Asking Clarification Questions in Knowledge-Based Question Answering

Jingjing Xu[1]*, **Yuechen Wang**[2], **Duyu Tang**[3], **Nan Duan**[3], **Pengcheng Yang**[1], **Qi Zeng**[1], **Ming Zhou**[3], **Xu Sun**[1]

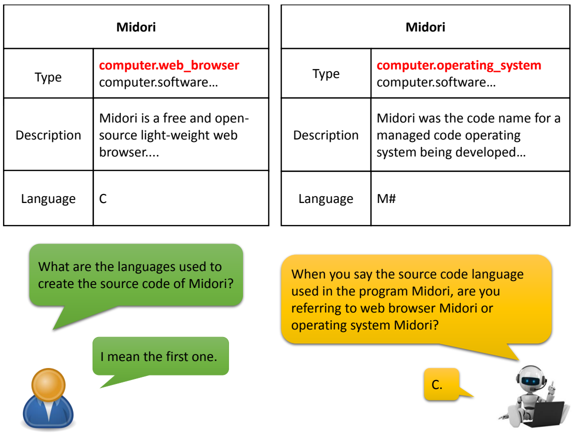
[1] MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2] University of Science and Technology of China
[3] Microsoft Research Asia, Beijing, China
{jingjingxu,yang_pc,pkuzengqi,xusun}@pku.edu.cn
wyc9725@mail.ustc.edu.cn
{dutang,nanduan,mingzhou}@microsoft.com

Abstract

The ability to ask clarification questions is essential for knowledge-based question answering (KBQA) systems, especially for handling ambiguous phenomena. Despite its importance, clarification has not been well explored in current KBQA systems. Further progress requires supervised resources for training and evaluation, and powerful models for clarification-related text understanding and generation. In this paper, we construct a new clarification dataset, CLAQUA, with nearly 40K open-domain examples. The dataset supports three serial tasks: given a question, identify whether clarification is needed; if yes, generate a clarification question; then predict answers base on external user feedback. We provide representative baselines for these tasks and further introduce a coarse-to-fine model for clarification question generation. Experiments show that the proposed model achieves better performance than strong baselines. The further analysis demonstrates that our dataset brings new challenges and there still remain several unsolved problems, like reasonable automatic evaluation metrics for clarification question generation and powerful models for handling entity sparsity.[1]

1 Introduction

Clarification is an essential ability for knowledge-based question answering, especially when handling ambiguous questions (Demetras et al., 1986). In real-world scenarios, ambiguity is a common phenomenon as many questions are not clearly articulated, e.g. *"What are the languages used to create the source code of Midori?"* in Figure 1. There are two "*Midori*" using different programming languages, which confuses the system. For these ambiguous or confusing questions, it is hard to directly give satisfactory responses unless systems can ask clarification questions to confirm the participant's intention. Therefore, this work explores how to use clarification to improve current KBQA systems.



Midori	
Type	computer.web_browser computer.software...
Description	Midori is a free and open-source light-weight web browser....
Language	C

Midori	
Type	computer.operating_system computer.software...
Description	Midori was the code name for a managed code operating system being developed...
Language	M#

What are the languages used to create the source code of Midori?
When you say the source code language used in the program Midori, are you referring to web browser Midori or operating system Midori?
I mean the first one.
C.

Figure 1: An example of a clarification question in KBQA. There are two entities named "Midori" using different programming languages, which confuses the system.

We introduce an open-domain clarification corpus, CLAQUA, for KBQA. Unlike previous clarification-related datasets with limited annotated examples (De Boni and Manandhar, 2003; Stoyanchev et al., 2014) or in specific domains (Li et al., 2017; Rao and III, 2018), our dataset covers various domains and supports three tasks. The comparison of our dataset with relevant datasets is shown in Table 1. Our dataset considers two kinds of ambiguity for single-turn and multi-turn questions. In the single-turn case, an entity name refers to multiple possible entities in a knowledge base while the current utterance lacks necessary identifying information. In the multi-turn case, ambiguity mainly comes from the omission where a pronoun refers to multiple possible entities from the previous conversation turn. Unlike CSQA

[*] The work was done while Jingjing Xu and Yuechen Wang were interns in Microsoft Research, Asia.

