NAMER: A Node-Based Multitasking Framework for Multi-Hop Knowledge Base Question Answering

Minhao Zhang[1] **Ruoyu Zhang**[1] **Lei Zou**[1,2] **Yinnian Lin**[1] **Sen Hu**[1]
[1]Peking University, China;
[2]National Engineering Laboratory for Big Data Analysis Technology and Application (PKU), China;
{zhangminhao,ry_zhang,zoulei,linyinnian,husen}@pku.edu.cn

Abstract

We present NAMER, an open-domain Chinese knowledge base question answering system based on a novel node-based framework that better grasps the structural mapping between questions and KB queries by aligning the nodes in a query with their corresponding mentions in question. Equipped with techniques including data augmentation and multitasking, we show that the proposed framework outperforms the previous SoTA on CCKS CKBQA dataset. Moreover, we develop a novel data annotation strategy that facilitates the node-to-mention alignment, a dataset[1] with such strategy is also published to promote further research. An online demo of NAMER[2] is provided to visualize our framework and supply extra information for users, a video illustration[3] of NAMER is also available.

1 Introduction

With the rapid popularization of knowledge bases (KB), knowledge base question answering (KBQA) (Unger et al., 2014) has witnessed much research effort to fulfill a robust system to simplify users' access to KBs. For any given factoid question in natural language, KBQA system utilizes its background KB for answers. Recently, many SoTA KBQA systems adopt a semantic parsing (Kwiatkowski et al., 2013; Yih et al., 2014) framework, in which they convert the question to a KB query (e.g. SPARQL, Prud'hommeaux, 2008) to get answers.

Since queries are highly structured, a robust KBQA system needs to grasp the structural mapping (Figure 1) between a question and its query. However, most previous works either adopted an end-to-end model that failed to directly use such mappings (Ge et al., 2019; Ji et al.) or devised a template or rule-based pipeline (Hu et al., 2018; Cui et al., 2017) that may lose generality in real-world applications. To preserve generality, Shen et al. (2019) incorporated a pointer generator (See et al., 2017) into the pipeline to learn the mapping between an entity and its mention. Nevertheless, the system failed to utilize the mappings of variables, literals, and types in a query.

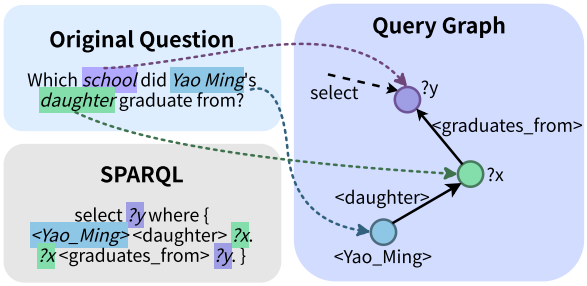

Figure 1: Structural mapping between a question and its corresponding SPARQL query.

In this paper, we argue that learning the complete question-query mapping (i.e. the alignments of all nodes to their mention, as in Figure 1) aids the system to achieve better performance. Hence, we supplement an open-domain complex Chinese KBQA dataset with annotations of all node mentions. Based on the additional data, we propose a novel node-based multi-hop KBQA framework that fully grasps the mappings of entities, variables, literals, and types. Unlike prior works, we generate the pointer of all query nodes to their mention to represent the mapping and exploit such mappings in downstream relation extraction task. Also, we explore techniques including multitasking to further improve model performance. Based on the framework, we implement a publicly available Chinese KBQA system, NAMER, for users to query KBs by natural language, which offers convenience for non-expert users to use KB and is thus fairly useful in practice. In short, the contributions of this work are: 1) we propose a novel KBQA framework with a strong ability to grasp structural mapping,

[1]https://github.com/ridiculouz/CKBQA
[2]http://kbqademo.gstore.cn
[3]https://youtu.be/yetnVye_hg4

