) on different scenarios. There are 1487 comparison questions and 5918 Bridge ones. Reasoning Types are estimated by Min et al. [2019c] via whether single-hop BERT has non-zero Ans F1 . There are 3426 single-hop questions and 3979 multi-hop questions.

leverages one BERT RC model to find hop entities and then another BERT RC to find candidate answer spans. After that, it ranks the candidate spans using a BERT based GAT, which is the only structure modeling step. In comparison, Transformer-XH is a unified model which directly represents structured texts and integrates BERT weights.

Table 4.2 further inspects model performances on the Dev set by question types and reasoning types. Transformer-XH significantly outperforms all baselines on bridge questions which require more multi-hop reasoning. And on the "multi-hop" questions, Transformer-XH has higher relative gains (39% over CogQA on EM) than the "single-hop" questions (27%), demonstrating its stronger multi-hop

	Dev		Test		Single Evidence		Multi Evidence	
	LA	FEVER	LA	FEVER	LA	FEVER	LA	FEVER
BERT Concat	73.67	68.89	71.01	65.64	-	-	-	-
GEAR/GAT	74.84	70.69	71.60	67.10	79.79	77.42	66.12	38.21
SR-MRS*	75.12	70.18	72.56	67.26	-	-	-	-
KGAT*	78.02	**75.88**	**72.81**	**69.40**	80.33	78.07	65.92	39.23
TRANSFORMER-XH	**78.05**	74.98	72.39	69.07	**81.84**	**81.31**	**86.58**	**58.47**

Table 4.3: FEVER Results. Contemporary work is marked by *. Single and Multi Evidence are results on Dev claims on which one or multiple sentences are labeled as evidence.

reasoning capability. We further study this in Section 4.6.3.

To further investigate the reasoning ability of TRANSFORMER-XH, we replace our retrieval pipeline with the top retrieved documents from the SR-MRS pipeline. More specifically, we use the top retrieved documents from SR-MRS to construct TRANSFORMER-XH's evidence graph while keeping all else constant. The resulting system, TRANSFORMER-XH (w. SR-MRS), outperforms SR-MRS's multi-step BERT based reasoning on all metrics and question types. TRANSFORMER-XH's effectiveness is robust with multiple IR systems.

FEVER fact verification results are shown in Table 4.3. TRANSFORMER-XH outperforms SR-MRS by 4 FEVER score on Dev and 1.8 on Test. It performs on par with KGAT. More importantly, TRANSFORMER-XH excels at verifying claims that require multiple pieces of evidence–outperforming the contemporary work KGAT by

20 FEVER scores on the multi-evidence claims, a 49% relative improvement. Compared to KGAT, TRANSFORMER-XH mainly loses on the `NOT ENOUGH INFO` category which is neither single nor multi evidence. This is an artifact the FEVER task which our system is not specifically designed for.

This result also demonstrates TRANSFORMER-XH's generality on tasks with multiple semi-structured text inputs. The only difference of TRANSFORMER-XH is the last (linear) task specific layer; it provides similar or better performances over contemporary approaches that were specifically designed for the fact verification task.

4.6.2 Ablation Studies

We use HOTPOTQA to analyze the behavior of TRANSFORMER-XH.

Document Retrieval. We first study the effectiveness and influence of different retrieval settings. We use different numbers of top K ranked documents from the BERT ranker, run TRANSFORMER-XH in the corresponding evidence graph, and evaluate its performance on Bridge questions in the Dev set. We also evaluate the Supporting facts Recall and Answer Recall. Supp Recall evaluates whether the document with the supporting fact is included in the first stage retrieved documents. Ans Recall evaluates whether there exists a document in the evidence graph that includes the ground truth answer. The results are in Table 4.4. Our BERT IR system is better than COGQA's TF-IDF and on par with SR-MRS, as expected. The latter uses a similar retrieval pipeline with our BERT IR system; TRANSFORMER-XH is

Method	Supp Recall	Ans Recall	Dev Ans EM	Dev Ans F1
Top 10 TF-IDF (CogQA)	70.8	n.a.	-	-
Top 2 w. BERT IR + Q Entites	72.3	88.1	**48.7**	**62.6**
Top 5 w. BERT IR + Q Entites	76.5	89.6	47.1	60.8
Top 10 w. BERT IR + Q Entites	**78.9**	**91.1**	46.6	60.3
SR-MRS Top 10 (All Together)	n.a.	86.1	47.9	62.3

Table 4.4: Ablation study on the retrieval systems. Top-10 TF-IDF is the one used by CogQA [Ding et al., 2019]. BERT IR is the retrieval system used by CogQA (w. BERT IR) and Transformer-XH. Top K refers to using the 2/5/10 highest ranked documents from the BERT ranker in the first stage. SR-MRS Top 10 uses the 10 retrieved documents per question provided by Nie et al. [2019a]. All retrieval methods include entities linked in the question and are expanded along Wiki links, except when evaluating the 1st stage Supp Recall.

robust on different retrieval settings and keeps its effectiveness when applied on top 10 documents from SR-MRS (including both stages).

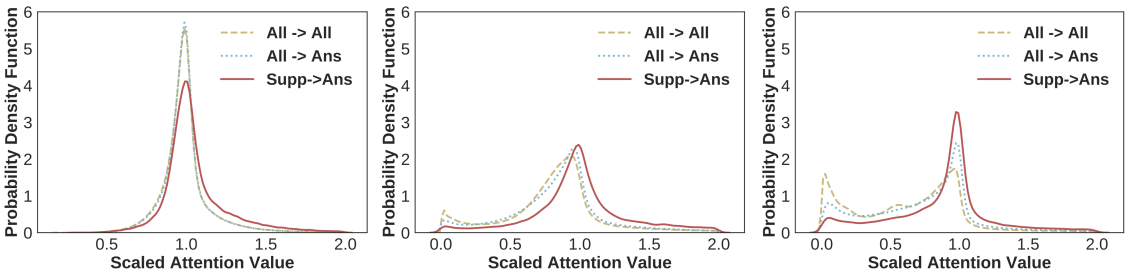
Model Variations. We show the results of different model variations on the top left of Table 4.5. *Single-Hop* BERT uses BERT RC model on each document individually, which significantly decreases the accuracy, confirming the importance of multi-hop reasoning in FullWiki setting [Min et al., 2019a]. GAT + BERT first uses Graph Attention Network [Veličković et al., 2017] on the evidence graph to predict the best node; then it uses BERT RC on the best document. It is 10% worse than

Model Ablation	Dev Ans		Hop Steps	Dev Ans	
	EM	F1		EM	F1
BERT RC on Individual Documents	31.3	42.2	One Hop	50.3	64.6
GAT (Node Prediction) + BERT (Best Node)	48.9	61.9	Two Hops	51.6	66.4
No Node Prediction Multi-Task	43.2	55.3	Four Hops	51.4	66.1
Bidirectional Edges on Hyperlinks	50.6	65.0	Five Hops	50.6	64.7
Fully Connected Graph	51.0	65.5	Six Hops	50.1	64.2
Node Sequence (TRANSFORMER-XL)	14.1	20.7	TRANSFORMER-XH	**52.4**	**66.3**

Table 4.5: Ablation studies on the bridge questions on Dev answer accuracy (%), including model components (top left), graph structures (bottom left), and hop steps (right). TRANSFORMER-XH's full model uses three hop step and unidirectional Wiki link graph.

TRANSFORMER-XH since the RC model has no access to the information from other documents. *No Node Prediction* eliminates the node prediction task and only trains on span prediction task; the accuracy difference shows node prediction task helps the model training.

Graph Structures. We show TRANSFORMER-XH's performance with different graph structures on the bottom left of Table 4.5. *Bidirectional Edges* adds reverse edges along the hyperlinks; *Fully Connected Graph* connects all document pairs; *Node Sequence* randomly permutes the documents and connects them into a sequence to simulate the TRANSFORMER-XL setting. Both *Bidirectional Links* and



(a) First Hop Attentions. (b) Second Hop Attentions. (c) Third Hop Attentions.

Figure 4.3: Distributions of learned attention weights of three hops on three groups: From All (Node) → (to) All, All → (to) Ans (ground truth answer node), and Supp (nodes with the supporting facts) → (to) Ans. X-axes are attention values scaled by number of nodes.

Fully Connected Graph have comparable performance with the original graph structure. TRANSFORMER-XH is able to learn meaningful connections using its hop attentions and is less dependent on the pre-existing graph structures. The fully connected graph can be used if there is no strong edge patterns available in the task. However, the performance drops significantly on *Node Sequence*, showing that structured texts cannot be treated as a linear sequence which cuts off many connections.

Hop Steps. Recall that a TRANSFORMER-XH layer with extra hop attention corresponds to one information propagation step in the graph. Thus TRANSFORMER-XH with last-K layers conducts K-step attention hops in the graph. We show results with different K on the right side of Table 4.5. TRANSFORMER-XH reaches its peak performance with three hops (our full-model). This is expected as most HOTPOTQA

questions can be answered by two documents [Yang et al., 2018].

4.6.3 Hop Attention Analysis

This experiment analyzes the hop attentions using our full-model (three-hop) on the fully connected graph to study their behavior without pre-defined structure. Figure 4.3 plots the distributions of the learned hop attentions on the Dev set. It shows a strong shift away from the normal distribution with more hops. TRANSFORMER-XH learns to distinguish different nodes after multi-hop attention: the attention score becomes a bimodal distribution after three hops, ignoring some non-useful nodes. TRANSFORMER-XH also learns to focus on meaningful edges: the score is higher on the path Supp→Ans than All→Ans. And the margin is larger as the hop step increases from one to three.

4.6.4 Case Study

Table 3.7 lists examples from TRANSFORMER-XH predictions. The second case has too many distractors in the first document. Without additional clues from document 2, it is likely that the single-hop hop entity extraction component in COGQA (w. BERT IR) misses the correct answer document in its candidate sets; and the later structural reasoning component can not recover from this cascade error. In comparison, TRANSFORMER-XH finds the correct answer by combining the evidence with the hop attentions between the two evidence pieces. However, in the second case, the document Algeria at the FIFA World Cup does not have

id	Example	Explanation
1(+)	**Q**: Which man who presented 2022 FIFA World Cup bid was born on October 22, 1930? **A**: Frank Lowy **P**: Frank Lowy ✓ **Document 1**: 2022 FIFA World Cup bid was presented by Frank Lowy, Ben Buckley, Quentin Bryce and Elle Macpherson. **Document 2**: **Frank Lowy** (born 22 October 1930), is an Australian-Israeli businessman and Chairman of Westfield Corporation.	TRANSFORMER-XH correctly finds the correct answer by combining the evidence with the hop attentions between the two evidence pieces.
2(-)	**Q**: Where was the world cup hosted that Algeria qualified for the first time into the round of 16? **A**: Brazil **P**: Spain ✗ **Document 1:**(Algeria at the FIFA World Cup) In 2014, Algeria qualified for the first time into the round of 16. **Document 2:** (2014 FIFA CUP) It took place in Brazil from 12 June to 13 July 2014, after the country was awarded the hosting rights in 2007.	TRANSFORMER-XH does not predict the correct answer, since document 1 does not link to any other documents.Thus, the information does not propagate to the correct answer document 2014 FIFA CUP.

Table 4.6: Examples for model prediction on HOTPOTQA dataset, the first example is the correct prediction (+), the second example is the wrong prediction (-).

links to any other documents, so the information can not be propagated, and using fully-connected graph solves this issue.

4.7 Other Related Work

Fact verification is a natural language inference task while also requires retrieving ("open-domain") and reasoning with multiple text pieces [Thorne et al., 2018, Nie et al., 2019a, Liu et al., 2019c]. Many recent FEVER systems leverage Graph Neural Networks to combine information from multiple text nodes, while each node text is represented by BERT encodings [Zhou et al., 2019, Liu et al., 2019c]. TRANSFORMER-XH is a more unified solution that simply includes language modeling as part of its joint reasoning.

In addition to TRANSFORMER-XL [Dai et al., 2019], other work is proposed to improve the Transformer architecture on long text sequence. For example, T-DMCA [Liu et al., 2018] splits the sequence into blocks and then the attention merges different blocks. Sparse Transformer [Child et al., 2019] introduces the sparse factorizations of the attention matrix. TRANSFORMER-XH shares similar motivation and focuses on multiple pieces of text that are not in sequential forms.

4.8 Conclusion

TRANSFORMER-XH and its eXtra Hop attention mechanism is a simple yet powerful adaptation of Transformer to learn better representations of semi-structured text data as it naturally occurs. It innately integrates with pre-trained language

models to allow for complex reasoning across multiple textual evidence pieces. When applied to HotpotQA, Transformer-XH significantly shrinks the typical multi-hop QA pipeline, eliminating many cascading errors that arise from the linear sequence input constraints of pre-trained Transformers. The same simplicity also applies to FEVER, with one Transformer-XH all we needed to obtain a much stronger answer accuracy.

Chapter 5: Multi-Step Reasoning Over Unstructured Text with Beam Dense Retrieval

In previous two chapters, we propose two QA systems for complex questions that rely on a free-text knowledge graph from Wikipedia. These QA systems first extract a question grounded sub-graph, then use graph neural networks (and further incorporate into Transformer architectures) to model the extracted sub-graph to predict the answer. However, these QA systems depend on a key assumption that only holds on benchmarks: assuming text corpora is semi-structured.

Building on dense retrieval methods, in this chapter, we propose a new multi-step retrieval approach (BeamDR) that iteratively forms an evidence chain through beam search in dense representations. When evaluated on multi-hop question answering, BeamDR is competitive to state-of-the-art systems, *without* using any semi-structured information. [1]

[1] This Chapter describes work published in NAACL 2021 [Zhao et al., 2021b].

5.1 Introduction

Answering complex questions requires combining knowledge pieces through multiple steps into an evidence chain (Ralph Hefferline → Columbia University in Figure 5.1). When the available knowledge sources are graphs or databases, constructing chains can use the sources' inherent structure. However, when the information needs to be pulled from unstructured text (which often has better coverage), standard information retrieval (IR) approaches only go "one hop": from a query to a single passage.

Recent approaches [Dhingra et al., 2020, Zhao et al., 2020a,b, Asai et al., 2020, *inter alia*] try to achieve the best of both worlds: use the unstructured text of Wikipedia with its structured hyperlinks. While they show promise on benchmarks, it's difficult to extend them beyond academic testbeds because real-world datasets often lack this structure. For example, medical records lack links between reports.

Dense retrieval [Lee et al., 2019, Guu et al., 2020, Karpukhin et al., 2020, *inter alia*] provides a promising path to overcome this limitation. It encodes the query and evidence (passage) into dense vectors and matches them in the embedding space. In addition to its efficiency—thanks to maximum inner-product search (MIPS)—Xiong et al. [2021a] show that dense retrieval rivals BERT [Devlin et al., 2019]-based (sparse) retrieve-then-rerank IR pipelines on single step retrieval. Unlike traditional term-based retrieval, fully learnable dense encodings provide flexibility for different tasks.