[1] The dataset and code will be released at https://github.com/msra-nlc/MSParS_V2.0

Dataset	Domain	Size	Task
De Boni and Manandhar (2003)	Open domain	253	Clarification question generation.
Stoyanchev et al. (2014)	Open domain	794	Clarification question generation.
Li et al. (2017)	Movie	180K	Learning to generate responses based on previous clarification questions and user feedback in dialogue.
Guo et al. (2017)	Synthetic	100K	Learning to ask clarification questions in reading comprehension.
Rao and III (2018)	Operating system	77K	Ranking clarification questions in an online QA forum, StackExchange[2].
CLAQUA (Our dataset)	Open domain	40K	Clarification in KBQA, supporting clarification identification, clarification question generation, and clarification-based question answering.

Table 1: The comparison of our dataset with relevant datasets. The first and second datasets focus on generating clarification questions while the small size limits their applications. The middle three tasks are either not designed for KBQA or limited in specific domains. Unlike them, we present an open-domain dataset for KBQA.

dataset (Saha et al., 2018) that constructs clarification questions base on predicate-independent templates, our clarification questions are predicate-aware and more diverse. Based on the clarification raising pipeline, we formulate three tasks in our dataset, including clarification identification, clarification question generation, and clarification-based question answering. These tasks can naturally be integrated into existing KBQA systems. Since asking too many clarification questions significantly lowers the conversation quality, we first design the task of identifying whether clarification is needed. If no clarification is needed, systems can directly respond. Otherwise, systems need to generate a clarification question to request more information from the participant. Then, external user feedback is used to predict answers.

Our main contribution is constructing a new clarification dataset for KBQA. To build high-quality resources, we elaborately design a data annotation pipeline, which can be divided into three steps: sub-graph extraction, ambiguous question annotation, and clarification question annotation. We design different annotation interfaces for single-turn and multi-turn cases. We first extract "ambiguous sub-graphs" from a knowledge base as raw materials. To enable systems to perform robustly across domains, the sub-graphs are extracted from an open-domain KB, covering domains like music, tv, book, film, etc. Based on the sub-graphs, annotators are required to write ambiguous questions and clarification questions.

We also contribute by implementing representative neural networks for the three tasks and further developing a new coarse-to-fine model for clarification question generation. Take multi-turn as an example. In the task of clarification identification, the best performing system obtains an accuracy of 86.6%. For clarification question generation, our proposed coarse-to-fine model achieves a BLEU score of 45.02, better than strong baseline models. For clarification-based question answering, the best accuracy is 74.7%. We also conduct a detailed analysis for three tasks and find that our dataset brings new challenges that need to be further explored, like reasonable automatic evaluation metrics and powerful models for handling the sparsity of entities.

2 Data Collection

The construction process consists of three steps, sub-graph extraction, ambiguous question annotation, and clarification question annotation. We design different annotation interfaces for single-turn and multi-turn cases.

2.1 Single-Turn Annotation

Sub-graph Extraction. As shown in Figure 2, we extract ambiguous sub-graphs from an open-domain knowledge base, Satori. For simplification, we set the maximum number of ambiguous entities to 2. In single-turn cases, we focus on shared-name ambiguity where two entities have the same name and there is a lack of necessary distinguishing information. To construct such cases, we extract two entities sharing the same entity name and the same predicate. Predicate represents the relation between two entities. The sub-graphs provide reference for human annotators to write diverse ambiguous questions based on the shared predicates.

[2] An online QA community.

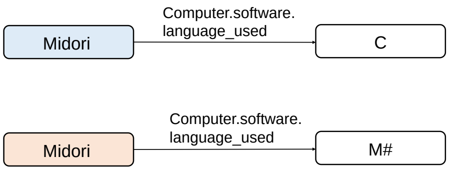


Figure 2: An extracted ambiguous sub-graph in the single-turn case. Two entities share the same name "Midori" and the same predicate "Computer.software,language_used".

Shared Name	Midori
Predicate # 1	computer.software.language_used
Predicate Description	A property relating an instance of computer.software to a programming language that was used to create the source code for the software.
Q_a: What are the languages used to create the source code of Midori?	

Table 2: An ambiguous question annotation example in the single-turn case.

E_1 Name	**Midori**
E_1 Type	computer.software **computer.web_browser** media_common.creative_work ... ...
E_1 Description	Midori is a free and open-source light-weight **web browser**. It uses the WebKit rendering engine and the GTK+ 2 or GTK+ 3 ... ...
E_2 Name	**Midori**
E_2 Type	**computer.operating_system** computer.software
E_2 Description	Midori was the code name for a managed code **operating system** being developed by Microsoft with joint effort of Microsoft Research... ...
Q_a	What are the languages used to create the source code of Midori?
Q_c: When you say the source code language used in the program Midori, are you referring to **[web browser Midori]** or **[operating system Midori]**?	