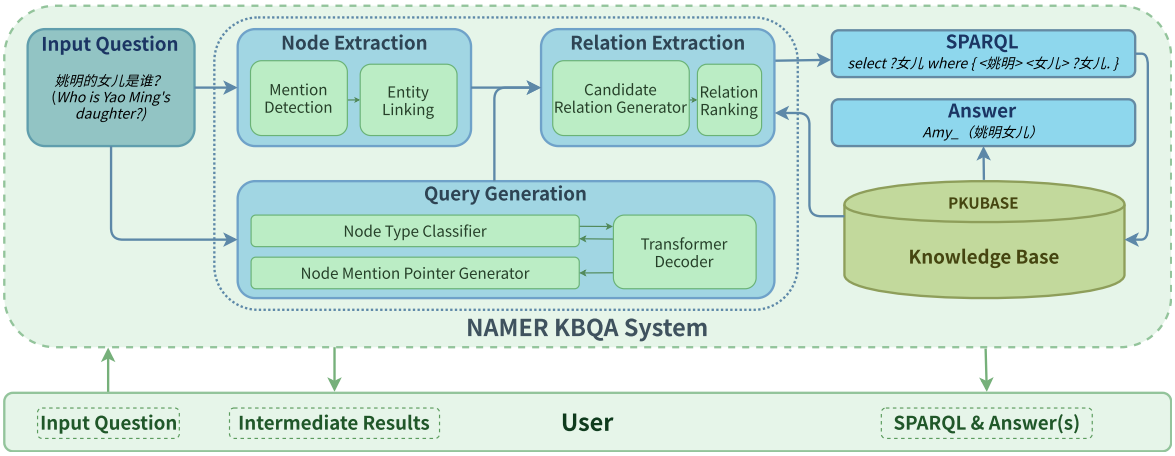

Figure 2: Architecture of the proposed system

the approach reaches SoTA results on a Chinese KBQA dataset, 2) we present a new data annotation format to better train KBQA models and publish a supplementary dataset of this format to prompt future research and 3) we implement an online demonstration of NAMER that can visualize our framework and aid users to explore KBs.

2 System Overview

This section explains the overall architecture of the proposed framework and the UI of the system.

2.1 Framework Architecture

Figure 2 illustrates the architecture of our framework. Basically, the framework can be divided into three modules, namely node extraction (NE), query generation (QG) and relation extraction (RE). Given a natural language question, NE extracts mentions of entities, variables, literals, types and performs entity linking. Meanwhile, QG generates a node sequence (i.e. vertices in the KB query, see Section 3.1 and 3.2 for more details) corresponding to the given question. Each node generated in QG consists of its type and the pointer to its mention in the input question, such pointer is replaced by the node extracted in NE when fusing NE and QG results. Up to now, we can generate the vertices of a SPARQL query, i.e. the head and tail of all its triples; to form a complete SPARQL output, RE (Section 3.3) is introduced to decide the edges (i.e. the relation of all triples). For each pair of nodes given by NE+QG, RE takes the raw question and mentions of the head and tail node as inputs to decide the relation between them. Combining all three modules, a SPARQL query is finally composed and sent to a knowledge base to get answers.

2.2 User Interface

An example of the interaction between users and our system is illustrated in Figure 3. With this UI, users can not only consult NAMER to answer their questions but also acquire more information around their interested entities and understand how NAMER works to compose the generated query.

3 Model

Consider the question "*Where was Yao Ming's daughter born?*", the following section elaborates how each module process the question to compose the correct SPARQL "*select ?y where { <Yao_Ming> <daughter> ?x. ?x <place_of_birth> ?y. }*".

