PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text

Haitian Sun Tania Bedrax-Weiss William W. Cohen
Google Research
{haitiansun,tbedrax,wcohen}@google.com

Abstract

We consider open-domain question answering (QA) where answers are drawn from either a corpus, a knowledge base (KB), or a combination of both of these. We focus on a setting in which a corpus is supplemented with a large but incomplete KB, and on questions that require non-trivial (e.g., "multi-hop") reasoning. We describe PullNet, an integrated framework for (1) learning what to retrieve and (2) reasoning with this heterogeneous information to find the best answer. PullNet uses an iterative process to construct a question-specific subgraph that contains information relevant to the question. In each iteration, a graph convolutional network (graph CNN) is used to identify subgraph nodes that should be expanded using retrieval (or "pull") operations on the corpus and/or KB. After the subgraph is complete, another graph CNN is used to extract the answer from the subgraph. This retrieve-and-reason process allows us to answer multi-hop questions using large KBs and corpora. PullNet is weakly supervised, requiring question-answer pairs but not gold inference paths. Experimentally PullNet improves over the prior state-of-the art, and in the setting where a corpus is used with incomplete KB these improvements are often dramatic. PullNet is also often superior to prior systems in a KB-only setting or a text-only setting.

1 Introduction

Open domain Question Answering (QA) is the task of finding answers to questions posed in natural language, usually using text from a corpus (Dhingra et al., 2017; Joshi et al., 2017; Dunn et al., 2017), or triples from a knowledge base (KB) (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Yih et al., 2015). Both of these approaches have limitations. Even the largest KBs are incomplete (Min et al., 2013), which limits recall of a KB-based QA system. On the other hand, while a large corpus may contain more answers than a KB, the diversity of natural language makes corpus-based QA difficult (Chen et al., 2017; Welbl et al., 2018; Kwiatkowski et al., 2019; Yang et al., 2018).

In this paper we follow previous research (Sawant et al., 2019; Sun et al., 2018) in deriving answers using both a corpus and a KB. We focus on tasks in which questions require compositional (sometimes called "multi-hop") reasoning, and a setting in which the KB is incomplete, and hence must be supplemented with information extracted from text. We also restrict ourselves in this paper to answers which correspond to KB entities. For this setting, we propose an integrated framework for (1) *learning what to retrieve*, from either a corpus, a KB, or a combination, and (2) *combining this heterogeneous information* into a single data structure that allows the system to reason and find the best answer. In prior work, this approach was termed an *early fusion* approach, and shown to improve over *late fusion* methods, in which two QA systems, one corpus-based and one KB-based, are combined in an ensemble.

Our system, PullNet, builds on the GRAFT-Net[1] (Sun et al., 2018) early fusion system. GRAFT-Net uses heuristics to build a question-specific subgraph which contains sentences from the corpus, and entities and facts from the KB. A graph CNN (Kipf and Welling, 2016; Li et al., 2016; Schlichtkrull et al., 2017) variant is then used to reason over this graph and select an answer. However, as we will show experimentally, GRAFT-Net's heuristics often produce subgraphs that are far from optimal–often they are much larger than necessary, and sometimes do not contain the answer.

Like GRAFT-Net, PullNet also uses a reason-

[1] Graphs of Relations Among Facts and Text Networks.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2380–2390, Hong Kong, China, November 3–7, 2019.

ing process based on a graph CNN to find answers. However, PullNet *learns* how to construct the subgraph, rather than using an *ad hoc* subgraph-building strategy. More specifically, PullNet relies on a small set of retrieval operations, each of which expands a graph node by retrieving new information from the KB or the corpus. PullNet learns when and where to apply these "pull" operations with another graph CNN classifier. The "pull" classifier is weakly supervised, using question-answer pairs.

The end result is a learned iterative process for subgraph construction, which begins with a small subgraph containing only the question text and the entities which it contains, and gradually expands the subgraph to contain information from the KB and corpus that are likely to be useful. The incremental question-guided subgraph construction process results in high-recall subgraphs that are much smaller than the ones created heuristically, making the final answer extraction process easier. The process is especially effective for multi-hop questions, which naively would require expanding the subgraph to include all corpus and KB elements that are k hops away from the question.

