Diverse and Specific Clarification Question Generation with Keywords

Zhiling Zhang
Shanghai Jiao Tong University
Shanghai, China
blmoistawinde@sjtu.edu.cn

Kenny Q. Zhu*
Shanghai Jiao Tong University
Shanghai, China
kzhu@cs.sjtu.edu.cn

ABSTRACT

Product descriptions on e-commerce websites often suffer from missing important aspects. *Clarification question generation* (CQGen) can be a promising approach to help alleviate the problem. Unlike traditional QGen assuming the existence of answers in the context and generating questions accordingly, CQGen mimics user behaviors of asking for unstated information. The generated CQs can serve as a sanity check or proofreading to help e-commerce merchant to identify potential missing information before advertising their product, and improve consumer experience consequently. Due to the variety of possible user backgrounds and use cases, the information need can be quite diverse but also specific to a detailed topic, while previous works assume generating one CQ per context and the results tend to be generic. We thus propose the task of *Diverse CQGen* and also tackle the challenge of specificity. We propose a new model named *KPCNet*, which generates CQs with Keyword Prediction and Conditioning, to deal with the tasks. Automatic and human evaluation on 2 datasets (`Home & Kitchen`, `Office`) showed that KPCNet can generate more specific questions and promote better group-level diversity than several competing baselines. [1]

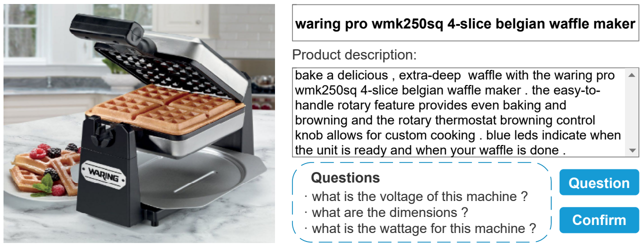
KEYWORDS

clarification question, text generation, diverse generation, keyword prediction, e-commerce

ACM Reference Format:
Zhiling Zhang and Kenny Q. Zhu. 2021. Diverse and Specific Clarification Question Generation with Keywords. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3449876

1 INTRODUCTION

The development of the Internet has spawned a number of task-oriented writings, such as product descriptions on Amazon. However, since merchants cannot always have a thorough understanding of consumers' need due to the variety of possible user backgrounds and use cases, their writings usually miss something deemed important by the customers. For example, a US merchant may assume his device be used on a 110V power line, and thus omit this in the product description. Customers from Asia and Europe, where 220V is used, might pay special attention to the voltage requirements in the description. On finding this absent from the description, some customers may ask CQs like "What is the voltage of this machine?" in customer QA, while others would turn to other products immediately, an unfortunate loss to the seller. It would be helpful if the platform can provide a service to remind the merchants of those potentially missing needs with its broader knowledge.


waring pro wmk250sq 4-slice belgian waffle maker

Product description:

bake a delicious , extra-deep waffle with the waring pro wmk250sq 4-slice belgian waffle maker . the easy-to-handle rotary feature provides even baking and browning and the rotary thermostat browning control knob allows for custom cooking . blue leds indicate when the unit is ready and when your waffle is done .

Questions
· what is the voltage of this machine ?
· what are the dimensions ?
· what is the wattage for this machine ?

Figure 1: A hypothetical writing assistant generating CQs.

Clarification question[2] generation (CQGen), which mimics the user engagement by raising questions, can be a promising option for such a service. Before publishing their writings, authors may request for CQs from somewhere like the hypothetical writing assistant we illustrated in Figure 1, and supplement missing information accordingly. CQGen is a challenging task for the following reasons:

First, it requires the question to be *specific* while not being *repetitive* to existing context. Questions pertaining to smaller set of products are considered more specific. For example, the first question in Figure 1 is more specific than the second one because it applies to only electric appliances, while the second one applies almost to every product. In contrast to the traditional QGen task which is typically evaluated on the SQuAD 1.1 dataset [26] and derives the specificity from the knowledge of answer, CQGen doesn't expect the existence of answer in the context. Therefore, QGen algorithms which require the answer span and its position as input [32, 34, 35] do not apply here. Vanilla seq2seq model has been shown to generate highly generic questions by Rao and Daumé III [29]. They then proposed GAN-Utility, which estimates the utility of answer with GAN as reward for RL to improve generation specificity. However, the answer used in the estimation is generated from the context and an already-generated question with another trained QA component, which may not be reliable here as the answers are inherently missing from context by definition. Consequently, this answer-based approach was shown to yield even worse results

*Corresponding author.

[1] Our code is available at https://github.com/blmoistawinde/KPCNet

[2] questions asking for what's missing from a given context

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449876

under some conditions [4]. We thus totally eliminate the need for answers in our work, which has the benefit of making use of more training data without answer.

Moreover, previous works on CQGen all assume generating one question per context. We claim that generating a group of diverse CQs (as is shown in Figure 1) can be more beneficial, because this allows the system to efficiently cover a variety of user needs at once, and tolerate occasional errors as the rest questions are still useful. We name this novel task as **Diverse CQGen**. We seek algorithms that can deal with the task, and adopt a new group-level evaluation protocol to properly evaluate the effectiveness of algorithms under this scenario.

To deal with the specificity challenge, we propose a novel model named Keyword Prediction and Conditioning Network (KPCNet). Keywords in CQs is one kind of prior knowledge that the platform can mine about user needs. They are usually product attributes or closely related concepts that make the questions specific, and thus the main semantic of a question can be captured by its keywords. For example, the keyword of "What's the *dimension*?" is "*dimension*", and the question can be comprehended even with a single word ("*dimension*?"). We can generate more detailed question like "Can you cook *rice* in this *cooker*?" with keywords "*cooker, rice*". Therefore, the proposed KPCNet first predicts the probability for a keyword to appear in the generated CQ, then selects keywords from the predicted distribution, and finally conditions on them to generate questions. We can also partially control the generation by operating on the conditioned keywords, which can be utilized to avoid repetitive questions and further improve the quality.

To promote diversity, we explore several diverse generation approaches for this problem, including model-based *Mixture of Experts* [30] and decoding-based *Diverse Beam Search* [36].

KPCNet's controllability through keywords, on the other hand, enables keywords-based approaches. We explore a novel use of classic clustering method on producing coherent keyword groups for keyword selection to generate correct, specific and diverse questions.

