Product Answer Generation from Heterogeneous Sources: A New Benchmark and Best Practices

Anonymous ACL submission

Abstract

It is of great value to answer product questions based on heterogeneous information sources available on web product pages, e.g., semi-structured attributes, text descriptions, user-provided contents, etc. However, these sources have different structures and writing styles, which poses challenges for (1) evidence ranking, (2) source selection, and (3) answer generation. In this paper, we build *a benchmark with annotations for both evidence selection and answer generation covering 6 information sources*. Based on this benchmark, we conduct a comprehensive study and present a set of best practices. We show that all sources are important and contribute to answering questions. Handling all sources within one single model can produce comparable confidence scores across sources and combining multiple sources for training always helps, even for sources with totally different structures. We further propose a novel data augmentation method to iteratively create training samples for answer generation, which achieves close-to-human performance with only a few thousand annotations. Finally, we perform an in-depth error analysis of model predictions and highlight the challenges for future research.

1 Introduction

Automatic answer generation for product-related questions is a hot topic in e-commerce applications. Previous approaches have leveraged information from sources like product specifications (Lai et al., 2018a, 2020), descriptions (Cui et al., 2017; Gao et al., 2019) or user reviews (McAuley and Yang, 2016; Yu et al., 2018; Zhang et al., 2019) to answer product questions. However, these works produce answers from only a single source. While a few works have utilized information from multiple sources (Cui et al., 2017; Gao et al., 2019; Feng et al., 2021), they lack a reliable benchmark and have to resort to noisy labels or small-scaled human evaluation (Zhang et al., 2020; Gao et al., 2021). Furthermore, none of the above make use of pretrained Transformer-based models, which are the current state-of-the-art (SOTA) across NLP tasks (Devlin et al., 2019; Clark et al., 2020).

In this work, we present a large-scale benchmark dataset for answering product questions from 6 heterogeneous sources and study best practices to overcome three major challenges: (1) evidence ranking, which finds most relevant information from each of the heterogeneous sources; (2) source selection, which chooses the most appropriate data source to answer each question; and (3) answer generation, which produces a fluent, natural-sounding answer based on the relevant information. It is necessary since the selected relevant information may not be written to naturally answer a question, and therefore not suitable for a conversational setting.

Most published research on product question answering is based on the AmazonQA dataset (McAuley and Yang, 2016), which takes the community question-answers (CQAs) as the ground truth. This leads to several problems. (1) CQAs, even the top-voted ones, are quite noisy. Many are generic answers or irrelevant jokes (Gao et al., 2021). (2) CQAs are based more on the opinion of the individual customer who wrote the answer rather than on accompanying sources such as product reviews and descriptions. As such, CQAs are not reliable references for judging the quality of answers generated from these sources (Gupta et al., 2019). (3) There are no annotations for assessing the relevance of the information across multiple data sources. This makes it difficult to evaluate the evidence ranker and generator separately. Some works collect annotations for evidence relevance, but only for a single source and with questions formulated post-hoc rather than naturally posed (Lai et al., 2018a; Xu et al., 2019). To address these shortcomings, we collect a benchmark dataset with the following features: (1) It provides clear annotations *for both evidence ranking and answer* *generation*, enabling us to perform in-depth evaluation of these two components separately. (2) We consider a *mix of 6 heterogeneous sources*, ranging from semi-structured specifications (jsons) to free sentences and (3) It represents *naturally-occurring questions*, unlike previous collections that elicited questions by showing answers explicitly.

As sources differ in their volume and contents, collecting training data covering all sources of natural questions and answers is challenging. To get enough positive training signals for each source, we propose filtering community questions based on the model score of a pretrained QA ranker. Questions are only passed for annotation when the confidence scores of top-1 evidence lie within some certain range. This greatly reduces annotation effort by removing most unanswerable questions.