PullNet improves over the current state-of-the-art for KB-only QA on several benchmark datasets, and is superior to, or competitive with, corpus-only QA on several others. For multi-hop questions, this improvement is often dramatic. For instance, MetaQA (Zhang et al., 2018) contains multi-hop questions based on a small movie KB, originally associated with the WikiMovies dataset (Miller et al., 2016). In a KB-only setting, PullNet improves hits-at-one performance for 3-hop MetaQA questions from 62.5% to 91.4%. Perhaps more interestingly, PullNet obtains performance of 85.2% hits-at-one with a KB from which half of the triples have been removed, if that KB is supplemented with a corpus. We note that this result improves by 7% (absolute improvement) over a pure corpus-based QA system, and by more than 25% over a pure KB-based QA system. In a similar incomplete-KB setting, PullNet improves over GRAFT-Net by 6.8% on the ComplexWebQuestions dataset (Talmor and Berant, 2018).

2 Related Work

This paper has focused on QA for multi-hop questions using large KBs and text corpora as the information sources from which answers can be drawn. The main technical contribution is an iterative question-guided retrieval mechanism that retrieves information from KBs, corpora, or combinations of both, which makes it possible to follow long paths of reasoning on large KB.

A long line of QA models have been developed which answer questions based on a single passage of text (Dhingra et al., 2016; Yu et al., 2018; Seo et al., 2016; Gao et al., 2018; Liu et al., 2017; Devlin et al., 2018). Generally, these "reading comprehension" systems are operated by encoding the passage and question into an embedding space, and due to memory limitations cannot be applied to a large corpus instead of a short passage. To address this limitation a number of systems have been designed which use a "retrieve and read" pipeline (Chen et al., 2017; Dhingra et al., 2017; Joshi et al., 2017; Dunn et al., 2017; Wang et al., 2018, 2017), in which a retrieval system with high recall is piped into a reading comprehension system that can find the answer. An alternative approach is "phrase-indexed" QA (Seo et al., 2018), where embedded phrases in a document are indexed and searched over. Existing systems are also not able to use both KB and text for QA. They also differ from PullNet in using only a single round of retrieval; however, for questions that require multi-hop reasoning, it is difficult for a single retrieval step to find the relevant information.

SplitQA (Talmor and Berant, 2018) is a text-based QA system that decomposes complex questions (e.g., with conjunction or composition) into simple subquestions, and performs retrieval sequentially for the subquestions. Although it uses iterative retrieval for multi-hop questions, unlike PullNet, SplitQA does not also use a KB as an information source. Also, SplitQA has been applied only to the Complex WebQuestions dataset, so it is unclear how general this approach is.

There has also been much work on QA from KBs alone, often using methods based on memory networks (Sukhbaatar et al., 2015), semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005) or reinforcement learning Das et al. (2017). Extending such KB-based work to also use text, however, is non-trivial. Another line of QA work from text and KBs is exemplified by AQQU (Bast and Haussmann, 2015) and its successors (Sawant et al., 2019). These systems focus on questions (e.g. SimpleWebQuestions), that can

be interpreted as identifying an entity based on a relationship and related entity, plus additional restrictions described in text, and it is unclear how to extend such approaches to multi-hop questions.

GRAFT-Net (Sun et al., 2018) supports multi-hop reasoning on both KBs and text by introducing a question subgraph built with facts and text, and uses a learned graph representation (Kipf and Welling, 2016; Li et al., 2016; Schlichtkrull et al., 2017; Scarselli et al., 2009) to perform the reasoning required to select the answer. We use the same representation and reasoning scheme as GRAFT-Net, but do not require that the entire graph be retrieved in a single step. In our experimental comparisons, this gives significant performance gains for multi-hop reasoning tasks.

Combinations of KBs and text have also been used for relation extraction and Knowledge Base Completion (KBC) (Lao et al., 2012; Toutanova et al., 2015; Das et al., 2017). The QA task differs from KBC in that in QA, the inference process must be conditioned on a natural-language question, which leads to different constraints on which methods can be used.

3 The PullNet Model

PullNet retrieves from two "knowledge sources", a text corpus and a KB. Given a question, PullNet will use these to construct a *question subgraph* that can be used to answer the question. The question subgraph is constructed iteratively. Initially the subgraph depends only on the question. PullNet then iteratively expands the subgraph by choosing nodes from which to "pull" information about, from the KB or corpus as appropriate. The question subgraph is heterogeneous, and contains both entities, KB triples, and entity-linked text.

In this section, we will first introduce notation defining the heterogeneous graph structure we use. Then we will introduce the general iterative retrieval process. Finally, we will discuss the retrieval operations used on the corpus and KB, and the classification operations on the graph which determine where to perform the retrievals.