3.1 Node Extraction (NE)

We define nodes as entities, variables, literals and types in a SPARQL query, namely the entity *<Yao_Ming>* and the variable *?x* and *?y* in the case above. The NE module aims to detect mentions of all nodes in a question, i.e. "Yao Ming", "daughter", and "where" respectively. To achieve this, we utilize a transformer (Vaswani et al., 2017) encoder with a sequence-tagging head of tag space {O, Eb, Ei, Vb, Vi, VTb, VTi, Tb, Ti, VLb, VLi, Lb, Li} (VL/VT denotes variable-literal/variable-type since mentions of multiple nodes may overlap) to tag the question. Afterward, NE performs entity linking on extracted entities via a mention-to-entity dictionary corresponding to the KB. For each entity

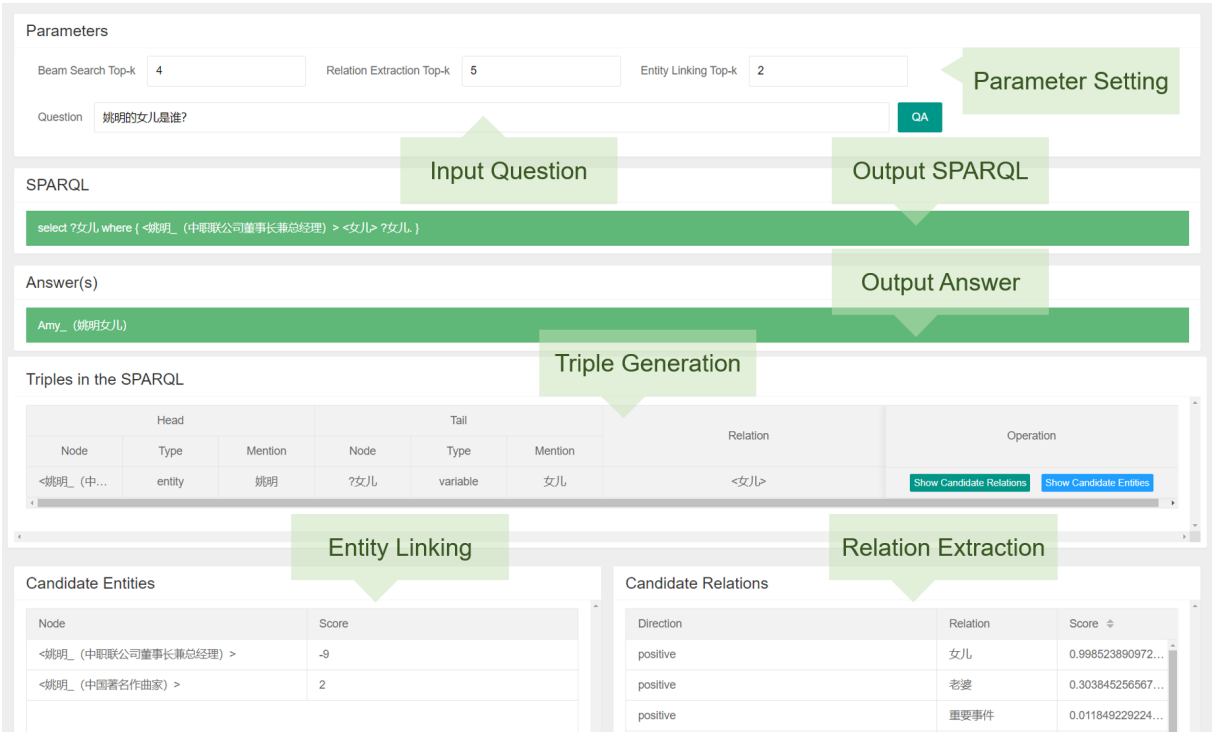

Figure 3: The user interface of NAMER. By entering a question and setting up a few parameters, a user can receive the output SPARQL and answer with intermediate results to visualize our framework. For instance, a user can check "Triples in the SPARQL" for the structure of the generated triples. Besides, after clicking the "Show Candidate Relations" button of each triple, its top score candidate relations would be displayed below; after clicking the "Show Candidate Entities", the scores of candidate entities in entity linking are also provided.

mention, we select its longest substring that appears in the dictionary and view the entities linked to such substring as candidate entities.

