Answering Complex Questions by Combining Information from Curated and Extracted Knowledge Bases

Nikita Bhutani[1]* **Xinyi Zheng**[1]* **Kun Qian**[2] **Yunyao Li**[2] **H V Jagadish**[1]
[1]University of Michigan, Ann Arbor
[2]IBM Research, Almaden
nbhutani@umich.edu, zxycarol@umich.edu, qian.kun@ibm.com,
yunyaoli@us.ibm.com jag@umich.edu

Abstract

Knowledge-based question answering (KB-QA) has long focused on simple questions that can be answered from a single knowledge source, a manually curated or an automatically extracted KB. In this work, we look at answering complex questions which often require combining information from multiple sources. We present a novel KB-QA system, MULTIQUE, which can map a complex question to a complex query pattern using a sequence of simple queries each targeted at a specific KB. It finds simple queries using a neural-network based model capable of collective inference over textual relations in extracted KB and ontological relations in curated KB. Experiments show that our proposed system outperforms previous KB-QA systems on benchmark datasets, ComplexWebQuestions and WebQuestionsSP.

1 Introduction

Knowledge-based question answering (KB-QA) computes answers to natural language questions based on a knowledge base. Some systems use a *curated* KB (Bollacker et al., 2008), and others use an *extracted* KB (Fader et al., 2014). The choice of the KB depends on its coverage and knowledge representation: a curated KB uses ontological relations but has limited coverage, while an extracted KB offers broad coverage with textual relations. Commonly, a KB-QA system finds answers by mapping the question to a structured query over the KB. For instance, example question 1 in Fig. 1 can be answered with a query (*Rihanna*, *place_of_birth*, ?) over a curated KB or (*Rihanna*, *'was born in'*, ?) over an extracted KB.

Most existing systems focus on simple questions answerable with a single KB. Limited efforts have been spent to support complex questions that require inference over multiple relations and entities. For instance, to answer question 2 in Fig. 1, we need to infer relations corresponding to expressions *'author of'* and *'attend'*. In practice, a single KB alone may not provide both high coverage and ontological knowledge to answer such questions. A curated KB might provide information about educational institutions, while an extracted KB might contain information about authorship. Leveraging multiple KBs to answer complex questions is an attractive approach but is seldom studied. Existing methods assume a simple abstraction (Fader et al., 2014) over the KBs and have limited ability to aggregate facts across KBs.

Figure 1: Simple vs Complex questions.

We aim to integrate inference over curated and extracted KBs for answering complex questions. Combining information from multiple sources offers two benefits: evidence scattered across multiple KBs can be aggregated, and evidence from different KBs can be used to complement each other. For instance, inference over ontological relation *book_author* can benefit from textual relation *'is written by'*. On the other hand, evidence matching *'attend'* may exclusively be in the curated KB.

Example 1 What college did the author of 'The Hobbit' attend?
Simple Queries:
G_1: *The_Hobbit 'is wrtten by' ?a.*
G_2: *?b person.education ?c . ?c institution ?x.*
Join: $G = G_1$ join$_{?a=?b}$ G_2
Evaluate: ans = University of Oxford

In this work, we propose a novel KB-QA sys-

* NB and XZ contributed equally to this work.

Proceedings of the First Workshop on Natural Language Interfaces, pages 1–10

https://doi.org/10.18653/v1/P17

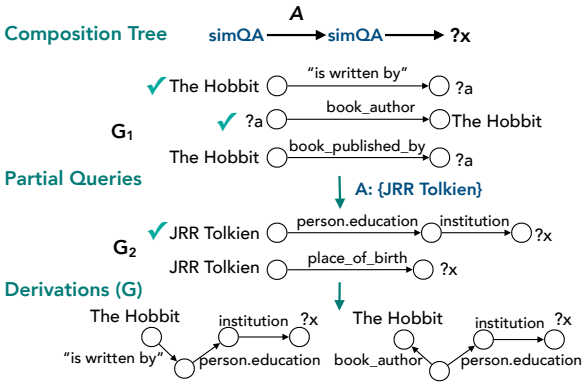

Figure 2: Partial queries and derivations.

