Improving Query Graph Generation for Complex Question Answering over Knowledge Base

Kechen Qin[1] **Cheng Li**[2] **Virgil Pavlu**[1] **Javed A. Aslam**[1]
[1]Khoury College of Computer Sciences, Northeastern University
[2] CodaMetrix
[1]qin.ke@northeastern.edu {vip,jaa}@ccs.neu.edu
[2]cheng@codametrix.com

Abstract

Most of the existing Knowledge-based Question Answering (KBQA) methods first learn to map the given question to a query graph, and then convert the graph to an executable query to find the answer. The query graph is typically expanded progressively from the topic entity based on a sequence prediction model. In this paper, we propose a new solution to query graph generation that works in the opposite manner: we start with the entire knowledge base and gradually shrink it to the desired query graph. This approach improves both the efficiency and the accuracy of query graph generation, especially for complex multi-hop questions. Experimental results show that our method achieves state-of-the-art performance on ComplexWebQuestion (CWQ) dataset.

1 Introduction

Knowledge-based question answering (KBQA) is the task of finding answers to questions by processing a structured knowledge base $\mathcal{KB}$. A $\mathcal{KB}$ graph consists of general facts which are organized as entity-relation-entity triplets, with entities as vertices and relations as edges. To answer a simple question such as: *"Who is the president of the United States?"*, a typical KBQA system first identifies the entity (i.e., "United States") and the relation (i.e., "president") asked in the question, and then searches for the answer entity by matching the triplet fact query <United States, president, ?> over $\mathcal{KB}$ (Bordes et al., 2015; Yin et al., 2016; Yu et al., 2017; Zhang et al., 2018; Zhao et al., 2019). To answer a multi-hop question, multiple facts are extracted to form a structured representation, namely, a query graph (Yih et al., 2015). For example, the question *"What was the name of the publisher for Disney Channel Magazine's first cartoon?"* corresponds to a query graph that consists of 3 facts with grounded (i.e., topic entity) and ungrounded entities (i.e., "?"): <?, publisher, Disney Channel Magazine>, <?, cartoon, ?>, <?, published date, ?>, and a constraint: `order by`. The query graph can be converted to an executable query to find the answer in $\mathcal{KB}$. Generating the query graph accurately and efficiently is the key challenge in KBQA.

While single-hop questions are easy to answer by searching for a single fact in $\mathcal{KB}$, multi-hop questions are much harder to answer because the search space grows exponentially as the number of hops increases. The most common way to solve a multi-hop question is to first generate candidate query graphs and then validate and rank them down to one. Previous works construct candidate query graphs by starting with the topic entity and progressively expanding the graph (Bao et al., 2016; Liang et al., 2017; Zhou et al., 2018; Luo et al., 2018; Chen et al., 2019). They greedily determine what relation best fits the current *incomplete* query graph at each step, but fail to capture global properties of the *complete* query graph. At each step as the query graph grows, these step-wise models need to query the $\mathcal{KB}$, measure semantic relevance, and update the model, which inevitably leads to high computational cost. For example, it takes more than two weeks to train the state-of-the-art model QGG (Lan and Jiang, 2020) on ComplexWebQuestion dataset (Talmor and Berant, 2018). To reduce the computation, these methods limit the maximum length of the search path to a small number, and use beam search to maintain only the top candidates at each step; thus causing some good candidates to be missed.

We instead propose a novel query graph generation method that works in quite the opposite manner: we start with the entire $\mathcal{KB}$ and gradually shrink it to the desired query graph. In the candidate query graph generation stage, in contrast to existing works that use expensive semantic relevance features, we only rely on cheap global features that capture syntactic matches with the query or structure matches with $\mathcal{KB}$. This allows us to quickly

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4201–4207
November 7–11, 2021.