3.1 The Question Subgraph

A *question subgraph* for question q, denoted $\mathcal{G}_q = \{\mathcal{V}, \mathcal{E}\}$, is a hetogeneous graph that contains information from both the text corpus and the KB relevant to q. Let $\mathcal{V}$ denote the set of vertices, which we also call nodes. Following GRAFT-Net (Sun et al., 2018), there are three types of nodes: entity nodes $\mathcal{V}_e$, text nodes $\mathcal{V}_d$, and fact nodes $\mathcal{V}_f$, with $\mathcal{V} = \mathcal{V}_e \cup \mathcal{V}_d \cup \mathcal{V}_f$. An entity node $v_e \in \mathcal{V}_e$ represents an entity from the knowledge base. A text node $v_d \in \mathcal{V}_d$ represents a document from the corpus, with a sequence of tokens denoted $(w_1, \ldots, w_{|d|})$. In this paper, a document is always a single sentence, to which an entity linker (Ji et al., 2014) has been applied to detect and ground entity mentions. A fact node $v_f \in \mathcal{V}_f$ represents a triplet (v_s, r, v_o) from the KB, with subject and objects $v_s, v_o \in \mathcal{V}_e$ and relation r. Let $\mathcal{E}$ denote the set of edges between nodes. An edge connects a fact node v_f and an entity node v_e iff fact v_f has v_e as its subject or object. An edge connects a text node v_d with entity node v_e iff the entity is mentioned in the text.

3.2 Iterative subgraph construction

3.2.1 Overview

We start with a question subgraph $\mathcal{G}_q^0 = \{\mathcal{V}^0, \mathcal{E}^0\}$ where $\mathcal{V}^0 = \{e_{q_i}\}$ is the list of entities in the question and $\mathcal{E}^0$ is an empty set. We iteratively expand the question subgraph $\mathcal{G}_q^0$ until it contains the information required to answer the question.

Algorithm 1 PullNet

1: Initialize question graph G_q^0 with question q and question entities, with $\mathcal{V}^0 = \{e_{q_i}\}$ and $\mathcal{E}^0 = \emptyset$.
2: **for** $t = 1, \cdots, T$ **do**
3: Classify the entity nodes in the graph and select those with probability larger than ϵ
$\{v_{e_i}\} = \texttt{classify\_pullnodes}\,(G_q^t, k)$
4: **for all** v_e in $\{v_{e_i}\}$ **do**
5: Perform pull operation on selected entity nodes
$\{v_{d_i}\} = \texttt{pull\_docs}\,(v_e, q)$
$\{v_{f_i}\} = \texttt{pull\_facts}\,(v_e, q)$
6: **for all** v_d in $\{v_{d_i}\}$ **do**
7: Extract entities from new document nodes
$\{v_{e(d)_i}\} = \texttt{pull\_entities}\,(v_d)$
8: **for all** v_f in $\{v_{f_i}\}$ **do**
9: Extract head and tail of new fact nodes
$\{v_{e(f)_i}\} = \texttt{pull\_headtail}\,(v_f)$
10: Add new nodes and edges to question graph
$G_q^{t+1} = \texttt{update}\,(G_q^t)$
11: Select entity node in final graph that is the best answer
$v_{\text{ans}} = \texttt{classify\_answer}\,(G_q^T)$

The algorithm is shown in Alg 1. Briefly, we expand the graph in T iterations. In each iteration, we choose entities whose probability is larger than ϵ to expand, and then for each selected entity,

we retrieve a set of related documents, and also a set of related facts. The new documents are then passed through an entity-linking system to identify entities that occur in them, and the head and tail entities of each fact are also extracted. The last stage in each iteration is to update the question graph by adding all these new edges. After the T-th iteration of expansion, an additional classification step is applied to the final question subgraph to predict the answer entity.

3.2.2 Pull Operations

Pull operations either retrieve information from a knowledge source, or extract entities from a fact or document.

The two extraction operations are relatively simple. The `pull_entities(`v_d`)` operation inputs a document node v_d, calls an entity linker and returns all entities mentioned in v_d. The `pull_headtail(`v_f`)` operation inputs a fact node v_f and returns the subject and object entity of fact v_f.

The retrieval operations are more complex. The `pull_docs(`v_e, q`))` operation retrieves relevant documents from the corpus. We use an IDF-based retrieval system, Lucene (McCandless et al., 2010) and assume that all sentences have been entity-linked prior to being indexed. The retrieved documents are constrained to link to entity v_e, and are ranked by their IDF similarity to the question q. Only the top N_d ranked documents are returned.