After collecting the data, we apply SOTA Transformer-based models for evidence ranking and answer generation, and present a set of data augmentation and domain adaptation techniques to improve the performance. We show that pretraining the model on the AmazonQA corpus can provide a better initialization and improve the ranker significantly. For evidence ranking, we apply question generation with consistency filtering (Alberti et al., 2019) to obtain large amounts of synthetic QA pairs from unannotated product sources. For answer generation, we propose a novel data augmentation algorithm that creates training examples iteratively. By first training on this augmented data and then finetuning on the human annotations, the model performance can be further enhanced.

As for the model design, we homogenize all sources by reducing them to the same form of input which is fed into a unified pretrained Transformer model, similarly to recent work in open-domain QA (Oguz et al., 2020). We show that combining all sources within a single framework outperforms handling individual sources separately and that training signals from different answer sources can benefit each other, even for sources with totally different structures. We also show that the unified approach is able to produce comparable scores across different sources which allows for simply using the model prediction score for data source selection, an approach that outperforms more complex cascade-based selection strategies. The resulting system is able to find the correct evidence for 69% of the questions in our test set. For answer generation, 94.4% of the generated answers are faithful to the extracted evidence and 95.5% of them are natural-sounding.

Question: how much weight will it safely hold?		
Source	Supporting Evidence	Relevance
Attribute	item_weight:{unit: pounds,value:2.2}	✖
Bullet Point	supports up to 115 pounds	✔
Description	weight limit: 115 lbs.	✔
OSP	if you're looking for an inexpensive way to change up ...	✖
CQA	we put ours on a swingset.	✖
Review	it is sturdy and well made.	✖
Annotated Answer: it can support up to 115 pounds.		

Table 1: Annotation example. **Relevance annotation**: Given one question and evidence from heterogeneous sources, judge if each one is relevant to the question. **Answer elicitation**: annotators produce a natural-sounding answer given the question and the evidence that was marked as relevant.

In summary, our contributions are four-fold: (1) We create a benchmark collections of natural product questions and answers from 6 heterogeneous sources covering 309,347 question-evidence pairs, annotated for both evidence ranking and answer generation. This collection will be released as open source. (2) We show that training signals from different sources can complement each other. Our system can handle diverse sources without source-specific design, (3) We propose a novel data augmentation method to iteratively create training samples for answer generation, which achieves close-to-human performance with only a few thousand annotations and (4) We perform an extensive study of design decisions for input representation, data augmentation, model design and source selection. Error analysis and human evaluation are conducted to suggest directions for future work.

2 Benchmark test set collection

We begin by explaining how we collect a benchmark test set for this problem. The benchmark collection is performed in 4 phases: question sourcing, supporting evidence collection, relevance annotation, and answer elicitation. An annotation example in shown in Table 1.

Question sourcing To create a question set that is diverse and representative of natural user questions, we consider two methods of question sourcing. The first method collects questions through Amazon Mechanical Turk, whereby annotators are shown a product image and title and instructed to ask 3 questions about it to help them make hypothetical purchase decisions. This mimics a scenario in which customers see a product for the first time, and questions collected in this way are often general and exploratory in nature. The second method samples questions from the AmazonQA corpus. These are real customer questions posted in the community forum and tend to be more specific and detailed, since they are usually asked after users have browsed, or even purchased, a product. We then filter duplicated and poorly-formed questions. This yields 914 questions from AmazonQA and 1853 questions from Mturk. These are combined to form the final question set.

Collecting Supporting Evidence We gather "supporting evidence" from 6 heterogeneous sources: (1) Attributes: Product attributes in json format extracted from the Amazon product database [1]. (2) Bullet points: Product summaries from the product page. (3) Descriptions: Product descriptions from the manufacturer and Amazon. (4) On-site-publishing (OSP): Publications about products (for example here). (5) CQA: Top-voted community answers. Answers directly replying to questions in our question set are discarded and (6) Review: User reviews written for the product.