Table 3: A clarification question annotation example in the single-turn case.

Ambiguous Question Annotation. In this step, the input is a table listing the shared entity name, predicate name, and predicate description. Based on this table, annotators need to write ambiguous questions, e.g., *"What are the languages used to create the source code of Midori?"*. An annotation example is shown in Table 2. For diversity, annotators are encouraged to paraphrase the intention words in the predicate description.

Clarification Question Annotation. Based on entities and the annotated ambiguous question, annotators are required to summarize distinguishing information and write a multi-choice clarification question with a special marker for separating entity information and pattern information, e.g., *"When you say the source code language used in the program Midori, are you referring to [web browser Midori] or [operating system Midori]?"*. Annotators are asked to write predicate-aware clarification questions instead of general questions, like *"Which one do you mean, A or B?"* or *"do you mean A?"* as adopted in (Saha et al., 2018). An example is shown in Table 3.

We use multi-choice as our basic type of clarification. There are three possible clarification question types, including zero-choice type (e.g., *I don't understand, can you provide more details?*), single-choice type (e.g., *Do you mean A?*), and multi-choice type (e.g., *Which one do you mean, A or B?*). The zero-choice type means that the system does not understand the question and expects more details from the participant. Though simple, it costs more user efforts as it pushes the participant to figure out what confuses the system. In comparison, the advantage of single-choice type lies in the control of conversation direction. However, if the system cannot provide user-expected choices, conversation may be longer. As for the multi-choice type, its advantage is less conversation turns to ask for more valuable information while it requires more annotation work.

2.2 Multi-Turn Annotation

Sub-graph Extraction. In multi-turn cases, ambiguity comes from the omission of the target entity name from the previous conversation turn. We first extract two connected entities. If they also share another predicate, we extract all related information as an ambiguous sub-graph, as shown in Figure 3. By asking the shared predicate, we get an ambiguous question.

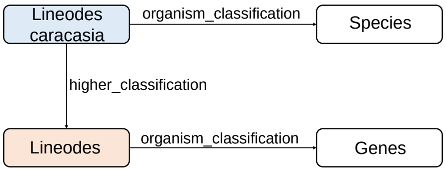


Figure 3: An extracted ambiguous sub-graph in the multi-turn case. "Lineodes caracasia" and "Lineodes" are linked and share the same relation "organism_classification".

E_1 Name	**Lineodes caracasia**
E_2 Name	**Lineodes**
Predicate #1	biology.higher_classification
Predicate Description	Property relating an biology.organism_classification to its higher or parent biology.organism_classification.
Predicate #2	biology.organism_classification
Predicate Description	Relating an biology.organism to its biology.organism_classification, the formal or informal taxonomy grouping for the organism.
Q_h: What is the higher classification for **Lineodes caracasia?** R_h: **Lineodes.** Q_a: What is the biological classification?	

Table 4: An ambiguous question annotation example in the multi-turn case.

E_1 Name	**Lineodes caracasia**
E_1 Type	media_common.cataloged_instance biology.organism_classification ... ...
E_1 Description	Lineodes caracasia is a moth in the family Crambidae... ...
E_2 Name	**Lineodes**
E_2 Type	media_common.cataloged_instance biology.organism_classification
E_2 Description	Lineodes is a genus of moths of the Crambidae family. ... ...
Q_h R_h Q_a	What is the higher classification for Lineodes caracasia? Lineodes. What is the biological classification?
Q_c: Are you referring to **[Lineodes caracasia]** or **[Lineodes]**, when you ask the biological classification?	

Table 5: A clarification question annotation example in the multi-turn case.

Ambiguous Question Annotation. An annotation example is shown in Table 4. Based on two entities in the extracted sub-graph, annotators construct a conversation turn and then write an ambiguous question where the entity name is omitted, e.g, *"What is the biological classification?"*.

Clarification Question Annotation. The annotation guideline of this step is the same as in single-turn annotation. The input includes two candidate entities and the annotated ambiguous question. The output is a clarification question. An annotation example is shown in Table 5.

3 Tasks

Our dataset aims to enable systems to ask clarification questions in open-domain question answering. Three tasks are designed in this work, including clarification identification, clarification question generation, and clarification-based question answering. Figure 4 shows the distribution of the top-10 domains in our dataset.