tem, MULTIQUE, which constructs query patterns to answer complex questions from simple queries each targeting a specific KB. We build upon recent work on semantic parsing using neural network models (Bao et al., 2016; Yih et al., 2015) to learn the simple queries for complex questions. These methods follow an *enumerate-encode-compare* approach, where candidate queries are first collected and encoded as semantic vectors, which are then compared to the vector representation of the question. The candidate with the highest semantic similarity is then executed over the KB. We propose two key modifications to adapt these models to leverage information from multiple KBs and support complex questions. First, to enable collective inference over ontological and textual relations from the KBs, we align the different relation forms and learn unified semantic representations. Second, due to the lack of availability of fully-annotated queries to train the model, we learn with implicit supervision signals in the form of answers for questions. Our main contributions are:

- We propose a novel KB-QA system, MULTIQUE, that combines information from curated and extracted knowledge bases to answer complex questions. To the best of our knowledge, this is the first attempt to answer complex questions from multiple knowledge sources.
- To leverage information from multiple KBs, we construct query patterns for complex questions using simple queries each targeting a specific KB. (Section 3 and 5).
- We propose a neural-network based model that aligns diverse relation forms from multiple KBs for collective inference. The model learns to score simple queries using implicit supervision from answers to complex questions (Section 4).
- We provide extensive evaluation on benchmarks demonstrating the effectiveness of proposed techniques on questions of varying complexity and KBs of different completeness (Section 6).

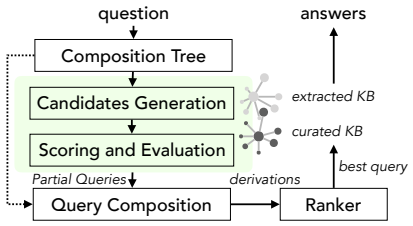

Figure 3: System Architecture

2 Task and Overview

Our goal is to map a complex question Q to a query G, which can be executed against a combination of curated KB $\mathcal{K}_c$ and extracted KB $\mathcal{K}_o$.

Knowledge Bases. The background knowledge source $\mathcal{K}$=$\bigcup\{\mathcal{K}_c, \mathcal{K}_o\}$ is denoted as $\mathcal{K}$=$(\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of entities and $\mathcal{E}$ is a set of triples (s, r, o). A triple denotes a relation $r \in \mathcal{R}$ between subject $s \in \mathcal{V}$ and object $o \in \mathcal{V}$. The relation set $\mathcal{R}$ is a collection of ontological relations $\mathcal{R}^o$ from $\mathcal{K}_c$ and textual relations $\mathcal{R}^t$ from $\mathcal{K}_o$. A higher order relation is expressed using multiple triples connected using a special CVT node.

Complex Question, Q corresponds to a query G which has more than one relation and a single query focus $?x$. G is a sequence of partial queries $G = (G_1, G_2, .., G_o)$ connected via different join conditions. A partial query has four basic elements: a *seed* entity s^r is the root of the query, a *variable* node o^v corresponds to an answer to the query, a *main relation path* (s^r, p, o^v) is the path that links s^r to o^v by one or two edges from either R^o or R^t, and *constraints* take the form of an entity linked to the main relation by a relation c. By definition, each partial query targets a specific KB.

A composition tree C describes how the query G is constructed and evaluated given the partial queries. It includes two functions, *simQA* and *join*. *simQA* is the model for finding simple queries. It enumerates candidates for a simple query, encodes and compares them with the question representation, and evaluates the best candidate. *join* describes how to join two partial queries i.e. whether they share the query focus or another variable node. Fig. 2 shows the partial queries and composition tree for the running example 1.

Overview. Given a complex input question, the task is to first compute a composition tree that describes how to break down the inference into simple partial queries. We then have to gather can-

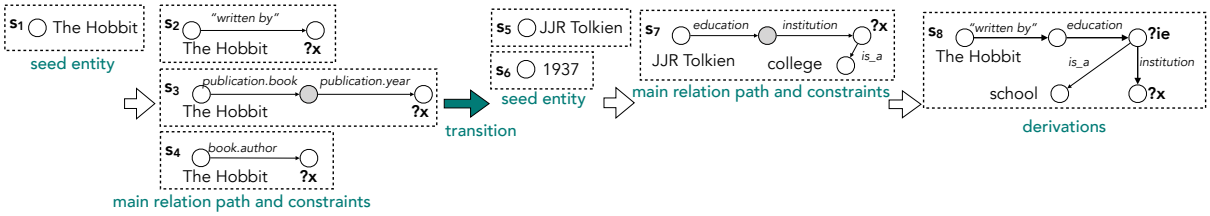

Figure 4: Example Candidate Generation for the running example 1.

didates for each partial query from both curated and extracted KBs. For each candidate, we have to measure its semantic similarity to the question using a neural-network based model that should be capable of inference over different forms of relations. We then have to join the different partial queries to find the complex query for the question. Since there can be multiple ways to answer a complex question, we derive several full query derivations. We rank them based on the semantic similarity scores of their partial queries, query structure and entity linking scores. We execute the *best* derivation over the multiple KBs. Fig. 3 shows the architecture of our proposed system, MULTIQUE.

