Complex Factoid Question Answering with a Free-Text Knowledge Graph

Chen Zhao
Computer Science, UMIACS
University of Maryland, College Park
chenz@cs.umd.edu

Chenyan Xiong
Microsoft AI & Research
cxiong@microsoft.com

Xin Qian
iSchool
University of Maryland, College Park
xinq@terpmail.umd.edu

Jordan Boyd-Graber†
Computer Science, iSchool, UMIACS, LSC
University of Maryland, College Park
jbg@umiacs.umd.edu

ABSTRACT

We introduce DELFT, a factoid question answering system which combines the nuance and depth of knowledge graph question answering approaches with the broader coverage of free-text. DELFT builds a free-text knowledge graph from Wikipedia, with entities as nodes and sentences in which entities co-occur as edges. For each question, DELFT finds the 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 edge sentences—to select a final answer. Experiments on three question answering datasets show DELFT can answer entity-rich questions better than machine reading based models, BERT-based answer ranking and memory networks. DELFT's advantage comes from both the high coverage of its free-text knowledge graph—more than double that of DBpedia relations—and the novel graph neural network which reasons on the rich but noisy free-text evidence.

KEYWORDS

Free-Text Knowledge Graph, Factoid Question Answering, Graph Neural Network

ACM Reference Format:
Chen Zhao, Chenyan Xiong, Xin Qian, and Jordan Boyd-Graber†. 2020. Complex Factoid Question Answering with a Free-Text Knowledge Graph. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3366423.3380197

†Now at Google Research Zürich.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380197

1 INTRODUCTION

Factoid question answering [22, 25, 54, 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) such as Freebase [3], DBpedia [34], YAGO [50], and Wikidata [53] have been fruitfully applied to factoid QA over knowledge graphs (KGQA) [4, 16, inter alia]. 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 [7, 30] or aligning questions to substructures of a knowledge graph [60, 61]. 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 [41]. For example, Google's Knowledge Vault has 570M entities, fourteen times more than Freebase, but only has 35K relation types: similar to Freebase [3, 17].

In the real world, human knowledge is often tested through competitions like Quizbowl [6] or *Jeopardy!* [21]. In these settings, the question clues are complex, diverse, and linguistically rich, most of 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. Figure 1 shows a real world factoid QA example. 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 machine reading (MR), which finds an answer directly from unstructured text [10]. Because it extracts answers from paragraphs and documents—pre-given or retrieved [12, 44, 63]—MR has much broader coverage [39]. However, current MR datasets mainly evaluate on reasoning within a single paragraph [37]; existing MR models focus on extracting from single evidence. Synthesizing multiple pieces of evidence to answer complex questions remains an on-going research topic [36, 59].

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). Each edge in our graph is associated with multiple sentences that give explain how entities interact with each other. 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 machine reading based model [11], BERT-based answer ranking [14], and a BERT-based memory network approach [56] 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 representation, 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.[1]

2 TASK DEFINITION

The problem we consider is general *factoid question answering* [2, 5, 7, 25, inter alia], in which the system answers natural language questions using facts, i.e., entities or attributes from knowledge graphs. 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 [2, 5], while in reality, minimal KG coverage preculdes broader deployment.

For example, WebQuestions [2] are written by crowdworkers targeting a triple in Freebase; LC-quad [18] 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 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 [27]. The datasets in our evaluation are solvable by humans using knowledge about real world entities; DELFT should also be able to find the answer entity without relying on explicit closed form relations.

Figure 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). 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 4).

3 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 [33, 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 indirect usings 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.

3.1 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 [8] 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 [20] finds entities mentioned in free text; we apply it to Wikipedia pages. Given the entity linker's

[1] The code, data, full free-text knowledge graph, and question grounded subgraph is at delft.qanta.org.