This chapter investigates a natural question: can we build a retrieval system

Question: Ralph Hefferline was a psychology professor at a university that is located in what city?
Evidence Chain: Ralph Hefferline -> Columbia University

P1: Ralph Hefferline
Ralph Franklin Hefferline was a psychology professor at Columbia University.

P2: Columbia University
Columbia University is a private Ivy League research university in Upper Manhattan, New York City.

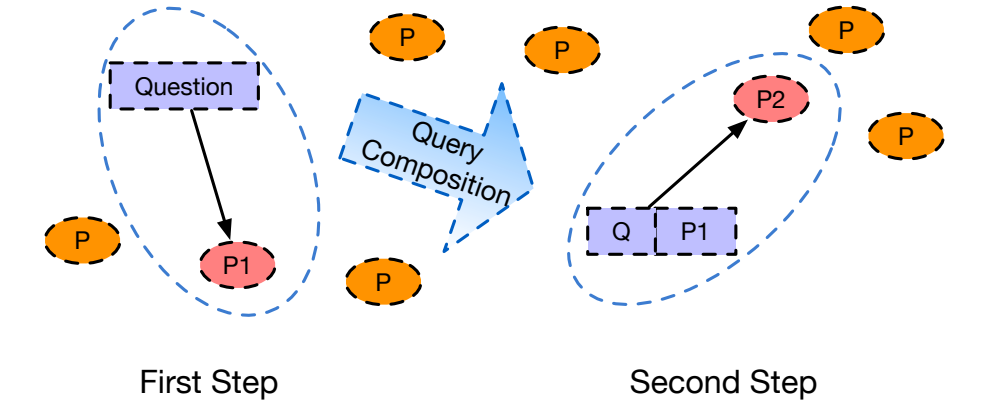


Figure 5.1: Top: A complex question example from HotPotQA that requires finding an evidence chain. Bottom: BeamDR iteratively composes the new query and retrieves evidence in dense space without the need for linked documents.

to find an evidence chain on unstructured text corpora? We propose a new multi-step dense retrieval method to model the implicit relationships between evidence pieces. We use beam search (Section 5.2) in the dense space to find and cache the most relevant candidate chains and iteratively compose the query by appending the retrieval history. We improve the retrieval by encouraging the representation to discriminate hard negative evidence chains from the correct chains, which are refreshed by the model.

We evaluate **Beam** **D**ense **R**etrieval (BeamDR) on HotPotQA [Yang et al., 2018], a multi-hop question answering benchmark. When retrieving evidence chains

directly from the corpus (full retrieval), BeamDR is competitive to the state-of-the-art cascade reranking systems that use Wikipedia links. Combined with standard reranking and answer span extraction modules, the gain from full retrieval propagates to findings answers (Section 5.3). By iteratively composing the query representation, BeamDR captures the hidden "semantic" relationships between the evidence pieces (Section 5.4).

5.2 BeamDR: Beam Dense Retriever

We focus on finding an evidence chain from an unstructured text corpus for a given question, often the hardest part of complex question answering. We formulate it as a multi-step retrieval problem. Formally, given a question q and a corpus C, the task is to form an ordered evidence chain $p_1 ... p_n$ from C, with each evidence a passage. We focus on the supervised setting, where the labeled evidence set is given during training (but not during testing).

Finding an evidence chain from the corpus is challenging because: 1) passages that do not share enough words are hard to retrieve (e.g., in Figure 5.1, the evidence Columbia University); 2) if you miss one evidence piece, you may err on all that come after.

Our proposed method BeamDR is based on dense retrieval, which we review in Section 2.4.3. We first introduce scoring a single evidence chain, then finding the top k chains with beam search, and finally training BeamDR.

Evidence Chain Scoring The score S_n of evidence chain $p_1, \ldots, p_n$ is the product of the (normalized) relevance scores of individual evidence pieces. At each retrieval step t, to incorporate the information from both the question and retrieval history, we compose a new query q_t by appending the tokens of retrieved chains $p_1, \ldots, p_{t-1}$ to query q ($q_t = [q; p_1; \ldots; p_{t-1}]$), we use MIPS to find relevant evidence piece p_t from the corpus and update the evidence chain score S_t by multiplying the current step t's relevance score $f(q_t, p_t) * S_{t-1}$.

Beam Search in Dense Space Since enumerating all evidence chains is computationally impossible, we instead maintain an evidence cache. In the structured search literature this is called a *beam*: the k-best scoring candidate chains we have found thus far. We select evidence chains with beam search in dense space. At step t, we enumerate each candidate chain j in the beam $p_{j,1}...p_{j,t-1}$, score the top k chains and update the beam. After n steps, the k highest-scored evidence chains with length n are finally retrieved.

Training BeamDR The goal of training is to learn embedding functions that differentiate positive (relevant) and negative evidence chains. Since the evidence pieces are unordered, we use heuristics to infer the order of evidence chains. A negative chain has at least one evidence piece that is not in the gold evidence set. For each step t, the input is the query q, a positive chain $P_t^+ = p_1^+, \ldots, p_t^+$ and m sampled negative chains $P_{j,t}^- = p_1^-, \ldots, p_t^-$. We update the negative log likelihood

(NLL) loss:

$$L(q, P^+, P_1^-, ..., P_m^-) \tag{5.1}$$
$$= \sum_t \frac{e^{f([q;P_{t-1}^+],p_t^+)}}{e^{f([q;P_{t-1}^+],p_t^+)} + \sum_{j=1}^m e^{f([q;P_{j,t-1}],p_{j,t}^-)}}.$$

Rather than using local in-batch or term matching negative samples, like Guu et al. [2020] we select negatives from the whole corpus, which can be more effective for single-step retrieval [Xiong et al., 2021a]. In multi-step retrieval, we select negative evidence chains from the corpus. Beam search on the training data finds the top k highest scored negative chains for each retrieval step. Since the model parameters are dynamically updated, we asynchronously refresh the negative chains with the up-to-date model checkpoint [Guu et al., 2020, Xiong et al., 2021a].

5.3 Experiments: Retrieval and Answering

Our experiments are on HotPotQA fullwiki setting [Yang et al., 2018], the multi-hop question answering benchmark. We mainly evaluate on retrieval that extracts evidence chains (passages) from the corpus; we further add a downstream evaluation on whether it finds the right answer.

5.3.1 Experimental Setup

Metrics Following Asai et al. [2020], we report four metrics on retrieval: answer recall (AR), if answer span is in the retrieved passages; passage recall (PR), if at least one gold passage is in the retrieved passages; Passage Exact Match (P EM), if both gold passages are included in the retrieved passages; and Exact Match (EM),

whether both gold passages are included in the top two retrieved passages (top one chain). We report exact match (EM) and F_1 on answer spans.

Implementation We use a BERT-base encoder for retrieval and report both BERT base and large for span extraction. We warm up BeamDR with TF-IDF negative chains. The retrieval is evaluated on ten passage chains (each chain has two passages). To compare with existing retrieve-then-rerank cascade systems, we train a standard BERT passage reranker [Nogueira and Cho, 2019], and evaluate on ten chains reranked from the top 100 retrieval outputs. We train BeamDR on six 2080Ti GPUs, three for training, three for refreshing negative chains. We do not search hyper-parameters and use suggested ones from Xiong et al. [2021a].

5.3.2 Passage Chain Retrieval Evaluation

Baselines We compare BeamDR with TF-IDF, Semantic Retrieval [Nie et al., 2019b, SR], which uses a cascade BERT pipeline, and the Graph recurrent retriever [Asai et al., 2020, GRR], our main baseline, which iteratively retrieves passages following the Wikipedia hyperlink structure, and is state-of-the-art on the leaderboard. We also compare against a contemporaneous model, multi-hop dense retrieval [Xiong et al., 2021b, MDR].

Results: Robust Evidence Retrieval without Document Links Table 5.1 presents retrieval results. On full retrieval, BeamDR is competitive to GRR, state-of-the-art *reranker* using Wikipedia hyperlinks. BeamDR also has better retrieval

Models	AR	PR	P EM	EM
Full Retrieval				
TF-IDF	39.7	66.9	10.0	18.2
MDR *	75.4	-	65.9	-
BeamDR (IR Neg)	76.8	86.4	64.1	40.4
BeamDR (Greedy)	83.6	90.7	72.7	34.1
BeamDR (Ours)	**87.0**	**92.9**	**79.2**	**60.7**
Reranking from Retrieval Outputs				
SR	77.9	93.2	63.9	46.5
GRR	87.8	93.3	77.9	61.1
MDR *	88.2	-	81.2	-
BeamDR (Ours)	**90.7**	**94.7**	**83.7**	**70.7**

Table 5.1: Compare BeamDR with other retrieval systems. Top: Retrieval from the whole corpus, bottom: Reranking from top 100 full retrieval outputs. * indicates parallel work.

than the contemporaneous MDR. Although both approaches build on dense retrieval, MDR is close to BeamDR with TF-IDF negatives. We instead refresh negative chains with intermediate representations, which help the model better discover evidence chains. Our ablation study (Greedy search) indicates the importance of maintaining the beam during inference. With the help of cross-attention between the question and the passage, using BERT to rerank BeamDR outperforms all baselines.

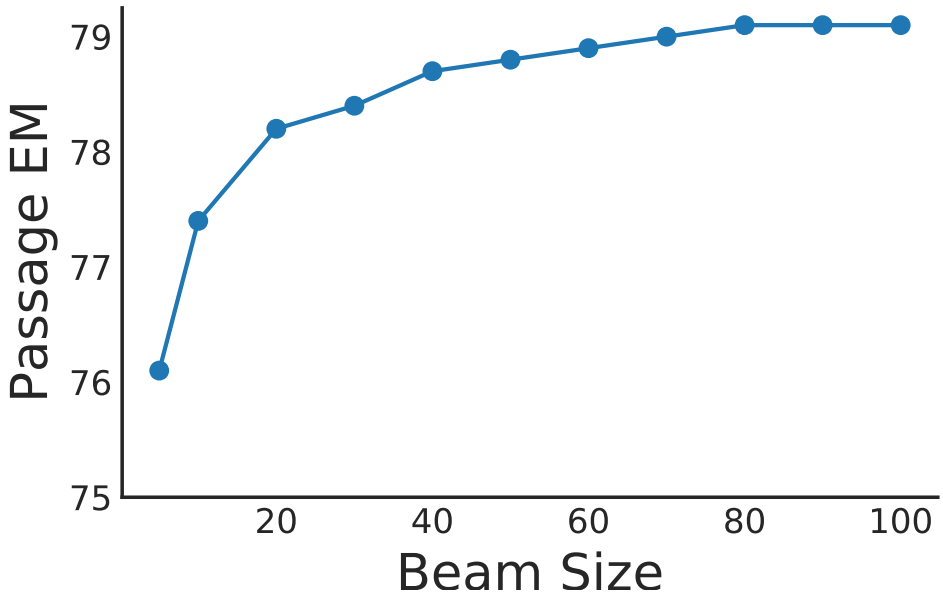


Figure 5.2: Passage retrieval accuracy on different beam size. Our system is robust to the increase of beam size.

Varying the Beam size Figure 5.2 plots the Passage EM with different beam sizes. While initially increasing the beam size improves Passage Exact Match, the marginal improvement decreases after a beam size of forty.

5.3.3 Answer Extraction Evaluation

Baselines We compare BeamDR with TXH (Chapter 4), GRR [Asai et al., 2020] and the contemporaneous MDR [Xiong et al., 2021b]. We use released code from GRR [Asai et al., 2020] following its settings on BERT base and large. We use four 2080Ti GPUs.

Retriever	Reader	Dev		Test	
		EM	F1	EM	F1
BERT base Reader					
TXH	TXH	54.0	66.2	51.6	64.1
GRR	GRR	52.7	65.8	-	-
BeamDR	GRR	**54.9**	**68.0**	-	-
BERT large wwm Reader					
GRR	GRR	60.5	73.3	60.0	73.0
BeamDR	GRR	61.3	74.1	60.4	73.2
MDR*	MDR*	**61.5**	**74.7**	-	-
ELECTRA large Reader					
MDR*	MDR*	63.4	76.2	62.3	75.3

Table 5.2: HotPotQA dev and test set answer exact match (EM) and F1 results.

* indicates parallel work.

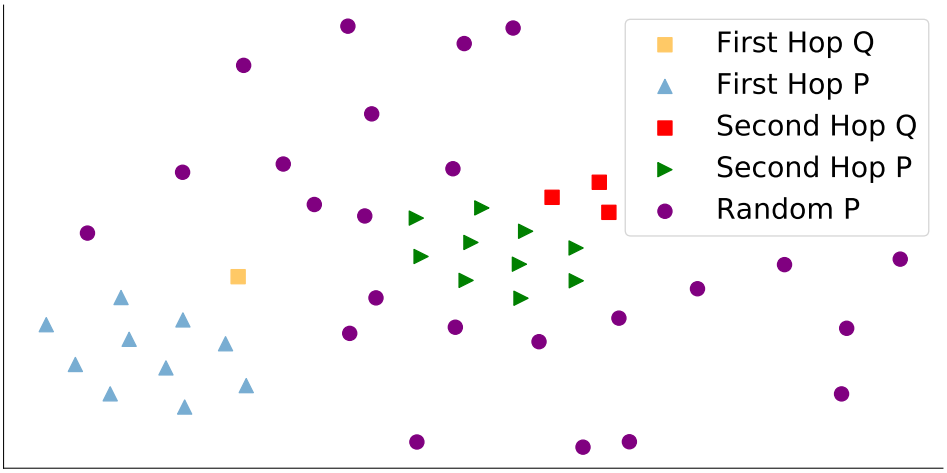


Figure 5.3: T-SNE visualization of query (Q) and passage (P) embeddings over different retrieval steps. BEAMDR conducts multi-step reasoning by hopping in the learned representation space.

Results Using the same implementation but on our reranked chains, BEAMDR outperforms GRR (Table 5.2), suggesting gains from retrieval could propagate to answer span extraction. BEAMDR is competitive with MDR but slightly lower; we speculate different reader implementations might be the cause.

5.4 Exploring How we Hop

In this section, we explore how BEAMDR constructs evidence chains.

5.4.1 Qualitative Analysis

Figure 5.3 shows query and passage representations with T-SNE [Maaten and Hinton, 2008]. Unsurprisingly, in the dense space, the first hop query (question) is close to its retrieved passages but far from second hop passages (with some negative

Models	Passage Recall		Overlap
	First hop	Second hop	
GRR	85.1	85.3	64.3
BeamDR	86.4	78.9	26.7
BeamDR†	88.0	87.1	14.7

Table 5.3: Passage Recall and overlap comparison between BeamDR and GRR with different hop passages. Systems with † filter second hop passages with links.

passages in between). After composing the question and first hop passages, the second hop queries indeed land closer to the second hop passages. Our quantitative analysis (Table 5.3) further shows BeamDR has little overlap between retrieved passages in two hops. BeamDR mimics multi-step reasoning by hopping in the learned representation space.