3.2 Query Generation (QG)

In QG, we want to generate the node sequence of the expected SPARQL, i.e. [*?y*, *<Yao_Ming>*, *?x*, *?x*, *?y*] for the instance above (the first node is the selected variable). One direct method to do so is to adopt a decoder that directly generates such sequence. However, as mentioned before, such an approach poses difficulties for models to grasp the query-question mapping. Hence, we adopt a pointer network (See et al., 2017) to generate a sequence of <type (entity, variable, etc.), pointer> to represent node sequence.

More specifically, QG model is based on a transformer encoder and decoder. Let $H_E \in R^{n \times d_h}$ be the encoder output given the question as input, let $T \in N^q$ be the previously generated node types (n is the question length, d_h denotes hidden dimension, q is the length of the node sequence) by the decoder. At each decoding step, hidden vector h_q of the current node is generated, which is then fed to an FFN to represent the type of next node $T_{next} \in \{E, V, L, T, Start, End\}$. We concatenate T_{next} to T for the next decoding step.

$$h_q = Decoder(T, H_E) \in R^{d_h}$$
$$P_{next} = softmax(FFN(h_q)) \in R^6$$
$$T_{next} = \arg\max_i P_{next}$$

An attention matrix $W^{att} \in R^{d_h \times d_h}$ is trained to calculate attention score of each input token and the pointer Ptr_{cur} being the input with max score.

$$S_{cur} = h_q * W^{att} * H_E^T \in R^n$$
$$Ptr_{cur} = \arg\max_i S_{cur}$$

When combining NE and QG results, we can replace each pointer with the node it points to given by NE, e.g. replacing <var, 5> with a variable node "?daughter" with mention "daughter". Consequently, the expected node sequence [*?where*, *<Yao_Ming>*, *?daughter*, *?daughter*, *?where*] can now be formed.

3.3 Relation Extraction (RE)

RE module aims to determine the relation of each node pair generated in QG, i.e. determining the

System	Dev Set			Test Set		
	P	R	F1	P	R	F1
Team JCHL	/	/	.707	.742	.752	.736
gAnswer	.593	.598	.589	.554	.560	.549
NAMER	**.774**	**.770**	**.761**	**.772**	**.771**	**.757**

Table 1: Performance on CCKS CKBQA dataset. The official metrics are average answer-level F1 while we also report Precision and Recall. "Team JCHL" refers to the contest winner whose dev P&R wasn't published.

relation <*daughter*> between <*Yao_Ming*> and *?x* and <*place_of_birth*> between *?x* and *?y* for the aforementioned case. We complete this in a ranking manner, which is, we first generate candidate relations for each node pair n_1 and n_2 (next paragraph), then, we concatenate each candidate with the raw question and the mentions of head and tail nodes to form model input. Such input is encoded by a transformer encoder and converted to a number $S \in [0, 1]$ to represent the score of such candidate relation. RE module selects top-scored candidates of each node pair to form output SPARQL. More specifically, since relations are directional in KB, we obtain candidates of both positive (from n_1 to n_2) and reversed (from n_2 to n_1) directions, marked as R_{pos} and R_{rev} respectively. Suppose a positive relation r^* is the correct choice and q is the question, for each r_1/r_2 in R_{pos}/R_{rev} excluding r^*, we construct $(q, n_1, n_2, r_1)/(q, n_2, n_1, r_2)$ as negative samples and (q, n_1, n_2, r^*) as a positive sample to train our model.

For each node pair, we query KB to obtain candidate relations. For pairs with an entity, literal or type (deterministic) node in it, we view those relations around that node in KB as candidates; for pairs merely consist of variables, we trace back the route from these variables to any deterministic node and view the relations k-hop away from the deterministic node as candidates. For instance, if three pairs (<*Yao_Ming*>, *?x*), (*?x*, *?y*) and (*?y*, *?z*) are generated, their candidates are 1, 2 and 3-hop away from entity <*Yao_Ming*> in KB respectively.