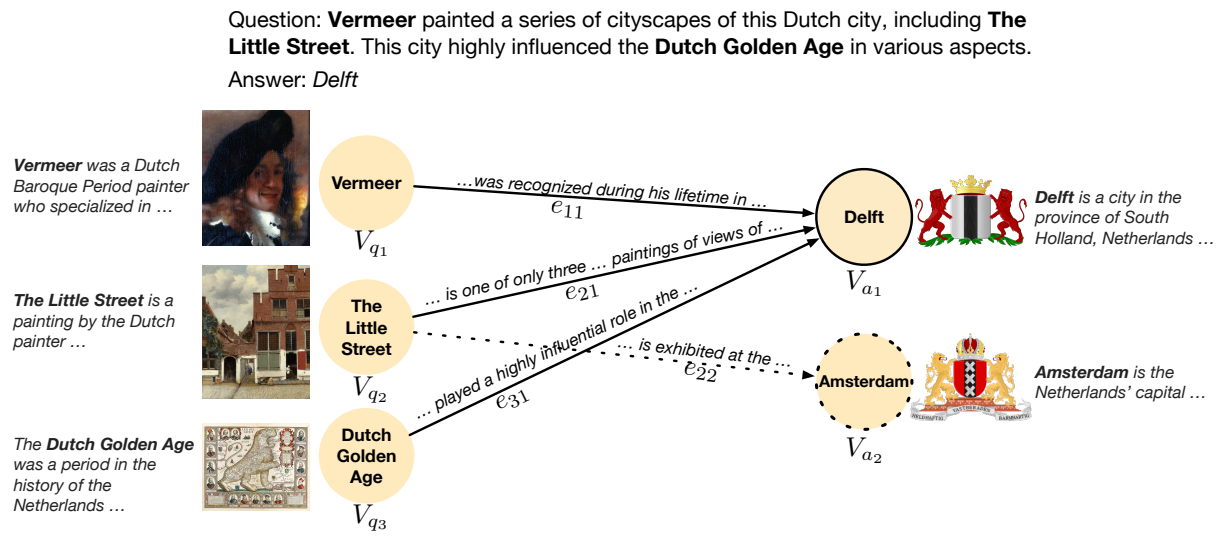


Figure 1: An example question grounded free-text knowledge graph in DELFT. The graph has both question entity nodes (left side) and candidate entity nodes (right side), and use natural language sentences from Wikipedia for both nodes and edges.

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.

3.2 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 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 $V_q = \{v_i \mid v_i \in \mathcal{X}_q\}$. In Figure 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 [24]), 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).

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 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.