5.4.2 Hop Analysis

To study model behaviors under different hops, we use heuristics[2] to infer the order of evidence passages. In Table 5.3, BeamDR slightly wins on first hop passages, with the help of hyperlinks, GRR outperforms BeamDR on second hop retrieval. Only 21.9% of the top-10 BeamDR chains are connected by links.

[2]We label the passage that contains the answer as the second hop passage, while the other one as the first hop passage. If both passages include the answer, passage title mentioned in the question is the first hop passage.

Errors	Type	%
GRR	Question entities	62
	Connect with reverse links	16
	Text matching	14
	Others	8
BeamDR	Text matching	46
	No links between passages	39
	Question entities	15

Table 5.4: We manually analyze 100 bridge questions and categorize model errors. BeamDR wins after using links to filter candidates.

5.4.3 Human Evaluation on Model Errors and Case Study

To understand the strengths and weaknesses of BeamDR compared with GRR, we manually analyze 100 bridge questions from the HotPotQA development set. BeamDR predicts fifty of them correctly and GRR predicts the other fifty correctly (Tables 5.4 and 5.5).

Strengths of BeamDR. Compared to GRR, the largest gain of BeamDR is to identify question entity passages. As there is often little context overlap besides the entity surface form, a term-based approach (TF-IDF used by GRR) falters. Some of the GRR errors also come from using reverse links to find second hop passages (i.e., the second hop passage links to the first hop passage).

Q: Chris Williams last played for which football club from the National League North?
Passage 1: Christopher Jonathan "Chris" Williams is an English semi-professional footballer who last played for Salford City as a forward.
Passage 2: Salford City Football Club is a professional football club in the Kersal area of Salford, Greater Manchester, England.
BeamDR: Chris Williams (English Footballer) → Salford City F.C. ✓
grr: Chris Williams (Wide Receiver) → Miami Dolphins ✗

Table 5.5: Case study of BeamDR and GRR retrieval. Term-based retrieval approaches (TF-IDF used by GRR) is unable to distinguish two players with same name. BeamDR correctly identifies the question entity.

Weaknesses of BeamDR. Like Karpukhin et al. [2020], many of BeamDR's errors could be avoided by simple term matching. For example, matching *"What screenwriter with credits for Evolution co-wrote a film starring Nicolas Cage and Téa Leoni?"* to the context *"The Family Man is a 2000 American film written by David Diamond and David Weissman, and starring Nicolas Cage and Téa Leoni."*.

5.5 Conclusion

We introduce a simple yet effective multi-step dense retrieval method, BeamDR. By conducting beam search and globally refreshing negative chains during training, BeamDR finds reasoning chains in dense space. BeamDR is competitive to more

complex SOTA systems albeit not using semi-structured information.

While BeamDR can uncover the relationship embedded within a single question, future work should investigate how to use these connections to resolve ambiguity in the question [Elgohary et al., 2019, Min et al., 2020a], resolve entity mentions [Guha et al., 2015], connect concepts across modalities [Lei et al., 2018], or to connect related questions to each other (Section 3.2).

Chapter 6: Distantly-Supervised Evidence Retrieval Enables Question Answering without Annotated Evidence Pieces

In the previous chapter, we propose a new retrieval method that is competitive to state-of-the-art approaches, without using corpus semi-structured metadata, a key assumption that does not hold in every text corpora. However, state-of-the-art neural approaches still require intermediate evidence annotations for training. such intermediate annotations are expensive, and methods that rely on them cannot transfer to the more common setting, where only question-answer pairs are available. This chapter investigates whether models can learn to find evidence from a large corpus, with only distant supervision from answer labels for model training, thereby generating no additional annotation cost. We introduce a novel approach (DistDR) that iteratively improves over a weak retriever by alternately finding evidence from the up-to-date model and encouraging the model to learn the most likely evidence. Without using any evidence labels, DistDR is on par with fully-supervised state-of-the-art methods on both multi-hop and single-hop QA benchmarks. Our analysis confirms that DistDR finds more accurate evidence over iterations, which leads to model improvements. [1]

[1]This Chapter describes work published in EMNLP 2021 [Zhao et al., 2021a].

6.1 Introduction

Open-domain question answering (ODQA) takes a question, retrieves evidence from a large corpus, and finds an answer based on that evidence [Voorhees et al., 1999]. With the help of large scale datasets, state-of-the-art approaches to QA [Karpukhin et al., 2020, *inter alia*] can answer both simple questions that require only a single evidence piece (i.e., one passage); and more challenging multi-hop questions: computers must jump or "hop" from passage to passage (we call these passages **evidence pieces**), building a reasoning chain to find the answer.

State-of-the-art (SOTA) methods, however, are trained with *all* of the intermediate evidence pieces (e.g., in Figure 6.1, the evidence pieces for Sang-Wook Cheong's workplace which point you to Rutgers University's location) needed for the answer. Creating such intricate training data is expensive. For example, Kwiatkowski et al. [2019] use additional experts to justify the correctness of annotated evidence. The annotation protocol is even more nuanced for multi-hop questions. For example, Yang et al. [2018] ask annotators to write multi-hop questions based on two linked Wikipedia passages as a pre-defined reasoning chain, which creates dataset artifacts [Min et al., 2019b]. While plenty of question-answer pairs are available without evidence labels, we cannot directly train SOTA models on such data.

Our work focuses on training ODQA systems without these expensive annotations (Section 6.2): we only start with a question-answer pair. With that starting point, we use distant supervision to infer which evidence helps us get to the answer. The technical challenge is how to find evidence from millions of candidates.

Question: What state does ***Sang-Wook Cheong*** work as a ***materials scientist***?
Answer: New Jersey
Annotated Evidence: Sang-Wook Cheong -> Rutgers University **Alternate Evidence:** Sang-Wook Cheong -> History of Rutgers University
Evidence Piece 1: Sang-Wook Cheong ***Sang-Wook Cheong*** is a Korean American ***materials scientist*** at Rutgers University.
Evidence Piece 2: Rutgers University Rutgers, The State University of New Jersey, commonly referred to as Rutgers University, Rutgers, or RU, is an American public research university and the largest institution for higher education in ***New Jersey***.
Evidence Piece 3: History of Rutgers University Rutgers University is an institution of higher learning with campuses across the State of ***New Jersey.*** Its main flagship campus locates in New Brunswick and Piscataway, ***New Jersey***.

Figure 6.1: A multi-hop question example from HotPotQA that requires finding multiple evidence pieces to form a reasoning chain (Sang-Wook Cheong → Rutgers University). Red: Text that overlaps between question and evidence piece; Blue: Span that matches the answer. State-of-the-art systems use evidence labels for training, but acquiring labeled evidence pieces is expensive.

Previous methods [Joshi et al., 2017a, Cheng et al., 2020] use term matching (e.g., TF-IDF) for evidence retrieval, but their goal is a single piece of evidence: linking a question to a passage. As shown in Figure 6.1, the key to finding some evidence pieces does not appear in the question: for example, you only know to figure out that Rutgers University is in New Jersey after learning where Professor Cheong works. Fortunately, navigating to an answer given a question from a search engine is not impossible: humans do it every day, building on their existing knowledge toward

the answer [Russell, 2019]. With each round of searching to find additional clues, human users accumulate information to find the right answer. This chapter creates a computational approach (DistDR) that can use similar techniques to find the evidence needed to answer a given question.

DistDR starts with a weak retriever then iteratively improves it by finding more useful evidence (Section 6.3). Specifically, we model evidence as a latent variable and develop a hard-EM algorithm that alternates between using the up-to-date retriever to find evidence (hard E-step) and updating the model parameters to further encourage the most useful evidence in the next iteration (M-step).

To implement this idea, we need a trainable retrieval system. We use dense retrieval [Lee et al., 2019] as our retriever, which uses a neural network to encode the evidence pieces we collect at each round into query vectors. DistDR provides iterative feedback within the context of a QA system to guide the encoder to better find evidence pieces.

We evaluate DistDR on ODQA benchmarks, including both single-hop questions [Kwiatkowski et al., 2019, NaturalQuestions], where the evidence is the target passage, and multi-hop questions [Yang et al., 2018, HotPotQA], where the evidence is a chain of passages. *Without* using any annotated evidence labels, DistDR's accuracy, according to several measures, is *on par* with fully-supervised state-of-the-art approaches on both benchmarks (Section 6.4).

Our analyses confirm the intuition that over iterations, DistDR selects more accurate evidence, which in turn improves the model. Although some retrieved evidence from DistDR does not match the annotation, it gives useful training signal, as

DistDR finds *alternative* evidence—another advantage of an automated approach (Section 6.5). For example, you can connect Sang-Wook Cheong to New Jersey through another Wikipedia page (History of Rutgers University in Figure 6.1).

6.2 Why Weakly-Supervised ODQA

Our task is to answer questions over large textual corpora. Our approach is generally applicable to both single-hop and multi-hop questions. We use Wikipedia as the knowledge source, but we do not use the metadata such as hyperlinks, to ensure our method can apply to any corpus (e.g., ClueWeb).

State-of-the-art approaches on ODQA mainly focus on the *fully-supervised* setting, where the question, answer, and evidence are given in training. Figure 6.1 shows an example multi-hop question with answer and annotated evidence. The fully-supervised setting simplifies model training but has three major challenges: (1) Annotation is costly: the question–answer pair is widely available in the real world, but the evidence (Sang-Wook Cheong → Rutgers University in Figure 6.1) requires human annotation, which is expensive to get, especially for complex questions; (2) Domain generalization: the labeled evidence is only on a single corpus, generalization to other corpora (e.g., moving from Wikipedia to medical records) is non-trivial; (3) Alternative evidence: there are often multiple correct evidence candidates in the corpora (e.g., in Figure 6.1, both Sang-Wook Cheong → Rutgers University and Sang-Wook Cheong → History of Rutgers University are correct), but only one of them is annotated as "gold evidence". Therefore, we explore the more common

but challenging *weakly-supervised* setting which only requires question–answer input pairs.

Formally, given a question q and a textual corpus with D passages, our task is to first find a small subset of relevant passages as evidence z to the answer, where each evidence combines evidence pieces $z = z_1, ..., z_n$ with length n ($n = 1$ for single-hop questions), then find a span a from evidence z as an answer. We focus on the weakly-supervised setting, where training examples consist only of question–answer pairs (q, a), we gather evidence $\hat{z}$ from corpus as the distant supervision signal, based on the assumption that the presence of the answer a (source of distant supervision) in an evidence piece implies that the evidence is needed to answer the question. This is contrast to the fully-supervised setting, where the gold evidence z^* is also given during training.

6.3 Weakly-Supervised ODQA with DistDR

We present DistDR, a unified framework for weakly-supervised ODQA. DistDR is trained by retrieving evidence from a large corpus with distant supervision.

DistDR follows the retriever–reader framework for ODQA, using dense retrieval to find evidence (Section 2.4.2). However, in our approach, we only have questions q and answers a,x so we need to induce evidence $\hat{z}$ for training our retriever. We fully expect that our initial retriever will struggle to find evidence. However, if it can find *some* useful evidence, we can encourage it to follow the same clues to more evidence that can answer questions. This intuition is the foundation

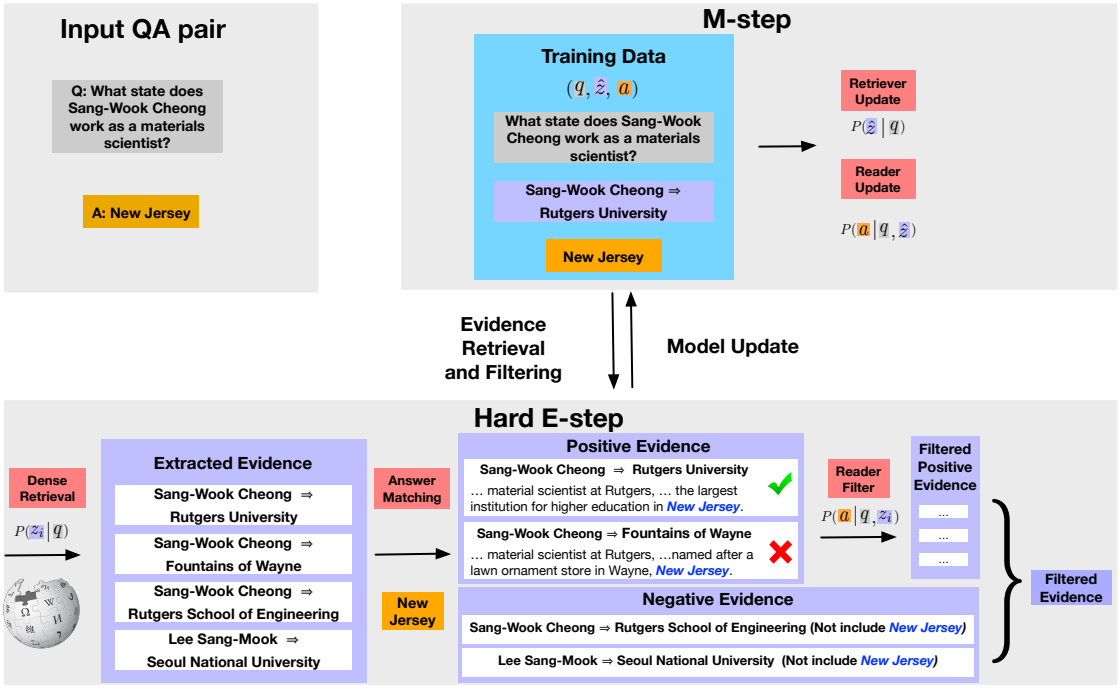


Figure 6.2: Our DistDR model, with question: q, evidence: z, answer: a, model component: Dense Retrieval. ✔indicates reader output matches the correct answer, thus the positive evidence is kept (otherwise filtered out, presented by ✗). Top: at M-step, DistDR updates both retriever and reader components using the training data from E-step as distant supervision. Bottom: at E-step, DistDR finds the most relevant evidence using the current dense retriever on the training examples, uses both answer string matching and reader filter to form positive and negative evidence.

for our iterative approach (Section 6.3.1) for evidence retrieval: an initial retriever attempts to find evidence (Dense Retrieval). If the evidence contains the answer (Answer Matching), we label it as positive evidence (otherwise it's negative evidence); then we use the retrieved evidence as labels to retrain the retriever and reader (Model Update). While this idea forms the basis of our algorithm, it can

be led astray by false-positives: evidence that contains the answer but is irrelevant to question. Returning to our running example, while "named after a lawn ornament store in Wayne, New Jersey" has the state where Professor Cheong works, it is irrelevant to condensed matter physics. Thus, we use a reader to filter spurious evidence (Reader Filter) and keep DistDR on target.