Relevance Annotation Annotators are presented with a question about a product and are instructed to mark all the items of supporting evidence that are relevant to answering the product question. Such evidence is defined as relevant *if it implies an answer, but it does not need to directly address or answer a question*. For evidence items from source 1, we directly present the attribute json to annotators. For sources 2∼6, we split the evidence into sentences and present each sentence as a separate item to be considered. There can be a very large number of CQA and Reviews for each product. As manual annotation of these would be impractical, we annotate only the top 40 and 20 evidence from each collection, respectively, as determined by a deep passage ranker pretrained on general-domain QA. Each item of evidence is inspected by 3 annotators and is marked as relevant if supported by at least two of them. In this way, items of evidence are paired with questions for review by annotators. Overall, annotators have inspected 309,347 question-evidence pairs, of which 20,233 were marked as relevant.

Answer Elicitation In the answer elicitation stage, annotators are presented with a question and an item of supporting evidence that has been marked as relevant. They are required to produce a *fluent, natural-sounding and well-formed* sentence (not short span) that *directly* answers the question. We sample 500 positive question-evidence pairs from each source for answer elicitation (if that many are available). The annotated answers are evaluated by another round of annotation to filter invalid ones. In the end, we obtain 2,319 question-evidence-answer triples for answer generation.

[1]We select 320 unique attributes that have diverse structures and hierarchies without standard schema.

Source	#words	available	answerable	N/P
Attribute	5.84	100%	36.10%	22.88
Bullet	12.55	100%	24.95%	5.59
Desc	12.86	98.37%	38.59%	23.97
OSP	17.75	18.98%	4.54%	11.16
CQA	13.32	99.39%	70.61%	13.85
Review	18.37	95.64%	61.16%	2.28
93.72% questions are answerable from at least 1 source.				

Table 2: Benchmark statistics: average number of words per evidence (**#words**), percentage of questions for which the source is available (**available**), percentage of answerable questions (**answerable**) and the negative-positive ratio (**N/P**).

Table 2 shows the collection statistics. Availability differs across sources. Only 19% of questions have available OSP articles, but all products have corresponding Attributes and Bullet Points. 93.72% of questions are answerable from at least 1 out of the 6 sources, indicating these sources are valuable as a whole to address most user questions.

3 Training data collection

For training data collection, a complete annotation of each set of evidence is not necessary; we need only a rich set of contrastive examples. Therefore, we propose to select questions for annotation based on the confidence score of a pretrained ranker (the same ranker we used to select top evidence for CQA and review). We sample 50k community questions about products in the toys and games domain. We first select questions whose top-1 item of supporting evidence returned by the pretrained ranker has a prediction score of > 0.8. In this way the selected questions have a good chance of being answerable from the available evidence and the approach should also yield enough positive samples from all sources to train the ranker. This selection step is crucial to ensure coverage of low-resource sources, like OSP, which otherwise might have zero positive samples. To avoid a selection process that is biased towards easy questions we further include questions whose top-1 evidence has a score within the range of 0.4∼0.6. Intuitively these questions will pose more of a challenge in ranking the evidence and their annotation should provide an informative signal.

From each out of the 6 sources, we sample 500 questions with prediction score > 0.8 and another 500 questions with scores in the range of $0.4 \sim 0.6$. For each question, we then annotate the top-5 (if available) evidence items returned by the pretrained ranker. This reduces annotation cost relative to the complete annotation that was done for the test set. The final dataset contains 6000 questions with 27,026 annotated question-evidence pairs being annotated, 6,667 of which were positive. We then submit the positive question-evidence pairs for answer elicitation. After filtering invalid annotations as was done for the benchmark collection, we obtain a set of 4,243 question-evidence-answer triples to train the answer generator. For both evidence ranking and answer generation, we split the collected data by 9:1 for train/validation.

4 Model

4.1 Evidence Ranking

Evidence ranking aims to get the best evidence from each of the sources. We build our evidence ranker with the Electra-base model (Clark et al., 2020). The question and evidence are concatenated together and fed into the model. We flatten the json structured from the attribute source into a string before feeding it to the encoder, whereas we split evidence from other sources into natural sentences, so it can be encoded as plain text (training detail in appendix G). We present comparison studies in figure 1 with the best model configuration. Due to space constraints we report only p@1 scores in Fig 1, with full results in appendix F.