3.3 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:

```
[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 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 [46] 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.

4 FREE-TEXT GRAPH MODELING

Given the question X_q and the grounded free-text graph G from Section 3, the goal is to find the correct answer node from the Candidate Entity Nodes. Recently several GNN based approaches [13, 52] 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 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 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.

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

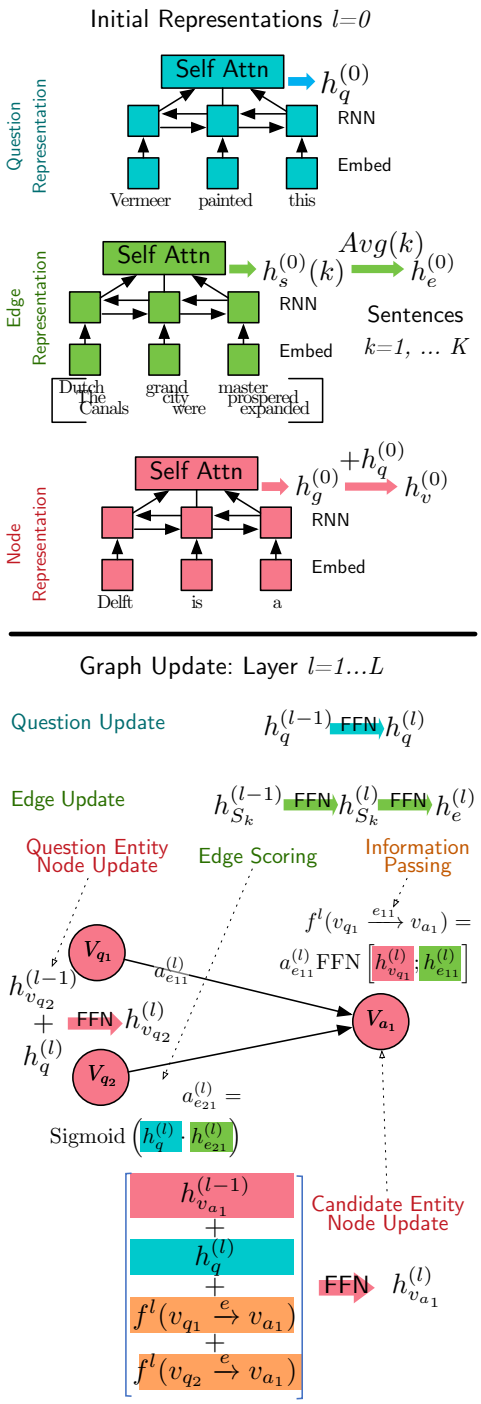


Figure 2: Model architecture of DELFT's GNN. The left side shows the initial representation of the network. The right side illustrates the graph update.

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 2 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 4.1). Each successive layer combines this information, focusing on the Evidence Edges and Candidate Entity Nodes that can best answer the question (Section 4.2). The final layer scores the Candidate Entity Nodes to produces an answer (Section 4.3).

4.1 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:

$$\mathbf{h}_{x_u} = \text{RNN}\left(\mathbf{E}_{x_1}, \ldots, \mathbf{E}_{x_n}\right) \tag{1}$$

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 $\mathbf{w}_x$

$$a_{x_u} = \text{softmax}\left(\mathbf{w}_x \cdot \mathbf{h}_{x_u}\right), \tag{2}$$

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

$$\mathbf{h}_x^{(0)} = \sum_u a_{x_u} \mathbf{h}_{x_u}. \tag{3}$$

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 $\mathbf{h}_q^{(0)}$.

$$\mathbf{h}_v^{(0)} = \mathbf{h}_g^{(0)} + \mathbf{h}_q^{(0)}. \tag{4}$$

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:

$$\mathbf{h}_{s_u} = \text{RNN}\left(\mathbf{E}_{s_1}, ..., \mathbf{E}_{s_n}\right). \tag{5}$$

Then we fuse the question information into each edge sentence. An inter attention layer [49] based on the question representation $\mathbf{h}_q^{(0)}$ first weights each sentence token position u

$$a_{s_u} = \text{softmax}\left(\mathbf{h}_{s_u} \cdot \mathbf{h}_q^{(0)}\right); \tag{6}$$

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

$$\mathbf{h}_s^{(0)}(k) = \sum_u a_{s_u} \mathbf{h}_{s_u}. \tag{7}$$

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

$$\mathbf{h}_e^{(0)} = \text{Avg}_k\left(\mathbf{h}_s^{(0)}(k)\right). \tag{8}$$

4.2 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 2.

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 $\mathbf{h}_{v_q}^{(l-1)}$,

$$\mathbf{h}_{v_q}^{(l)} = \text{FFN}\left(\mathbf{h}_{v_q}^{(l-1)} + \mathbf{h}_q^{(l)}\right). \tag{9}$$

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,

$$\mathbf{h}_q^{(l)} = \text{FFN}\left(\mathbf{h}_q^{(l-1)}\right), \tag{10}$$

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,

$$\mathbf{h}_s^{(l)}(k) = \text{FFN}\left(\mathbf{h}_s^{(l-1)}(k)\right), \tag{11}$$

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

$$\mathbf{h}_e^{(l)} = \text{Avg}_k \left(\mathbf{h}_s^{(l)}(k) \right). \quad (12)$$

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:

$$a_e^{(l)} = \text{Sigmoid} \left(\mathbf{h}_q^{(l)} \cdot \mathbf{h}_e^{(l)} \right). \quad (13)$$

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 13),

$$f^{(l)} \left(v_q \xrightarrow{e} v_a \right) = a_e^{(l)} \text{FFN} \left(\left[\mathbf{h}_{v_q}^{(l)}; \mathbf{h}_e^{(l)} \right] \right). \quad (14)$$

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.

$$\mathbf{h}_{v_a}^{(l)} = \text{FFN} \Big(\underbrace{\mathbf{h}_{v_a}^{(l-1)}}_{\text{previous}} + \underbrace{\mathbf{h}_q^{(l)}}_{\text{question}} + \underbrace{\sum_{v_q \in V_q} f^{(l)}(v_q \xrightarrow{e} v_a)}_{\text{Information Passing}} \Big). \quad (15)$$

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

4.3 Answer Scoring

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

$$p(v_a; G) = \text{Sigmoid} \left(\text{MLP}(\mathbf{h}_{v_a}^{(L)}) \right). \quad (16)$$

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 $p(v_a, G)$.

5 EXPERIMENTS

We evaluate on three datasets with expert-authored (as opposed to crowd-worker) questions. QbLink [19] 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 [25] is a QA dataset collected from Quizbowl competitions. Each question is a sequence of sentences providing increased information about the answer entity. TriviaQA [28] includes questions from trivia games and is a benchmark dataset for MR. 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 1: 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.

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 2: Parameters in DELFT's GNN.

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 1 shows the statistics of these three datasets sand the fraction of questions with entities.