3 Partial Query Candidate Generation

We first describe how we find candidates for partial queries given an input question. We use a staged generation method with *staged* states and actions. Compared to previous methods (Yih et al., 2015; Luo et al., 2018) which assume a question has one main relation, our strategy can handle complex questions which have multiple main relations (and hence partial queries). We include a new action A_t that denotes the end of the search for a partial query and transition to a state S_t. State S_t refers back to the composition tree to determine the join condition between the current partial query and the next query. If they share an answer node, candidate generation for the subsequent query can resume independently. Otherwise, it waits for the answers to the current query.

We generate (entity, mention) pairs for a question using entity linking (Bast and Haussmann, 2015) and then find elements for query candidates. Fig. 4 depicts our staged generation process.

Identify seed entity. The seed s^r for a partial query is a linked entity in the question or an answer of a previously evaluated partial query.

Identify main relation path. Given a seed entity, we consider all 1-hop and 2-hop paths p. These include both ontological and textual relations. The other end of the path is the variable node o^v.

Identify constraints. We next find entity and type constraints. We consider entities that can be connected using constraint relations *is_a* relations[1] to the variable node o^v. We also consider entities connected to the variables on the relation path via a single relation. We consider all subsets of constraints to enable queries with multiple constraints.

Transition to next partial query. Once candidates of a partial query G_i are collected, we refer to the composition tree to determine the start state of the next partial query G_{i+1}. If the next operation is *simQA*, we compute the semantic similarity of the candidates of G_i using our semantic matching model and evaluate K-best candidates. The answers form the seed for collecting candidates for G_{i+1}. Otherwise, candidate generation resumes with non-overlapping entity links in G_i.

4 Semantic Matching

We now describe our neural-network based model which infers over different relation forms and computes the semantic similarity of a partial query candidate to the question.

4.1 Model Architecture

Fig. 5 shows the architecture of our model. To encode the question, we replace all seed (constraint) entity mentions used in the query by dummy tokens w_E (w_C). To encode the partial query, we consider its query elements, namely the main relation path and constraint relations. Given the vector representations q for the question Q and g for the partial query G_i, we concatenate them and feed a multi-layer perceptron (MLP). The MLP outputs a scalar which we use as the semantic similarity $S_{sem}(Q, G_i)$. We describe in detail the encoding methods for the question and different relation forms in the main relation path. We also describe other design elements and the learning objective.

Encoding question. We encode a question Q using its token sequence and dependency structure.