Individual and group-level evaluation showed that KPCNet is capable of producing more diverse and specific questions than strong baselines. Our contributions are:

(1) To our best knowledge, we are the first to propose the task of *Diverse CQGen*, which requires generating a group of diverse CQs per context, to cover a wide range of information needs. (§1)
(2) We propose KPCNet, which first predicts keywords that focus on the specific aspects of the question, before generating the whole question to promote specificity. (§2.2, §3.1)
(3) Based on KPCNet's keyword conditioned generation, we propose keyword selection methods to produce multiple keywords groups for generation diversity. (§2.3, §3.2, §3.3)
(4) We show with probing tests that KPCNet can be further enhanced with external knowledge to alleviate the problem of asking existing information in the context, an under-explored yet fundamental problem in CQGen, and improve generation quality. (§3.4)

2 PRELIMINARIES

2.1 Keyword-based Diverse CQGen

Given a textual *context* $\mathbf{x} = (x_1, x_2, ..., x_{T_1})$, our aim is to generate a *clarification question* $\mathbf{y} = (y_1, y_2, ..., y_{T_2})$, so that y asks for relevant but not repetitive information to $\mathbf{x}$. In the setting of *Diverse CQGen*, we should generate a group of CQs for the same context such that they are semantically different from each other. In this work, we additionally consider *keywords* $\mathbf{z} = (z_1, z_2, ..., z_k)$ that are expected to capture the main semantic of $\mathbf{y}$. The definition of keywords may vary across domains, and here for e-commerce, we empirically define keywords as lemmatized, non-stopping nouns, verbs and adjectives appearing in *questions*, according to our observations on specificity (§2.2). Note that keywords are different from *answers*, and we don't assume the existence of an answer in our approach. We extract ground truth keywords and a keyword dictionary Z of size C from the CQs in the training set using this definition.

With keywords introduced, the marginal likelihood of a question are decomposed as:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \sum_{\mathbf{z} \subseteq Z} p(\mathbf{y}, \mathbf{z}|\mathbf{x}) \\ &= \sum_{\mathbf{z} \subseteq Z} p(\mathbf{y}|\mathbf{x}, \mathbf{z}) p(\mathbf{z}|\mathbf{x}) \end{aligned} \tag{1}$$

where $p(\mathbf{z}|\mathbf{x})$ corresponds to the keyword prediction part, and $p(\mathbf{y}|\mathbf{x}, \mathbf{z})$ refers to the keyword conditioned generation. The range of $\mathbf{z} \subseteq Z$ is very large, so in practice, we sample portions of them to get an approximation, as will be discussed later (§3.3).

2.2 Specificity

In this work, the specificity of a question is determined by the size of its applicable range. Question that can only be raised against one particular context is considered more specific than universal questions. First, *relevance* of the question is the basic requirement of specificity. Traditional MLE training may generate generic but not relevant question for higher likelihood. We conjecture that the additional task of keyword prediction will help focus on relevant aspects. Moreover, by observation, we discover that *specificity* of e-commerce questions can be further promoted by:

(1) Focusing on certain aspects, like the type, brand and attributes.
(2) Mentioning components of a product, e.g. blade of a food grinder.
(3) Describing a using experience specific to the product, such as cooking rice in a cooker.

We hypothesize that many of them can be captured by keywords, with nouns and adjectives covering aspects and components, and verbs constituting the using experience.

2.3 Diversity

Diverse CQGen requires a group of questions to be generated about the same context, to cover various information needs as well as improve the robustness to problematic generations. This setting differs from some previous literature [29, 37], where they generate only one response at a time, and *Diversity* is used to measure the expected variety among *all generated response*. We call it *global diversity*. Our setting is referred to as *local diversity*, measuring the diversity *within one usage*. This is also adopted by another line of literatures [30, 36]. If not specified, we mean *local diversity* by using *diversity*. Global diversity is also desired, as it increases the likelihood of the questions to be specific to various contexts.

To meet the diversity requirement as well as to promote specificity, we propose KPCNet below.

3 MODEL DESCRIPTION

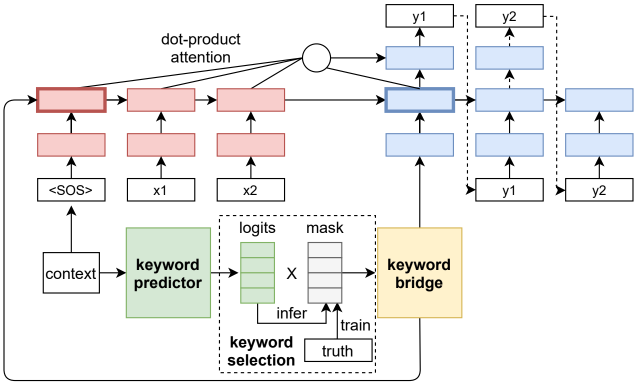


Figure 2: Illustration of KPCNet.

In Equation 1, $p(\mathbf{z}|\mathbf{x})$ corresponds to the keyword prediction part, and $p(\mathbf{y}|\mathbf{x}, \mathbf{z})$ refers to the keyword conditioned generation. Our model is thus divided into 2 logical parts. The whole model pipeline is illustrated in Figure 2.

3.1 Keyword Prediction

For the *Keyword Predictor*, we assume the probability of each keyword z are independent from each other given context $\mathbf{x}$, i.e. $p(\mathbf{z}|\mathbf{x}) = \Pi_{z \in \mathbf{z}} p(z|\mathbf{x})$, to simplify the modeling. We parameterize $p(z|\mathbf{x})$ with TextCNN [14]. The training loss is binary cross entropy over each keyword:

$$L_{pred} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} z_{n,c}^{t} log(p_{n,c}) \tag{2}$$

Here we use $z_{n,c}^{t}$ as a binary indicator that shows if c_{th} keyword in keyword dictionary Z is among the ground truth keywords of the n_{th} sample, and $p_{n,c}$ is the predicted probability for it.

3.2 Keyword Conditioned Generation

The main structure of our generator is based on a standard sequence-to-sequence model [19]. We will focus on our specific design to condition the generation on keywords.

Keyword Selection. We take the unnormalized keyword logits $\hat{p} \in \mathbb{R}^{C}$ from the keyword predictor, and then we select a conditioning keyword set $\mathbf{z}^s$ to mask out irrelevant dimensions to get a masked logits $\tilde{p} = [\hat{p}_1 z_1^s, \hat{p}_2 z_2^s, ..., \hat{p}_C z_C^s]$. This procedure allows us to control the generation with the selected keywords. Specific methods for this part will be discussed in §3.3.

Keyword Bridge. After getting the masked logits $\tilde{p}$, we pass them through a dropout layer, and then transform them to another distributed representation using a Multi-Layer Perceptron (MLP). They are then transformed into encoder features and decoder features with 2 MLPs respectively. The encoder feature will replace the hidden state of the first encoder step as memory to guide the generation via attention. The decoder feature will be fed as the input word embedding of the first decoder step to influence the generation.