5.1 Question Answering Methods

We compare the following methods:

- QUEST [33] is an unsupervised factoid QA system over text. For fairness, instead of Google results we apply QUEST on IR-retrieved Wikipedia documents.
- DrQA [11] is a machine reading model for open-domain QA that retrieves documents and extracts answers.
- DocQA [12] improves multi-paragraph machine reading, and is among the strongest on TriviaQA. Their suggested settings and pre-trained model on TriviaQA are used.
- BERT-Entity fine-tunes BERT [14] 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 candidate entities in DELFT's graph.
- BERT-MemNN is a memory network [56] 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 [42] and BERT embeddings.

5.2 Implementation

Our implementation uses PyTorch [40] 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 2. For DELFT-BERT, we use BERT output as contextualized 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.

6 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.

6.1 Graph Coverage

DELFT's graph has high coverage (Table 3). 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.

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

6.2 Answer Accuracy

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

QUEST falters on these questions, suggesting that some level of supervision is required. On more complicated factoid QA datasets QbLink and QANTA, DELFT improves over DrQA, the machine reading 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 MR 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 MR 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.

6.3 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(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(b)), since it brings in uninformative entities (e.g., linking "name the first woman in space" to "Name"). Filtering too agressively, 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.

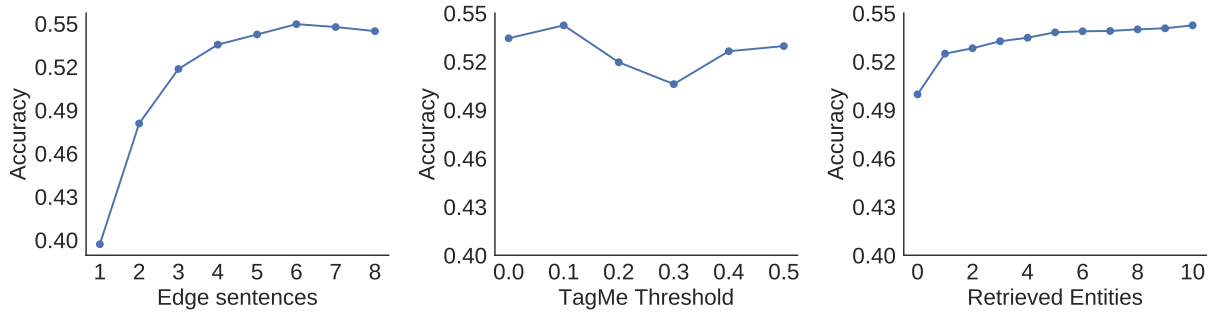
[4] Recall, however, that we exclude questions that with no entities or whose answer is not an entity.

	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 Graph*	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: 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.

	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 4: 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.



(a) Number of Edge Sentences **(b) TagMe Threshold on Question Entities** **(c) Number of Retrieved Entities**

Figure 3: Accuracy of DELFT on different variations of Free-Text Knowledge Graphs.

To see how senstitive 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.

Recall that DELFT uses not just the entities in the question but also searches for edges similar to question text. Figure 3(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 5. As expected, both node (gloss) and edge evidence help; each contributes ∼ 10% accuracy. Edge importance scoring, which controls the weights of the information flow from Question Entity Node to Candidate Entity Nodes, provides ∼ 3% accuracy. The input representation is important as well; the self-attention layer contributes ∼ 2.2% accuracy.

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

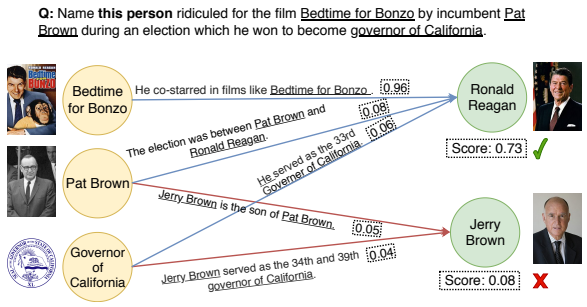


Figure 4: An example DELFT subgraph. The learned edge weights in GNN are in the brackets.

6.4 Graph Visualization

Figure 4 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 Bonzo` 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.

6.5 Case Study

To gain more insights into DELFT model's behavior, we further sample some examples from QbLink. Table 6 shows two positive examples (1 and 2) and two negative examples (3 and 4). DELFT succeeds on these cases: with direct evidence sentences, DELFT finds the correct answer with high confidence (Example 1) or with multiple pieces of evidence, DELFT could aggregate different pieces together, and make more accurate predictions (Example 2). However some common sources of error include: too few informative entities in the question (Example 3) or evidence that overlaps too much between two Candidate Entity Nodes.

7 RELATED WORK: KNOWLEDGE REPRESENTATION FOR QA

DELFT is impossible without the insights of traditional knowledge bases for question answering and question answering from natural language, which we combine using graph neural networks. This section describes how we build on these subfields.

7.1 Knowledge Graph Question Answering

With knowledge graphs (KG) like Freebase [3] and DBpedia [34] enable question answering using their rich, dependable structure. This has spurred KG-specific QA datasets on general domain large scale knowledge graphs: WebQuestions [2], SimpleQuestions [5], and special-domain KGs, such as WikiMovies [35]. In turn, these new datasets have prompted special-purpose KGQA algorithms. Some convert questions to semantic parsing problems and execute the logical forms on the graph [7, 30, 45, 61]. Others use information extraction to first extract question related information in KG and then find the answer [1, 23, 60].