The `pull_facts(`v_e, q`)` operation retrieves the top N_f facts from the KB about entity v_e. The retrieved facts are constrained to have v_e as their subject or object, and are ranked based on the similarity $S(r, q)$ between the fact's relation r and the question q. Since it is not obvious how to assess relevance of a fact to a question q, we learn $S(r, q)$ as follows. Let h_r be an embedding of relation r, which is looked up from an embedding table, and let $q = (w_1, \ldots, w_{|q|})$ be the sequence of words for question q. Similarity is defined as the dot-product of the last-state LSTM representation for q with the embedding for r. This dot-product is then passed through a sigmoid function to bring it into a range of $[0, 1]$: as we explain below, we will train this similarity function as a classifier which predicts which retrieved facts are relevant to the question q. The final ranking method for facts is

$$h_q = \text{LSTM}(w_1, \ldots, w_{|q|}) \in \mathbb{R}^n$$
$$S(r, q) = \text{sigmoid}(h_r^T \, h_q)$$

3.2.3 Classify Operations

Two types of `classify` operations are applied to the nodes in a subgraph G_q^t. These operations are applied only to the entity nodes in the graph, but they are based on node representations computed by the graph CNN, so the non-entity nodes and edges also affect the classification results.

During subgraph construction, the `classify_pullnodes(`G_q^t`)` operation returns the probability entity node v_e should be expanded in the next iteration. We choose the k nodes with the highest probability in each iteration. After the subgraph is complete, the `classify_answer(`G_q^t`)` operation predicts whether an entity node answers the question. The highest-scoring entity node is returned as the final answer.

We use the same CNN architecture used by GRAFT-Net (Sun et al., 2018) for classification. GRAFT-Net supports node classification on heterogeneous graphs containing facts, entities, and documents. GRAFT-Net differs from other graph CNN implementations in using special mechanisms to distribute representations across different types of nodes and edges: notably, document nodes are represented with an LSTM encoding, extended by mechanisms that allow the representations for entity nodes v_e to be passed into a document v_d that mentions v_e, and mechanisms that allow the LSTM hidden states associated with an entity mention to be passed out of v_d to the associated entity node v_e.

3.2.4 The Update Operation

The `update` operation takes the question subgraph G_q^{t-1} from the previous iteration and updates it by adding the newly retrieved entity nodes $\{v_{e(f)_i}\} \cup \{v_{e(d)_i}\}$, the text nodes $\{v_{d_i}\}$, and the fact nodes $\{v_{f_i}\}$. It also updates the set of edges $\mathcal{E}$ based on the definitions of Section 3.1. Note that some new edges are derived when pull operations are performed on text and fact nodes, but other new edges may connect newly-added nodes with nodes that already exist in the previous subgraph.

3.3 Training

To train PullNet, we assume that we only observe question and answer pairs, i.e., the actual inference chain required to answer the question is latent. We thus need to use weak supervision to train the classifiers described above.

To train these models, we form an approximation of the ideal question subgraph for question q as follows. Note that in training, the answer entities are available. We use these to find all shortest paths in the KB between the question entities and answer entities. Each entity e that appears in such a shortest path will be marked as a *candidate intermediate entities*. For each candidate intermediate entity e we record its minimal distance t_e from the question nodes.

When we train the `classify_pullnodes` classifier in iteration t, we treat as positive examples only those entities e' that are connected to a candidate intermediate entity e with distance $e_t = t+1$. Likewise in training the similarity function $S(h_r, q)$ we treat as positive relations leading to candidate intermediate entities e at distance $e_t = t + 1$. This encourages the retrieval to focus on nodes that lie on shortest paths to an answer.

In training we use a variant of teacher forcing. We pull from all entity nodes with a predicted score larger than some threshold ϵ, rather than only the top k nodes. If, during training, a candidate intermediate entity is not retrieved in iteration t_e, we add it to the graph anyway. The values T and ϵ are hyperparameters, but here we always pick for T the maximum length of the inference chain needed to ensure full coverage of the answers.

We use the same classifier in the retrieval step as in answer selection, except that we change the last fully-connected layer. The classifiers used for retrieval in the different iterations are identical.

The learned parts of the model are implemented in PyTorch, using an ADAM optimizer (Kingma and Ba, 2014), and the full retrieval process of Alg 1 is performed on each minibatch.

4 Experiments and Results

4.1 Datasets

MetaQA (Zhang et al., 2018) contains more than 400k single and multi-hop (up to 3-hop) questions in the domain of movies. The questions were constructed using the knowledge base provided with the WikiMovies (Miller et al., 2016) dataset. We use the "vanilla" version of the queries[2] We use the KB and text corpus supplied with the WikiMovies dataset, and use exact match on surface forms to perform entity linking. The KB used here is relatively small, with about 43k entities and 135k triples.