Additionally, we propose an augmentation method when training RE model. Back to the case above, we also add $(q, n_2, n_1, r_1)/(q, n_1, n_2, r_2)$ and (q, n_2, n_1, r^*) to negative samples when training. Consequently, the model learns the effects of mention order to the prediction, through which it may learn a better scoring policy. See further analysis in Section 4.4.

3.4 Multitasking

Clearly, since all modules above have an encoder, we can share it across different models in the hope of better comprehension and less error propagation. Let $loss_{NE}, loss_{type}, loss_{ptr}, loss_{RE}$ be the losses of NE, QG-type, QG-pointer, and RE respectively, we can co-train the models by minimizing the weighted sum over all losses.

$$loss = \gamma * loss_{NE} + \alpha * loss_{type} + \beta * loss_{ptr} + \theta * loss_{RE}$$

We can also multitask on a subset of modules by setting some hyperparameters $(\gamma, \alpha, \beta, \theta)$ to zero.

4 Experiments

4.1 Experimental Setup

Dataset We utilize the dataset published in CCKS Chinese KBQA Contest[4] for evaluation. The dataset consists of various Chinese open-domain complex (multi-hop) questions that require deep comprehension of questions and strong generalization ability, its background KB is PKUBASE[5], a Chinese KB based on Baidu Baike. We follow the raw separation of 2.2k/0.76k/0.76k train/dev/test data, note that no information in dev or test set are used when training.

Annotations We manually label the mention of all SPARQL nodes in the question required by our framework in the train and dev set. When multiple mentions co-refer a node, all mentions are accepted but we recommend annotators to choose a more informative one, e.g. for the question "*Who is Yao Ming's daughter?*" and SPARQL "*select ?x where {<Yao_Ming> <daughter> ?x.}*", both "*daughter*" and "*who*" refer to *?x*, but the former is preferred. When no mention refers to a node, annotators leave the mention as "*None*". We perform a brief double-check on 420 randomly selected questions and >93% of which are annotated correctly. See more details of the annotation process in Ethical Considerations.

Baselines We compare our results with the top ranking team *"jchl"*[6](Luo et al., 2019) in the contest and a competitive KBQA system gAnswer[7] (Hu

[4] https://www.biendata.xyz/competition/ccks_2019_6/data/

[5] A KB endpoint: http://pkubase.gstore.cn/

[6] Team *"luoxiao1"* was disqualified in final ranking so we compare with the team that won the contest

[7] https://github.com/pkumod/gAnswer

Methods	NE			QG		RE		Overall F1	
	P	R	F1	EM Acc.	Actual Acc.	Hit@5	MRR	Dev Set	Test Set
Separate	.840	.855	.843	.654	.773	.971	.895	.730	.715
NE+QG+RE	.839	**.862**	.846	.668	**.795**	.957	.866	.726	.705
NE+QG	**.843**	.861	**.847**	**.678**	.792	**.971**	**.895**	**.761**	**.757**

Table 2: Performance details of different multitasking strategies. "Methods" refer to co-trained modules, "Separate" means no multitasking. Metrics in NE refers to the P/R/F1 of the extracted node list. In QG, EM (exact-match) and Actual Acc. means the accuracy of generated node sequence (yield of NE&QG); the former counts when the generated sequence is identical to gold sequence while the latter counts when two sequences are equivalent semantically, e.g. when gold and generated sequence are [*?daughter*, *<Yao_Ming>*, *?daughter*] and [*?who*, *<Yao_Ming>*, *?who*], EM Acc. does not count due to false pointer of the variable but they are semantically equal since the name of a variable does not effect query results. For RE, Hit@5 denotes the ratio of node pairs whose score of gold relation is among top-5 in all candidates; MRR was defined in Craswell, 2009. Overall F1 is explained in Table 1. We evaluate all NE, QG, and RE-related metrics on dev set.

et al., 2018) that reached first place in QALD-9 (Ngomo, 2018). Since the NE and RE module in gAnswer does not officially support Chinese, we replace them with those in our system. Hence, the gAnswer evaluated can be partly viewed as our system with a rule-based QG module and its comparison with us indicates the effectiveness of our generative QG module.