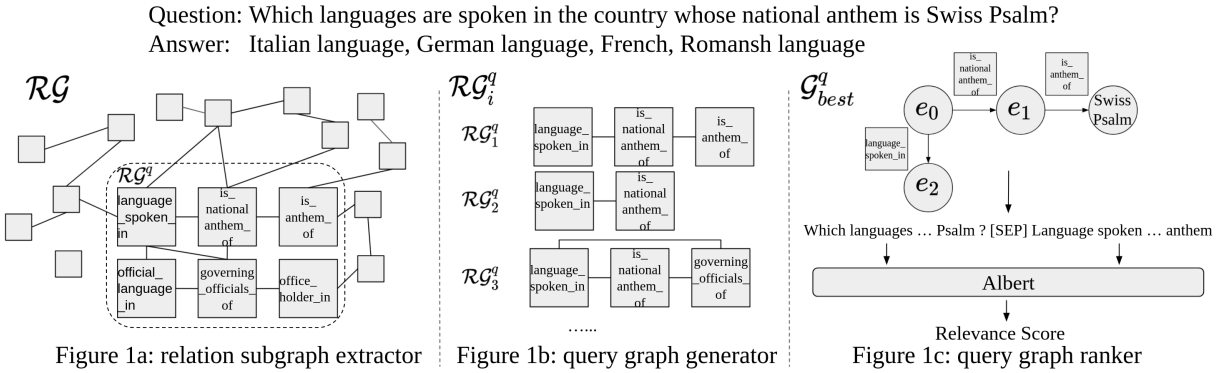

Figure 1a: relation subgraph extractor Figure 1b: query graph generator Figure 1c: query graph ranker

Figure 1: Different stages of the proposed method on an example question with topic entity "Swiss Psalm". Rectangles represent relations and circles represent entities. The graph transformation can be summarized as: $\mathcal{KB} \xrightarrow{\text{1a}} \mathcal{RG} \xrightarrow{\text{1a}} (\mathcal{RG}^q \subset \mathcal{RG}) \xrightarrow{\text{1b}} (\mathcal{RG}_i^q \subset \mathcal{RG}^q) \xrightarrow{\text{1b}} (\mathcal{G}_j^q \subset \mathcal{KB}) \xrightarrow{\text{1c}} (\mathcal{G}_{best}^q \subset \mathcal{KB}) \xrightarrow{\text{1c}} answer$. The index above the arrow denotes in which subfigure (stage) the transformation happens.

filter out a large number of low-quality candidate graphs, and run the computation-intensive ranking stage on a relatively-small number of promising graphs. Compared to previous approaches, ours is computationally efficient and could explore, in early stages, candidates graphs missed by previous methods. Experimental results show that our method delivers consistent performance on two KBQA datasets: it improves the state-of-the-art results by an absolute 5.8% in F1 on the multi-hop KBQA task CWQ and produces competitive result on the single-hop / two-hop KBQA task WQSP. In contrast, while some baseline methods work somewhat better on simple single-hop / two-hop questions, their performance drops dramatically on complex multi-hop questions.

2 Methodology

First we introduce some definitions and notations. For a given knowledge base $\mathcal{KB}$, its associated **relation graph** $\mathcal{RG}$ is the undirected graph whose vertices correspond to the edges in $\mathcal{KB}$, and whose edges correspond to the edge adjacencies in $\mathcal{KB}$ (two nodes in $\mathcal{RG}$ are connected if the corresponding two edges in $\mathcal{KB}$ share a common entity node). A **query graph** $\mathcal{G}^q$ is a subgraph of $\mathcal{KB}$, which reveals the semantic structure and the topic entities of the input question q. One can translate a query graph into an executable SPARQL query and execute it against the $\mathcal{KB}$ to obtain the answer entities. A typical query graph consists of four types of nodes (Yih et al., 2015): the **grounded entity** corresponding to e^{topic}; **existential variables** e^{ung}, which are ungrounded entities; the **lambda variable** e^{ans}, which is an ungrounded entity representing the answer. It is also a common practice to define some **aggregation functions nodes** for $\mathcal{G}^q$, which represent set operations (e.g., `argmax` and `order by`) on e^{ung} or e^{ans}.

To find the answer to a question q, our method shrinks $\mathcal{KB}$ down to $\mathcal{G}^q$ in three stages: **relation subgraph extractor** predicts a relation subgraph $\mathcal{RG}^q \subset \mathcal{RG}$ for an input question q; **query graph generator** generates a set of candidate query graphs $\mathcal{G}_j^q \subset \mathcal{KB}, j = 1, 2, \cdots$; **query graph ranker** ranks $\mathcal{G}_j^q$ and selects the top one as the answer.