WebQuestionsSP (Yih et al., 2015) contains 4737 natural language questions that are answerable using Freebase.[3] The questions require up to 2-hop reasoning from knowledge base, and 1-hop reasoning using the corpus. We use Freebase as our knowledge base but for ease of experimentation restrict it to a subset of Freebase which contains all facts that are within 2-hops of any entity mentioned in the questions of WebQuestionsSP. We also exclude a few "very common" entities, e.g. the ones that describe the hierarchical structure of KB. This smaller KB contains 43 million facts and 12 million entities. We use Wikipedia as our corpus and use a simple entity-linker: we link entities by exact matching to any surface form annotated the FACC1 dataset (Gabrilovich et al., 2013).[4]

Complex WebQuestions 1.1 (Complex WebQ) (Talmor and Berant, 2018) is generated from WebQuestionsSP by extending the question entities or adding constraints to answers, in order to construct multi-hop questions.[5] There are four types of question: composition (45%), conjunction (45%), comparative (5%), and superlative (5%). The questions require up to 4-hops of reasoning on the KB and thr 2-hops on corpus. We use the same KB and corpus as for WebQuestionsSP.

	Train	Dev	Test
MetaQA 1-hop	96,106	9,992	9,947
MetaQA 2-hop	118,980	14,872	14,872
MetaQA 3-hop	114,196	14,274	14,274
WebQuestionsSP	2,848	250	1,639
Complex WebQ	27,623	3,518	3,531

Table 1: Statistics of all datasets.

4.2 Tasks

We explored several different QA settings: complete KB only, corpus only, incomplete KB only, and incomplete KB paired with the corpus.

In the *complete KB only* setting, the answer always exists in knowledge base: for all of these datasets, this is true because the questions were crowd-sourced to enforce this conditions. This is the easiest setting for QA, but arguably unrealistic, since with a more natural distribution of ques-

[2]For this version, the 1-hop questions in MetaQA are exactly the same as WikiMovies.

[3]We use the same train/dev/test splits as GRAFT-Net (Sun et al., 2018).

[4]If two overlapping spans are possible matches, we match only the longer of them.

[5]We use Complex WebQuestions v1.1, where the author re-partition the train/dev/test data to prevent the leakage of information.

tions, a KB is likely to be incomplete. In the *text only* setting we use only the corpus. In the *incomplete KB* setting, we simulate KB-based QA on an incomplete KB by randomly discarding some of the triples in the KB: specifically, we randomly drop a fact from the knowledge base with probability $p = 50\%$. This setting is presented mainly as a baseline for the *incomplete KB plus text setting*, where we pair the same incomplete knowledge base with the corpus. In principle this allows a learned QA system to adopt many different hybrid strategies: e.g., for 1-hop queries, a model "back off" to text when the KB is missing information, while in more complex queries, reasoning can involve combining inferences done with text and inferences done with KB triples.

Comment. One point to note is that our training procedure is based on finding shortest paths in a complete KB, and we use this same procedure on the incomplete KB setting. Thus the weak training that we use should, in the incomplete-KB settings, be viewed as a form of weak supervision, with labels that are intermediate in informativeness between pure distant training (with only question-answer pairs) and gold inference paths (a setting that has been extensively investigated on some of these datasets, in particular WebQuestionsSP).

4.3 Baselines

We choose Key-Value Memory Networks (Miller et al., 2016) and GRAFT-Net (Sun et al., 2018) as our baseline models: to the best of our knowledge, these are the only ones that can use both text and KBs for question answering. However, both models are limited by the number of facts and text that can fit into memory. Thus, we create a separate retrieval process as a pre-processing step, which will be discussed below.

Key-Value Memory Networks (KVMem) (Miller et al., 2016) maintain a memory table which stores KB facts and text encoded into key-value pairs. Our encoding of KB facts is the same as is presented in Miller et al. (2016). For text, we use an bi-directional LSTM to encode the text for the key and take the entities mentioned in the text as values. Our implementation shows comparable performance on the WikiMovies (MetaQA 1-hop) dataset, as shown in Table 3.

For **GRAFT-Net** (Sun et al., 2018), we use the implementation published by the author;[6] however, we retrieve data with a simpler process, as described in Section 4.4.

[6] https://github.com/OceanskySun/GraftNet

	WikiM-KB	WikiM-KB (50%)
MetaQA-1hop	0.995 / 9.17	0.544 / 4.58
MetaQA-2hop	0.983 / 47.3	0.344 / 28.6
MetaQA-3hop	0.923 / 459.2	0.522 / 316.6
	Freebase	Freebase (50%)
WebQuestionsSP	0.927 / 1876.9	0.485 / 1212.5
ComplexWebQ	0.644 / 1948.7	0.542 / 1849.2

Table 2: Retrieval results (recall / # entities in graph) on MetaQA ($m = 500$), and WebQuestionsSP and Complex WebQuestions with Freebase ($m = 2000$).