6.3.1 Learning DistDR: Hard-EM

This section introduces training DistDR with weak supervision. Our QA system follows state-of-the-art retrieval-reader pipeline. We use dense retrieval model as retrieval, it's a standard dense retrieval model for simple questions (Section 2.4.3), or BeamDR for complex questions (Chapter 5). Our reader—multi-tasked with evidence reranking and span extraction—is a fine-tuning BERT model (Section 2.4.4).

We treat the evidence z as a latent variable (Figure 6.2) with a multinomial distribution (parameterized by probability of selecting correct evidence from the retriever) over solution set $Z = D \times n$, where D is the corpus size and n is the number of retrieval steps. Formally, given the question q and answer a, our goal is to maximize the probability of finding correct evidence $P(z \mid q)$ (For single-hop questions, z is retrieved with a single step; for multi-hop questions, it takes multiple steps to find z: after the first retrieval step, the query q is not just the original question but also appended evidence.), and selecting an answer from the question $P(a \mid q, z)$. We use expectation maximization (EM) to infer the latent variable z. We compute the likelihood of each z given q, yielding a vector of estimates $\hat{z}$ (E-

step); then update the model parameters based on $\hat{z}$ (M-step). Since it's intractable to enumerate all evidence candidates to compute the expectation, we adopt hard-EM [Samdani et al., 2012] and approximate the E-step by picking the most likely solution as $\hat{z}$. We pass over all questions in the training set and repeat this process for multiple iterations until the model converges.

Hard E-step At the hard E-step, for each question q in the training set we find the most likely estimated evidence $\hat{z}$. This is implemented by multiple retrieval steps for multi-hop questions, specifically at each step t:

$$\hat{z}_t = \arg\max_{\hat{z} \in Z} P(\hat{z}_t \,|\, q, \hat{z}_1, ..., \hat{z}_{t-1}); \tag{6.1}$$

and single-step retrieval for single-hop questions. We use an up-to-date retriever to find the top-k evidence from corpus (with beam search for the k-highest scoring chains, Chapter 5).

Given the retrieval output, we will eventually need to retrain the retriever. This requires knowing which evidence is useful and which is not. As a proxy, we look for the *answer* to split the top-k candidate evidence into positive (has the answer) $\hat{Z}^+$ and negative $\hat{Z}^-$ evidence (lacks the answer) sets. And as a by-product, we generate the most challenging negative evidence at each iteration, which makes training more robust [Guu et al., 2020].

Evidence Filter Although using the answer to filter evidence ensures that the positive evidence contains the answer, it does not always mean this evidence is relevant. For example, the answer to "Who played in the most world series MLB

games" is New York Yankees, but "New York Yankees is an American professional baseball team" is not the *correct* evidence. This issue is more pronounced at the beginning of the process when the retriever is weaker. To mitigate this issue, the reader filters spurious positives: if it does not believe New York Yankees is the answer to the question, the evidence is not usable. Specifically, for each evidence $\hat{z}^+$ in the positive evidence set, the current reader model outputs the most likely answer $\hat{a}$. We only keep positive evidence if $\hat{a}$ matches the correct answer.

M-step Now that we have our estimated evidence $\hat{z}$, including both the highest scored positive evidence $\hat{z}^+$ (after filtering), which we assume is a true solution, and $\hat{Z}^-$ as negative evidence set. We have $(q, \hat{z}, a)$ for each training example to update both the retriever $P(\hat{z} \mid q)$ and reader $P(a \mid q, \hat{z})$.

6.4 Experiments

In this section, we evaluate DistDR on both multi-hop and single-hop QA benchmarks. DistDR is generally applicable to both questions by adopting different evidence retrieval steps.

6.4.1 Datasets

We evaluate on two datasets, HotPotQA and NaturalQuestions.

HotPotQA [Yang et al., 2018] is a multi-hop QA benchmark. We focus on the fullwiki setting. We do not use its supporting facts (evidence) annotation in our

setting. We only use its bridge questions subset, which is designed to be multi-hop. HotPotQA also includes comparison questions that compare properties of two question entities, but its yes/no answers are beyond the scope of this paper, as we cannot get distant supervision from them.

NaturalQuestions [Kwiatkowski et al., 2019] is a QA benchmark, which mainly includes single-hop questions. Besides questions and answers, NaturalQuestions also annotates passages as evidence, but we do not use it in the weakly-supervised setting. We follow Karpukhin et al. [2020] and use all of Wikipedia as a corpus, split into 100-token chunks (21 million passages).

6.4.2 Evaluation Metrics

On HotPotQA, we evaluate the retrieval component on ten evidence (chains), where each sequence has two passages. For retrieval, we follow Zhao et al. [2021b] and report **answer recall** (the fraction of questions with the answer string in the retrieved passages), **passage recall** (if at least one gold passage is in the retrieved passages), and **chain recall** (if both gold passages are included in the retrieved passages) on the dev set. For the reader, we first use the same metrics as above on the top ten chains reranked from top-100 retrieval results for reranking. Then we report an exact match (EM) score on answer spans. On NaturalQuestions, we report answer recall on top-k passages ($k = 1, 20$) from the retriever, and exact match (EM) on answer spans on test set, following Karpukhin et al. [2020].

6.4.3 Compared Methods

On HotPotQA retrieval, we compare DistDR with unsupervised tf-idf, and two recent state-of-the-art multi-step dense retrieval methods—BeamDR (Chapter 5) and mdr [Xiong et al., 2021b]—under full supervision. For the reader, we first compare DistDR with BeamDR and two other top leaderboard entries, Transformer-xh (Chapter 4) and grr [Asai et al., 2020], both of which use Wikipedia hyperlinks to find candidates. For fair comparison, all approaches use bert-base as pre-trained model, and we use the released model checkpoints to do inference on bridge question subsets.[2]

On NaturalQuestions, we compare DistDR with two state-of-the-art dense retrieval methods, dpr [Karpukhin et al., 2020] and ance [Xiong et al., 2021a], which use same model architecture as dpr, with asynchronous negative evidence updates during training. We evaluate on fully-supervised and weakly-supervised settings. We directly use the released model checkpoint on full supervision and train models from published code and data[3] on distant supervision. All approaches use bert-base as pre-trained model.

[2] mdr uses RoBERTa-base for retrieval, and bert/electra-large for reranking and span extraction. We include the retrieval results (though it gives slight gains) but do not compare the reader. We expect the reader results are close to BeamDR, as both use similar models.

[3] The released data uses answers to match top bm-25 results for distant supervision.

6.4.4 Implementation details

On HotPotQA, we initialize our retriever with a dense retrieval checkpoint from NaturalQuestions for both hops.[4] To initialize the reader, we first run DistDR's retrieval for one iteration (without the reader filter), and train the reader from scratch, using the top-50 retrieval outputs.[5] We use the same hyper-parameters as BeamDR, and train DistDR for eight iterations.[6] On NaturalQuestions, we initialize DistDR from a DPR checkpoint under distant supervision (so the evidence label is not used). We train DistDR for ten iterations, using the same hyper-parameters as DPR [Karpukhin et al., 2020]. We run DistDR on eight 2080Ti GPUs, and training takes three days.

6.4.5 Main Results

Table 6.1 presents retrieval results on HotPotQA. Here, TF-IDF is only able to find one evidence piece (usually the first hop which overlaps with question), but fails to find all the evidence pieces. Hence, using evidence from TF-IDF as distant supervision cannot effectively train DistDR. DistDR is slightly better

[4]We make this choice because we need a cold starting point for our EM process (i.e., at the first iteration, find some training signals from the retrieved evidence). The initialization on HotPotQA does not use any evidence labels (which would be cheating in our setting). There are potentially alternative approaches to initializations that could be considered for future work.

[5]We follow the reader setup from Karpukhin et al. [2020]: if multiple positives are in top-50 outputs, one of them is sampled as the positive instance in each training iteration. Our pilot experiment shows that reader model is robust: such initialization gives reasonable accuracy.

[6]DistDR converges after five iterations (Figure 6.3).

Models	Ans	Passage	Chain
No Supervision			
TF-IDF	60.7	90.8	35.8
Full Supervision			
MDR	85.2	89.3	**75.3**
BeamDR	84.9	91.1	73.6
Distant Supervision			
DistDR w/ TF-IDF chains	55.6	52.3	22.3
DistDR	**86.2**	**92.2**	75.1

Table 6.1: Compare DistDR's retrieval (based on top-10 chains) with unsupervised and fully-supervised methods on HotPotQA dev set over answer, passage and chain recall. DistDR matches fully-supervised dense retrieval approaches.

than BeamDR and MDR—both are state-of-the-art dense retrieval methods with full supervision—even though DistDR is only trained on (question, answer) pairs. When using the same model implementation but with distant supervision, DistDR is competitive to BeamDR on reader results (Table 6.2). On NaturalQuestions (Table 6.3), unlike multi-hop questions, using IR to find evidence as distant supervision provides helpful training signals, which is confirmed by a small gap between distant and full supervision.[7] Building on top of distantly-supervised DPR, DistDR

[7]Compared to retrieval, the gap is larger for the reader. This is due to training data processing for DPR (and ANCE). On retrieval, DPR replaces annotated gold passage with the corresponding

Models	Ans	Passage	Chain	Span
Full Supervision				
Transformer-XH	**91.8**	**97.0**	81.3	52.4
GRR	87.5	92.2	79.0	50.4
BeamDR	90.5	94.7	**83.0**	**52.6**
Distant Supervision				
DistDR	91.4	95.3	81.7	51.6

Table 6.2: Compare DistDR with other fully-supervised methods with reranking over answer, passage and chain recall, and span extraction over span exact match. DistDR is competitive to BeamDR.

beats weakly-supervised systems, and is competitive with fully-supervised models.

6.5 Analysis

This section explores HotPotQA results to understand why DistDR is on par with fully-supervised approaches.

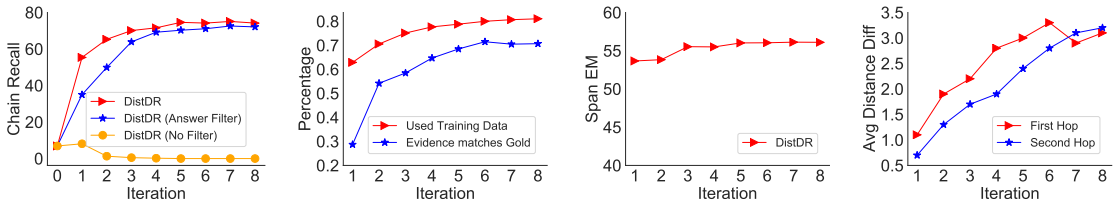
100-token passage in the candidate pool, and discard the questions if the matching is failed (25% of questions). On reader, DPR uses annotated passage as positive, and top retrieved passages that do not contain the answer are the negatives, thus entire training data is used.

Models	Top 1	Top 20	Span
Full Supervision			
DPR	46.3	78.4	41.5
ANCE	**50.9**	**81.9**	**46.0**
Distant Supervision			
DPR	45.2	78.2	37.9
ANCE	45.7	79.1	38.3
DistDR	50.4	80.1	40.5

Table 6.3: Compare DistDR with other dense retrieval systems on NaturalQuestions dataset over answer recall @1, 20, and span exact match. DistDR is competitive to fully-supervised approaches.

Models	Ans	Passage	Chain
DistDR	**86.2**	**92.2**	**75.1**
DistDR w/ Sampled Pos	83.9	90.9	72.1
Remove Reader Filter	85.4	91.9	72.1
Remove Non-gold Evidence	66.7	74.1	53.1

Table 6.4: Ablation Studies evaluated on answer, passage and chain recall over top-10 chains.



(a) (b) (c) (d)

Figure 6.3: Quantitive analysis on DistDR by iteration. (a): Compare different evidence filter strategies on dev set; (b): Statistics on training set extracted evidence; (c): Compare span extraction component over gold evidence; (d): Average distance difference on dev set from question to top-10 negative passages (Average) and positive passage. DistDR finds better evidence over iterations, the improved evidence further helps model training.

6.5.1 Analysis on Model Training

We first study how DistDR finds better evidence from a large corpus with hard EM.

Effect of Hard EM We study the effect of hard EM by going through accuracy and extracted evidence statistics at every iteration. Chain recall for DistDR on the dev set increases over the first five iterations (Figure 6.3a) and then converges. As the evidence finder improves, the E-step extracts better evidence for training examples (Figure 6.3b): both the percentage of used training examples and gold evidence in training examples increase with additional EM iterations. The improved evidence quality further helps DistDR training. We compare multiple positive

sampling strategies: using hard EM (top-1 as positive evidence) is slightly better than randomly sampling from top-k positives for the M-step update (Table 6.4, second row).

The Effect of Evidence Filtering Filtering positive evidence is crucial for ensuring high-quality evidence. We ablate different filtering methods in Table 6.4 and Figure 6.3a: No filter, only an answer filter, and an additional reader filter (DistDR). No filter fails completely: evidence should not be considered correct if it does not contain the answer. With an additional reader filter, DistDR outperforms using only the answer matching filter and converges faster (since the reader filter reduces false-positives and makes training more robust).

Despite rapidly changing evidence, the reader's span extraction component is robust over iterations (Figure 6.3c), even on the first iteration, suggesting it is robust against spurious false-positive evidence.

6.5.2 Analysis on Retrieved Evidence

DistDR is competitive to fully-supervised approaches on HotPotQA. However, even at the last iteration, only 65% of extracted evidence matches the gold. We ablate evidence and contrasts gold evidence with that retrieved by DistDR.

Non-gold Evidence is Helpful If we only retain training examples that matches the labeled gold evidence, answer accuracy falters and underperform DistDR (Table 6.4, last row). Instead of providing noise, non-gold evidence gives useful signal

for model training.

Human Analysis and Case Study We manually annotate fifty training examples where the extracted evidence from DistDR does not match the labeled evidence. We confirm that most of the extracted evidence is helpful for model training. Specifically, 38% of the questions are answerable via a single piece of evidence, even though this dataset is supposed to require multiple hops (first case of Table 6.5). In another 28% of cases, DistDR finds alternate valid evidence. In second case of Table 6.5, the question mentions both films The Mist and The Green Mile; therefore the reasoning chain from either film to the director Frank Darabont is correct (though only one is annotated). In the final of 34% of cases, DistDR finds the wrong evidence, often because the extracted evidence only includes one span that matches the answer type, therefore the reader confidently outputs the span for the wrong reason. In third case of Table 6.5, Nassau County is the only county it sees, and therefore the model has stumbled upon the right answer erroneously. Building a model with faithful predictions is an important ongoing research topic [Jacovi and Goldberg, 2020a].

6.6 Conclusion

We present DistDR, a distantly-supervised ODQA system that improves over a weak retriever by iteratively finding evidence from a corpus, and using the evidence as distant supervision for model training. Without using any evidence labels, DistDR matches the fully-supervised SOTA approaches on both multi-hop and

single-hop QA benchmarks.