[1] common.topic.notable_types,common.topic.notable_for

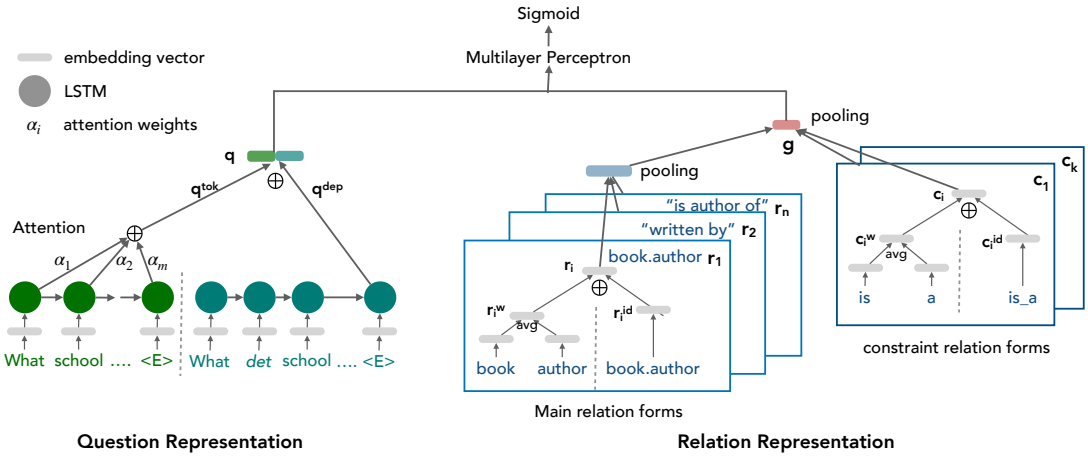

Figure 5: Semantic Matching Model

Since a complex question tends to be long, encoding its dependency tree captures any long-range dependencies. Let $\langle w_1, w_2, \ldots, w_n \rangle$ be the tokens in Q, where seed (constraint) entity mentions have been replaced with w_E (w_C). We map the tokens to vectors $\langle q_1^w, q_2^w, \ldots, q_n^w \rangle$ using an embedding matrix E_w and use an LSTM to encode the sequence to a latent vector q^w. Similarly, we encode the dependency tree into a latent vector q^{dep}.

Encoding main relation path. The main relation path can have different forms, a textual relation from $\mathcal{K}_o$ or an ontological relation from $\mathcal{K}_c$. In order to collectively infer over them in the same space, we first align the textual relations to ontological relations. For instance, we find textual relations *'is author of'*, *'written by'* can be aligned to ontological relation *book.author*. We describe how we derive the relation alignments in Sec. 4.2. Given a relation alignment, we encode each relation form i in the alignment to a latent vector r_i. We apply a max pooling over the latent vectors of different relations in the alignment to obtain a unified semantic representation over the different relation forms. Doing so enables the model to learn better representations of an ontological relation which has complementary textual relations.

To encode each relation form into vector r_i, we consider both sequence of tokens and ids (Luo et al., 2018). For instance, the id sequence of the relation in Fig. 5 is $\{book\_author\}$, while its token sequence is $\{$*'book'*, *'author'*$\}$. We embed the tokens into vectors using an embedding matrix and use average embedding r^w as the token-level representation. We translate the relation directly using another embedding matrix E_r of relation paths to derive its id-level representation r_i^{id}. The vector representation of a path then is $r_i = [r_i^w; r_i^{id}]$.

Encoding constraints. Similarly, we encode the constraint relations c_i in by combining its token-level representation c_i^w and id-level representation c_i^{id}. Given the unified vector representation of a relation path, and the latent vectors of the constraint relations, we apply max pooling to obtain the compositional semantic representation g of the query.

Attention mechanism. Simple questions contain expressions for matching one main relation path. A complex question, however, has expressions for matching multiple relation paths, which could interfere with each other. For instance, words *'college'* and *'attend'* can distract the matching of the phrase *'author of'* to the relation *book.author*. We mitigate this issue by improving the question representation using an attention mechanism (Luong et al., 2015). The idea is to learn to emphasize parts of the question that are relevant to a context derived using the partial query vector g. Formally, given all hidden vectors h_t at time step $t \in \{1, 2, \ldots, n\}$ of the token-level representation of the question, we derive a context vector c as the weighted sum of all the hidden states:

$$c = \sum_{t=1}^{n} \alpha_t h_t$$

where α_t corresponds to an attention weight. The attention weights are computed as:

$$\alpha = softmax(W tanh(W_q q^w + W_g g))$$

where W, W_g, W_q are network parameters. The attention weights indicate how much the model focuses on each token given a partial query.

Objective function. We concatenate the context vector c, question dependency vector q^{dep} and query vector g and feed to a multi-layer perceptron (MLP). It is a feed-forward neural network with

two hidden layers and a scalar output neuron indicating the semantic similarity score $S_{sem}(q, G_i)$. We train the model using cross entropy loss,

$$loss = ylog(S_{sem}) + (1-y)log(1-S_{sem})$$

where $y \in \{0, 1\}$ is a label indicating whether G_i is correct or not. Training the model requires a) an alignment of equivalent relation forms, and b) examples (question, partial query) pairs. We describe how we generate them given QA pairs.

4.2 Relation Alignment

An open KB has a huge vocabulary of relations. Aligning the textual relations to ontological relations for collective inference can become challenging if the textual relations are not canonicalized. We, first learn embeddings for the textual relations and cluster them to obtain canonicalized relation clusters (Vashishth et al., 2018). For instance, a cluster can include both *'is author of'* and *'authored'*. We use the canonicalized textual relations to derive an alignment to the ontological relations. We derive this alignment based on the support entity pairs (s, o) for a pair of ontological relation and canonicalized textual relation. For instance, relations *'is author of'* and *book.author* in our example question will share more entities than relations *'is author of'* and *education.institution*. The alignment is based on a support threshold i.e. minimum number of support entity pairs for a pair of relations. In our experiments, we set the threshold to 5 to avoid sparse and noisy signals in the alignment.