3.3 Keyword Selection

At training, the ground truth keywords set $\mathbf{z}^t$ is selected as $\mathbf{z}^s$, and the training objective is to maximize the log-likelihood of all questions given context $\mathbf{x}$ and keywords $\mathbf{z}^t$. This equals to minimize:

$$L_{mle} = -\frac{1}{N}\sum_{n=1}^{N} log(p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{z}_n^t)) \tag{3}$$

At inference, we select $\mathbf{z}^s$ from keyword predictor's predicted distribution as condition for generation. This process was done once at a time, and can be done several times to fully explore the diversity in $p(\mathbf{y}|\mathbf{x})$ (Equation 1) with different keyword sets. We come up with 3 methods for keyword selection:

Threshold. We select all keywords whose predicted probability are above a threshold α as $\mathbf{z}^s$. If not specified, this is the default selection method at inference.

Sampling. The threshold selection approach is deterministic and thus limited to one conditioning keyword set. We may encourage more diverse generation via diversifying the keyword set. An intuitive solution is to introduce randomness. Inspired by the Top-K [8, 25] and Top-p (nucleus) sampling [10], we also adopted a similar approach, sampling k keywords from softmax-normalized prediction distribution after Top-K, Top-p filtering.

Clustering. Both the threshold and sampling selection strategies run the risk of putting semantically uncoherent keywords together, which is the drawback of the independence assumption used by keyword predictor. For example, if *"voltage, machine, long, waffle"* are selected as the keywords for the waffle maker in Figure 1, we may generate an illogical question "what are the *voltage* of the *waffle*". To get more coherent keyword sets, we explore the use of clustering technique. For the above example, the keywords can form 2 semantic groups, which lead to "What is the *voltage* of the *machine*" and "How *long* does it take to cook *waffle*", respectively.

In practice, we first mine a keyword co-occurrence graph from the training set. We then take the Top-K likely keywords, and run Spectral clustering [31] on the induced subgraph of them. The resulting g disjoint groups are then used as generation conditions respectively.

3.4 Keyword Controllability Probing

One potential benefit that KPCNet brings is the controllability over generation by providing different conditioning keywords. To probe into this, we propose 2 approaches to operate on the keywords besides the 3 keywords selection methods. The operations are designed with hypotheses that will be tested with experiments.

Product	iliving organic buckwheat pillow with authentic japanese pillow cover, 14 by 20-inch, green
KPCNet	what is the **size** of this **pillow** case? (size, cover, pillow, wash, zipper)
+Filter	does this **pillow** have a **zipper**? (cover, pillow, wash, zipper)

Table 1: Example on the effect of keyword filtering. Predicted keywords for a question are shown in the parentheses below. "size" was filtered as it has already been covered in product description.

3.4.1 *Keyword Filtering.*

Asking only about things **not** in the context is the basic requirement of CQGen. However, none of existing methods in the literature have specific solution for this. In preliminary experiments of KPCNet, we found that some of the repetitive cases came with repetitive keywords. Therefore, we conjecture that we may alleviate the problem by filtering out such repetitive keywords. Table 1 provided a concrete example. This would be especially useful for iterative generation, as we will explicitly exclude repeating keywords if user triggers CQGen for the second time with some information vacancy already filled.

Here we use a simple matching method for keyword filtering. We first select a set of keywords that tends to lead to repetition. Then for each keyword in the set, we maintain a blacklist of words or patterns so that we filter the keyword if the pattern is matched. For example, we would filter words like "height"/"width" from the predicted keywords, if we can match "height"/"width" in the context. This process is currently done manually, so it doesn't scale. However, we find that a small set of frequent keywords is already enough to cover a relatively large number of repetitive cases and demonstrate the effect of this approach, as will be shown in §4. We leave automatic repeating keyword detection and filtering for future works.

3.4.2 *External Knowledge.*

It is a common practice for e-commerce platforms to build knowledge graph to manage their products [7, 18]. As a result, products are attached to highly related tags, concepts, or keywords in our terms. Since the keywords used here is just a simple kind of knowledge, we believe that such richer external knowledge may further improve the generation by directly providing high-quality keywords, or helping the keyword prediction. Nevertheless, since we don't have access to such knowledge, we simulate such scenario where we have higher quality keywords by directly feeding ground truth keywords to the model[KPCNet(truth)]. This establishes an upper bound to what extent can KPCNet be improved with knowledge.

3.5 Deduplication Postprocessing

All algorithms will more or less produce semantically similar questions in their initial generation group. Therefore, we will first generate more candidates than needed (say, produce 6 questions for 3 displaying slots), so that at least certain level of diversity can be guaranteed for the initial group. We then apply a simple, model-agnostic heuristic for deduplicating question selection. We first add the top generation into the current group, then we will iterative through the remaining questions. If the question's Jaccard similarity with any currently selected question is below 0.5, it will be added into the current group, otherwise it will be discarded.

4 EXPERIMENTS

In this section we try to answer the following research questions:

(1) Can KPCNet generate more specific CQs than previous baselines?
(2) To what extent can we control the generation of KPCNet by operating on the keywords with methods like keyword selection and filtering (§3.3, §3.4) ?
(3) How well can our proposed keyword selection methods promote local diversity, compared to existing diverse generation approaches?

4.1 Evaluation metrics

Most previous works on question generation [11, 13, 29] adopts *Individual-level* evaluation protocol, where only the best generated question of a group is evaluated (thus also named *Oracle* metrics). Specially, for proper evaluation of the novel *Diverse CQGen* task, we need to evaluate the overall quality and diversity of CQ groups. We refer to this as *Group-level* evaluation. We adopt automatic metrics as well as human judgements on both level.

4.1.1 *Automatic Metrics.*

We use **Distinct-3** (DIVERSITY), **BLEU**[3] [23] and **METEOR** [3] for individual-level automatic evaluation. For group-level evaluation, we adopt the evaluation protocol proposed by Shen et al. [30] for diverse machine translation, and use **Pairwise-BLEU** and **Avg BLEU** as the evaluation metric. We report them in percentage.

4.1.2 *Human Judgements.*

For individual-level human judgements, we show every annotator one context and one generated question for each system (including reference). The system name is invisible to the annotator and the order is randomly shuffled. The selected candidate is the one that achieved the highest BLEU in the generation group. We ask human to judge the **Grammaticality(G), Relevance(R), Seeking New Information(N) and Specificity(S)** of the questions. Also, noting that the system generations are also prone to make logical errors like improper repetition ("does the lid have a lid ?") or asking for relevant but not exactly the correct object (asking "what is the thickness of the bed ?" for a mattress), we further judge the **Logicality(L)** of the candidate. Futher descriptions of these metrics can be found in Appendix B.