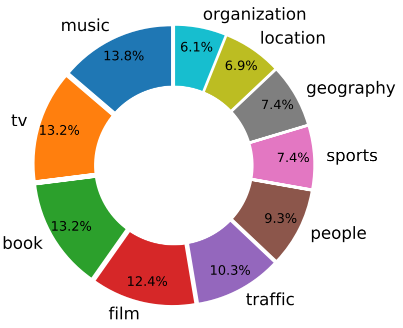


Figure 4: The distribution of top-10 domains.

3.1 Clarification Identification

Clarification identification can be regarded as a classification problem. It takes conversation context and candidate entity information as input. The output is a label identifying whether clarification is needed. Specifically, the input is $\{Q_a, E_1, E_2\}$ for single-turn cases or $\{Q_p, R_p, Q_a, E_1, E_2\}$ for multi-turn cases. Q_a represents the current question. E_1 and E_2 represent the candidate entities. Q_p and R_p are question and response from the previous conversation turn. The output is a binary label from set $Y = \{0, 1\}$ where 1 indicates that the input question is ambiguous and 0 indicates the opposite.

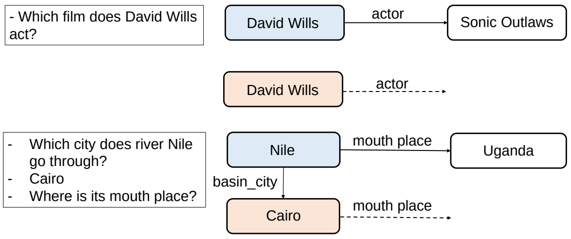

- Which film does David Wills act?
David Wills
actor
Sonic Outlaws
David Wills
actor
- Which city does river Nile go through?
- Cairo
- Where is its mouth place?
Nile
mouth place
Uganda
basin_city
Cairo
mouth place

Figure 5: Negative examples of unambiguous sub-graphs and annotated conversations. The dotted line means the non-existent relation. The upper figure is in the single-turn case and the lower one is in the multi-turn case.

As Figure 5 shows, negative examples are annotated in the same way as the positive (ambiguous) examples do, but without the clarification related steps. In their sub-graphs, one of the entities has its unique predicate. The unique predicates are included in the user questions like "Where is its mouth place?". As only river "Nile" has mouth

	Single-Turn		**Multi-Turn**	
	Positive	Negative	Positive	Negative
Train	8,139	6,841	12,173	8,289
Dev	487	422	372	601
Test	637	673	384	444

Table 6: Statistics of the dataset. It is important to note that three tasks share the same split. Since clarification question generation and clarification-based question answering are built upon ambiguous situations, they only use the positive data in their training, development, and test sets.

places, this question is unambiguous.[3]

The statistics of our dataset are shown in Table 6. It is important to note that three tasks share the same split. Since clarification question generation and clarification-based question answering are built upon ambiguous situations, they only use the positive (ambiguous) data in their training, development, and test sets. For generalization, we add some examples with unseen entities and predicates into the development and test sets.

3.2 Clarification Question Generation

This text generation task takes ambiguous context and entity information as input, and then outputs a clarification question. For single-turn cases, the input is $\{Q_a, E_1, E_2\}$. For multi-turn cases, the input is $\{Q_p, R_p, Q_a, E_1, E_2\}$. In both cases, the output is a clarification question Q_c. We use BLEU as the automatic evaluation metric.

3.3 Clarification-Based Question Answering

As our dataset focuses on simple questions, this task can be simplified as the combination of entity identification and predicate identification. Entity identification is to extract entity e from candidate entities based on current context. Predicate identification is to choose predicate p that describes the ambiguous question. The extracted entity e and predicate p are combined to query the knowledge triple (e, p, o). The model is correct when both tasks are successfully completed.

Additional user responses toward clarification questions are also necessary for this task. Due to the strong pattern in responses, we use a template-based method to generate the feedback. We first design four templates, “I mean the first one”, “I mean the second one”, “I mean [entity name]”, “I mean [entity type] [entity name]”. Then for each clarification example, we randomly select a candidate entity and fill its information into the template as the user response, e.g., “I mean web browser Midori.” for the clarification question in Table 3. For entity identification, we use the selected entity as the gold label. For predicate prediction, we use the predicate in the current ambiguous question as the gold label. In muli-turn cases, the predicate label is *Predicate #1* as shown in Table 2. In single-turn cases, the predicate label is *Predicate #2* as shown in Table 4.