Setup We adopt Chinese RoBERTa-large (Cui et al., 2020) in transformers library (Wolf et al., 2020) released by HFL[8] as encoder and a 6-layer 8-head transformer as decoder. For our best results, we co-train the NE and QG models, remaining RE as a separate model. For NEQG, we train the encoder and decoder with learning rate 1e-6 and 4e-6 respectively with an Adam (Kingma and Ba, 2015) optimizer, setting hyperparameters to $\gamma = 1, \alpha = \beta = 2.5, \theta = 0$ and batch size to 40. For RE model, we set the learning rate and batch size to 1e-5 and 96 respectively with $\gamma = \alpha = \beta = 0, \theta = 1$. Both models are trained until no progress on validation accuracy for at most 10k steps.

4.2 Overall Performance Evaluation

Table 1 compares the performance of our system and the baselines on official F1 metrics as well as precision and recall. As shown, our system consistently outperforms the contest winner *"jchl"* on dev and test set while significantly surpass the modified version of gAnswer, setting up a new state-of-the-art performance on the evaluated dataset.

We attribute the improvement to the effectiveness of the proposed framework. With the cooperation of NE and QG, NAMER learns the direct mapping between question and query, making it possible for models to deeply grasp their supervision signals even in case of complex questions and insufficient training data, which is exactly the case for the current dataset. Since the evaluated gAnswer can be viewed as replacing QG with a rule-based subgraph matching module, our advantage over it also implies the superiority of a trainable generative module in KBQA which, we speculate, has better generalization ability facing the highly diversified questions. Finally, based on NEQG, our RE module can naturally deal with complex multi-hop questions by processing a triple (instead of a question) at a time, resulting in an accurate relation scoring for every node pair.

4.3 Analysis of Multitasking

In his section, we try to discuss the impact of different multitasking strategies (Section 3.4) on the framework performance. The results of each module and the overall metrics are given in Table 2. Evidently, multitasking NE and QG consistently improves performance over no multitasking; this is probably due to the shared supervise signals across NE and QG offer extra information for models to better comprehend their tasks. E.g., the supervision in NE tells QG model the semantics of a pointer (since it provides the node mention of a pointer) which assists QG to predict pointers. However, when multitasking all three modules, the performance fails to improve. In detail, although NE and QG metrics resemble our best results, RE encounters a considerable drop on both metrics. We

[8]Pretrained weights: https://github.com/ymcui/Chinese-BERT-wwm

Methods	RE		Overall F1	
	Hit@5	MRR	Dev-set	Test-set
Ours	**.971**	**.895**	**.761**	**.757**
w/o Aug.	.943	.863	.736	.740

Table 3: Effects of RE data augmentation. "w/o Aug." denotes the RE model trained without augmentation.

Methods	n_1 to n_2		n_2 to n_1	
	Relation	Score	Relation	Score
Ours	*Elder_Brother*	.998	*Younger_Brother*	.801
w/o Aug.	*Elder_Brother*	.999	***Elder_Brother***	.999

Table 4: A case study for the top-scored relation in both directions between a node pair. False answer is in bold.

speculate that the different input format of NEQG and RE results in a different semantic space on the tasks, which harms the performance when we forcibly co-train them. Anyway, multitasking notably reduces the storage cost of our system by sharing one encoder across various tasks, which is significant for a system in practice.