For group-level human judgements, we run the deduplication procedure (§3.5) to get 3 top questions for each system. And annotators are showed one context and the 3 selected questions for each group. The groups are also anonymized and shuffled.

For each question in a group, we score the same metrics as those for individual-level judgements. To evaluate the valid variety of each group produced by local generation diversity, we introduced an additional and important group-specific metric: **#Useful**. This is the number of useful questions after excluding problematic (ungrammatical, irrelevant, illogical, etc.) and semantically equivalent questions within a group. And we further calculate **#Redundant**

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

as (the number of unproblematic questions - **#Useful**) to measure local redundancy.

Individual-level and group-level evaluation was conducted on the same set of 100 sample products for 8 systems and every group has 3 questions. They are distributed to 4 annotators so that each of the 2400 questions are annotated twice. We report inter-annotator agreement in Appendix B.

4.2 Dataset

We evaluate our model on the Home & Kitchen category of the Amazon Dataset [20, 21] preprocessed by Rao and Daumé III [29]. We apply extra preprocessing on the raw data to remove noises in dataset (see Appendix A). In this dataset, *context* is the product title concatenated with the product description, and *question* is the CQ asked by customers to the product. It consists of 19,119 training, 2,435 validation and 2,305 test examples (product descriptions), with 3 to 10 questions (average: 7) per description. The inherent diversity of questions in the dataset allows the proper evaluation of group-level generation diversity. We process another category, Office, in a similar way. Office is a much smaller dataset, consisting of 2,190 training, 285 validation and 256 test examples, with 3 to 10 questions (average: 6) per description. We will first analyze the results on Home & Kitchen in detail, then briefly discuss the results on Office.

4.3 Baselines

For individual-level generation, we compare KPCNet with the following models:

MLE. Vanilla seq2seq model trained on (context, question) pairs using maximum likelihood objective.

hMup. A representative of the family of mixture models proposed by Shen et al. [30], which achieved a good balance of overall quality and diversity.

Since we don't assume the availability of answers, we don't include traditional QGen methods and GAN-Utility [29] in the comparison. For a fair comparison, we control the encoder and decoder for all the above methods to have a similar 2-layer GRU [5] or LSTM [9] architecture and close amount of parameters.

For group-level generation, we compare across 3 categories of diverse generation methods:

Decoding based. Classical beam search naturally produces different generation on each beam. Therefore, we evaluate the effect of beam search combined with MLE and KPCNet with threshold selection [KPCNet(beam)]. Recently, several decoding approaches [12] are proposed to further promote diversity in generation, among which *Diverse Beam Search*[36] and *Biased Sampling* like top-K, top-p sampling [8, 10] are representative methods. So we also evaluate KPCNet with them [KPCNet(divbeam), KPCNet(BSP)].

Model based. hMup is designed for diversity at the model level. It provides a discrete latent variable called *expert* to control the generation. We thus take the top beam-searched candidate of each expert to form a generation group for evaluation.

Keywords based. This is dedicated to KPCNet. We evaluate the *Sampling*[KPCNet(sample)] and *Clustering*[KPCNet(cluster)] methods for keyword selection. We also estimate the potential of KPCNet with knowledge (§3.4.2) by providing the ground truth keyword set [KPCNet(truth)].

All systems using beam search have a beam size of 6, we also set number of experts for hMup to 6, and we use beam size of 6 with 3 diverse groups for *diverse beam search*. We select 2 keyword groups for KPCNet(sample) and KPCNet(cluster). To produce the final generation group for evaluation, outputs of all systems will go through the same deduplication postprocessing (§3.5) to get 3 questions for each group.

4.4 Home & Kitchen Dataset Results

	Distinct-3	BLEU	METEOR
ref	69.34	-	-
MLE	7.77	**18.13**	14.86
hMup	11.11	17.76	15.40
KPCNet	**15.30**	17.77	**16.18**
KPCNet(truth)	37.38	23.63	19.38

Table 2: Individual-level automatic evaluation results on Home & Kitchen dataset.

4.4.1 Individual-level Evaluation.

Table 2 shows the automatic evaluation results. KPCNet and hMup outperform MLE in METEOR but not in BLEU. We claim that it is due to the shorter and the safer generation of MLE, which is naturally favored by precision-based BLEU but not F-based METEOR. The average generation length is 5.957 for MLE, 8.231 for hMup, and 7.263 for KPCNet. KPCNet significantly outperform all the other baselines in Distinct-3 and METEOR, showing that KPCNet potentially promote higher global diversity and generation quality. We note that KPCNet(truth) has a great advantage over KPCNet, indicating the controllability of keywords and the potential of KPCNet to be further strengthened by improving the conditioned keyword set with other helpers like external knowledge (§3.4.2).

	G	R	L	N	S
ref	0.98	1.00	1.00	0.94	2.68
MLE	**0.99**	0.92	**0.98**	**0.85**	1.45
hMup	**0.99**	0.92	0.86	0.81	1.81
KPCNet	**0.99**	**0.99**	0.95	0.80	1.81
KPCNet(filter)	**0.99**	**0.99**	0.94	**0.85**	**1.84**

Table 3: Individual-level human evaluation metrics on 100 sample products from Home & Kitchen. G/R/L/N/S stand for Grammaticality, Relevance, Logicality, New Info and Specificity respectively.

Table 3 shows the individual-level human evaluation results. We can see that all systems perform well in *Grammaticality*, KPCNet significantly outperforms other systems in *Relevance* and achieved the best *Specificity*, while performs slightly worse in *Logicality*. The

	Relevant$_{[0-1]}$	Logical$_{[0-1]}$	New Info$_{[0-1]}$	Specific$_{[0-4]}$	#Useful$_{[0-3]}$	#Redundant$_{[0-2]}$	Avg Rank
ref	0.990	1.000	0.947	2.530	2.680	0.120	-
MLE	0.907	**0.943**	**0.863**	1.457	1.550	0.590	3.667
hMup	0.900	0.793	0.833	1.727	1.530	**0.130**	4.667
KPCNet(-filter)	$\underline{\textbf{0.987}}$	$\underline{0.870}$	$\underline{0.830}$	1.757	1.280	0.750	4.500
KPCNet(beam)	$\underline{\textbf{0.987}}$	$\underline{0.853}$	$\underline{\textbf{0.863}}$	1.793	1.330	0.750	3.667
KPCNet(divbeam)	$\underline{0.963}$	0.780	$\underline{0.860}$	1.760	1.480	$\underline{0.310}$	4.167
KPCNet(sample)	$\underline{0.963}$	$\underline{0.837}$	$\underline{0.850}$	$\underline{\textbf{1.890}}$	1.500	0.450	3.500
KPCNet(cluster)	$\underline{0.963}$	$\underline{0.863}$	$\underline{0.823}$	$\underline{1.877}$	$\underline{\textbf{1.760}}$	$\underline{0.190}$	**3.000**