Relation Subgraph Extractor: we first extract a relation subgraph $\mathcal{RG}^q \subset \mathcal{RG}$ for an input question q. $\mathcal{RG}^q$ only captures relations relevant to q (see a running example in Figure 1a). The reason of considering a relation graph (with relations as nodes) instead of an entity graph (with entities as nodes) is that we want the graph size to be small, and in typical $\mathcal{KB}$ the number of relations is much smaller than the number of entities. We identify the relations in $\mathcal{RG}^q$ for the given question with a multi-label classifier. Specifically, we consider all annotated relations in the training data as potential labels and train binary relevance (Tsoumakas and Katakis, 2007; Wang et al., 2018) with logistic regression to determine the relevance of each relation. Unigrams from the questions are used as features. Because the given questions are very short, both extracting unigrams and running logistic regressions on sparse features are very fast. Next we extract a subgraph $\mathcal{RG}^q \subset \mathcal{RG}$ whose nodes correspond to the predicted relevant relations. In effect, we narrow down the search space from the entire $\mathcal{KB}$ (38M entities) to a small relation graph of about

10-20 relations.

Query Graph Generator: we aim to get a set of candidate query graphs $\mathcal{G}_j^q \subset \mathcal{KB}, j = 1, 2, \cdots$. We start with the relation subgraph $\mathcal{RG}^q \subset \mathcal{RG}$ from previous stage, and further narrow down the search space by selecting some of its high-quality subgraphs $\mathcal{RG}_i^q \subset \mathcal{RG}^q, i = 1, 2, \cdots$, which represent relevant relation query structures. We propose to find such $\mathcal{RG}_i^q$ that leads to answers with high F1 scores using a logistic regression model. In practice, we consider all subgraphs of $\mathcal{RG}^q$ with up to 5 nodes as candidates (see examples of the top three $\mathcal{RG}_i^q$ in Figure 1b). Four types of features are used to characterize each candidate subgraph $\mathcal{RG}_i^q$: (1) Shape of the graph (categorical): two graphs have the same shape if they are isomorphic; 15 different shapes are identified. (2) Relation relevance score (numeric): for relations in $\mathcal{RG}_i^q$, we use their relevance scores generated in the previous stage. For relations not in $\mathcal{RG}_i^q$, the scores are set to 0. (3) Relation pairs: the presence of each adjacent relation pair is used as a binary feature. (4) (Relation, entity type, relation) triplets (binary): for each adjacent relation pair, we further combine it with the most common entity type of the connected node.

In training, we integrate $\mathcal{RG}_i^q$ with labeled topic entities to build a query graph, and execute it to get the answer and measure answer F1 score. We generate training data for logistic regression by including all positive subgraphs with high F1 scores and randomly sample 30 negative subgraphs. We turn F1 scores into binary training labels based on a threshold 0.9. Once the logistic regression model is trained using the extracted features, we can use it to score and rank $\mathcal{RG}_i^q$. After selecting the top 50 $\mathcal{RG}_i^q$, we couple them with topic entities e^{topic} to build query graph candidates $\mathcal{G}_j^q$. Each $\mathcal{RG}_i^q$ can be mapped to multiple $\mathcal{G}_j^q$ by inserting different topic entities at different positions and assigning different relation directions (see an example of $\mathcal{G}_j^q$ in Figure 1c). After this step, the generated $\mathcal{G}_j^q$ only contains one topic entity. To build a query graph for a question with multiple topic entities, we merge the generated query graphs with similar relation structures but different entities. For example, $e_0^{ung} \xrightarrow{\text{r1}} e_1^{ung} \xrightarrow{\text{r2}} e_1^{topic}$ and $e_2^{topic} \xrightarrow{\text{r1}} e_1^{ung} \xrightarrow{\text{r2}} e_2^{ung}$ will be merged into $e_1^{topic} \xrightarrow{\text{r1}} e_1^{ung} \xrightarrow{\text{r2}} e_2^{topic}$, where $r1$ and $r2$ correspond to two different relations.