Pre-tuning on AmazonQA Pre-tuning the evidence ranker on similar domains has shown to be important when limited in-domain training data is available (Hazen et al., 2019; Garg et al., 2020). For our product-specific questions, the AmazonQA corpus is a natural option to pre-tune the model (Lai et al., 2018b). The corpus contains 1.4M question-answer pairs crawled from the CQA forum. We remove answers containing "I don't know" and "I'm not sure", and filter questions of more than 32 words and answers of more than 64 words. We construct negative evidence with answers to different questions for the same product. The filtered corpus contains 1,065,407 community questions

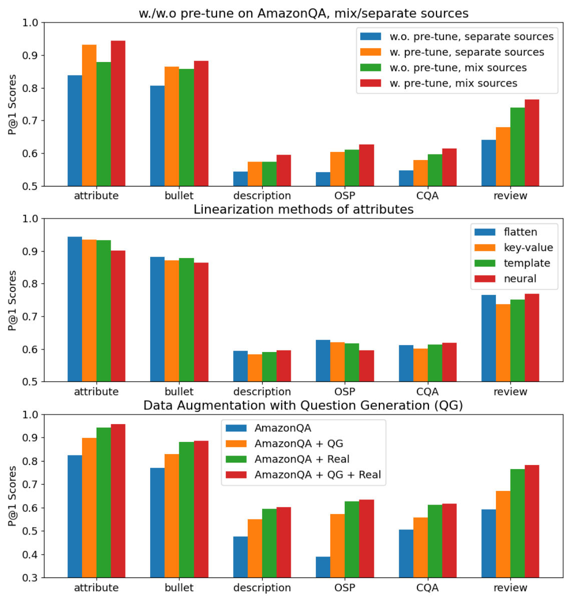

Figure 1: Ablation studies of evidence ranker. From up to down (1) effects of pre-tuning on AmazonQA, mix/separate sources, (2) effects of linearization methods of attributes, and (3) Effects of data augmentation by question generation.

for training. In the training stage, we first finetune the Electra-base model on the filtered AmazonQA corpus and then finetune on our collected training data. As can be seen, *pre-tuning on the AmazonQA corpus improves the p@1 on all sources*. The conclusion holds for both training on mixed sources and individual sources separately.

Mixed sources vs split sources We investigate whether different sources conflict with each other by (1) training a single model on the mixed data from all sources, and (2) training a separate model for each individual source. For the second case, we obtain 6 different models, one from each source. The resulting models are tested on 6 sources individually. We can observe that *mixing all answer sources into a single training set improves the performance on each individual source*. The training signals from heterogeneous sources complement each other, even for sources with totally different structures. p@1 on the semi-structured attribute improves consistently through adding training data of unstructured text. This holds for models with and without pre-tuning on AmazonQA.

Linearization methods In the above experiment, we use a simple linearization method that flattens the json-formatted attributes into a string. We also compare it with 3 other different linearization methods: (1) key-value pairs: Transform the hierarchical json format into a sequence of key-value

ranker / selector	BM25	AmazonQA	our best
perfect	0.4709	0.7546	0.8338
best-score-based	**0.2880**	**0.5370**	**0.6986**
highest-score	0.2696	0.5089	0.6888
cascade 1	0.2653	0.5298	0.6791
cascade 2	0.2638	0.5110	0.6715

Table 3: p@1 using different rankers and source selectors.

pairs. For example, the attribute in Table 1 will be transformed into "item_weight unit pounds | item_weight value 2.2". (2) templates: Transform the json by pre-defined templates, e.g. "The [attribute_name] of it is [value] [unit]" and (3) NLG: Transform the json into a sentence by a neural data-to-text model. Details of the template and NLG model are in appendix C D. The results show that *the best performance is achieved by simply linearizing the json into a string*. Although applying the template or neural data-to-text model is closer to a natural sentence, this did not lead to an improvement in p@1. Nonetheless, all these methods have rather similar performance, suggesting *the model can adapt quickly to different representations by finetuning on limited training data and that more complex linearization methods are unnecessary*.