4.4 Subgraph Retrieval for Baseline Models

Text is retrieved (non-iteratively) using IDF-based similarity to the question. It is not obvious how to perform KB retrieval: we would like to retrieve as many facts as possible to maximize the recall of answers, but it is infeasible to take all facts that are within k-hops of question entities since the number grows exponentially. Based on prior work in the database community on sampling from graphs (Leskovec and Faloutsos, 2006) and local partitioning (Andersen et al., 2006a), we ran an approximate version of personalized PageRank (aka random walk with reset) to find the KB entities closest to the entities in the question—specifically, we used the PageRank-Nibble algorithm (Andersen et al., 2006b) with $\epsilon = 1e^{-6}$ and the picked the m top-scoring entities.[7] We then eliminate all top-m entities that are more than k-hops of the question entities, and finally, retrieve all facts from the KB connecting retrieved entities.

The results for several tasks are shown in Table 2, which gives the answer recall, and the average number of entities retrieved. For the smaller MetaQA KB with $m = 500$, the retrieval method finds high-coverage graphs when the KB is complete. For ComplexWebQuestions, even with $m = 2000$, the recall is 64%—which is expected, since retrieving relevant entities for a multi-hop question from a KB with millions of entities is difficult.

4.5 Main Results

4.5.1 MetaQA

The experimental results for MetaQA are shown in Table 3. For 1-hop questions (which is identical to the WikiMovies dataset), PullNet is comparable

[7] Notice that in this process we do not use answer entities which are of course not available at test time.

	MetaQA (1-hop) / wikimovies				MetaQA (2-hop)				MetaQA (3-hop)			
	KB	Text	50% KB	50% KB + Text	KB	Text	50% KB	50%KB + Text	KB	Text	50% KB	50% KB + Text
KV-Mem*	96.2	75.4	(63.6)	75.7	82.7	7.0	(41.8)	48.4	48.9	19.5	(37.6)	35.2
GRAFT-Net*	97.0	82.5	(64.0)	91.5	94.8	36.2	(52.6)	69.5	77.7	40.2	(59.2)	66.4
PullNet (Ours)	97.0	84.4	(65.1)	92.4	**99.9**	**81.0**	(52.1)	**90.4**	**91.4**	**78.2**	(59.7)	**85.2**
KV-Mem	93.9	76.2	–	–	–	–	–	–	–	–	–	–
GRAFT-Net	96.8	**86.6**	(68.0)	**92.6**	–	–	–	–	–	–	–	–
VRN	**97.5**	–	–	–	89.9	–	–	–	62.5	–	–	–

Table 3: Hits@1 on MetaQA compared to baseline models. Number below the double line are from original papers: KV-Mem (KB) (Miller et al., 2016), KV-Mem (Text) (Watanabe et al., 2017), GRAFT-Net (Sun et al., 2018), and VRN (Zhang et al., 2018). *Reimplemented or different retrieval process.

to the state-of-the-art.[8] We also see that our simplified retrieval pipeline has slightly lower performance than the original (Sun et al., 2018) method. However, for multi-hop questions, PullNet generally shows a large improvement over the baseline models. In the KB-only setting, the absolute performance improvement over the best previously published results is 10 points for 2-hop and almost 30 points for 3-hop questions—and there are even larger improvements relative to the baseline models in the settings using text. In the incomplete KB plus text setting, PullNet also consistently improves over either the text-only or incomplete-KB-only setting.[9] This supports our claim that PullNet is able to effectively combine a KB and text.[10]

4.5.2 WebQuestionsSP

Table 4 presents similar results on the WebQuestionsSP dataset. PullNet improves over GRAFT-Net in the incomplete KB plus text setting, and the addition of text to the incomplete KB also improves performance for PullNet.

4.5.3 Complex WebQuestions

Complex WebQuestions contains multi-hop questions against Freebase: intuitively, one would expect that single-shot retrieval of facts and text would not be able to always find sufficient information to answer such questions. Table 5 shows our results for Complex WebQuestions on the development set. As expected, PullNet shows significant improvement over GRAFT-Net and KV-Mem on all four settings. Once again we see some improvement when pairing the incomplete knowledge base with text, compared to using the incomplete knowledge base only or the text only.