Table 4: Group-level human evaluation results on 100 sample products (300 questions each system) from `Home & Kitchen`. Grammaticality is omitted as the results are similar to Table 3 where all systems performs well. *Avg Rank* is the average ranking among all 7 methods across the 6 metrics. We perform hypothesis test among KPCNet variants, and the difference between underlined and non-underlined numbers at each column is statistically significant with $p \leq 0.05$.

superior *Relevance* score validates our hypothesis that independently trained keyword predictor help focus on relevant keywords instead of irrelevant but generic words preferred by MLE (§2.2). KPCNet(filter) gets a much higher *New Info* at the cost of only slight drop in *Logicality*. It shows that the Keyword Filtering step (§3.4.1) can truly utilize the controllability of keywords to help avoid repetition on the basis of KPCNet. Therefore, we by default incorporate the step with all the KPCNet variants in the next group-level evaluation stage while keeping the vanilla KPCNet for comparison as KPCNet(-filter).

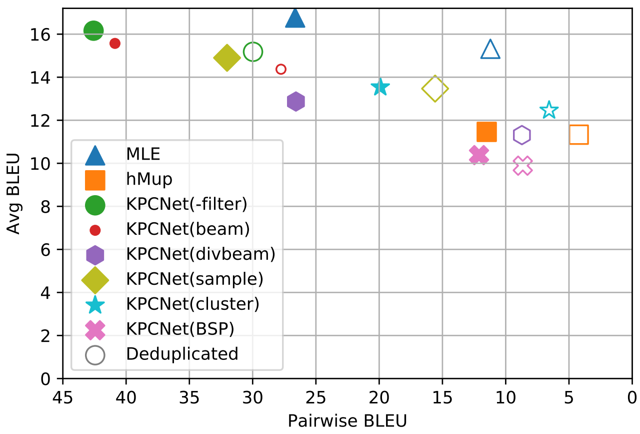


Figure 3: Group-level Automatic metrics on the whole test set of `Home & Kitchen`. The lower Pairwise BLEU, the more diverse the generated group. Solid markers are the results for the top 3 candidates in the original group, while hollow markers measures the remaining 3 after deduplication. Points located near top-right are preferred as they achieve a good tradeoff between the 2 metrics.

4.4.2 Group-level Evaluation. The group-level automatic evaluation metrics before and after deduplication for each system are shown in Figure 3. Original results are shown in solid markers. KPCNet(BSP) has the poorest Avg BLEU and we found the results very likely to be ungrammatical and illogical, and we thus omit it in the following evaluation. hMup has the highest local diversity while has the second poorest Avg BLEU. MLE has moderate level of local diversity and the highest Avg BLEU, and we found that Keyword Filtering slightly harmed Avg BLEU, which is against our intuition. But we later found Avg BLEU doesn't correlate well with most human judgements (discuss later). Several diversity-promoting variants of KPCNet improved local diversity at the cost of Avg BLEU, among which KPCNet(cluster) achieved a best tradeoff between the two. Comparing the original and deduplicated results (hollow markers), we can see that our simple heuristic can effectively eliminate redundancy at the cost of slight degradation of Avg BLEU, as only nearly identical hypotheses with high BLEU are excluded.

Group-level human evaluation results are shown in Table 4. We can see that all KPCNet variants clearly outperform baselines in *Relevant* and *Specific* while have a competitive performance in *New info*. MLE rated best for *Logical* for its conservative generations (low *Specific*), and the questions tend to overlap with each other, as is reflected in high *#Redundant*. KPCNet(beam) has a even higher redundancy since its searching space is further limited by the conditioned keyword set. Diverse generation variants can help overcome this drawback. Especially, KPCNet(cluster) achieved the best *#Useful*, *Avg Rank*, and its performance on all metrics is among the best of KPCNet variants. This shows that the semantically-coherent keyword sets produced by clustering can effectively improve the generation diversity and quality of KPCNet.

We also study the system-level Pearson correlation between the automatic metrics and human judgements. Pairwise-BLEU has a correlation of 0.915 with *#Redundant* ($p < 0.01$), -0.835 with *#Useful* ($p < 0.05$). Avg BLEU is shown only correlates well with *Logical* (correlation: 0.849, $p < 0.05$). This result validates the use of Pairwise-BLEU as an automatic proxy metric for local diversity.

4.4.3 Case Study. Table 6 provides 2 example generation groups of KPCNet(cluster). For each group, the 6 predicted keywords captured specific aspects of the product. Then they are divided into 2 coherent groups (as they formed natural phrases such as "firm pillow" and "stomach sleeper") by clustering. Finally, the different conditioned keyword sets are reflected in the generation. In the first case, specific and diverse generations are successfully produced with precisely predicted keywords. We can see that the separation of keywords as controlling factors allows the novel use of classical clustering technique to help generate high-quality question groups by first producing coherent keyword sets. There are also bad cases like

product	homelegance 2588s accent dining chair, blue grey, set of 2		
system (#Useful)	generation group	specific	problem
ref (3)	can any of the recent reviewers confirm the seat height ? i see the question was posted in april ...	2	
	would u please send me the box dimensions (when buy in a set of 2) and the weight ?	3	
	can someone please tell me the depth of the chair seat from the end of the curved back to the end of the seat ?	3	
MLE (1)	what is the seat height ?	2	
	what are the dimensions of the chair ?	2	
	what are the dimensions ?	1	
hMup (1)	what is the weight limit for the chair ?	2	
	i have a table that is a [UNK]. will this chair be able to fit on a table ?	2	illogical
	is this a set of 2 chairs or just one ?	2	repetitive
KPCNet (2)	what is the **color** of the **chair** ?	2	repetitive
	what are the **dimensions** of the **seat** ?	2	
	what is the **weight** limit ?	2	

Table 5: Example generation group and the human judgements for each system. Here we use KPCNet to stand for KPC-Net(cluster) for brevity, and the appeared keywords of KPCNet are in bold.

Product	Novaform memory foam comfort curve pillow
KPCNet (cluster)	is this a **firm pillow**? (pillow, foam, sleep, firm) is this pillow good for **stomach sleepers**? (stomach, sleeper)
Product	full-sized headboard in solid wood
KPCNet (cluster)	what is the height of this **headboard** ? (bed frame headboard) does it have a **box spring** ? (mattress box spring)

Table 6: Example generation groups for KPCNet(cluster). Keywords in the parentheses.

the second question in another group. The possible reason is that keyword predictor produced related but unsuitable keywords "box spring", which can be asked for a whole bed but not for headboard alone. This shows that predictor is the performance bottleneck of KPCNet.