Question Generation Question generation has been a popular data augmentation technique in question-answering. We collect $\sim 50k$ unannotated pieces of evidence from the 6 sources and apply a question generator to generate corresponding questions. The question generator is finetuned first on the AmazonQA corpus and then on our collected training data. We apply nucleus sampling with $p = 0.8$ to balance the diversity and generation quality (Sultan et al., 2020). We further filter the generated questions with our evidence ranker by only keeping those with model prediction scores of > 0.5, which has been shown crucial to get high-quality augmented data (Alberti et al., 2019). We try different finetuning methods and report the results on the bottom of Fig 1, where the "+" means the finetuning order. As can be observed, *finetuning on the augmented data brings further improvement to the model*. A three-step finetuning to gradually bring the model to our interested domain leads to the best performance over all sources.

4.2 Source Selection

Source aims to select the best source to answer after we obtain the top-1 item of evidence from each source. We show results for the following source selectors: (1) **perfect selector**: oracle selection of the correct item of evidence (if any) in the top-1 pieces of evidence provided from the 6 sources. (2) **best-score**: evidence item with the highest empirical accuracy in its score range which should yield the *upper-bound performance for a selector based on model prediction scores*. (3) **highest-score**: evidence with the highest model prediction score. (4) **cascade 1**: prioritizes evidence from the attribute/bullet sources since they have the highest p@1 scores. If the top-1 evidence item from those two sources has a score of more than ϵ, it is selected. Otherwise, the evidence item with the highest prediction score is selected from the remaining sources and (5) **cascade 2**: prioritizes evidence from attribute, bullet, and descriptions sources since these have better official provenance than user-generated data sources. The selection logic is the same as **cascade 1**. **highest-score** is the most straightforward choice but relies on a comparable score across sources. **cascades 1/2** are also commonly used to merge results from sub-systems. For the **best-score** selector, we split the prediction score range into 100 buckets and estimate the empirical accuracy on the test data. For example the prediction score of 0.924 for the top-1 evidence from an attribute source will fall into the bucket 0.92$\sim$0.93. In our test set, evidence items from each source will have an empirical accuracy within each score bin [2]. This will lead to an upper-bound approximation of a selector based on prediction scores since we explicitly "sneak a peep" at the test set accuracy. We combine these selectors with 3 evidence rankers: BM25, Electra-based tuned on AmazonQA, and our best ranker (AmazonQA + QG + Real in Figure 1). The results are in Table 3. The thresholds for cascade 1/2 are tuned to maximize the p@1 on the testset.

As our best "fair" ranker, the highest-score selector performs remarkably well, with p@1 only 1% lower than that of the best-score-based selectors. It also outperforms the two cascade-based selectors which prioritize official and high-precision sources. This implies the *the prediction scores across different sources are comparable* in our model, which might be because our model is trained on a combination of all sources with the same representation. For the model tuned on AmazonQA, where

[2]By continuing to split the confidence range into more buckets we can make an arbitarily exact approximation to the perfect selector for the test set, but with significant over-fitting.

evidence comes solely from the CQA source, the highest-score selector is not as effective as the cascade selectors. For all rankers, even with the best-score-based selector, there is still a large p@1 gap with the perfect selector, suggesting *a further improvement must take into account evidence content*, in addition to the prediction scores.

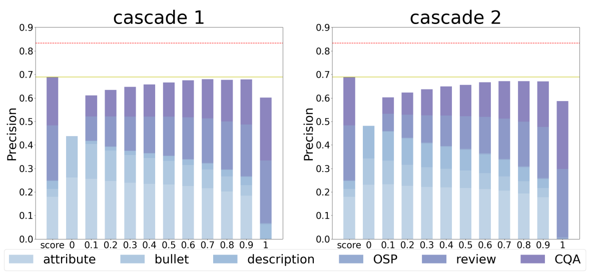

Figure 2: Answer source distribution as the threshold changes when using the cascade selection. Yellow line is with highest-score selector and red line is with a perfect selector.