	KB	Text	50% KB	50% KB + Text
KV-Mem*	46.7	23.2	(32.7)	31.6
GRAFT-Net*	66.4	24.9	(48.2)	49.7
PullNet (Ours)	**68.1**	24.8	(50.3)	**51.9**
GRAFT-Net	67.8	**25.3**	(47.7)	49.9
NSM	69.0 (F1)	–	–	–

Table 4: Hits@1 on WebQuestionsSP compared to baseline models. Number below the double line are from the original papers: GRAFT-Net (Sun et al., 2018), and NSM (Liang et al., 2017) (which only reports F1 in their paper). * Reimplemented or different retrieval process.

	KB	Text (Wikipedia)	50% KB	50% KB + Text
KV-Mem*	21.1	7.4	(14.8)	15.2
GRAFT-Net*	32.8	10.6	(26.1)	26.9
PullNet (Ours)	**47.2**	**13.1**	(31.5)	**33.7**
		(Snippets)		
SplitQA	–	34.2	–	–
PullNet (Ours)	45.9	29.7	–	–

Table 5: Hits@1 on Complex WebQuestions compared to baseline models. Above the double line results are on the dev set, below on the test set. *Reimplemented or different retrieval process.

Researchers are only allowed limited submissions on the Complex WebQuestions test set, however, some results for the test set are shown in Table 5. On the test set, our model has 45.9% Hits@1 in the KB only setting and 13.8% Hits@1 (not listed in Table 5) in the text only setting with Wikipedia corpus, which are comparable to the dev set results.

For completeness, we also compare to SplitQA (Talmor and Berant, 2018). SplitQA decomposes a multi-hop question into several simpler sub-questions, and sends each sub-question to the Google search engine. In then applies a reading comprehension model to these web snippets to find answers. The snippet corpus is made available

[8] 6.7% questions in Wikimovies are ambiguous, e.g. questions about movies with remakes without specifying years.

[9] The implement KB results are parenthesized because it presented only to compare against the incomplete KB plus text setting.

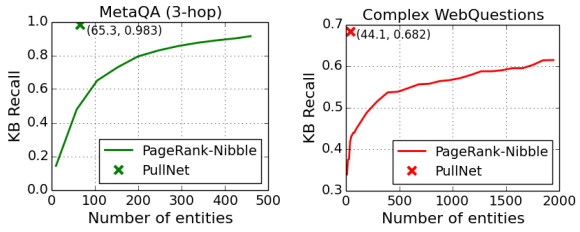
[10] Although we do not repeat the results here, we note that prior work (Sun et al., 2018) showed GRAFT-Net outperforms "late fusion" approaches for combining information: in late fusion, two QA systems are ensembled together, one which uses the incomplete KB, and one which uses the text.

with the Complex WebQuestions dataset, but it is arguably biased toward the SplitQA model, since it was collected specifically to support it, and also relies on non-reproducible, non-open-source components. With this corpus our model has 4.5% lower Hits@1 than SplitQA. However, using the KB, PullNet has much better performance than SplitQA, with an absolute gain of 11.7%.

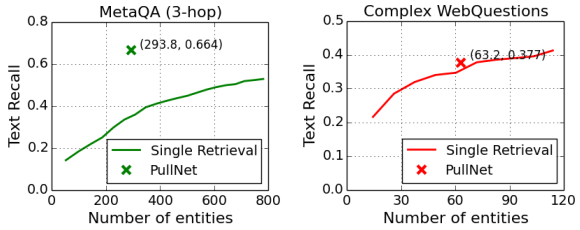
4.6 Additional Results

4.6.1 Retrieval Performance of PullNet

In Figure 1a we compare the retrieval performance of PullNet to that of PageRank-Nibble on multi-hop questions with a complete KB, on two multi-hop problems, varying the number of entities retrieved by PageRank-Nibble. PullNet retrieves far fewer entities but obtains higher recall.



(a) Recall of graphs retrieved by PageRank-Nibble compared with PullNet.



(b) Recall of a single round of retrieval with Apache Lucene compared with PullNet.

Figure 1: Retrieval performance on different datasets.

In Figure 1b we evaluate the effectiveness of iterative retrieval for multi-hop questions on a text corpus. PullNet, with multiple iterations of retrieval, greatly outperfoms a single iteration of IDF-based retrieval on 3-hop MetaQA, and slighly outperforms IDF on Complex WebQuestions.

Figure 2 shows the recall of question subgraphs on 3-hop MetaQA questions as training proceeds. Performance of the retrieval components of PullNet converges relatively quickly, with recall saturating after 10-20,000 examples (about 10-20% of a single epoch).