Entity identification and predicate identification are both classification tasks. They take the same information as input, including ambiguous context, clarification question, and user response. For single-turn cases, the input is $\{Q_a, E_1, E_2, Q_c, R_c\}$ where R_c is the feedback of the participant toward clarification question Q_c. For multi-turn cases, the input is $\{Q_p, R_p, Q_a, E_1, E_2, Q_c, R_c\}$. For entity identification, the output is a label from set $Y_E = \{E_1, E_2\}$. For predicate identification, we randomly choose 1 negative predicates from the training set and combine them with the gold predicate together as the candidate set Y_P. The output of predicate identification is the index of the gold predicate in Y_P.

4 Approach

For three tasks, we implement several representative neural networks as baselines. Here we introduce these models in detail.

4.1 Classification Identification Models

The input of classification models is the ambiguous context and entity information. For simplification, we use a special symbol, `[S]`, to concatenate all the entity information together. These two parts can be regarded as different source information. Based on whether the inter-relation between different source information is explicitly explored, the classification models can be classified into two categories, unstructured models and structured models.

Unstructured models concatenate different source inputs into a long sequence with a special separation symbol `[SEL]`. We implement several widely-used sequence classification models, including Convolutional Neural Network (CNN) (Kim, 2014), Long-Short Term Memory Network (LSTM) (Schuster and Paliwal, 1997),

[3] It brings a little noise because some predicates may be omitted in the used knowledge base.

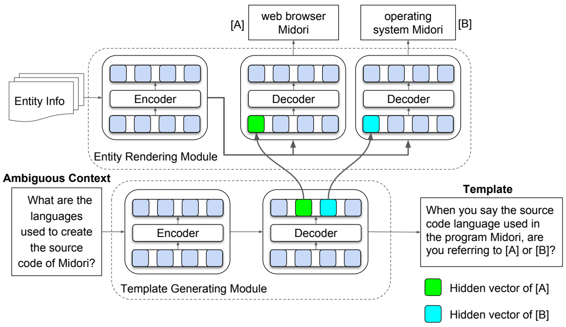

[A]
web browser Midori
operating system Midori
[B]
Entity Info
Encoder
Decoder
Decoder
Entity Rendering Module
Ambiguous Context
What are the languages used to create the source code of Midori?
Encoder
Decoder
Template Generating Module
Template
When you say the source code language used in the program Midori, are you referring to [A] or [B]?
Hidden vector of [A]
Hidden vector of [B]

Figure 6: An illustration of the proposed coarse-to-fine model. The proposed model consists of a template generating module and an entity rendering module. The former is used to generate a clarification template based on ambiguous question, e.g., "When you say the source code language used in the program Midori, are you referring to [A] or [B]?". The latter is used to fill up the generated template with detailed entity information.

and Recurrent Convolutional Neural Network (RCNN) (Lai et al., 2015), Transformer. The details of the models are shown in Supplementary Materials.

Structured models use separate structures to encode different source information and adopt an additional structure to model the inter-relation of the source information. Specifically, we use two representative neural networks, Hierarchical Attention Network (HAN) (Yang et al., 2016) and Dynamic Memory Network (DMN) (Kumar et al., 2016), as our structured baselines. The details of the models are shown in Supplementary Materials.

4.2 Clarification Question Generation Models

The input of the generation model is the ambiguous context and entity information. In single-turn cases, the ambiguous context is current question Q_a. In multi-cases, the input is current question and the previous conversation turn $\{Q_p, R_p, Q_a\}$. We use `[S]` to concatenate all entity information together, and use `[SEL]` to concatenate entity information and context information into a long sequence. We adopt Seq2Seq (Bahdanau et al., 2015) and Transformer (Vaswani et al., 2017) as our baselines and further develop a new generation model. The details of baselines can be found in Supplementary Materials.

Coarse-to-fine Model. Generally speaking, a clarification question contains two parts, entity phrases (e.g., "*web browser Midori*") and pattern phrases (e.g. *"When you say the source code language used in the program Midori, are you referring to [A] or [B]?"*). The entity phrase is summarized from the given entity information for distinguishing between two entities. The pattern phrase is used to locate the position of ambiguity, which is closely related with the context. In summary, two kinds of phrases refer to different source information. Based on this feature, we propose a new coarse-to-fine model, as shown in Figure 6. Similar ideas have been successfully applied to semantic parsing (Dong and Lapata, 2018).