4.4 Analysis of Data Augmentation

A data augmentation technique is introduced in Section 3.3, we inspect its effect in Table 3 and provide further discussion in this section. As illustrated, removing augmentation from RE results in a drop on both RE metrics and overall performance, indicating the positive effect of augmentation. To explain, we perform a case study on the question *What's the nickname of Tom's elder brother?*[9]. Consider the node pair $n_1 = Tom, n_2 = elder\ brother$, we compare the top-scored relation from n_1 to n_2 and from n_2 to n_1 given by the model with and without augmentation. As shown in Table 4, the augmented model outputs two antonymous relations in two directions while its counterpart makes two same predictions. Hence, we argue that the augmented training data help the model to concurrently learn 1) the topic-level relationship between a relation and a node pair in question and 2) the effect of node direction to relation (i.e. r^* is only the correct choice from n_1 to n_2, not conversely). The extra supervise signal enables a deeper comprehension of RE which, in turn, improves model performance.

Interestingly, we find a similar discussion in Lan et al. (2019) on the advantage of SOP over NSP, since sentence order provides additional supervision on discourse-level coherence (which largely resembles the coherence between node direction and relation in our case). Thus, we speculate that similar augmentation methods may work in more scenarios in future research.

[9]We translate raw Chinese input to English in this case.

5 Related Work

Semantic parsing-based KBQA A semantic parser in KBQA converts a question to a KB query. Previously, some works (Petrochuk and Zettlemoyer, 2018; Mohammed et al., 2018) only focus on answering one-hop questions. To process multi-hop questions, Cui et al. (2017) proposed a template-based pipeline in which a question is converted to a template to further decide its predicate. Hu et al. (2018) and Jin et al. (2019) adopted a subgraph-matching-based model in the pipeline to form a query graph. Ge et al. (2019) used a seq2seq transformer to directly generate queries. To help models directly comprehend the structural mapping, Wang et al. adopted a query template generator as well as an entity and relation extractor to represent the mentions of entities and relations; however, they failed to utilize the mention of variables and literals. Similar to our approach, Shen et al. (2019) used a pointer-generator and entity extractor to grasp the mapping between an entity and its mention, but the mappings of other types of nodes are omitted in their work, also, unlike us, the mappings failed to directly assist downstream RE task. Different from the above, we propose a framework that grasps the mappings of all node types and use them to aid downstream tasks.

Public KBQA systems Prior to us, several online KBQA systems are available for the public. However, most systems focused on domain-specific KBs, e.g. E-commerce (Li et al., 2019) and food (Haussmann et al., 2019). On open-domain KBs, Cui et al. (2016) published a system that can answer complex questions and visualize their answers. However, few systems on open-domain Chinese KBQA provide a publicly available web page with detailed visualization of the framework pipeline as in NAMER.

6 Conclusion

We present a robust Chinese KBQA system, NAMER, based on a novel node-based multitasking framework. With three cooperative modules, our system grasps the structural mapping between

a question and its corresponding query. Hence, NAMER reaches superior performance compared to previous SoTA on an open-domain Chinese complex KBQA dataset. Further experiments also demonstrate the effectiveness of the architecture and the techniques adopted in NAMER. As a system intended for easier access to KB for all users, the UI of NAMER provides not only the answers to a given question but also the query structure accompanied by a series of intermediate results (e.g. candidates & scores), assisting users to visualize our system pipeline and explore more KB information to their interest.

For the future, we will incorporate more visualization functions into NAMER to further reduce the barrier to KB for nonspecialist users. Since extra data annotations are required to support NAMER, we also plan to study the effects of the scale of annotated data on system performance. Moreover, we expect to implement and optimize NAMER in multilingual scenarios.

Ethical Considerations

Data Collection

Annotation Guideline SPARQL queries usually include several triples, restricting the range of target answers. Nodes are defined as the entities, variables, literals, and types in a SPARQL (including the select variable). For instance, the SPARQL *select ?x where {<Yao_Ming> <daughter> ?x.}* corresponds to the node sequence [*?x*, *<Yao_Ming>*, *?x*]. Given a natural language question "*Who is Yao Ming's daughter?*" and its corresponding SPARQL, annotators are asked to annotate the mention span of every node in the question, i.e. "*Yao Ming*" for *<Yao_Ming>* and "*daughter*" for *?x*.