Query Graph Ranker: we rank the candidate query graphs $\mathcal{G}_j^q, j = 1, 2, \cdots$, and use the top one $\mathcal{G}_{best}^q$ to produce the final answer to the input question q. Since there are only a small number of generated candidate query graphs, we can afford to evaluate them with a powerful and expensive model. We use the Albert (Lan et al., 2020) to compute the matching score between $\mathcal{G}_j^q$ and q. To represent $\mathcal{G}_j^q$ as a sequence of tokens and concatenate it with q, we locate the source node[1] and concatenate all paths starting from the source node to make a sequence. The example in Figure 1c has source node e_0 and two paths that point to e^{topic} and e_2 separately. We represent the ungrounded entities as their entity types. Then the concatenation is sent to the Albert model to get a matching score.

Keywords	SPAQRL constraints
prior, before,	FILTER (var_1 <"var_2"^^xsd:dateTime)
after	FILTER (var_1 >"var_2"^^xsd:dateTime)
less	FILTER (xsd:integer(var_1) <var_2)
greater, more	FILTER (xsd:integer(var_1) >var_2)
earliest, smallest, first	ORDER BY var_1 LIMIT 1
largest, most, last	ORDER BY DESC var_1 LIMIT 1

Table 1: Mapping rules that translate keywords into SPAQRL constraints.

Following previous work (Luo et al., 2018), we further augment $\mathcal{G}_{best}^q$ with constraints based on a set of predefined rules. This is necessary for detecting time and number constraints in superlative and comparative questions. The rules consist of mappings from keywords to SPARQL constraints as shown in Table 1. Take as an example the input question *"Who were the presidents of the United States before 2020?"* and the predicted query graph "SELECT distinct * from <cwq> WHERE { <US> <president> ?e . }". The model first detects the keyword "before" from the question, and then learns "var_1" and "var_2" to be "?e" and "2020" based on question and the predicted graph. At the end, it couples the generated SPARQL constraints with the predicted query to generate the final query.

Then, we execute the query against the $\mathcal{KB}$ to obtain the answer. Because all nodes in $\mathcal{G}_{best}^q$ except the grounded entities and CVT nodes[2] can potentially be the answer node, in the last step we resolve the answer node using a simple heuristic: we compare $\mathcal{G}_{best}^q$ with all annotated query graphs in the training set to select graphs which are isomorphic to $\mathcal{G}_{best}^q$. From the selected graphs, we choose

[1] Source node is a node without incoming edges.

[2] Compound value type (CVT) is a special entity category in Freebase, which does not represent real-world entity, but helps with connecting other nodes.

	CWQ	WQSP
HR-BiLSTM (Yu et al., 2017)	31.2 †	62.3 †
GRAFT-Net (Sun et al., 2018)	26.0 †	62.8
KBQA-GST (Lan et al., 2019)	36.5	67.9
TEXTRAY (Bhutani et al., 2019)	33.9	60.3
UHop (Chen et al., 2019)	29.8 †	68.5 †
QGG (Lan and Jiang, 2020)	40.4	**74.0**
Our Method	**46.2**	66.0

(a)

Query graph generator	Δ Coverage
w/o shape of the graph	−2.4%
w/o relation relevance score	−1.3%
w/o relation pairs	−0.4%
w/o (relation, entity type, relation) triplets	−4.5%
Query graph ranker	**Δ F1**
w/o query graph merge	−3.5%
w/o path encoding	−1.4%
w/o entity type encoding	−4.5%
w/o constraints modeling	−1.3%
w/ top 5 prediction	+14.5%

(b)

	CWQ			WQSP		
	Coverage	Upper bound F1	Cardinality	Coverage	Upper bound F1	Cardinality
Entity Linking	0.732	-	25.7 (#entities)	0.961	-	2.3 (#entities)
Relation Subgraph Extractor	-	0.859	13.1 (#relations)	-	0.855	16.2 (#relations)
Query Graph Generator	-	0.741	83.6 (#query graphs)	-	0.771	20.6 (#query graphs)
Query Graph Ranker	-	0.462	1(#query graphs)	-	0.660	1(#query graphs)

(c)

Table 2: (a): F1 scores (%) on WQSP and CWQ test sets. † denotes re-implementation. (b): Ablation study results on CWQ development set. We report changes of query graph coverage rate for query graph generator and F1 score for query graph ranker. (c): Result of each stage.

the node (e.g., the top left node) which has most often been the answer node in the dataset. This heuristic achieves over 90% accuracy in practice.