Annotating evidence for existing question-answer pairs is generally expensive, especially for complex questions. While DistDR can accurately find evidence for arbitrary complex machine reading-style questions, future work needs to validate whether this can work for other types of questions. This could improve the reader to answer numerical reasoning [Dua et al., 2019], temporal reasoning [Ning et al., 2020], multi-model reasoning [Lei et al., 2018], or combination of these skills [Bartolo et al., 2020].

Q: Jo Ann Terry won the 80m hurdles event at what Sao Paulo-based event from 1963? **A**: Pan American Games

Human Annotation: Jo Ann Terry → 1963 pan american games

DistDR: Jo Ann Terry → Jovem Pan

Jo Ann Terry: Jo Ann Terry won the 80 m hurdles event at the 1963 Pan American Games.

1963 Pan American Games: The 4th Pan American Games were held in São Paulo.

Jovem Pan: Jovem Pan is the main Brazilian radio station based in São Paulo, Brazil.

Q: Who's the Hungarian-born US film director renowned for adapting Stephen King novellas to the screen, including The Mist and The Green Mile? **A**: Frank Darabont

Human Annotation: The Mist (film) → Frank Darabont

DistDR: The Green Mile (film) → Frank Darabont

The Mist (film): The Mist is a 2007 American film written and directed by Frank Darabont.

The Green Mile (film): The Green Mile is a 1999 film written and directed by Frank Darabont.

Frank Darabont: Frank Arpad Darabont is a film director, screenwriter and producer.

Q: Zimbabwe's Guwe Secondary School has a sister school in what New York county?**A**: Nassau County

Human Annotation: Guwe Secondary School → Carle Place High School

DistDR: University of Zimbabwe → East Rockaway High School

Guwe Secondary School: Guwe Secondary School has a sister school in New York.

Carle Place High School: Carle Place School is located in Nassau County, New York.

University of Zimbabwe: The University of Zimbabwe (UZ) is the oldest and formerly largest university in Zimbabwe.

East Rockaway High School: East Rockaway Junior-Senior High School is the sole high school in Nassau County, New York.

Table 6.5: Examples of DistDR on "false positive" evidence from HotPotQA that does not match gold evidence.

Chapter 7: Bridging the Generalization Gap in Text-to-SQL Parsing with Schema Expansion

In this chpater, we switch our attention to question answering over tables. Using tables as a knowledge source, QA is often reframed as text-to-SQL semantic parsing task, in which a parser maps natural language questions to programs that are executable over tables to generate answers, and are typically evaluated on large-scale datasets like SPIDER [Yu et al., 2018]. We argue that existing benchmarks fail to capture a certain *out-of-domain generalization* problem that is of significant practical importance: matching domain specific phrases to composite operation over columns. To study this problem, we first propose a synthetic dataset along with a re-purposed train/test split of the SQUALL dataset [Shi et al., 2020] as new benchmarks to quantify domain generalization over column operations, and find existing state-of-the-art parsers struggle in these benchmarks. We then propose to address this problem by incorporating prior domain knowledge through preprocessing table schemas, and design a method that consists of two components: *schema expansion* and *schema pruning*, which can be easily applied to different base parsers. We show that on this domain generalization over column operations problem, our proposed method significantly outperforms baseline parsers, and as a result boosting

the underlying parsers' overall performance. Our goal is to direct attention on a challenging aspect of out-of-domain generalization by providing a new evaluation benchmark, as well as an initial direction for solving this problem, and reference point for future work. [1]

[1]This Chapter describes work in submission [Zhao et al.].

7.1 Introduction

Text-to-SQL parsing is the task of translating natural language questions over provided tables to SQL queries which can be executed to produce answers. With the availability of large-scale datasets [e.g., Zhong et al., 2017, Yu et al., 2018], neural semantic parsers have witnessed significant success on this task. However, recent work [Suhr et al., 2020, Lee et al., 2021] has suggested that these state-of-the-art parsers are far from successful in terms of *out-of-domain generalization* in real scenarios, where users may ask questions related to potentially very large tables with the goal of improving their productivity (e.g., while they are viewing or editing a large Excel spreadsheet). In such scenarios, it is common to encounter tables specific to new domains that were never encountered before while training a parser. Perhaps the most challenging aspect of domain generalization is that models need to understand domain-specific phrases that they have not seen before, and translate them into logical form segments that involve references to table elements (e.g., column names or aggregation operations over columns). This is mainly because of two kinds of abstract operations, shown in Figure 7.1, that are challenging for new domains:

1. Column Matching: The task of mapping natural language phrases to the most relevant columns (e.g., mapping "Income" to the `"Wages"` column). This can be challenging because some mappings may be implicit or may require domain knowledge.

2. Column Operations: The task of mapping natural language phrases to com-

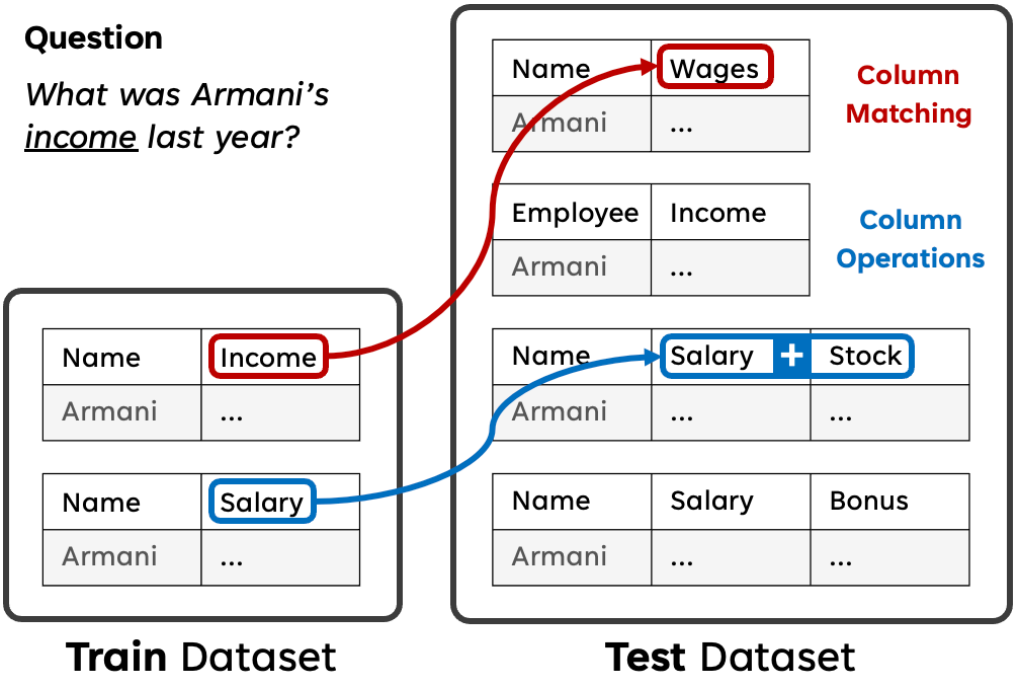


Figure 7.1: Illustration of two aspects of out-of-domain generalization that are challenging for text-to-SQL parsers. While existing methods partially address the "column matching" issue, they still suffer when it comes to "column operations". Note that there are more tables on the right to illustrate the fact that there are a variety of settings the parser may run into at test time.

posite expressions over table columns. For example, in Figure 7.1, we need to map *income* to just `"Wages"` for one table, and to `"Salary" + "Stock"` for another table. Similarly, consider the `"Term"` column in Figure 7.2. Some questions may ask about the *term duration* while others may ask about the *term start*. Each of these questions requires mapping the corresponding phrase to an expression that refers to this column (e.g., `"Term"._2 - "Term"._1` for the former and `"Term"._1` for the latter).

While recent approaches rely on pre-trained language models [e.g., Yin et al., 2020,

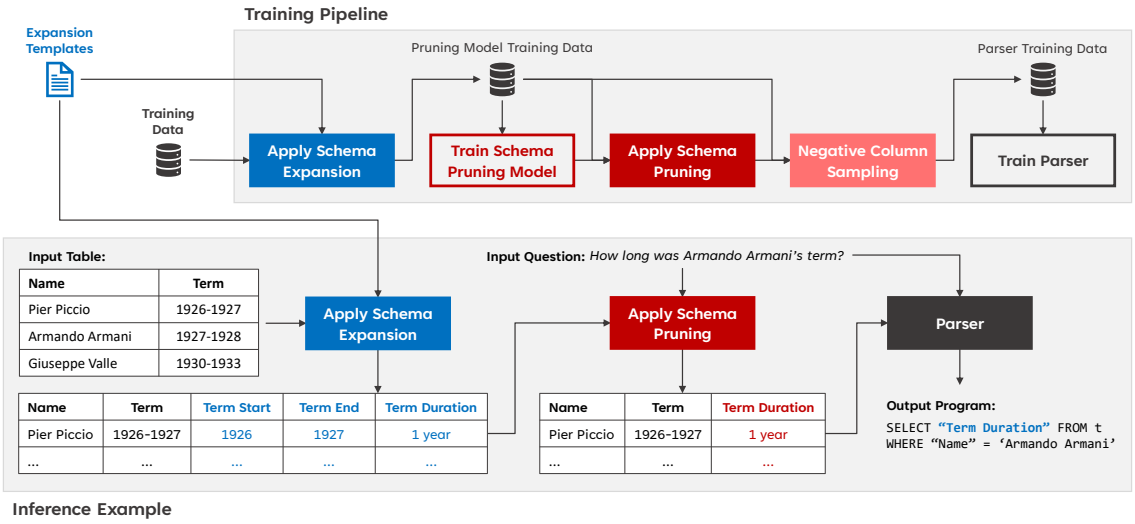


Figure 7.2: Illustration of the training pipeline for the proposed method and the inference process for an example. The proposed method is described in detail in §7.4. Note that the proposed components interact with the parser by modifying the table that is fed to it as input, as well as the target program during training in the case of *schema expansion*.

Deng et al., 2021] for addressing the column matching challenge, column operations remain relatively unexplored due to the lack of evaluation benchmarks.

To this end, we first propose two new benchmarks: a synthetic dataset and a train/test repartitioning of the SQUALL dataset [Shi et al., 2020]; both capable of quantifying out-of-domain generalization on column operations. We then show that existing neural parsers underperform on both benchmarks, because they require an impractically large amount of in-domain training data—which is not available in our setting—to effectively "memorize" mappings from natural language phrases to program fragments. Finally, we propose a new method for making any existing text-to-SQL parsers aware of prior information that may be available about the domains

of interest. Specifically, we propose two new components: *schema expansion* and *schema pruning*.

Schema expansion uses heuristics to expand columns into sets of derived columns based solely on their *types* (all schemas are assumed to be typed which tends to be true for both relational databases and Excel spreadsheets in practice; Excel uses a built-in type inference mechanism). Relying on generic types enables this method to apply to new domains, as long as they make use of similar underlying types. This process allows us to transform complex program fragments (e.g., `"Term"._2 - "Term"._1`) into simpler ones (e.g., `"Term Duration"`) that are better aligned with the natural language questions, thus making the underlying parser's job easier. While schema expansion may result in a large number of unnecessary expanded columns, schema pruning then prunes the set of relevant columns that final parser is allowed to look at, based on both the questions and the expanded schemas.

Our experiments show that schema expansion and schema pruning can boost the underlying parsers' performance by up to 13.8% relative accuracy (5.1% absolute) on the new SQUALL data split. Furthermore, they also boost performance over the original SQUALL data splits by up to 4.2% relative (1.9% absolute). One of our main goals in this chapter is to put attention on the difficult problem of domain generalization by providing a new evaluation benchmark, as well as an initial direction for solving this problem.

7.2 Background

We present background of text-to-sql parsing in Section 2.5, including task, benchmarks and state-of-the-art-models.

Out-of-Domain Generalization. Generalization in machine learning is often defined as the ability to do well on a test set after learning from a training set, where all examples in both sets are drawn independently from the same distribution (*i.i.d. generalization*). However, as Gu et al. [2021] argue, in real-world applications such as semantic parsing, the test data may involve new compositional structures (*compositional generalization*), or new domains (*domain generalization*) that are not encountered during training. Existing work in compositional generalization for semantic parsing has focused on using synthetic datasets [e.g., Keysers et al., 2020, Lake and Baroni, 2018], or repartitioning existing text-to-SQL datasets into new train and test splits [e.g., Finegan-Dollak et al., 2018]. Both approaches have generally shown that compositional generalization remains an important challenge [e.g., Shaw et al., 2021]. We focus on the arguably even more challenging domain generalization problem, also known as *domain adaptation*, where entire domains may never be encountered during training or may only be encountered a small number of times [Motiian et al., 2017]. Even though this problem has been studied extensively in the context of classification [Daumé III and Marcu, 2006], machine translation [Daumé III and Jagarlamudi, 2011], and question answering [Talmor and Berant, 2019], it remains underexplored for semantic parsing. However, to be applicable in real scenarios, semantic parsers should be able to generalize to new domains,

since collecting domain-specific labeled data is often prohibitively expensive. Recent approaches have focused on data synthesis [Yin et al., 2021], meta-learning [Wang et al., 2021], relation-aware schema encoding [Wang et al., 2020], and encoder pre-training [Yin et al., 2020, Herzig et al., 2020, Yu et al., 2020, Deng et al., 2021]. In this chapter, we hone in on one aspect of domain generalization that we shall broadly refer to as *column operations* and which was introduced in §7.1 and illustrated in Figure 7.1.

7.3 Proposed Evaluation Benchmarks

Our goal is to design an evaluation benchmark that has the following out-of-domain generalization properties: (i) the training data involves a different set of domains than the test data, (ii) the questions and tables that appear in the train and test data are non-overlapping, not only in terms of the domains they belong to, but also in terms of the program fragments that they contain, and (iii) To simulate the more challenging setting that often encountered in real applications, the test data is biased to contain more examples that involve both nested *column access operations*, like getting the start of a `"term"` in Figure 7.2, as well as *composite column expressions*, like getting the duration of a `"term"`. To this end, we propose a new synthetic dataset and a repartitioning of the SQUALL dataset into new train/test splits.

7.3.1 Synthetic Dataset

We consider three fictional domains inspired by common uses of tables: *finance*, *sports*, and *science*. We explain our synthetic dataset generation process through a running example as follows:

1. For each domain, we declare a set of formulas that relate different quantities (e.g., `"Income" = "Salary" + "Stock"`). The primitives used in these formulas define the set of available columns.

2. For each column we declare a set of noun phrases that can be used to refer to it (e.g., "wages" for `"Income"`, and "base salary" for "salary"). And we define the SQL query template that shall be used for all programs: `SELECT <column> FROM t WHERE "Year" = <year>`, and a question template `What was <column> in <year>`? Note that the `"Year"` column is special and is included in all examples of this synthetic dataset.

3. We first sample a formula, and a variable from that formula (e.g., sample `"Income"` from formula `"Income" = "Salary" + "Stock"`). We then generate a question asking for this variable, randomly replacing the variable with a noun phrase in the corresponding referring set, and randomly generate a year value (e.g., use "wages" to replace "income" and generate a question "What was `[wages]` in `[2011]`?").