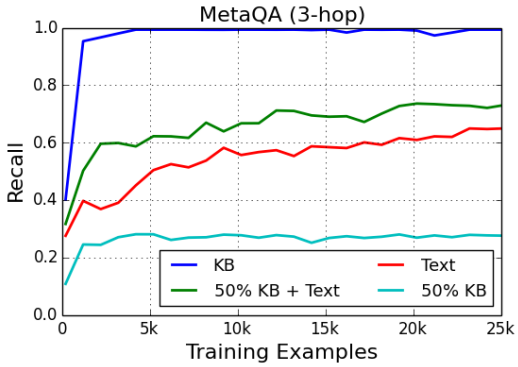


Figure 2: Recall of question subgraph on MetaQA 3-hop questions.

4.6.2 Training Time

PullNet's algorithm is quite different from prior systems, since learning and retrieval are interleaved: in most prior systems, including GRAFT-Net, retrieval is performed only once, before learning. Intuitively, interleaving learning with the relatively slow operation of retrieval is potentially slower; on the other hand, PullNet's final question subgraph is smaller than GRAFT-Net, which makes learning potentially faster.

To study these issues, we plot the Hits@1 performance of learned model versus wall clock time in Figure 3. This experiment is run on Complex WebQuestions in the KB-only setting, using one high-end GPU.[11] GRAFT-Net takes an average of 31.9 minutes per epoch, while PullNet takes an average of 114 minutes per epoch, about 3.5 times slower.

As the graph shows, initially PullNet's performance is better, since GRAFT-Net cannot start learning until the preprocessing finishes. GRAFT-Net's faster learning speed then allows it to dominates for some time. GRAFT-Net reaches its peak in about 3.6 hours. PullNet passes GRAFT-Net after around 6 hours (about 3 epochs for PullNet).

5 Conclusions

PullNet is a novel integrated QA framework for (1) learning what to retrieve from a KB and/or corpus and (2) reasoning with this heterogeneous data to find the best answer. Unlike prior work, PullNet uses an iterative process to construct a question-specific subgraph that contains information relevant to the question. In each iteration, a

[11] To be fair to GRAFT-Net, we used a fast in-memory implementation of PageRank-Nibble (based on SciPy sparse matrices), which takes about 40 minutes.

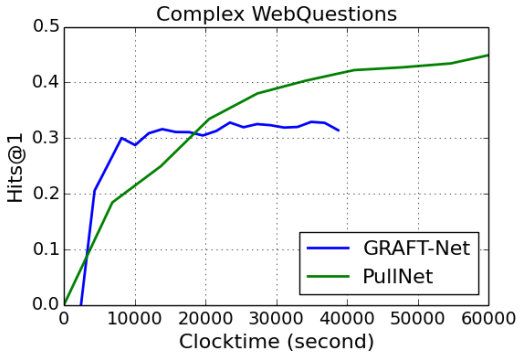


Figure 3: Performance of PullNet and GRAFT-Net under wall clock training time.

graph CNN is used to identify subgraph nodes that should be expanded using "pull" operations on the corpus and/or KB. This iterative process makes it possible to retrieve a small graph that contains just the information relevant to a multi-hop question.

Experimentally PullNet improves over the prior state-of-the-art for the setting in which questions are answered with a corpus plus an incomplete KB, or in settings in which questions need "multi-hop" reasoning. Sometimes the performance improvements are dramatic: e.g., an improvement from 62.5% Hits@1 to 91.4% Hits@1 for 3-hop MetaQa with a KB, or improvements from 32.8% to 47.2% for Complex WebQuestions with a KB.

References

Reid Andersen, Fan Chung, and Kevin Lang. 2006a. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE.

Reid Andersen, Fan Chung, and Kevin Lang. 2006b. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. 2018. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6926–6935.

Heng Ji, Joel Nothman, Ben Hachey, et al. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*, pages 1333–1339.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics.

Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. *ICLR*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *ACL*.

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *EMNLP*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference*, pages 777–782. Association for Computational Linguistics (ACL).

Uma Sawant, Saurabh Garg, Soumen Chakrabarti, and Ganesh Ramakrishnan. 2019. Neural architecture for question answering using a knowledge graph and web corpus. *Information Retrieval Journal*, pages 1–26.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

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*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 559–564.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *EMNLP*.

A. Talmor and J. Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017. Evidence aggregation for answer re-ranking in open-domain question answering. *CoRR*, abs/1711.05116.

Yusuke Watanabe, Bhuwan Dhingra, and Ruslan Salakhutdinov. 2017. Question answering from unstructured text by retrieval and comprehension. *arXiv preprint arXiv:1703.08885*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

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 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

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. *arXiv preprint arXiv:1804.09541*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National conference on Artificial intelligence-Volume 2*, pages 1050–1055. AAAI Press.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666. AUAI Press.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.