3 Results and Analysis

We conduct experiments on two popular multi-hop KBQA datasets, COMPLEXWEBQUESTION-1.1 (CWQ) (Talmor and Berant, 2018) and WEBQUESTIONSP (WQSP) (Yih et al., 2015). CWQ dataset has 34,689 complex questions (2-5 hops), while WQSP dataset contains 4,737 simple questions (1 or 2 hops). In this work, we use CWQ for the main evaluation because our method is designed for complex questions. Both datasets use Freebase (Google, 2013) as the supporting knowledge base.

We implement our model using NETWORKX (Hagberg et al., 2008), PYTORCH-1.6.0 (Paszke et al., 2019), and Huggingface (Wolf et al., 2019). For entity linking, we take a union of AllenNLP (Gardner et al., 2017) and Stanford NER (Finkel et al., 2005) outputs in CWQ experiments and use S-MART (Yang and Chang, 2016) in WQSP experiments. We further build an uppercase detector to add uppercase words to the ensembling results. For entity type linking, we search for entity types from the Freebase via two relations, `ns:common.topic.notable_types` and `ns:type.object.name`. For Albert training, we initialize the model with pre-trained weights and fine-tune it on the corresponding KBQA dataset for 5 epochs. The model has 12 layers, 4096 hidden dimensions, and 64 attention heads. We set learning rate to $1e^{-5}$ and limit the maximum length of input sequence to 128 tokens.

3.1 Experimental Results

Table 2a compares our method with state-of-the-art models. We adopt the F1 score between the predicted answer set and the ground truth answer set as our main evaluation metric. Experimental results show that our method outperforms existing methods on CWQ, while staying competitive on WQSP. We can see that most previous methods perform very well on WQSP but poorly on CWQ. This is because the "step-wise growing" methods have to restrict search space in order to be tractable on complex question datasets, and that causes good query graph candidates to be missed, ultimately hurting the performance on CWQ. In the query graph generation stage during training, the search space of previous methods is $\Theta(n^t)$ without beam search or $\Theta(\sum_t bn)$ with beam search, while ours is $\Theta(\binom{k}{t})$, where t is the maximum number of hops, n is the average number of degrees in $\mathcal{KB}$, b is the beam size, and k is the number of nodes in $\mathcal{RG}^q$. In practice, we have roughly $n = 70$, $3 \leq b \leq 8$, $k = 15$, and $t = 5$ on CWQ and $t = 2$ on WQSP. Our search space is not as restricted as the previous methods using beam search but is still tractable. On CWQ our model only took 1 day to train while the second best model QGG (Lan and Jiang, 2020) took 2 weeks.

To further disentangle the contributions of different factors in our method, we present an ablation test on CWQ in Table 2b. Among four features used in query graph generator, the (relation, entity type, relation) triplet feature has the biggest impact on the performance. Features such as shape and entity type are global features that capture important priors about the graph. Without them, performance drops in both query generation and ranking stages. It is also necessary to extract paths to make a sequential input to Albert. If instead, we simply concatenate triplet facts into a sequence, even though this saves time to detect source nodes and paths, the performance drops by 1.4%. We also observe a significant performance boost by predicting top 5 predictions instead of just the top 1, implying that correct answers are still ranked high when they are not at rank 1.

Table 2c shows separate performance of each component and how overall F1 score changes through the three stages. We show the cardinality of the output (# of predictions) in each stage. On CWQ, the entity linking models do not work well, as almost half of the questions in CWQ have multiple topic entities which makes the linking task difficult. On WQSP, the bottleneck is extracting relation graph $\mathcal{RG}^q$, likely due to overfitting on small training data. By adding gold relations to $\mathcal{RG}^q$ and using the same setup as the current WQSP experiment, we got 84.1% upper bound F1 in query graph generator and 73.3% F1 in the final output. This score is comparable to the state-of-the-art model.