4. To generate the target program π^*, we randomly drop a variable from the sampled formula in step 3. If the asked value corresponds to this variable, we

transform its reference in the SQL query so that it is expressed as a function of the columns that are kept (e.g., `"Salary" + "Stock"`), otherwise we use the column name (e.g., `"Income"`).

5. To generate a table schema we first add a `"Year"` column and two of the columns that were not sampled from the formula (e.g., `"Salary"` and `"Stock"`). We then sample k other columns and add them to schema ($k = 15$ in our experiments) as *distractor* columns. Note that we do not generate full tables for this synthetic dataset since we do not evaluate on table cell selections.

We construct benchmark datasets by first generating 1,000 examples per domain and then iterating over the domains and keeping the data generated for the current domain as our test data, while using the data of the remaining two domains for training. This results in three datasets, each with 2,000 train examples and 1,000 test examples. More details on the declarations for our domains can be found in **??**.

7.3.2 SQUALL Repartitioning

Aside from the synthetic dataset we also propose to repartition SQUALL into new train and test data splits, with a focus on the aforementioned out-of-domain generalization properties. The original splits for SQUALL were produced by uniformly sampling 20% of the tables to produce the test set and using the remaining 80% as the train set. This process was repeated five times and the evaluation metric results were averaged over the results obtained for each repetition. This resulted in similar tables being included in both the train and test sets (e.g., tables refer-

ring to two different basketball matches, but having identical schemas), and few examples in the test set required column operations. In order to avoid this, we propose the following algorithm for automatically constructing data splits focused on out-of-domain generalization on column operations:

1. Collect the table schemas used across all examples in the train and dev splits of the dataset (there are about 1,600 schemas; note that the test set is not annotated with SQL queries).

2. Construct a graph by treating each schema as a node, and adding an edge for each pair of schemas that share more than 33% of their columns.

3. Find all the connected components of the graph. Each defines a cluster of table schemas.

Data Category	**# Examples**	
	Train	**Test**
All	8,956	2,320
w/ `Score` Accessors	91	86
w/ `Score` Expressions	47	53
w/ `Date` Accessors	81	173
w/ `Date` Expressions	18	95

Table 7.1: Statistics for our repartitioned version of SQUALL, including categories that we use in our empirical analysis and which are presented in § 7.3.2.

4. Each table has a set of SQL queries associated with it: one for each example that uses this table. For each query we check if it is a `SELECT` of a single column or if it is a `SELECT` that involves column operations such as field accessors or arithmetic operations. We associate each cluster with the number of queries that involve such column operations.

5. We sort the clusters based on this number, in decreasing order, and then use the first 20% as the test set and the remaining as the train set. Note that adding a cluster to the train/test set is equivalent to adding all examples that use tables included in this cluster. This step will result in disproportionally more column operations being used in our test set than the train set, which means the model will need to learn to generalize well in this setting to do well in this dataset.

In the following sections we pay special attention to four data subcategories that are representative of the out-of-domain generalization setting for SQUALL:

- `Score` Expressions: Represents SQL queries that include expressions over columns of type `Score` (e.g., a query selecting the score difference for a basketball game).

- `Score` Accessors: Represents SQL queries that include field accessors for columns of type `Score` (e.g., a column with the results of a basketball game, like "89-72", and a query that requires accessing the first element of this score; i.e., "89").

- `Date` Expressions: Similar to `Score` expressions except using the `Date` and `TimeSpan` type (e.g., a query asking for the duration of a presidency term).

- `Date` Accessors: Similar to `Score` Accessors, except using the `Date` and `TimeSpan` type (e.g., a query asking for the start of a term).

We shall refer to these categories when reporting experimental results in §7.5. We provide statistics for the resulting dataset in Table 7.1.

7.4 Proposed Method

In this section we propose a simple approach for tackling this specific out-of-domain generalization problem that ought to serve as evidence that it is a real problem and that it is solvable, as well as a reference point for evaluating future approaches. Our approach consists of two new components that can be used in combination with any existing text-to-SQL parser: *schema expansion* and *schema pruning*. These components interact with the parser by preprocessing the table that is fed to it as input. This is illustrated in Figure 7.2.

As discussed in §7.1, there are two kinds of challenges related to out-of-domain generalization in text-to-SQL parsing, *column matching* and *column operations*, with the latter being more challenging. The goal of *schema expansion* is to reduce column operations to column matching, by adding synthetic columns to the table schema, which correspond to expressions or accessors over existing columns (e.g., it may add a column that represents the sum of two columns). This is based on the intuition that learning (or rather memorizing) the ways in which different types of columns can be

composed together requires a large amount of in-domain training data. Instead, we propose to inject prior knowledge as to what kind of symbolic operations are possible based solely on the column types in a schema. This reduces column operations to column matching by effectively bringing the target programs closer to their surface form in the natural language question (e.g., `"Income"` can now map to a synthetic column that corresponds to the sum of `"Salary"` and `"Stock"`, instead of having the parser produce the sum expression directly). Since our expansion is based on column types, we argue that it's reasonable to assume that all schemas are typed and our expansion could be applied to any new domain. It's also worth noting that even though our templates may not cover all cases, [2] when applying to new domains, developers can declare a few templates of their interest and apply schema expansion on these templates, which is a more cost-effective way compared with collecting large in-domain training data to train the parser.

Naturally, having a component that expands the table schema means that we may end up with large schemas that the parser has to deal with, which will often involve a lot of irrelevant columns (partially because the schema expansion component does not peek at the question). This can result in increased latency which is not desirable in real-world systems. To this end, we introduce a *schema pruning* component which looks at both the expanded table schema and the question and decides which columns to prune before invoking the parser. It can be argued that this pruning is as hard as parsing itself, but there is evidence from other areas

[2] An interesting future work is to automatically expand schema using large pre-trained language models

that it can indeed be helpful [e.g., vocabulary selection; Chen et al., 2019, Xu et al., 2021]. As we shall show schema pruning can actually provide an additional boost in accuracy, depending on architecture of the underlying parser.

7.4.1 Schema Expansion

A domain developer first declares a set of *templates* that specify the ways in which different column types can interact (e.g., it specifies that given a typed `TimeSpan` column that contains two subfields, `_1` and `_2`, the expression `TimeSpan._2 - TimeSpan._1` can be constructed that represents a duration), and the names for each such interaction (e.g., `"Duration"`).[3] The schema expansion component receives as input this set of templates along with the table schema and returns an expanded schema that includes additional columns generated by using all applicable templates. For our SQUALL experiments, we declared the templates shown in Table 7.2. Although these templates are somewhat tailored to this dataset, our main goal is to show that there is considerable room for improvement in this challenging generalization scenario, and that even a very simple approach requiring minimal manual effort can result in significant boosts.

7.4.2 Schema Pruning

We propose a simple schema pruning approach that is inspired by vocabulary selection methods in machine translation. Let us denote the input question by q and

[3]Having a name that accurately represents the meaning of column operation results is desired, as semantic parser is sensitive to column names for column matching.

Column : Type : Fields	Synthetic Column		Example	
	Column Name	Expression	Original Column Name(s)	Synthetic Column Name(s)
Expressions				
`x:TimeSpan:_1,_2`	`[x]` Duration	`x._2 - x._1`	Term	Term Duration
`x:Date,y:Date`	`[x & y]` Duration	`y - x`	Term Start; Term End	Term Duration
`x:Score:_1,_2`	`[x]` Difference	`x._2 - x._1`	Result	Result Difference
`x:Score:_1,_2`	`[x]` Sum	`x._2 + x._1`	Result	Result Sum
Accessors				
`x:TimeSpan:_1,_2`	`[x]` Start; `[x]` End	`x._1`; `x._2`	Term	Term Start; Term End
`x:Score:_1,_2`	Home `[x]`; Away `[x]`	`x._1`; `x._2`	Result	Home Result; Away Result
`x:Score:_1,_2,_3`	Win Record; Loss Record; Tie Record	`x._1`; `x._2`; `x._3`	Result	Win Record; Loss Record; Tie Record
`x:Score:_1,_2,_3`	First Round `[x]`; Second Round `[x]`; Total `[x]`	`x._1`; `x._2`; `x._3`	Score	First Round Score; Second Round Score; Total Score

Table 7.2: Templates used for schema expansion over `TimeSpan`-, `Date`-, `Score`-valued columns, including column expressions and accessor operations. `x` and `y` denote columns in the original schema, `_1`, `_2`, and `_3` refer to tuple field accessors, `&` denotes overlapping column name tokens, and `;` is used as a column separator.

the input column names after expansion by $c_1, \ldots, c_M$. We concatenate the question and the column names as `[CLS]` q `[SEP]` c_1 `[SEP]` ... `[SEP]` c_M `[SEP]` and feed the resulting sequence to a BERT encoder [Devlin et al., 2019]. We then define the embedding of each column, $\boldsymbol{c}_i$, as the final-layer representation of the last token of

that column's name. Finally, we define the probability that a column should be kept as $p_i = \text{Softmax}\left(\text{MLP}(\boldsymbol{c}_i)\right)$. We train this model based on whether each column is used in the corresponding SQL program. At inference time, we need to choose a threshold on the predicted probabilities for deciding whether to prune a column or not. We assume a transductive setting and choose this threshold such that the ratio of pruned columns over the test set equals to the ratio of pruned columns over the train set plus a constant hyper-parameter to account for fact that accuracy will likely be lower for the test set than the train set. Note that assuming a transductive setting is fine because in a real-world system we could be tuning this threshold based on the last t requests made to the model. While this is not equivalent, assuming a large enough t, we should be able to adapt this threshold using the same approach.

Negative Column Sampling. As is evident from Figure 7.2, we also introduce a negative column sampling component. This is because we train our pruning model on the same data that we use to train the underlying parser (aside from the modified table schemas) and thus the pruning model can become good at pruning all irrelevant columns over this dataset. This will result in the underlying parser being unable to handle situations where irrelevant columns are left in by the pruning model. To this end, during training we introduce some irrelevant columns (i.e., negative column sampling) to improve the robustness of the underlying parser. We found that making sure to always include at least 3 columns in the schemas was sufficient and equivalent to randomly sampling 1 or 2 additional columns for each training example, and so that is what we did in our experiments.

Dataset Split	Seq2Seq				SmBop			
	Base	Base + P	Base + E	Base + E + P	Base	Base + P	Base + E	Base + E + P
Synthetic Dataset								
I.I.D.	52.5 ± 2.2	86.5 ± 0.7	93.4 ± 0.6	**96.2** ± 0.3	85.3 ± 0.5	90.9 ± 0.2	**97.3** ± 0.1	**97.3** ± 0.1
Finance	17.4 ± 0.6	18.3 ± 0.8	58.7 ± 0.3	**65.9** ± 0.7	23.6 ± 0.2	24.4 ± 0.3	**69.7** ± 0.3	68.7 ± 0.1
Sports	16.8 ± 0.6	26.7 ± 0.4	69.0 ± 0.4	**71.8** ± 0.5	28.8 ± 0.9	28.8 ± 0.2	77.3 ± 0.5	**79.5** ± 0.2
Science	12.8 ± 0.1	17.5 ± 0.8	64.8 ± 0.2	**69.9** ± 0.6	20.1 ± 2.0	26.3 ± 0.2	69.7 ± 0.6	**71.2** ± 0.2
Squall Dataset								
I.I.D.	45.1 ± 0.6	46.3 ± 1.0	47.2 ± 0.9	**47.5** ± 1.0	46.4 ± 0.8	46.7 ± 0.6	48.1 ± 0.8	**48.2** ± 0.5
Repartitioning	35.0 ± 0.3	36.8 ± 0.2	38.0 ± 0.3	**39.3** ± 0.2	37.0 ± 0.1	39.8 ± 0.3	**42.1** ± 0.3	**42.1** ± 0.2
`Date` Expressions	1.4 ± 0.7	4.5 ± 0.3	28.0 ± 2.3	**43.2** ± 1.0	3.2 ± 0.9	10.6 ± 1.9	**50.2** ± 2.5	46.3 ± 0.6
`Score` Expressions	9.4 ± 1.1	5.0 ± 2.7	30.9 ± 3.5	**33.9** ± 1.7	26.4 ± 3.1	30.0 ± 3.0	47.2 ± 1.3	**51.6** ± 1.3
`Date` Accessors	19.1 ± 0.7	**26.3** ± 0.4	24.9 ± 1.0	26.0 ± 0.5	21.4 ± 0.3	**24.8** ± 1.4	23.5 ± 0.8	**24.8** ± 0.3
`Score` Accessors	18.1 ± 2.3	**22.6** ± 1.1	21.8 ± 1.0	18.1 ± 0.9	21.8 ± 1.3	**31.7** ± 0.8	26.4 ± 1.8	25.3 ± 2.0

Table 7.3: Mean accuracy and standard error for 3 experiment runs, computed over multiple different splits for each dataset. The best results in each row are shown in **bold red font**. Note that, when compared with the Base model, all gains statistically significant. + P stands for using the schema pruning model and + E for the schema expansion model.

7.5 Experiments

We performed experiments on the two proposed benchmarks (as well as the existing version of the Squall benchmark), using the two current state-of-the-art parser architectures presented in §7.2 in combination with our proposed schema

expansion and pruning components.[4]

7.5.1 Experimental Setup

As described in §7.3, our synthetic benchmark consists of three domains, *finance*, *sports* and *science*. We repeat our experiments once for each domain. For each repetition we test on one of the domains, while training on the other two. For SQUALL, we present results on our repartitioned split from § 7.3.2. For both datasets, we also include results for three i.i.d. splits. In each experiment, we compare four different configurations for the parsers: (1) Base: the underlying parser (i.e., SEQ2SEQ or SMBOP), (2) Base + P: Base while also using our schema pruning component, (3) Base + E: Base while also using our schema expansion component, and (4) Base + P + E: Base while also using both of our schema expansion and pruning components. We repeat each experiment 3 times using different random seeds and report mean *exact match accuracy* (i.e., fraction of examples where the predicted SQL queries exactly match the gold queries), and standard error for this mean.[5]

[4]Our evaluation benchmarks along with code for reproducing our experiments are available at `http://anonymous`.

[5] For SQUALL researchers often also report *execution accuracy*, which measures the fraction of examples for which executing the predicted SQL queries results in the correct answer to the input question. However, we found that for 7% of the examples that are representative of out-of-domain generalization, executing the gold SQL queries does not yield the correct answer (e.g., in cases where the correct answer is a sub-string of a cell value). Therefore we chose to only report exact match accuracy in our experiments.

7.5.2 Results

Synthetic Benchmark Results. Our results for this benchmark are presented in the top part of Table 7.3. A first observation is that performance on the i.i.d. split for the baseline parsers is significantly better than on the domain-based splits. Interestingly, our expansion and pruning components still provide a significant boost over baseline performance in this setting (up to 43.7% absolute accuracy / 83.2% relative). However, the baseline parsers are practically unusable in the domain-based splits. In this case, our approach provides a very significant accuracy boost, rendering them useful (*up to 55.0% absolute / 327.4% relative*).