4.3 Implicit Supervision

Obtaining questions with fully-annotated queries is expensive, especially when queries are complex. In contrast, obtaining answers is easier. In such a setting, the quality of a query candidate is often measured indirectly by computing the F_1 score of its answers to the labeled answers (Peng et al., 2017a). However, for complex questions, answers to the partial queries may have little or no overlap with the labeled answers. We, therefore, adopt an alternative scoring strategy where we estimate the quality of a partial query as the best F_1 score of all its full query derivations. Formally, we compute a score $V(G_i^{(k)})$ for a partial query as:

$$V(G_i^{(k)}) = \max_{i \le t \le n-1} F_1(D_{t+1}^{(k)})$$

where D_t denotes the derivation at level t and n denotes the number of partial queries.

Such implicit supervision can be susceptible to spurious derivations which happen to evaluate to the correct answers but do not capture the semantic meaning of a question. We, thus, consider additional priors to promote true positive and false negative examples in the training data. We use $L(Q, G_i^{(k)})$ as the ratio of number of words in the relations of $G_i^{(k)}$ that are mentioned in the question Q. We also use $C(Q, G_i^{(k)})$ as the fraction of relation words that hit a small hand-crafted lexicon of co-occurring relation and question words. We estimate the quality of a candidate as: $V(G_i^{(k)}) + \gamma\, L(Q, G_i^{(k)}) + \delta\, C(Q, G_i^{(k)})$. We consider a candidate a positive example if its score is larger than a threshold (0.5) and negative otherwise.

5 Query Composition

In this work, we focus on constructing complex queries using a sequence of simple partial queries, each with one main relation path. Since the original question does not have to be chunked into simple questions, constructing composition trees for such questions is fairly simple. Heuristically, a composition tree can simply be derived by estimating the number of main relations (verb phrases) in the question and the dependency between them (subordinating or coordinating). We use a more sophisticated model (Talmor and Berant, 2018) to derive the composition tree. The post-order traversal of the tree yields the order in which partial queries should be executed.

Given a computation tree, we adopt a beam search and evaluate best k candidates for a partial query at each level. This helps maintain tractability in the large space of possible complex query derivations. The semantic matching model only independently scores the partial queries and not complete derivations. We, thus, need to find the best derivation that captures the meaning of the complex input question. To determine the best derivation, we aggregate the scores over the partial queries and consider additional features such as entities and structure of the query. We train a log-linear model on a set of (question-answer) pairs using features such as semantic similarity scores, entity linking scores, number of constraints in the query, number of variables, number of relations and number of answer entities. Given the best scoring derivation, we translate it to a KB query and evaluate it to return answers to the ques-

tion. Such an approach has been shown to be successful in answering complex questions over a single knowledge base (Bhutani et al., 2019). In this work, we extend that approach to scenarios when multiple KBs are available.

6 Experiments

We present experiments that show MULTIQUE outperforms existing KB-QA systems on complex questions. Our approach to construct queries from simple queries and aggregate multiple KBs is superior to methods which map questions directly to queries and use raw text instead.

6.1 Experimental Setup

Datasets. We use two benchmark QA datasets:

- **CompQWeb** (Talmor and Berant, 2018): A recent dataset with highly complex questions with compositions, conjunctions, superlatives and comparatives. It contains 34,689 questions, split into 27,734 train, 3,480 dev and 3,475 test cases. For simplicity of evaluation, we only reserve questions with compositions and conjunctions (90% of the dataset).
- **WebQSP** (Yih et al., 2016): It contains 4,737 questions split into 3,098 train and 1,639 test cases. Most of the questions are simple; only 4% questions have multiple constraints (Yin et al., 2015). We evaluate on this dataset to demonstrate our proposed methods are effective on questions of varying complexity.

Knowledge Bases. We use the Freebase[2] dump as the curated KB. We construct an extracted KB using StanfordOpenIE (Angeli et al., 2015) over the snippets released by (Talmor and Berant, 2018) for CompQWeb and (Sun et al., 2018) for WebQSP.