These work well on questions tailored for the underlying KG. For example, WebQuestions guarantee its questions can be answered by Freebase [2]. Though modern knowledge graphs have good coverage on entities [57], adding relations takes time and money [41], often requiring human effort [3] or scraping human-edited structured resources [34]. These lacunærepresent impede broader use and adoption.

Like DELFT, QUEST [33] seeks to address this by building a noisy quasi-KG with nodes and edges, consisting of dynamically retrieved entity names and relational phrases from raw text. Unlike DELFT, this graph is built using existing Open Information Extraction (IE). Then it answers questions on the extracted graph. Unlike DELFT, which is geared toward recall, IE errs toward precision and require regular, clean text. In contrast, many real-world factoid questions contain linguistically rich structures, making relation extraction challenging. We instead directly extract free-text sentences as indirect relations between entities, which ensures high coverage of evidence information to the question.

Similarly, GRAFT-Net [51] extends an existing KG with text information. It grafts text evidence onto KG nodes but retains the original KG relations. It then reasons over this graph to answer KG-specific questions. DELFT, in contrast, grafts text evidence onto both nodes and edges to enrich the relationships between nodes, building on the success of unconstrained "machine reading" QA systems.

7.2 Question Answering over Text

Compared with highly structured KG, unstructured text collections (e.g., Wikipedia, newswire, or Web scrapes) is cheaper but noisier for QA [10]. Recent datasets such as SQuAD [44], TriviaQA [28], MS MARCO [39] and natural questions [31] are typically solved via a coarse search for a passage (if the passage isn't given) and then finding a fine-grained span to answer the question.

A rich vein of neural readers match the questions to the given passages and extract answer spans from them [49, 62, inter alia]. Its popular solutions include BIDAF, which matches the question and document passages by bi-directional attention flows [49], QANet, which enriches the local contexts with global self-attention [62], and pre-training methods such as BERT [14] and XLNet [58].

id	Example	Explanation
1(+)	**Q**: This general was the leader of the nationalist side in the civil war and ruled until his death in 1975. He kept Spain neutral in the Second World War and is still dead. **A**: Francisco Franco **P**: Francisco Franco	The question directly maps to evidence sentence "After the nationalist victory in the Spanish Civil War, until his (Francisco Franco) death in 1975".
2(+)	**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 (correct): "Tom Brady plays for New England Patriots", and "Tom Brady had touchdown passes with Deion Branch". DELFT aggregates evidence and makes the correct prediction. Without the graph structure, BERT-MEMNN instead predicts Stephon Gilmore (Another New England Patriot).
3(-)	**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".
4(-)	**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 6: Four examples from QbLink dataset with DELFT output. Each example has a question (Q), gold answer (A) and DELFT predcition (P), along with an explanation of what happened. The first two are correct (+), while the last two are wrong (-).

The most realistic models are those that also search for a passage: DrQA [11] retrieves documents from Wikipedia and use MR to predict the top span as the answer, and ORCA [32] trains the retriever via an inverse cloze task. Nonetheless, questions mainly answerable by DrQA and ORCA only require single evidence from the candidate sets [37]. DELFT in contrast searches for edge evidence and nodes that can answer the question; this subgraph often corresponds to the same documents found by machine reading models. Ideally, it would help synthesize information across *multiple* passages.

Multi-hop QA, where answers require require assembling information [55, 59], is a *task* to test whether machine reading systems can synthesize information. HotpotQA [59] is the multi-hop QA benchmark: each answer is a text span requiring one or two hops. Several models [15, 38, 43] solve this problem using multiple MR models to extract multi-hop evidence. While we focus on datasets with Wikipedia entities as answers, expanding DELFT to span-based answers (like HotpotQA) is a natural future direction.

7.3 Graph Networks for QA