We provide a group-level evaluation example in Table 5. We can see that the diversity of MLE is very limited (it gets *#Useful* of only 1, though all 3 questions are valid, and thus *#Redundant* is 2), and it produces highly generic question. The generations are more diverse for hMup. However, we find that a certain expert of hMup has a style of long and illogical generation, like the second one demonstrated here. (It's abnormal to put chairs *on* a table, and the text is not coherent as it doesn't use a pronoun in the second sentence.) This may attribute to its focus on *style* instead of aspects of the products, as it is originally proposed for translation of diverse styles. This significantly harms hMup's group-level performance (Table 4) compared to its best single model (Table 3). KPCNet(cluster) produces a diverse and specific generation, and we can clearly see the effect of keyword in its generation.

4.5 `Office` Dataset Results

For brevity, we only show the individual-level automatic evaluation and group-level human judgement results. All the experimental settings are the same with the previous experiments, except that we apply no keyword filtering here.

	Distinct-3	BLEU	METEOR
ref	75.54	-	-
MLE	20.33	**14.73**	13.81
hMup	15.31	10.45	12.52
KPCNet	**30.99**	13.84	**15.29**

Table 7: Individual-level automatic evaluation results on the `Office` dataset.

Table 7 shows that KPCNet still outperforms MLE in Distinct-3 and METEOR, while falls behind at BLEU. Both the automatic metrics and our manual check indicate that hMup fails to give comparable results for the small dataset, so we exclude it in group-level evaluation.

Table 8 shows that the performance of both models degraded here possibly due to the smaller data size. However, the observation is similar. KPCNet(cluster) outperforms MLE in most metrics especially at *Relevant*, *Specific* and *#Useful* despite a weakness at *Logical*. This shows that KPCNet(cluster) can consistently improve the diversity and specificity of the generation.

5 RELATED WORK

Clarification Question Generation. The concept of CQ can be naturally raised in a dialogue system where the speech recognition results tend to be erroneous so that we raise CQs for sanity check [33], or the intents for a task is incomplete or ambiguous in a first short utterance and further CQs are needed to fill in the slots [6].

	Grammatical$_{[0-1]}$	Relevant$_{[0-1]}$	Logical$_{[0-1]}$	New Info$_{[0-1]}$	Specific$_{[0-4]}$	#Useful$_{[0-3]}$	#Redundant$_{[0-2]}$
ref	0.993	0.997	0.993	0.933	2.713	2.420	0.330
MLE	0.970	0.843	**0.883**	0.797	1.470	1.070	0.420
KPCNet	**0.993**	**0.940**	0.817	**0.803**	**1.903**	**1.470**	**0.190**

Table 8: Group-level human judgments on 100 samples from the `Office` dataset. KPCNet here uses keyword clustering.

The concept is then extended to IR to clarify ambiguous queries [2], and has been successfully put into practice [39]. Other application areas including KBQA [38] and open-domain dialogue systems [1]. CQGen can also be applied to help refine posts on websites like StackExchange [15] and Amazon [29]. In this context, our work closely follows the research line of [4, 28, 29]. Rao and Daumé III [28] first adopted a retrieval-then-rank approach. They [29] then proposed a generation approach to train the model to maximize the utility of the hypothetical answer for the questions with GAN, to better promote specificity. Cao et al. [4] propose to control the specificity by training on data with explicit indicator of specificity, but it requires additional specificity annotation. Towards the similar specificity goal, we adopted a different keyword-based approach. They also assume generating one question per context, which we claim is not sufficient to cover various possible information needs, and thus propose the task of the diverse CQGen.

Diverse Generation. The demand for diverse generation exists in many other fields [17, 30, 36], and we've drawn inspirations from these literatures. For image captioning, we may use multiple descriptions for different focusing points of a scene. *Diverse Beam Search* [36] was proposed to broaden the searching space to catch such diversity by dividing groups in decoding and imposing repetition penalty between them. For machine translation, a context can be translated with different styles. Shen et al. [30] thus proposed *Mixture of Expert* models including hMup to reflect various styles with a discrete latent variable (*expert*). And here for CQGen, diversity is required to cover various potentially missing aspects, so we come up with the idea to use keywords as a controlling variable like *expert* to promote diversity.

6 CONCLUSION

To tackle the problem of missing information in product descriptions on e-commerce websites, we propose the task of Diverse CQGen to request for various unstated aspects in the writing with a group of semantically different questions. We then propose KPCNet to deal with this novel task as well as improve the specificity of the questions with the prior knowledge on user needs in the form of keywords. Human judgements showed that KPCNet is able to generate more specific questions and promote better group-level diversity. Oracle tests with ground truth keywords provided in keyword selection showed strong performance, indicating the great potential to be exploited from improving keyword prediction possibly with external knowledge. Future works may include utilizing richer external knowledge to improve the keyword prediction, and solutions for the occasionally illogical generations. We also believe that our approach can be applied to other scenarios with slight domain-specific modifications on the utilized knowledge.

ACKNOWLEDGMENTS

This work was partly supported by the SJTU-CMBCC Joint Research Scheme and SJTU Medicine-Engineering Cross-disciplinary Research Scheme.

REFERENCES

[1] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2020. ConvAI3: Generating Clarifying Questions for Open-Domain Dialogue Systems (ClariQ). *arXiv preprint arXiv:2009.11352* (2020).

[2] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). 475–484.

[3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization.* 65–72.

[4] Yang Trista Cao, Sudha Rao, and Hal Daumé III. 2019. Controlling the Specificity of Clarification Question Generation. In *Proceedings of the 2019 Workshop on Widening NLP.* 53–56.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 1724–1734.

[6] Kaustubh D Dhole. 2020. Resolving Intent Ambiguities by Retrieving Discriminative Clarifying Questions. *arXiv preprint arXiv:2008.07559* (2020).

[7] Xin Luna Dong. 2018. Challenges and Innovations in Building a Product Knowledge Graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). 2869.

[8] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 889–898.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

[11] Wenpeng Hu, Bing Liu, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Aspect-based Question Generation. (2018).

[12] Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of Diverse Decoding Methods from Conditional Language Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* 3752–3762.

[13] Unnat Jain, Ziyu Zhang, and Alexander G. Schwing. 2017. Creativity: Generating Diverse Questions Using Variational Autoencoders. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* 5415–5424.

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 1746–1751.

[15] Vaibhav Kumar and Alan W Black. 2020. ClarQ: A large-scale and diverse dataset for Clarification Question Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* 7296–7301.