Evaluation Metric. We report averaged F_1 scores of the predicted answers. We additionally compute precision@1 as the fraction of questions that were answered with the exact gold answer.

Baseline systems. We compare against two systems that can handle multiple knowledge sources.

- **GraftNet+** (Sun et al., 2018): Given a question, it identifies a KB subgraph potentially containing the answer, annotates it with text and performs a binary classification over the nodes in the subgraph to identify the answer node(s). We point that it collects subgraphs using 2-hop paths from a seed entity. Since this cannot scale for complex questions which can have arbitrary length paths, we follow our query composition strategy to generate subgraphs. We annotate the subgraphs with snippets released with the datasets. We call this approach GraftNet+.
- **OQA** (Fader et al., 2014): It is the first KB-QA system to combine curated KB and extracted KB. It uses a cascade of operators to paraphrase and parse questions to queries, and to rewrite and execute queries. It does not generate a unified representation of relation forms across the KBs. For comparison, we augment its knowledge source with our extracted KB and evaluate the model released by the authors.

Several other KB-QA systems (Cui et al., 2017; Abujabal et al., 2017; Bao et al., 2016) use only Freebase and handle simple questions with a few constraints. SplitQA (Talmor and Berant, 2018) and MHQA (Song et al., 2018) handle complex questions, but use web as the knowledge source.

Implementation Details. We used NVIDIA GeForce GTX 1080 Ti GPU for our experiments. We initialize word embeddings using GloVe (Pennington et al., 2014) word vectors of dimension 300. We use BiLSTMs to encode the question token and dependency sequences. We use 1024 as the size of hidden layer of MLP and *sigmoid* as the activation function.

6.2 Results and Discussion

We evaluate several configurations. We consider candidates from curated KB as the only available knowledge source to answer questions and use it as a baseline (*cKB-only*). To demonstrate that inference over curated KB can benefit from open KB, we consider diverse relation forms of curated KB facts from open KB (*cKB+oKB*). Lastly, we downsample the curated KB candidates to 90%, 75% and 50% to simulate incompleteness in KB.

Effectiveness on complex questions. Our proposed system outperforms existing approaches on answering complex questions (Table 1). Even though both MULTIQUE and GraftNet+ use the same information sources, our semantic matching model outperforms node classification. Also, using extracted facts instead of raw text enables us to exploit the relations between entities in the text. We also achieve significantly higher F_1 than OQA that uses multiple KB but relies on templates for parsing questions to queries directly and does not deeply integrate information from multiple KBs.

[2] http://commondatastorage.googleapis.com/freebase-public/rdf/freebase-rdf-2015-08-02-00-00.gz

Method	CompQWeb	WebQSP
MULTIQUE (cKB-only)	31.24/37.61	**61.16/69.84**
MULTIQUE (cKB+oKB)	**34.62/41.23**	57.49/67.51
MULTIQUE (90%cKB+oKB)	27.15/30.21	55.47/65.42
MULTIQUE (75%cKB+oKB)	25.54/28.09	50.64/60.17
MULTIQUE (50%cKB+oKB)	18.57/20.51	41.72/50.82
GraftNet+ (Sun et al., 2018)	31.96/44.78	57.21/68.98
OQA (Fader et al., 2014)	0.42/42.85	21.78/32.63
SplitQA(Talmor and Berant, 2018)	-/27.50	-
MHQA (Song et al., 2018)	-/30.10	-

Table 1: Average F_1 / precision@1 of baseline systems and MULTIQUE in different configurations.

In contrast, we can construct complex query patterns from simple queries, and can infer over diverse relation forms in the KB facts. SplitQA (Talmor and Berant, 2018) and MHQA (Song et al., 2018) use a similar approach to answer complex questions using a sequence of simpler questions, but rely solely on noisy web data. Clearly, by combining the knowledge from curated KB, we can answer complex questions more reliably.

Effectiveness on simple questions. An evaluation on simpler questions demonstrates that MULTIQUE can adapt to questions of varying complexity. We achieve the comparable F_1 score on the as other KB-QA systems that adopt an enumerate-encode-compare strategy. STAGG (Yih et al., 2015), a popular KB-QA system uses a similar approach for candidate generation but improves the results using feature engineering and by augmenting entity linking with external knowledge and only uses curated KB. MULTIQUE uses multiple KBs, and can be integrated with a better entity linking and scoring scheme for derivations.