The proposed model consists of a template generating module T_θ and an entity rendering module R_ϕ. T_θ first generates a template containing pattern phrases and the symbolic representation of the entity phrases. Then, the symbolized entities contained in the generated template are further properly rendered by the entity rendering module R_ϕ to reconstruct complete entity information. Since the annotated clarification questions explicitly separate entity phrases and pattern phrases, we can easily build training data for these two modules. For clarity, the template is constructed by replacing entity phrases in a clarification question with special symbols, `[A]` and `[B]`, which represent the positions of two entities.

The template generating module T_θ uses Transformer (Vaswani et al., 2017) as implementation. The input is the ambiguous context and the output is a clarification template.

Similar to T_θ, the entity rendering module R_ϕ is also implemented with a Transformer structure. More specifically, for each symbolized entity in the template, we add the hidden state of the decoder of T_θ corresponding to this symbolized entity into the decoder input.

4.3 Clarification-Based Question Answering Models

This task contains two sub-tasks, entity identification and predicate identification. They share the same input. In single-turn cases, the input is $\{Q_a, Q_c, R_c, E_1, E_2\}$. In multi-cases, the input is $\{Q_p, R_p, Q_a, Q_c, R_c, E_1, E_2\}$. Considering these two sub-tasks both belong to classification tasks, we use the same models as used in the task of clarification identification for implementation.

5 Experiments

We report the experiment results on three formulated tasks and give a detailed analysis. More

Model	Single-Turn	Multi-Turn
CNN	83.3	80.0
RNN	83.8	72.7
Transformer	84.0	72.8
HAN	83.1	73.0
DMN	**86.6**	**80.5**

Table 7: Results of clarification identification models.

details of hyper-parameters are in Supplementary Materials.

5.1 Experiment Results

Clarification Identification. As shown in Table 7, the structured models have the best performance, with a 86.6% accuracy on single-turn cases and a 80.5% accuracy on multi-turn cases. Compared to the best-performing unstructured model, the structured architecture brings obvious performance improvements, with 2.6% and 6.7% accuracy increases. It indicates that the structured architectures have learned the inter-relation between the context and entity details, which is a vital part for reasoning whether a question is ambiguous. We also find that the models all achieve better results on multi-turn cases. Multi-turn cases contain additionally previous conversation turn, which can help the models capture more key phrases from the entity information.

Clarification Question Generation. Table 8 shows that Seq2Seq achieves low BLEU scores, which indicates its tendency to generate irrelevant text. Transformer achieves higher performance than Seq2Seq. Our proposed coarse-to-fine model demonstrates a new state of the art, improving the current highest baseline result by 3.35 and 0.60 BLEU scores, respectively.

We also find that multi-turn cases generally obtain higher BLEU scores than single-turn cases. In single-turn cases, the same-name characteristic makes it hard to summarize key distinguishing information. As a comparison, in multi-turn cases, two candidate entities usually have different names and it is easier to generate clarification questions. It is due to the flexible structure of ambiguous questions, which makes it hard to define universal rules to identify the asked objects. Table 9 shows a an example generated by the proposed model.

Clarification-Based Question Answering. As Table 10 shows, the unstructured models perform better than the structured models, indicating that this task tends to be over-fitting and small models have better the generalization ability.

Model	Single-Turn	Multi-Turn
Seq2Seq	18.84	31.62
Transformer	20.69	44.42
The proposed Model	**24.04**	**45.02**

Table 8: Results of clarification question generation models.

E_1 Name	Lineodes interrupta
E_1 Type	biology.organism_classification
E_1 Description	Lineodes interrupta is a moth in the Crambidae family. It was described by Zeller in 1873. It is found in Mexico and the United States, where it has been recorded from Florida, Louisiana, Oklahoma and Texas.
E_2 Name	Lineodes
E_2 Type	biology.organism_classification
E_2 Description	Lineodes is a genus of moths of the Crambidae family.
Q_p	What is higher classification of Lineodes interrupta?
R_p	Lineodes.
Q_a	Biologically speaking, what is the classification?
Output: Are you referring to Lineodes interrupta or Lineodes, when you ask the biological classification?	

Table 9: An example of the generated clarification questions.

5.2 Discussion

Automatic Evaluation. In the task of clarification question generation, we use BLEU, a widely used evaluation metric on language generation tasks, as the automatic evaluation metric. BLEU evaluates the generated text by computing its similarity with the gold text. However, we find this metric not suitable for our task. A good answer can ask different aspects as long as it can distinguish the entities. The given answer in our dataset is one of possible answers but not the only correct answer. A good answer may get a low BLEU score. Therefore, a more reasonable evaluation metric needs to be explored in the future, like paraphrase-based BLEU.