Squall Benchmark Results. Our results for this benchmark are presented in the bottom part of Table 7.3. Similar to the synthetic benchmark, we observe that both parsers perform reasonably well on the i.i.d. split, but significantly underperform in our repartitioned benchmark. This is consistent with earlier observations by Suhr et al. [2020] and Lee et al. [2021]. Furthermore, we observe that our expansion component helps boost the accuracy of both parsers significantly (up to 5.1% absolute / 13.8% relative) and the pruning component provides some small further improvements on top of that. However, we notice that the pruning component is not as helpful for SmBop as it is for Seq2Seq, which we provide detailed analysis in § 7.5.4. Drilling down a bit further, we observe that most gains are due to the data categories we defined in § 7.3.2. Perhaps most importantly, we get a *47.0% absolute accuracy gain (1,468.8% relative)* for SmBop on the "`Date` Expressions" category alone. This can be largely attributed to our schema expansion component, where

by incorporating prior domain knowledge we are effectively reducing the original column operations problem to a column matching problem, which is significantly easier. As a result, we get significant improvements on both "Expression" data categories. We do not observe the same for "Accessor" categories, which we address in the following section.

7.5.3 When is Schema Expansion Helpful?

From Table 7.3, schema expansion does not seem to help much for "Accessor" expressions (i.e., Base + P performs as well as or slightly better than Base + E + P on those categories). In order to further understand the contribution of schema expansion, we conducted an ablation study where we compare the proposed Base + P + E with three more approaches: (1) E Expressions: the schema expansion component only uses "Expression" templates, (2) E Accessors: the schema expansion component only uses "Accessor" templates, (3) P Oracle: the schema pruning model is replaced with an oracle model that always only keeps the columns that are used in the gold SQL queries (so the parser only has to figure out how to use them, rather than also figuring out which ones to use). Note that (3) will be discussed in the following section. We present the results for this ablation study in Table 7.4. We observe that expanding "Expressions" but not "Accessors" boosts performance on the "Expressions" categories, and similarly for "Accessors". More importantly though we see that using either one alone performs worse than using both types of expansion, indicating that they both provide value and that they work well together.

Dataset Split	SEQ2SEQ				SMBOP			
	Base + E + P	E Accessors	E Expressions	P Oracle	Base + E + P	E Accessors	E Expressions	P Oracle
Repartitioning	39.3 ± 0.2	37.4 ± 0.1	37.6 ± 0.1	**52.9** ± 0.3	42.1 ± 0.2	40.6 ± 0.2	40.4 ± 0.2	**57.9** ± 0.4
`Date` Expressions	43.2 ± 1.0	2.8 ± 0.7	39.6 ± 1.3	**81.0** ± 2.5	46.3 ± 0.6	16.3 ± 1.3	41.1 ± 1.0	**75.4** ± 1.7
`Score` Expressions	33.9 ± 1.7	14.2 ± 3.6	25.7 ± 1.7	**48.8** ± 1.7	51.6 ± 1.3	36.5 ± 4.1	48.4 ± 1.6	**74.2** ± 1.7
`Date` Accessors	26.0 ± 0.5	27.6 ± 0.6	26.6 ± 0.7	**33.1** ± 1.5	24.8 ± 0.3	23.5 ± 0.8	22.6 ± 2.1	**32.4** ± 1.0
`Score` Accessors	18.1 ± 0.9	20.6 ± 1.5	9.9 ± 0.8	**29.2** ± 1.1	25.3 ± 2.0	30.7 ± 2.8	15.3 ± 3.2	**58.1** ± 3.2

Table 7.4: Mean accuracy and standard error for 3 runs of our ablation studies on SQUALL repartitioning split for domain generalization, with the best results in each row colored **red**.

7.5.4 When is Schema Pruning Helpful?

It is evident from Table 7.3 that schema pruning is useful both on its own (i.e., Base + P), but also on top of schema expansion (i.e., Base + E + P). For SMBOP, we observe that Base + P is more or less on par with Base. Though this may seem inconsistent with the SEQ2SEQ results at first, it is not actually surprising because SMBOP keeps the most relevant columns in the beam during bottom-up decoding, and thus it is implicitly already using a schema pruning component. Furthermore, we observe that schema pruning is especially useful on top of schema expansion for the column operation data categories ("Expressions" and "Accessors"). This is because in the corresponding examples we end up with a significantly larger number of expanded columns that labeled as negatives when training the pruning model. Schema pruning then filters most of these irrelevant columns before training

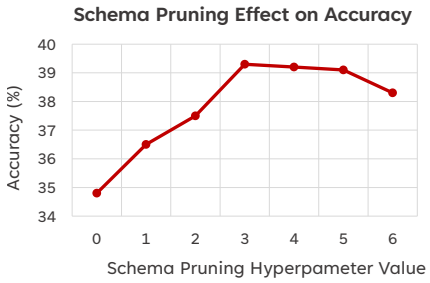


Figure 7.3: Accuracy (%) while varying the schema pruning model hyperparameter of §7.4. Pruning more than necessary has a significant negative impact on accuracy, while pruning less does not.

Title: Adam Szalai International Goals

score before goal → Score, Home Score, Away Score

final score → Result, Home Result, Away Result

Goal	Date	Opponent	Score	Home Score	Away Score	Result	Home Result	Away Result
1	10/08/2010	San Marino	2-0	2	0	8-0	8	0
2	10/08/2010	San Marino	4-0	4	0	8-0	8	0
3	12/08/2010	Finland	0-1	0	1	1-2	1	2

Question: *How many games did he score in, where his team lost?*

Program: `SELECT COUNT(DISTINCT “Date”) FROM t WHERE “Home Result” < “Away Result”`

Figure 7.4: Example that showcases some of the challenges that are not addressed by our approach, but which are accounted for in the evaluation benchmarks that we propose. In this case, the `"Score"` and `"Result"` columns have domain-specific semantics that are hard for the model to learn, and the question also depends on the title of the table, which current models do not take into account.

the underlying parser, resulting in a more robust training procedure. Finally, in Table 7.4 we observe that P Oracle performs really well, indicating that investing in a good schema pruning model would be meaningful for improving generalization performance.

Schema Pruning Decision Threshold. As discussed in §7.4, the proposed schema pruning component requires setting a decision threshold hyperparameter. We already described the way we do this in §7.4, but it is also worth analyzing the impact of this decision on the overall parser accuracy. This is because, intuitively we expect that too aggressive pruning will likely cause cascading errors, while too conservative

pruning would not be very effective and end up being equivalent to not using any pruning at all. To this end, we conducted a study for how the parser accuracy varies as a function of the schema pruning model hyperparameter which was discussed in §7.4. We performed this experiment using the SEQ2SEQ model which is more affected by the pruning component, over our repartitioned SQUALL benchmark. The results are shown in Figure 7.3. It is evident that aggressive pruning has a more significant negative impact on accuracy then conservative pruning.

7.5.5 Limitations

The proposed method is of course not without any limitations and in this section we would like to put attention on some of them. While schema expansion does help significantly when tackling out-of-domain generalization on column operations, there are a lot of cases that it cannot directly handle as currently designed. For example, consider the question-table pair shown in Figure 7.4. In this case the original table contains a `"Score"` column and a `"Result"` column. The interpretation of these columns is very domain-specific and in this case, `"Score"` refers to the score in a game right before the player of that row scored a goal, while `"Result"` refers to the final score of the game. Our schema expansion component cannot help with resolving distinctions of this kind. Arguably, one might say this is a challenge inherently related to column matching, but putting details aside, our approach coupled with the proposed benchmarks does help show that column operations pose a significant challenge for existing text-to-SQL parsers, and this chapter provides a

reference point that future work can build upon. Also, note that while constructing expansion templates requires some effort and may initially seem like a limitation of our approach, we have shown that this effort can be small relative to the amount of training data that would need to be annotated otherwise.

7.6 Conclusion

In this chapter we introduced and focused on *column operations*, an important challenge related to out-of-domain generalization for Text-to-SQL parsing. We proposed two new evaluation benchmarks—one based on a new synthetic dataset and one based on a repartitioning of the SQUALL dataset—and showed that current state-of-the-art parsers significantly underperform when it comes to this form of generalization. We then proposed a simple way to incorporate prior domain knowledge to the parser via a new component called *schema expansion* that allows us to reduce the column operations challenge to column matching; an arguably easier challenge. We also introduced a *schema pruning* component allowing us to scale schema expansion, and showed that when paired together, these two components can boost the performance of arbitrary underlying Text-to-SQL parsers by a significant amount (up to 13.8% relative accuracy gain / 5.1% absolute in our experiments). We hope that this work puts attention on this important challenge and provides a reference point for future work to build upon. Possible directions include making the schema expansion component learnable or prompting models like GPT-3 [Brown et al., 2020] to address it.

Chapter 8: Conclusion

8.1 Summary

In this dissertation, we study complex question answering—questions that require multiple reasoning steps to find the answers—under two knowledge sources: text and tables.

For complex question answering over text, we build new datasets and QA systems. First, we construct a human-authored dataset QBLINK from Quizbowl competitions, where each question contains multiple parallel clues of the answer entity. We build a QA system DELFT that combines the advantages of knowledge graph QA and reading comprehension. DELFT extracts free-text sentences as the evidence signals and constructs an evidence graph, then a graph neural network models this semi-structured text graph, combines the evidence pieces and selects the answer (Chapter 3). We further build TRANSFORMER-XH to model such semi-structured text with an extra hop attention in the layers to aggregate the representations from multiple connected text pieces (Chpater 4). These QA systems are state-of-the-art on multiple complex QA benchmarks.

We then present two approaches to adapt these state-of-the-art QA systems into real scenarios. First, we present a new retrieval model BEAMDR that is based

on dense retrieval, and represents the implicit relationships between evidence pieces. BeamDR is on par with existing state-of-the-art retrievers, but not using Wikipedia hyperlinks to find evidence piece candidates, a key assumption that does not always hold on every text corpora (Chpater 5). Second, we present another retrieval model DistDR under weakly-supervised setting that does not require intermediate evidence annotations during training, since evidence annotation is usually prohibitively expansive to get, and creates dataset artifacts. DistDR iteratively improves over a weak retriever by alternately finding evidence from the up-to-date model and encouraging the model to learn the most likely evidence. Without using evidence labels, DistDR is on par with BeamDR, its fully-supervised counterpart (Chpater 6).

We next shifted our attention to complex question answering over tables, that is often reframed as text-to-SQL semantic parsing problem. We focus on out-of-domain generalization, and specifically a practical problem over complex questions: matching domain specific phrases to composite operation over columns. We first construct new benchmarks to quantify this problem, then we address this problem by incorporating prior domain knowledge through preprocessing table schemas, and designing a method that consists of two components: *schema expansion* and *schema pruning* that are applicable to any existing state-of-the-art parsers. We find that baseline parsers falter on this problem, and our method significantly outperforms baseline parsers, and as a result boosting the overall performance (Chpater 7).

8.2 Future Directions

In this section, we discuss some potential future research directions in the area of question answering, including a unified open-domain QA system, calibration, faithfulness, interaction with humans and multi-model reasoning.

8.2.1 A Unified Open-domain QA System

According to Chapter 6, DistDR can accurately retrieve evidence for both simple questions and complex questions, without requiring evidence labels. However, a limitation is that the number of evidence retrieval steps is pre-defined, i.e., simple questions take one retrieval step, and complex questions in the benchmarks (e.g., HotpotQA) take two retrieval steps. And the QA system cannot answer questions where the answer is not a passage span. Following this motivation, I propose to build a unified QA system that targets at factoid questions of any complexity. The proposed system should have three features: adaptive retrieval steps; advanced answer prediction modules; training under weak supervision.

First, we argue that the complexity of questions relies on the text corpora. As shown in Figure 1.1, to answer "which city was Facebook launched?", two connected evidence pieces are required in Wikipedia. However, another text collections could directly include the following evidence sentence: "Facebook was launched in Cambridge, Massachusetts", and as a result, only a single retrieval step is necessary. Following this intuition, QA system should adapt to varying number of evidence retrieval steps, which can be framed as a sequential decision problem. Qi et al.

[2021] also considers flexible retrieval steps, after each retrieval step, the model decides whether it should stop (as necessary information is retrieved), or continue for another step. The disadvantage of this approach is that the model is trained on the combination of simple QA and multi-hop QA datasets, where the questions are easily distinguishable, as multi-hop questions are usually much longer; However, in real world (as shown in Figure 1.1), these questions are usually indistinguishable, and making training a retriever still challenging.

Second, the proposed system should have more advanced reader modules to predict answers from evidence. Most current open-domain QA systems predict a span in a passage as the answer. However, the actual answer can be an equivalent entity that is not mentioned in the passage [Si et al., 2021], multiple discontinuous spans [Segal et al., 2020], or rather mathematical operations (e.g., sum of two numbers) over multiple spans [Dua et al., 2019]. The proposed unified QA system should tackle these problems.

Finally, as mentioned in Chapter 6, annotating evidence for questions is non-trivial, and more challenging if we don't know how many evidence pieces will be used. Therefore, it's more beneficial to focus on the weakly-supervised setting and use iterative approaches for evidence discovery.

8.2.2 Calibration in Open-domain QA

One end goal of QA systems is to increase user trust of system output. To avoid wrong answers, QA systems should know when to abstain. Previous approaches on

QA calibration [Kamath et al., 2020, Jiang et al., 2020] mostly focus on generalization on new domains under closed-domain, but an important problem is missing: the model needs to abstain if the reasoning skills required for the question exceeds the model capacities. More specifically, we focus on a QA calibration from a different angle: given an open-domain QA dataset such as NaturalQuestions [Kwiatkowski et al., 2019], most examples are simple questions, with a small number of complex questions (but we don't know them beforehand). We argue that for existing QA systems focused on simple questions, these small number of complex questions should be abstained to answer.

Having a good calibration model also plays an important role in the proposed unified QA system (Section 8.2.1). Since after each evidence retrieval step, the system needs to decide whether it should continue to retrieve evidence, or just stop. And it is the calibration model's job to notify the system whether already retrieved evidence is enough.

There are several challenges that make this problem challenging: The dataset itself does not tell us which questions are complex (the assumption for the dataset is that all questions are answerable by the existing system), getting the training signal is non-trivial; Neural network is known to be overly confident [Desai and Durrett, 2020]. Unlike previous calibration approaches [Kamath et al., 2020, Jiang et al., 2020] that only calibrate on the final answer prediction score (since they work on closed-domain QA), we propose to combine the retrieval and answer prediction scores, and make the joint prediction.

8.2.3 Towards Faithful QA Systems

To increase the user trust, QA systems should also be faithful [Jacovi and Goldberg, 2020b] to the predictions, which accurately represents the reasoning process behind the predictions. According to Chapter 6, one problem for existing QA systems is that these systems predict the correct answer, but for the incorrect reason, i.e., the evidence is incorrect. This is also a known problem in weakly-supervised semantic parsing, as multiple denotations could execute to the correct answer, but only some of them match the meaning of the questions. A possible solution is to introduce adversarial features [Feng et al., 2018] to increase the model robustness.