KB completeness. Our results show that including information from extracted KB helps improve inference over ontological relations and facts for complex questions (as indicated by 3.38 F_1 gain in cKB+oKB). It instead hurts the performance on WebQSP dataset. This can be attributed to the coverage of the accompanying textual data sources of the two datasets. We found that for only 26% of the questions in WebQSP, an extracted fact could be aligned with a curated KB candidate. In contrast, there were 55% such questions in the CompQWeb. This illustrates that considering irrelevant, noisy facts does not benefit when curated KB is complete. Such issues can be mitigated by using a more robust retrieval mechanism for text snippets or facts from extracted KB.

A KB-QA system must rely on an extracted KB when curated KB is incomplete. This is reflected in the dramatic increase in the percentage of hybrid queries when curated KB candidates were downsampled (e.g., from 17% to 40% at 90% completeness). As expected, the overall F_1 drops because the precise curated KB facts become unavailable. Despite the noise in extracted KBs, we found 5-15% of the hybrid queries found a correct answer. Surprisingly, we find 55% of the queries changed when the KB is downsampled to 90%, but 89% of them did not hurt the average F_1. This indicates that the system could find alternative queries when KB candidates are dropped.

Setup	CompQWeb	WebQSP
No constraints	31.23/37.87	52.53/60.84
No attention	26.92/31.24	40.29/51.86
No re-ranking	29.39/36.14	55.13/62.78
No prior	30.88/36.68	57.54/64.63

Table 2: Ablation results, average F_1 / precision@1, of MULTIQUE (cKB+oKB).

Ablation Study. Queries for complex questions often have additional constraints on the main relation path. 35% of the queries in CompQWeb had at least one constraint, while most of the queries (85%) in WebQSP are simple. Ignoring constraints in candidate generation and in semantic matching drops the overall F_1 score by 9.8% (8.6%) on CompQWeb (WebQSP) (see Table 2). Complex questions also are long and contain expressions for matching different relation paths. Including the attention mechanism helps focus on relevant parts of the question and improves the relation inference. We found F_1 drops significantly on CompQWeb when attention is disabled.

Re-ranking complete query derivations by additionally considering entity linking scores and query structure consistently helps find better queries. We examined the quality of top-k query derivations (see Table 3). For a large majority of the questions, query with the highest F_1 score was among the top-10 candidates. A better re-ranking model, thus, could help achieve higher F_1 score. We also observed that incorporating prior domain knowledge in deriving labels for partial queries at training was useful for complex questions.

Qualitative Analysis. The datasets also provide queries over Freebase. We used them to analyze the quality of our training data and the queries generated by our system. We derive labels for each partial query candidate by comparing it to the labeled query. On an average, 4 candidates per ques-

	CompQWeb		WebQSP	
	%	Avg. *best* F_1	%	Avg. *best* F_1
Top-1	35.11	34.62	69.12	57.49
Top-2	39.73	37.02	76.21	63.74
Top-5	51.12	42.08	85.05	70.00
Top-10	59.19	46.39	89.63	73.37

Table 3: Percentage of questions with the highest F_1 score in the top-k derivations, and the average best F_1.

tion were labeled correct. We then compare them with the labels derived using implicit supervision. We found on average 3.06 partial queries were true positives and 103.08 were true negatives, with few false positives (1.72) and false negatives (0.78).

We further examined if the queries which achieve a non-zero F_1 were spurious. We compared the query components (entities, relations, filter clauses, ordering constraints) of such queries with labeled queries. We found high precision (81.89%) and recall (76.19%) of query components, indicating the queries were indeed precise.

Error Analysis. We randomly sampled 50 questions which achieve low F_1 score (< 0.1) and analyzed the queries manually. We found 38% errors were made because of incorrect entities in the query. 92% of the entity linking errors were made at the first partial query. These errors get propagated because we find candidate queries using a staged generation. A better entity linking system can help boost the overall performance. 12% of the queries had an incorrect curated KB relation and 18% of the queries had an incorrect extracted KB relation. In a large fraction of cases (32%) the predicted and true relation paths were ambiguous given the question (e.g., *kingdom.rulers* vs *government* for *"Which queen presides over the location..."*). This indicates that relation inference is difficult for highly similar relation forms.