Error Analysis. In clarification question generation, although our proposed model has achieved the best performance, it still generates some low-quality questions. We conduct a human evaluation on 100 generated clarification questions and summarize two error categories. The first one is the entity error where the generated clarification question has a correct multi-choice structure but with

Model	Single-Turn	Multi-Turn
CNN	**60.8**	74.7
RNN	56.4	**76.0**
Transformer	52.1	58.9
DMN	50.9	54.5
HAN	51.7	54.7

Table 10: Results of clarification-based question answering models.

E_1 Name	**Come Back, Little Sheba**
E_1 Type	award.winning_work award.nominated_work ... ...
E_1 Description	Come Back, Little Sheba is a 1950 theater production of the play by William Inge... ...
E_2 Name	**Come Back, Little Sheba**
E_2 Type	theater.production award.nominated_work
E_2 Description	Come Back, Little Sheba is a 2008 theater production of the play by William Inge.... ...
Q_h: Who directed Come Back, Little Sheba? **Output**: Which one do you mean, winning work visual artist Claudia Coleman or winning work Elmer Gantry, when you ask the directors?	

Table 11: A bad generated case.

irrelevant entity information. Here we present one example in Table 11. In this example, our model generates entity phrases irrelevant to the input ambiguous question. This failure is mainly due to the unseen entity name "Come Back, Little Sheba", which leads the model to wrong generation decisions. Since there are lots of entities with different names and descriptions, how to handle the sparse information is a non-trivial problem. The second one is the grammar error where models sometimes generate low-quality phrases after generating a low-frequency entity word. These errors both can be attributed to the sparsity of entities to some extent. How to deal with the sparsity of entity information still needs further exploration.

6 Related Work

Our work is related to clarification question and question generation.

6.1 Clarification Question in Other Tasks

There are several studies on asking clarification questions. Stoyanchev et al. (2014) randomly drop one phrase from a question and require annotators to ask a clarification question toward the dropped information, e.g.*"Do you know the birth data of XXX"*. However, the small dataset size makes it hard to help downstream tasks. Following this work, Guo et al. (2017) provide a larger synthetic dataset QRAQ by replacing some of entities with variables. Li et al. (2017) use misspelled words to replace entities in questions to build ambiguous questions. Although these studies are good pioneering studies, the synthetic constructing way makes them unnatural and far from real world questions.

Different from these studies, Braslavski et al. (2017) investigate a community question answering (CQA) dataset and study how to predict the specific subject of a clarification question. Similarly, Rao and III (2018) focus on learning to rank human-written clarification questions in an online QA forum, StackExchange. Our work differs from these two studies in that we have a knowledge graph at the backend, and the clarification-related components are able to facilitate KBQA.

6.2 Question Generation

For different purposes, there are various question generation tasks. Hu et al. (2018) aim to ask questions to play the 20 question game. Dhingra et al. (2017) teach models to ask questions to limit the number of answer candidates in task-oriented dialogues. Ke et al. (2018) train models to ask questions in open-domain conversational systems to better interact with people. Guo et al. (2018) develop a sequence-to-sequence model to generate natural language questions.

7 Conclusion and Future Work

In this work, we construct a clarification question dataset for KBQA. The dataset supports three tasks, clarification identification, clarification question generation, and clarification-based question answering. We implement representative neural networks as baselines for three tasks and propose a new generation model. The detailed analysis shows that our dataset brings new challenges. More powerful models and reasonable evaluation metrics need further explored.

In the future, we plan to improve our dataset by including more complex ambiguous situations for both single-turn and multi-turn questions, such as multi-hop questions, aggregation questions, etc. We also plan to integrate the clarification-based models into existing KBQA system, and study how to iteratively improve the model based on human feedback.

Acknowledgments

We thank all reviewers for providing the thoughtful and constructive suggestions. This work was supported in part by National Natural Science Foundation of China (No. 61673028). Jingjing Xu and Xu Sun are the corresponding authors of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Pavel Braslavski, Denis Savenkov, Eugene Agichtein, and Alina Dubatovka. 2017. What do you mean exactly?: Analyzing clarification questions in cqa. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 345–348. ACM.

Marco De Boni and Suresh Manandhar. 2003. An analysis of clarification dialogue for question answering. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–55. Association for Computational Linguistics.