DELFT is not the first to use graph neural networks [29, 47, 48, inter alia] for question answering. Entity-GCN [13], DFGN [43], and HDE [52] build the entity graph with entity co-reference and co-occurrence in documents and apply GNN to the graph to rank the top entity as the answer. CogQA [15] builds the graph starting with entities from the question, then expanding the graph using extracted spans from multiple MR models as candidate span nodes and adopt GNN over the graph to predict the answer from span nodes. All these methods' edges are co-reference between entities or binary scores about their co-occurrence in documents; DELFT's primary distinction is using free-text as graph edges which we then represent and aggregate via a GNN.

Other methods have learned representations of relationships between entities. NUBBI [9] used an admixture over relationship prototypes, while Iyyer et al. [26] used neural dictionary learning for analyzing literature. DELFT draws on these ideas to find similarities between passages and questions.

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. Future work should explore whether these approachs 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 6: while both Odysseus and Penelope have Telemachus as a son, only Odysseus made a long journey and should thus be the answer. Recognizing specific properties of nodes either in a traditional KG or in free text could resolve these issues.

ACKNOWLEDGEMENTS

This work was supported by NSF Grants IIS-1748663 (Zhao) and IIS-1822494 (Boyd-Graber). Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

REFERENCES

[1] Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based Question Answering as Machine Tsranslation. In *Proceedings of the Association for Computational Linguistics*.

[2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of Empirical Methods in Natural Language Processing*.

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human knowledge. In *Proceedings of the ACM SIGMOD international conference on Management of data*.

[4] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question Answering with Subgraph Embeddings. In *Proceedings of Empirical Methods in Natural Language Processing*.

[5] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale Simple Question Answering with Memory Networks. *arXiv preprint arXiv:1506.02075* (2015).

[6] Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daume III. 2012. Besting the Quiz Master: Crowdsourcing Incremental Classification Games. In *Proceedings of Empirical Methods in Natural Language Processing*.

[7] Qingqing Cai and Alexander Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Association for Computational Linguistics*.

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

[9] Jonathan Chang, Jordan Boyd-Graber, and David M. Blei. 2009. Connections between the Lines: Augmenting Social Networks with Text. In *Knowledge Discovery and Data Mining*.

[10] Danqi Chen. 2018. *Neural Reading Comprehension and Beyond.* Ph.D. Dissertation. Stanford University.

[11] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the Association for Computational Linguistics*.

[12] Christopher Clark and Matt Gardner. 2018. Simple and Effective Multi-Paragraph Reading Comprehension. In *Proceedings of the Association for Computational Linguistics*.

[13] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question Answering by Reasoning Across Documents with Graph Convolutional Networks. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

[15] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. In *Proceedings of the Association for Computational Linguistics*. Proceedings of the Association for Computational Linguistics.

[16] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *Proceedings of the Association for Computational Linguistics*.

[17] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Knowledge Discovery and Data Mining*.

[18] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A Large Dataset for Complex Question Answering over Wikidata and Dbpedia. In *International Semantic Web Conference*.

[19] Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. Dataset and Baselines for Sequential Open-Domain Question Answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

[20] Paolo Ferragina and Ugo Scaiella. 2010. Tagme: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

[21] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine* 31, 3 (2010).

[22] Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. Question Answering is a Format; When is it Useful? *arXiv preprint arXiv:1909.11291* (2019).

[23] Matt Gardner and Jayant Krishnamurthy. 2017. Open-Vocabulary Semantic Parsing with Both Distributional Statistics and Formal Knowledge. In *Association for the Advancement of Artificial Intelligence*.

[24] Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

[25] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A Neural Network for Factoid Question Answering over Paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*.