Annotation Details The questions were distributed evenly to seven annotators with substantial knowledge of NLP. To ensure that the annotators were comfortable with the task, annotation guidance was given before the task began. After the primary annotation, two annotators double-checked the annotation to ensure consistency. All annotators worked part-time on the task.

System Output

We provide an online Chinese KBQA system as shown in Figure 3. The system uses PKUBASE as its supportive KB and accepts Chinese questions as possible input. Despite our efforts to eliminate biased and offensive output, NAMER retains the potential to generate answers that may be wrong or trigger offense. This failure may be induced by the deficiency of PKUBASE, implicit bias of the pretrained model and the limitation of training data. These are known issues in current state-of-the-art neural network-based language models and automatically constructed knowledge base. In no case should inappropriate answers generated by NAMER be construed to reflect the views or values of the authors.

Acknowledgements

We would like to thank Yanzeng Li and Wenjie Li for the valuable assistance on system design and implementation. We also appreciate anonymous reviewers for their insightful and constructive comments. This work was supported by NSFC under grants 61932001, 61961130390, U20A20174. This work was also partially supported by Beijing Academy of Artificial Intelligence (BAAI). The corresponding author of this work is Lei Zou (zoulei@pku.edu.cn).

References

Nick Craswell. 2009. Mean reciprocal rank. *Encyclopedia of database systems*, 1703.

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. *Proceedings of the VLDB Endowment*, 10(5).

Wanyun Cui, Yanghua Xiao, and Wei Wang. 2016. Kbqa: An online template based question answering system over freebase. In *IJCAI*, volume 16, pages 09–15.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online. Association for Computational Linguistics.

Donglai Ge, Junhui Li, and Muhua Zhu. 2019. A transformer-based semantic parser for nlpcc-2019 shared task 2. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 772–781. Springer.

Steven Haussmann, Yu Chen, Oshani Seneviratne, Nidhi Rastogi, James Codella, Ching-Hua Chen, Deborah L McGuinness, and Mohammed J Zaki. 2019. Foodkg enabled q&a application. In *ISWC Satellites*, pages 273–276.

S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.

Guangxi Ji, Shujun Wang, Ding Zhang, Xiaowang Zhang, and Zhiyong Feng. A fine-grained complex question translation for kbqa.

Hai Jin, Yi Luo, Chenjing Gao, Xunzhu Tang, and Pingpeng Yuan. 2019. Comqa: Question answering over knowledge base via semantic matching. *IEEE Access*, 7:75235–75246.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1545–1556.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Feng-Lin Li, Weijia Chen, Qi Huang, and Yikun Guo. 2019. Alime kbqa: Question answering over structured knowledge for e-commerce customer service. In *China Conference on Knowledge Graph and Semantic Computing*, pages 136–148. Springer.

Jinchang Luo, Cunxiang Yin, Xiaohui Wu, Lifang Zhou, and Huiqiang Zhong. 2019. Hunhe yuyi xiangsidu de zhongwen zhishitupu wenda xitong [a chinese knowledge base question answering system based on mixed semantic similarity]. *Proceedings of the 2019 China Conference on Knowledge Graph and Semantic Computing: Evaluation Papers*.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296.

Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (qald-9). *language*, 7(1).

Michael Petrochuk and Luke Zettlemoyer. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 554–558.

Eric Prud'hommeaux. 2008. Sparql query language for rdf, w3c recommendation. *http://www. w3. org/TR/rdf-sparql-query/*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Tao Shen, Xiubo Geng, QIN Tao, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-task learning for conversational question answering over a large-scale knowledge base. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2442–2451.

Christina Unger, André Freitas, and Philipp Cimiano. 2014. An introduction to question answering over linked data. In *Reasoning Web International Summer School*, pages 100–140. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shujun Wang, Jie Jiao, Yuhan Li, Xiaowang Zhang, and Zhiyong Feng. Answering questions over rdf by neural machine translating.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648.