Marty J Demetras, Kathryn Nolan Post, and Catherine E Snow. 1986. Feedback to first language learners: The role of repetitions and clarification questions. *Journal of child language*, 13(2):275–292.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. 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 484–495.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from sql queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1607. Association for Computational Linguistics.

Xiaoxiao Guo, Tim Klinger, Clemens Rosenbaum, Joseph P Bigus, Murray Campbell, Ban Kawas, Kartik Talamadupula, Gerry Tesauro, and Satinder Singh. 2017. Learning to query, reason, and answer questions on ambiguous texts. In *ICLR 2017*.

Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen. 2018. Playing 20 question game with policy-based reinforcement learning. *arXiv preprint arXiv:1808.07645*.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1499–1508.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Dialogue learning with human-in-the-loop. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2736–2745.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg. 2014. Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*, volume 20.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

A Supplementary Materials

A.1 Classification Models

Model	Hyper-parameter	Setting
CNN	Batch size	64
	Embedding size	128
	Hidden size	128
	Filter size	128
	Kernel size	100
	Learning rate	0.0003
	Optimizer	Adam
RNN	Batch size	8
	Embedding size	128
	Hidden size	128
	Learning rate	2
	Optimizer	Adam
Transformer	Batch size	8
	Embedding size	128
	Hidden size	128
	Learning rate	2
	Dropout	0.1
	Optimizer	Adam
HAN	Batch size	64
	Embedding size	100
	Hidden size	100
	Learning rate	0.01
	Optimizer	Adam
DMN	Batch size	64
	Embedding size	300
	Hidden size	300
	Learning rate	0.005
	Optimizer	Adam

Table 12: Settings of classification models.

Table 12 shows the detailed hyper-parameter settings of these classification models. Here we introduce the classification models used in our experiments.

CNN. It first feeds the input embedding sequence into a convolutional layer. The convolutionial layer contains K filters $f \in R^{s,d}$ where s is the size of filters and d is the dimension of word embeddings. We use different sizes of filters to get rich features. Then, a max pooling layer takes K vectors as input and generates a distributed input representation $\boldsymbol{h}$. Finally, the vector $\boldsymbol{h}$ is projected to the probability of labels.

RNN. It is built on a traditional bi-directional gated recurrent unit (GRU) structure, which is used to capture global and local dependencies inside the input sequence. The last output of GRU is then fed into the output layer to generate the probability of labels.

Transformer. A model similar with RNNs. It removes the recurrent unit in RNNs and replaces it with a special attention mechanism. called multi-head attention. The model is composed of a stack of 2 identical layers. Each identical layer has two sub-layers. The first sub-layer is a multi-head self-attention layer, and the second is a position-wise fully connected feed-forward layer. Since input nodes are not sequentially in order, the positional embedding layer is removed from the model. The decoder is also composed of a stack of 2 identical layers.

Model	Hyper-parameter	Setting
Seq2Seq	Batch size	16
	Embedding size	32
	Hidden size	32
	Learning rate	2
	Beam size	1
	Dropout	0.1
	Optimizer	Adam
Transformer	Batch size	16
	Embedding size	32
	Hidden size	32
	Learning rate	2
	Beam size	1
	Dropout	0.1
	Optimizer	Adam
Our Model	Batch size	16
	Embedding size	32
	Hidden size	32
	Learning rate	2
	Beam size	1
	Dropout	0.1
	Optimizer	Adam

Table 13: Settings of generation models.

HAN. It is built upon two self-attention layers. The first layer is responsible for encoding context information and entity information. Then, the second layer uses a self-attention mechanism to learn the inter-relations between them. Finally, the output of the second layer is fed into the output layer for predicting labels.

DMN. This model regards context information as query and entity information as input to learn their inter-relations. It consists of four modules, an input module, a question module, an episodic memory module, and an answer module. The input module and thed question module encode query and input into distributed vector representations. The episodic memory module decodes which parts of the entity information to focus on through the attention mechanism. Finally, the answer module generates an answer based on the final memory vector of the memory module.

A.2 Clarification Question Generation Models

Here is the detailed introduction about the generation baselines.

Seq2Seq (Bahdanau et al., 2015). This model is based on a traditional encoder-decoder framework which encodes the input sequence into a dense vector and then decodes the target sequence word by word. Attention is used to capture global dependencies between input and output.

Transformer (Vaswani et al., 2017). It is based solely on attention mechanism to capture global dependencies. Similar to Seq2Seq, it uses an encoder-decoder framework but with different implementation.

Table 13 shows the detailed hyper-parameter settings of generation models.