Future Work. Future KB-QA systems targeting multiple KBs should address two key challenges. They should model whether a simple query is answerable from a given a KB or not. It should query the reliable, extracted KBs only when the curated KB lacks sufficient evidence. This could help improve overall precision. Second, while resolving multiple query components simultaneous is beneficial, the inference could be improved if the question representation reflected all prior inferences.

7 Related Work

KB-QA methods can be broadly classified into: retrieval-based methods, template-based methods and semantic parsing-based methods. Retrieval-based methods use relation extraction (Feng et al., 2016) or distributed representations (Bordes et al., 2014; Xu et al., 2016) to identify answers from the KB but cannot handle questions where multiple entities and relations have to be identified and aggregated. Template-based methods rely on manually-crafted templates which can encode very complex query logic (Unger et al., 2012; Zou et al., 2014), but suffer from the limited coverage of templates. Our approach is inspired by (Abujabal et al., 2017), which decomposes complex questions to simple questions answerable from simple templates. However, we learn solely from question-answer pairs and leverage multiple KBs.

Modern KB-QA systems use neural network models for semantic matching. These use an encode-compare approach (Luo et al., 2018; Yih et al., 2015; Yu et al., 2017), wherein continuous representations of question and query candidates are compared to pick a candidate which is executed to find answers. These methods require question-answer pairs as training data and focus on a single knowledge source. Combining multiple knowledge sources in KB-QA has been studied before, but predominantly for textual data. (Das et al., 2017b) uses memory networks and universal schema to support inference on the union of KB and text. (Sun et al., 2018) enriches KB subgraphs with entity links from text documents and formulates KB-QA as a node classification task. The key limitations for these methods are that a) they cannot handle highly compositional questions and b) they ignore the relational structure between the entities in the text. Our proposed system additionally uses an extracted KB that explicitly models the relations between entities and can compose complex queries from simple queries.

We formulate complex query construction as a search problem. This is broadly related to structured output prediction (Peng et al., 2017b) and path finding (Xiong et al., 2017; Das et al., 2017a) methods which learn to navigate the search space using supervision from question-answer pairs. These methods are effective for answering simple questions because the search space is small and the rewards to guide the search can be estimated reliably. We extend the ideas of learning from implicit supervision (Liang et al., 2016) and integrate it with partial query evaluation and priors to

preserve the supervision signals.

8 Conclusion

We have presented a new KB-QA system that uses both curated and extracted KBs to answer complex questions. It composes complex queries using simpler queries each targeting a KB. It integrates an enumerate-encode-compare approach and a novel neural-network based semantic matching model to find partial queries. Our system outperforms existing state-of-the-art systems on highly compositional questions, while achieving comparable performance on simple questions.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proc. WWW '17*, pages 1191–1200. International World Wide Web Conferences Steering Committee.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proc. ACL '15*, volume 1, pages 344–354.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proc. COLING '16*, pages 2503–2514.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proc. CIKM '15*, pages 1431–1440. ACM.

Nikita Bhutani, Xinyi Zheng, and HV Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 739–748.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. SIGMOD '08*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Proc. ECML '14*, volume 8724, pages 165–180. Springer.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *Proc. VLDB '17*, 10(5):565–576.

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

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017b. Question answering on knowledge bases and text using universal schema and memory networks. In *Proc. ACL '17*, pages 358–365.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proc. SIGKDD ' 14*, pages 1156–1165. ACM.

Yansong Feng, Songfang Huang, Dongyan Zhao, et al. 2016. Hybrid question answering over knowledge base and free text. In *Proc. COLING '16*, pages 2397–2407.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Q. Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proc. EMNLP '18*, pages 2185–2194.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP '15*, pages 1412–1421.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017a. Maximum margin reward networks for learning from explicit and implicit supervision. In *Proc. EMNLP '17*, pages 2368–2378.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017b. Maximum margin reward networks for learning from explicit and implicit supervision. In *Proc. EMNLP '17*, pages 2368–2378.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP '14*, pages 1532–1543.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

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 *Proc. EMNLP '18*, pages 4231–4242.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proc. NAACL '18*, pages 641–651.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *Proc. WWW '12*, pages 639–648. ACM.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proc. WWW '18*, pages 1317–1327. International World Wide Web Conferences Steering Committee.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proc. EMNLP '17*, pages 564–573.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proc. ACL '16*.

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 *Proc. ACL '15*, pages 1321–1331.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proc. ACL '16*, volume 2, pages 201–206.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194*.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proc. SIGMOD '14*, pages 313–324. ACM.