[26] Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding Families and Former Friends: Unsupervised Learning for Dynamic Fictional Relationships. In *North American Association for Computational Linguistics*.

[27] Ken Jennings. 2006. *Brainiac: adventures in the curious, competitive, compulsive world of trivia buffs.* Villard.

[28] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the Association for Computational Linguistics*.

[29] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

[30] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of Empirical Methods in Natural Language Processing*.

[31] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural Questions: A Benchmark for Question Answering Research. In *Transactions of the Association for Computational Linguistics*.

[32] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6086–6096. https://doi.org/10.18653/v1/P19-1612

[33] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. 2019. 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*.

[34] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the International Conference on Semantic Systems*.

[35] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of Empirical Methods in Natural Language Processing*. Proceedings of the Association for Computational Linguistics.

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

[37] Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and Robust Question Answering from Minimal Context over Documents. In *Proceedings of the Association for Computational Linguistics*.

[38] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In *Proceedings of the Association for Computational Linguistics*.

[39] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms MARCO: A Human Generated Machine Reading Comprehension Dataset. *arXiv preprint arXiv:1611.09268* (2016).

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. (2017).

[41] Heiko Paulheim. 2018. How much is a Triple? Estimating the Cost of Knowledge Graph Creation. In *International Semantic Web Conference*.

[42] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

[43] Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically Fused Graph Network for Multi-hop Reasoning. In *Proceedings of the Association for Computational Linguistics*.

[44] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of Empirical Methods in Natural Language Processing*.

[45] Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-Scale Semantic Parsing without Question-Answer Pairs. In *Transactions of the Association for Computational Linguistics*.

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

[47] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. In *IEEE Transactions on Neural Networks*.

[48] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *arXiv preprint arXiv:1703.06103* (2017).

[49] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional Attention Flow for Machine Comprehension. In *Proceedings of the International Conference on Learning Representations*.

[50] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proceedings of the World Wide Web Conference*.

[51] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. In *Proceedings of Empirical Methods in Natural Language Processing*.

[52] Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs. In *Proceedings of the Association for Computational Linguistics*.

[53] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledge Base. (2014).

[54] Mengqiu Wang. 2006. A Survey of Answer Extraction Techniques in Factoid Question Answering. *Computational Linguistics* 1 (2006).

[55] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing Datasets for Multi-Hop Reading Comprehension Across Documents. *Transactions of the Association for Computational Linguistics*.

[56] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory Networks. *arXiv preprint arXiv:1410.3916* (2014).

[57] Chenyan Xiong. 2018. *Text Representation, Retrieval, and Understanding with Knowledge Graphs*. Ph.D. Dissertation. Carnegie Mellon University.

[58] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of Advances in Neural Information Processing Systems*.

[59] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

[60] Xuchen Yao and Benjamin Van Durme. 2014. Information Extraction Over Structured Data: Question Answering with Freebase. In *Proceedings of the Association for Computational Linguistics*.

[61] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the Association for Computational Linguistics*.

[62] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *Proceedings of the International Conference on Learning Representations*.

[63] Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. 2019. Coarse-grain Fine-grain Coattention Network for Multi-evidence Question Answering. In *Proceedings of the International Conference on Learning Representations*.