8.2.4 Evidence Discovery by Interacting with Humans

Finding evidence for complex questions is challenging for machines. A promising idea is find evidence by interacting with human users. Specifically, given a complex question, users try to answer it with the help of a search engine. Usually users will type a query, get some information from the search engine, then refine the query until it reaches the correct answer. For example, given a question "Who organized a conference that initiated the scramble for Africa?", users first search "Scramble for Africa", then search "Berlin Conference", and highlight "organize" over the search results to find the answer Otto von Bismarck. In addition to collecting high quality data from expert users, we can also compare human and machine strategies, and further build an agent to combine the best of both worlds.

8.2.5 Multimodal QA

Another promising direction is to focus on multimodal QA. In this thesis, we use text or tables as knowledge source. However, images or audios also include useful information, and more importantly, as complex questions usually require multiple evidence pieces, it's more likely that these pieces appear in multiple knowledge sources. A promising research problem is to make use of cross-modality information to tackle ambiguous questions [Min et al., 2020b]. For example, an ambiguous question "Who has the record for most super bowl losses?" might ask about a team, a player, or a coach. However, if the related table is about records for all NFL teams, this question is automatically disambiguated.

8.3 Last Word: Towards More Practical Complex QA

Throughout this chapter, we work towards a coherent vision of how to further make complex QA systems into real scenarios. Rather than a separate complex QA system, we believe it is more beneficial to have a unified QA system that can answer both simple and complex questions. To achieve that goal, the system should make a sequence of decisions based on the complexity of the questions, and it should know when to stop making additional decision steps, and be faithful to the decisions. We also believe future QA systems should be able to interact with humans, and rely on different forms of world knowledge.

Bibliography

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1995. URL https://arxiv.org/abs/cmp-lg/9503016.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over Wikipedia graph for question answering. In *Proceedings of the International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgVHkrYDH.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. Beat the ai: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 2020. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00338/96474/Beat-the-AI-Investigating-Adversarial-Human.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. doi: 10.1109/TPAMI.2013.50.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*, 2013.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 2008.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé III. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of Empirical Methods in Natural Language Processing*, 2012.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of Advances in Neural Information Processing Systems*, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Association for Computational Linguistics*, 2013.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. Clueweb09 data set, 2009.

Danqi Chen. *Neural Reading Comprehension and Beyond*. PhD thesis, Stanford University, 2018.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the Association for Computational Linguistics*, 2017.

Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Yang Wang. How large a vocabulary does text classification need? a variational approach to vocabulary selection. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/N19-1352. URL `https://aclanthology.org/N19-1352`.

Hao Cheng, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Probabilistic assumptions matter: Improved models for distantly-supervised document-level question answering. In *Proceedings of the Association for Computational Linguistics*, 2020. doi: 10.18653/v1/2020.acl-main.501. URL `https://www.aclweb.org/anthology/2020.acl-main.501`.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the Association for Computational Linguistics*, 2018.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September

2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1070.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the Association for Computational Linguistics*, 2019.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of Empirical Methods in Natural Language Processing*, 2019. doi: 10.18653/v1/D19-1606. URL https://aclanthology.org/D19-1606.

Hal Daumé III and Jagadeesh Jagarlamudi. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the Association for Computational Linguistics*, 2011. URL https://aclanthology.org/P11-2071/.

Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 2006. URL https://www.aaai.org/Papers/JAIR/Vol26/JAIR-2603.pdf.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-sql. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021. URL https://aclanthology.org/2021.naacl-main.105/.

Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 295–302, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxstlHFPH.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the Association for Computational Linguistics*, 2019.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Knowledge Discovery and Data Mining*, 2014.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/N19-1246. URL `https://aclanthology.org/N19-1246`.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International Semantic Web Conference*, 2019.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. Dataset and baselines for sequential open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. Can you unpack that? learning to rewrite questions-in-context. In *Proceedings of Empirical Methods in Natural Language Processing*, 2019. URL `https://www.aclweb.org/anthology/D19-1605/`.

Yair Feldman and Ran El-Yaniv. Multi-Hop Paragraph Retrieval for Open-Domain Question Answering. In *Proceedings of the Association for Computational Linguistics*, 2019.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2010.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3): 59–79, 2010.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving text-to-SQL evaluation methodology. In *Proceedings of the Association for Computational Linguistics*, 2018. doi: 10.18653/v1/P18-1033. URL `https://www.aclweb.org/anthology/P18-1033`.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. Question answering is a format; when is it useful? *arXiv preprint arXiv:1909.11291*, 2019.

Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2015. ISBN 1449358543, 9781449358549.

Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the World Wide Web Conference*, 2021. URL `https://arxiv.org/abs/2011.07743`.

Anupam Guha, Mohit Iyyer, Danny Bouman, and Jordan Boyd-Graber. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2015. URL `https://www.aclweb.org/anthology/N15-1117`.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the International Conference of Machine Learning*, 2020. URL `https://arxiv.org/abs/2002.08909`.

Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Entity Linking in Queries: Efficiency vs.Effectiveness. In *European Conference on Information Retrieval*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision*, 2015.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the Association for Computational Linguistics*, 2020. URL `https://aclanthology.org/2020.acl-main.398/`.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the Association for Computational Linguistics*, 2020a. URL `https://www.aclweb.org/anthology/2020.acl-main.386`.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, 2020b.

Ken Jennings. *Brainiac: adventures in the curious, competitive, compulsive world of trivia buffs*. Villard, 2006. ISBN 9781400064458.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of Empirical Methods in Natural Language Processing*, 2017.

Yichen Jiang and Mohit Bansal. Avoiding Reasoning Shortcuts: Adversarial Evaluation, Training, and Model Development for Multi-Hop QA. In *Proceedings of the Association for Computational Linguistics*, 2019.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 2020. doi: 10.1162/tacl_a_00324. URL https://aclanthology.org/2020.tacl-1.28.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*, 2017. URL https://arxiv.org/abs/1702.08734.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*, 2017a. doi: 10.18653/v1/P17-1147. URL https://www.aclweb.org/anthology/P17-1147.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*, 2017b.

Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In *Proceedings of the Association for Computational Linguistics*, 2020.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2020. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of the International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygcCnNKwr.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2017.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of Empirical Methods in Natural Language Processing*, 2013.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. In *Transactions of the Association for Computational Linguistics*, 2019.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the International Conference of Machine Learning*, 2018. URL `https://arxiv.org/abs/1711.00350`.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the Association for Computational Linguistics*, 2021. doi: 10.18653/v1/2021.acl-long.176. URL `https://aclanthology.org/2021.acl-long.176`.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the Association for Computational Linguistics*, 2019. URL `https://www.aclweb.org/anthology/P19-1612`.

Wendy Grace Lehnert. *The process of question answering.* Yale University, 1977a.

Wendy Grace Lehnert. *The Process of Question Answering.* PhD thesis, USA, 1977b. AAI7728146.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara Berg. TVQA: Localized, compositional video question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018. URL `https://www.aclweb.org/anthology/D18-1167/`.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations*, 2018.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the Association for Computational Linguistics*, 2019a.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b.

Zhenghao Liu, Chenyan Xiong, and Maosong Sun. Kernel Graph Attention Network for Fact Verification. *arXiv preprint arXiv:1910.09796*, 2019c.

Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008. URL `https://www.jmlr.org/papers/v9/vandermaaten08a.html`.

Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: Shedding light on the web of documents. In *Proceedings of the International Conference on Semantic Systems*, 2011.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. Efficient and robust question answering from minimal context over documents. In *Proceedings of the Association for Computational Linguistics*, 2018.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional Questions Do Not Necessitate Multi-hop Reasoning. In *Proceedings of the Association for Computational Linguistics*, 2019a.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the Association for Computational Linguistics*, 2019b.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the Association for Computational Linguistics*, 2019c.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of Empirical Methods in Natural Language Processing*, 2020a. URL `https://www.aclweb.org/anthology/2020.emnlp-main.466/`.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of Empirical Methods in Natural Language Processing*, 2020b.

Michael Minock, Peter Olofsson, and Alexander Näslund. Towards building robust natural language interfaces to databases. In *International Conference on Application of Natural Language to Information Systems*, 2008. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.158.70&rep=rep1&type=pdf.

Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Proceedings of Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/21c5bba1dd6aed9ab48c2b34c1a0adde-Paper.pdf.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Yixin Nie, Songhe Wang, and Mohit Bansal. Revealing the Importance of Semantic Retrieval for Machine Reading at Scale. In *Proceedings of Empirical Methods in Natural Language Processing*, 2019a.

Yixin Nie, Songhe Wang, and Mohit Bansal. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of Empirical Methods in Natural Language Processing*, 2019b. URL https://www.aclweb.org/anthology/D19-1258/.

Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. Torque: A reading comprehension dataset of temporal ordering questions. In *Proceedings of Empirical Methods in Natural Language Processing*, 2020. URL https://aclanthology.org/2020.emnlp-main.88.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. Answering while Summarizing: Multi-task Learning for Multi-hop QA with Evidence Extraction. In *Proceedings of the Association for Computational Linguistics*, 2019.

Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019.

Takashi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of Empirical Methods in Natural Language Processing*, 2016.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the Association for Computational Linguistics*, 2015. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Heiko Paulheim. How much is a triple? estimating the cost of knowledge graph creation. In *International Semantic Web Conference*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

Peng Qi, Haejun Lee, Oghenetegiri "TG" Sido, and Christopher D. Manning. Answering open-domain questions of varying reasoning steps from text. In *Proceedings of Empirical Methods in Natural Language Processing*, 2021.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*, 2016.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*, 2013.

Pedro Rodriguez and Jordan Boyd-Graber. Evaluation paradigms in question answering. In *Empirical Methods in Natural Language Processing*, page 5, 2021. URL `http://umiacs.umd.edu/~jbg//docs/2021_emnlp_paradigms.pdf`.

Ohad Rubin and Jonathan Berant. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021. doi: 10.18653/v1/2021.naacl-main.29. URL `https://aclanthology.org/2021.naacl-main.29`.

Daniel M Russell. *The Joy of Search: A Google Insider's Guide to Going Beyond the Basics*. MIT Press, 2019. URL `https://mitpress.mit.edu/books/joy-search`.

Gerard. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968. ISBN 0070544859.

Gerard Salton and Michael J McGill. *Introduction to Modern Information Retrieval.* mcgraw-hill, 1983.

Rajhans Samdani, Ming-Wei Chang, and Dan Roth. Unified expectation maximization. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2012. URL `https://www.aclweb.org/anthology/N12-1087.pdf`.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. A simple and effective model for answering multi-span questions. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 3074–3080, 2020.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations*, 2016.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? *Proceedings of the Association for Computational Linguistics*, 2021. URL https://aclanthology.org/2021.acl-long.75/.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP*, 2020. URL https://aclanthology.org/2020.findings-emnlp.167/.

Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Proceedings of Advances in Neural Information Processing Systems*, 2014. URL https://dl.acm.org/doi/10.5555/2969033.2969086.

Chenglei Si, Chen Zhao, and Jordan L. Boyd-Graber. What's in a name? answer equivalence for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2021. URL https://aclanthology.org/2021.emnlp-main.757.

Robert F Simmons, Sheldon Klein, and Keren McConlogue. Indexing and dependency logic for answering english questions. *American Documentation*, 15(3): 196–204, 1964.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the World Wide Web Conference*, 2007.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the Association for Computational Linguistics*, 2020. doi: 10.18653/v1/2020.acl-main.742. URL https://www.aclweb.org/anthology/2020.acl-main.742.

Alon Talmor and Jonathan Berant. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the Association for Computational Linguistics*, 2019. URL https://aclanthology.org/P19-1485/.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. The Fact Extraction and VERification (FEVER) Shared Task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Ellen M Voorhees, Dawn M Tice, et al. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82. Citeseer, 1999.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledge base. 2014.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the Association for Computational Linguistics*, 2020. URL `https://aclanthology.org/2020.acl-main.677/`.

Bailin Wang, Mirella Lapata, and Ivan Titov. Meta-learning for domain generalization in semantic parsing. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021. URL `https://aclanthology.org/2021.naacl-main.33/`.

Mengqiu Wang. A survey of answer extraction techniques in factoid question answering. *Computational Linguistics*, (1), 2006.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

William A Woods and James G Schmolze. The kl-one family. *Computers & Mathematics with Applications*, 23(2-5):133–177, 1992.

William A Woods and WOODS WA. Lunar rocks in natural english: Explorations in natural language question answering. 1977.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of the International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=zeFrfgyZln`.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. Answering complex open-domain questions with multi-hop dense retrieval. In *Proceedings of the International Conference on*

Learning Representations, 2021b. URL https://openreview.net/forum?id=EMHoBG0avc1.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the Association for Computational Linguistics*, 2021. URL https://aclanthology.org/2021.acl-long.571/.

Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017. URL https://arxiv.org/abs/1711.04436.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the Association for Computational Linguistics*, 2014.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Association for Computational Linguistics*, 2015.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the Association for Computational Linguistics*, 2020. URL https://aclanthology.org/2020.acl-main.745/.

Pengcheng Yin, John Wieting, Avirup Sil, and Graham Neubig. On the ingredients of an effective zero-shot semantic parser. *arXiv preprint arXiv:2110.08381*, 2021. URL https://arxiv.org/abs/2110.08381.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018. doi: 10.18653/v1/D18-1425. URL https://www.aclweb.org/anthology/D18-1425.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Caiming Xiong, et al. Grappa: Grammar-augmented pre-training for table

semantic parsing. In *Proceedings of the International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=kyaIeYj4zZ.

Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of Empirical Methods in Natural Language Processing*, 2018.

Chen Zhao, Yu Su, Adam Pauls, and Emmanouil Antonios Platanios. Bridging the generalization gap in text-to-sql parsing with schema expansion. In *In Submission*.

Chen Zhao, Chenyan Xiong, Xin Qian, and Jordan Boyd-Graber. Complex factoid question answering with a free-text knowledge graph. In *Proceedings of the World Wide Web Conference*, 2020a. URL https://arxiv.org/abs/2103.12876.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *Proceedings of the International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=r1eIiCNYwS.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. Distantly-supervised dense retrieval enables open-domain question answering without evidence annotation. In *Empirical Methods in Natural Language Processing*, 2021a. URL http://umiacs.umd.edu/~jbg//docs/2021_emnlp_weak_dpr.pdf.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. Multi-step reasoning over unstructured text with beam dense retrieval. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021b. URL https://arxiv.org/abs/2104.05883.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017. URL https://arxiv.org/abs/1709.00103.

Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. Coarse-grain fine-grain coattention network for multi-evidence question answering. In *Proceedings of the International Conference on Learning Representations*, 2019.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. In *Proceedings of the Association for Computational Linguistics*, 2019.