[16] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[17] Yuding Liang and Kenny Qili Zhu. 2018. Automatic Generation of Text Descriptive Comments for Code Blocks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* 5229–5236.

[18] Xusheng Luo, Luxin Liu, Yonghua Yang, Le Bo, Yuanpeng Cao, Jinghang Wu, Qiang Li, Keping Yang, and Kenny Q. Zhu. 2020. AliCoCo: Alibaba E-commerce

Cognitive Concept Net. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). 313–327.

[19] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

[20] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto (Eds.). 43–52.

[21] Julian J. McAuley and Alex Yang. 2016. Addressing Complex and Subjective Product-Related Queries with Customer Reviews. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao (Eds.). 625–635.

[22] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 3111–3119.

[23] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 311–318.

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. (2019).

[26] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2383–2392.

[27] Justus J Randolph. 2005. Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss' Fixed-Marginal Multirater Kappa. *Online submission* (2005).

[28] 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 (Volume 1: Long Papers)*. 2737–2746.

[29] Sudha Rao and Hal Daumé III. 2019. Answer-based Adversarial Training for Generating Clarification Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 143–155.

[30] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture Models for Diverse Machine Translation: Tricks of the Trade. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). 5719–5728.

[31] Jianbo Shi and Jitendra Malik. 2000. *Normalized cuts and image segmentation*. Technical Report.

[32] Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging Context Information for Natural Question Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 569–574.

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

[34] Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio. 2018. Neural Models for Key Phrase Extraction and Question Generation. In *Proceedings of the Workshop on Machine Reading for Question Answering*. 78–88.

[35] Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and Position-aware Neural Question Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3930–3939.

[36] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2018. Diverse Beam Search for Improved Description of Complex Scenes. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). 7371–7379.

[37] Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. 2018. Learning to Ask Questions in Open-domain Conversational Systems with Typed Decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2193–2203.

[38] Jingjing Xu, Yuechen Wang, Duyu Tang, Nan Duan, Pengcheng Yang, Qi Zeng, Ming Zhou, and Xu Sun. 2019. Asking Clarification Questions in Knowledge-Based Question Answering. 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)*. 1618–1629.

[39] Hamed Zamani, Susan T. Dumais, Nick Craswell, Paul N. Bennett, and Gord Lueck. 2020. Generating Clarifying Questions for Information Retrieval. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). 418–428.

A EXPERIMENTAL DETAILS

A.1 Data Cleaning

The following steps are enforced to remove noises as well as remove unhelpful parts for the CQGen task in the original data:

Fixing Unescaped HTML characters. We noticed that there are unescaped HTML special characters in both context and the question. (e.g. "does it slice like zucchini **& amp ;** cucumbers?" is changed to "does it slice like zucchini **&** cucumbers?")

Remove non-question parts. Sometimes there are declarative sentences following the question, which is not the focus of our task. We thus removed them. (e.g. where is this product made ? *i contacted customer service and the representative was uninformed and could not offer any information* .)

Remove noise questions. Some questions contain the comparison between 2 specific entities, which is unlikely to be tackled by our model, so we dropped them. And some questions are too universal ("Does it ship to Canada?"). We consider them as noise and also dropped them.

Note that the data cleaning was only imposed on the training set and the validation set. We preserve exactly the same test set as Rao and Daumé III [29] for fair comparison.

A.2 Hyperparameters and other settings

For all models, we set the max length of context to be 100, question to be 20. For all variants of KPCNet, we use 2-layer GRU [5] with 100 hidden units for both the encoder and decoder. We use a learning rate of 0.0003 to train at most 60 epochs. For MLE, the model structure and parameters are identical to KPCNet, and we follow the setting of Rao and Daumé III [29], using dropout=0.5, learning rate=0.0001 to train 100 epochs. To improve the generation quality, we block bigrams from appearing more than once, and also forbid 2 same words to appear within 3 steps. For sampling-based keyword selection, we sampled 3 keywords from top-K top-p filtered keywords distribution with $K = 6, p = 0.9$ for 2 times. For clustering-based keyword selection, we produce 2 clusters from the top 6 predicted keywords. For hMup, we use the implementation in fairseq[4]. The architecture is set to 2-layer LSTM [9] with 100 hidden units, and other settings are identical to KPCNet for fair comparison. The threshold α for the default keyword selection method of KPCNet is manually tuned within range [0.05, 0.1]. The dropout strength is shared among all components of KPCNet and is manually tuned within range [0.2, 0.5]. MLE and KPCNet is implemented in PyTorch. For all manually tuned hyperparameters, we

[4] https://github.com/pytorch/fairseq/blob/master/examples/translation_moe

fix all other hyperparameters and random search for value within given range that can achieve the best BLEU on our validation set. The models are trained on a Ubuntu 18.04.4 LTS server with one NVIDIA GeForce RTX 2080 Ti.

For `Home & Kitchen` dataset, all models are operated on 200D word embeddings borrowed from Rao and Daumé III [29], which are pretrained from in-domain data with Glove [24] and are frozen during training, except for hMup, which uses unique embedding to distinguish between experts and thus the embeddings are trained from scratch. The selected threshold α is 0.07, after 3 trials, and the selected dropout is 0.3 after 4 trials.

For `Office` dataset, all models are operated on 200D word embeddings that we pretrained from in-domain data with Word2vec[22] in gensim[5], except for hMup. The selected threshold α is 0.07, after 3 trials, and the dropout is initially selected as 0.3 based on the result of `Home & Kitchen`.

For hypothesis test in Table 4, we use `proportions_ztest` of scipy for the first 3 columns whose range is binary, and `ttest_rel` for the other 3 columns. The procedure we assign the underline are: First, we underline the best number at each column. Then we run hypothesis test against every other number. If the difference is not significant, we also underline it, otherwise we don't underline it.

B HUMAN JUDGEMENT DETAILS

B.1 Metric Descriptions

For human evaluation, we show each annotator a detailed annotation guideline with definitions and examples. Here we provide some brief explanations:

- Grammaticality=0, if there is syntax error, or the generation result is not a question
- Relevance=0, if the problem is not related to the context
- Logicality=0, if there is clear nonsense within the question it self (does the lid have a lid ?), or the question is not suitable for the context (asking "how many bottles does it hold ?" for a bottle).
- Seeking New Information=0, if the question is asking for information already contained in the context, like asking color for a product titled "blue chair".