3.2 Error Analysis

As shown in Table 2c, the entity linking model does not perform very well on CWQ dataset. We note that most of the questions in CWQ contain multiple topic entities, which makes the prediction job more challenging than it is on WQSP. This is the main reason why there is a big performance gap between CWQ and WQSP. In addition to that, we notice several difficulties of doing entity linking on CWQ. (1) Typo in the dataset: "Bill Clinton" is mistakenly spelled as "Bill Clnton". (2) Name is not unique: there are more than one "Michael Jordan" in the knowledge graph. There is no automatic way to determine which "Michael Jordan" the question refers to. (3) Topic entity can be a generic word: in question *"What art movement do the artists who study perspective belong to?"*, the topic entity is "perspective". It is difficult to detect it with a regular entity linking tool. (4) Disambiguation: similar to (2), the model needs to map the extracted words to corresponding entities in the knowledge graph. Even if the entity is unique in $\mathcal{KB}$, it is not always easy to perfectly perform the mapping.

We see a significant F1 score drop on CWQ dataset in the last stage (from 0.741 to 0.462). We take a closer look at failure cases and observe that the model has difficulty in distinguishing very similar relations. For example, for the question *"Where did the subject of the movie 'I'm Not There live'?"*, the model predicts a graph with the relation "place_of_birth", while the graph with the correct relation "places_lived" is ranked second. This is because, in training set, a similar question *"In the film 'Lydia Bailey', where did the subject live?"* is linked with relation "place_of_birth", whereas the better relation "places_lived" is not annotated. This kind of issue could be alleviated by annotating more positive samples or encouraging the model to explore unlabeled data in training (Qin et al., 2020). A rather tricky issue is that the relation "ns:location.country.languages_spoken" is usually mistakenly predicted as "ns:location.country.official_language", or the other way around. These two relations are represented by similar features in the embedding space and thus easily confuse the model. Specifically, they appear 1,707 and 825 times in the training set, and in more than half of the cases they are perfectly interchangeable or generate very close answers. To distinguish such similar relations, the model needs a large number of samples to learn the subtle difference between the two.

4 Conclusion

We propose a novel query graph generation method by gradually shrinking a $\mathcal{KB}$ to a desired query graph. Compared to previous approaches, our approach is more computationally efficient. Experiments show that our method delivers consistent performance on two KBQA datasets: it improves the state-of-the-art results by an absolute 5.8% in F1 on the multi-hop KBQA task CWQ and produce competitive result on the single-hop / two-hop KBQA task WQSP. In contrast, while some baseline methods work somewhat better on simple single-hop / two-hop questions, their performance drops dramatically on complex multi-hop questions.

References

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2503–2514. ACL.

Nikita Bhutani, Xinyi Zheng, and H. V. 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, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 739–748. ACM.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. Uhop: An unrestricted-hop relation extraction framework for knowledge-based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 345–356.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 363–370. The Association for Computer Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Google. 2013. Freebase data dumps. https://developers.google.com/freebase/data.

Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 969–974. Association for Computational Linguistics.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Knowledge base question answering with topic units. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5046–5052.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 23–33.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Q. Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2185–2194.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Kechen Qin, Yu Wang, Cheng Li, Kalpa Gunaratna, Hongxia Jin, Virgil Pavlu, and Javed A Aslam. 2020. A complex kbqa system using multiple reasoning paths. *arXiv preprint arXiv:2005.10970*.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4231–4242.

Alon Talmor and Jonathan Berant. 2018. Repartitioning of the complexwebquestions dataset. *CoRR*, abs/1807.09623.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Bingyu Wang, Cheng Li, Virgil Pavlu, and Jay Aslam. 2018. A pipeline for optimizing f1-measure in multi-label text classification. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 913–918. IEEE.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yi Yang and Ming-Wei Chang. 2016. S-MART: novel tree-based structured learning algorithms applied to tweet entity linking. *CoRR*, abs/1609.08075.

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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 571–581.

Hongzhi Zhang, Guandong Xu, Xiao Liang, Tinglei Huang, and Kun Fu. 2018. An attention-based word-level interaction model: Relation detection for knowledge base question answering. *CoRR*, abs/1801.09893.

Wenbo Zhao, Tagyoung Chung, Anuj Kumar Goyal, and Angeliki Metallinou. 2019. Simple question answering with subgraph ranking and joint-scoring. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 324–334. Association for Computational Linguistics.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2010–2022.