For Specificity, we ask "How specific is the question?" and let annotators choose from:

- 4: Specific pretty much only to this product (or same product from different manufacturer)
- 3: Specific to this and other very similar products
- 2: Generic enough to be applicable to many other products of this type
- 1: Generic enough to be applicable to any product under this category (H&K or Office)
- 0: N/A (Not applicable) i.e. Question is ungrammatically, irrelevant or illogical

B.2 Inter-annotator Agreement

We report the inter-annotator agreement measured by Randolph's κ [27] in Table 9. It can be seen that *Grammatical* and *Relevant* have high agreement as they are easy to judge. *New Info* has lower agreement possibly because it is harder to decide. For the example in Table 5, the question "what is the color of the chair ?" may have not been annotated as repetitive as the word "color" doesn't appear in the context, though it is actually covered by the specific value "blue grey". *Logical* and *Specific* have the lowest degree of agreement as they are more subjective criteria. According to the table suggested by Landis and Koch [16], all the criteria achieved at least moderate agreement.

Criteria	Agreement
Grammatical$_{[0\text{-}1]}$	0.933
Relevant$_{[0\text{-}1]}$	0.853
Logical$_{[0\text{-}1]}$	0.659
New Info$_{[0\text{-}1]}$	0.701
Specific$_{[0\text{-}4]}$	0.546

Table 9: Inter-annotator Agreement measured by Randolph's κ [27]

C ABLATION TEST

Below we describe the ablation test to check the influence of the components and hyperparameters of the model. These tests are all conducted on the `Home & Kitchen` dataset.

C.1 Additional Metrics

To evaluate the quality of our keyword predictor and keyword bridge, we propose these additional automatic metrics:

P@5. Since the number of keywords in ground truth questions are different across each sample. We take the top 5 keywords with the highest predicted probability as selected keyword set $\mathbf{z}^s$, and calculates precision@5 by:

$$P@5 = \frac{|\mathbf{z}^s \cap \mathbf{z}^T|}{5} \tag{4}$$

where $\mathbf{z}^T$ is the union of keywords extracted from all ground truth questions of a sample.

Response Rate. which is the proportion of conditioned keywords that appears in the corresponding generation, and we report the macro average on all the records. We use this to evaluate the controllability of the keyword conditions.

We also report the average generation length(**Length**) as it is related to almost all metrics proposed above, but neither long or short generation should be considered an indicator of good performance.

	Distinct-3	BLEU	P@5	Response	Length
KPCNet(C, S)	15.30	17.77	0.47	0.40	7.26
-C	16.51	15.88	0.51	0.35	7.52
-S, +E	12.00	9.04	0.22	0.50	7.66
+H	29.97	12.85	0.47	0.66	9.17

Table 10: Ablation test results on `Home & Kitchen` for data and keyword predictor at individual-level. The first line is final adopted setting.

[5] https://radimrehurek.com/gensim/models/word2vec.html

	Distinct-3	BLEU	Response	length
Dropout = 0.2	17.29	17.11	0.45	7.16
Dropout = 0.3	15.30	17.77	0.40	7.26
Dropout = 0.4	13.02	18.33	0.35	6.95
Dropout = 0.4, NE	15.04	18.19	0.34	6.78
Dropout = 0.4, ND	12.19	17.47	0.32	6.53
Dropout = 0.5	11.77	18.53	0.32	6.66

Table 11: Ablation test results for keyword bridge at individual-level on `Home & Kitchen`.

	G	R	L	N	S
KPCNet	**0.99**	**0.99**	**0.95**	0.80	1.81
KPCNet(filter)	**0.99**	**0.99**	0.94	0.85	**1.84**
KPCNet	0.98	0.97	0.88	0.84	1.77
KPCNet(filter)	0.98	0.97	0.89	**0.88**	1.80

Table 12: Comparison between KPCNet with Dropout=0.3 (upper half) and Dropout=0.2 (lower half) with individual-level human judgements on 100 sample products from `Home & Kitchen`. G/R/L/N/S stand for Grammaticality, Relevance, Logicality, New Info and Specificity respectively.

C.2 Ablation Factors

These are many important factors and parameters in our model. So we divide the ablation test into 2 logical parts: one for keyword predictor (and the effect of data cleaning on it), and another for keyword bridge.

The ablation factors for keyword predictor are as follows (abbreviated for readability):

- **E**: End2end training of keyword predictor with other component. The training objective is a weighted sum of the 2 objectives (Equation 2 & 3).
- **S**: Separate training, first train predictor, and then freeze its parameters to train other parts.
- **H**: Hard label fed to bridge instead of masked soft logits. The label can be provided from ground truth in training and is decided with threshold filtering in inference. If this setting works well, we can then completely separate the parameters of predictor from other parts.
- **C**: Cleaned dataset.

The ablation factors for keyword bridge are:

- **NE**: No encoder feature fed back to encoder
- **ND**: No decoder feature fed to decoder
- **Dropout**: We add a dropout layer for the unmasked keywords logits before it passes the latter transformation. Due to the nature of dropout, this part may help ease the noise introduced by the error of keyword predictor. And we study the effect of the strength of this layer.

C.3 Results

The ablation test result for data and keyword predictor at individual-level is shown in Table 10. The setting for keyword bridge is fixed: dropout=0.3, both encoder and decoder feature are used. After data cleaning(C), *P@5* dropped because of the reduction of the number of ground-truth keywords. The decreasing of *Distinct-3* and *Length* shows the effect of irrelevant part removing. The improvement on *BLEU* and *Response* indicates the overall benefits brought by the cleaning. End2end training(-S, +E) leads to significant performance degradation on all metrics except slight increase on *Response*. The possible reason is that keyword prediction skews highly towards frequent keywords under this condition. Finally, feeding hard label instead of logits also produce worse result. We can see from the extremely high *Response* and *Length* that this setting suffers severely from over-generation of keywords: model generates illogical long questions to contain as much keywords as possible. We hypothesize that the soft logits can reflect subtle difference on the importance of each conditioned keyword and thus can lead to more robust performance. Moreover, we can achieve a *P@5* of 0.628 with one group of group truth keywords, as compared to 0.472 of the current model, which shows a huge room for improvement of the keyword predictor.

The ablation test result for keyword bridge at individual-level is shown in Table 11. The setting for keyword predictor is fixed as KPCNet(C, S). We can clearly witness the trend that the higher dropout, the higher controllability keywords will have over generation (Response). As a result, the behavior of KPCNet will be more and more like MLE when dropout grows, with lower generation length, lower keyword response and higher BLEU. We speculate that the dropout imposed on the keywords logits to be masked forces the model to make prediction with incomplete keyword set. Therefore, proper level of dropout can make the model robust to the noise introduced by keyword predictor. Furthermore, the ablation of either encoder bridge or decoder bridge would harm BLEU, response and length, which proved the effect of KPCNet's double-bridge design to guide the generation via attention between the two sides.

We also conducted human evaluation for different value of dropout (Table 12), and found that lower dropout trades logicality for new information. We selected Dropout=0.3 as the final setting for its good balance of all metrics.