In Figure 2, we visualize the distribution of selected sources by varying the threshold of two cascade-based selectors. We also show the distribution by using the highest-score selector (score) on the left. As the threshold grows, model precision first grows and then degrades, suggesting *all sources can contribute to answering product questions*. There is no single source that dominates. Although the cascade selection strategy underperforms the highest-confidence selector, it provides us with a flexible way to adjust the source distribution by threshold tuning. In practice, one may want to bias the use of information from official providers, even with a slight reduction in precision.

4.3 Answer Generation

After selecting an evidential item from one source, the role of answer generation is to *generate a natural-sounding answer based on both the question and the evidence*. We build our answer generator with the Bart-large model (Lewis et al., 2020). Similar to the evidence ranker, we take a unified approach for all sources by concatenating both the question and the evidence together (split by the token "|") as the model input. The model is then finetuned on the collected question-evidence-answer (q-e-a) triples. As in training the ranker, we flatten the json structures into strings and process them in the same way as the other sources.

Mixed sources vs split sources We experimented with training the generative model on each individual source separately as well as mixing the training data from all sources and training a unified model.

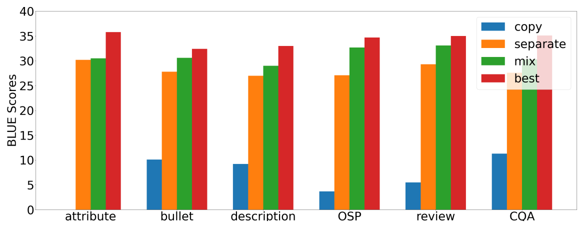

Figure 3: Ablation studies of answer generation. copy evidence vs separate sources/combine sources vs our best model.

We measured the BLEU scores of these systems with results shown in Figure 3, where we also include the results of directly copying the evidence. We can see that *training a unified model to handle all sources improves the performance on all sources*, as is consistent with our findings in evidence ranking. This is not surprising since previous research on data-to-text has also found that text-to-text generative models are quite robust to different variants of input formats (Kale and Rastogi, 2020; Chang et al., 2021). Directly copying the evidence as the answer leads to very low BLEU scores, especially for json-formatted attributes. This indicates *we must significantly rewrite the raw evidence to produce a natural answer*.

Conditional Back-translation (CBT) In our scenario, the AmazonQA contains a large amount of q-a pairs but these do not have corresponding evidence. We can apply a similar idea as back-translation (Sennrich et al., 2016) but further "condition" on the question. Firstly, we train an evidence generator based on our annotated q-e-a triples. The model is trained to generate the evidence by taking the q-a pairs as input. We then apply the model to generate pseudo-evidence e' from the $q-a$ pairs in AmazonQA. The answer generator is then first finetuned on the pseudo $q-e'-a$ triples and then finetuned further on the real $q-e-a$ annotations. It can be considered as a "conditional" version of back-translation where the model is additionally conditioned on the questions. We use nucleus sampling with p=0.8 to generate the evidence e' since a diversity of inputs is important for back-translation (Edunov et al., 2018). The results are displayed in Table 4. We can see that *adding the conditional back-translation step improves the BLEU score by nearly 3 points*.

Noisy Self-training (NST) Self-training is another popular technique in semi-supervised learning (Scudder, 1965). It uses a trained model to generate outputs for unlabeled data, then uses the generated outputs as the training target. In our

Method	BLEU	B-1	B-2	B-3	B-4
Copy	4.0	47.3	22.4	15.9	12.6
Bart-large	30.9	57.6	36.1	24.9	17.6
CBT	33.5	60.3	39.0	27.6	20.5
NST	32.5	59.5	37.3	26.2	19.2
NST + noise	33.2	59.8	38.0	26.9	19.9
Iteration-1	34.3	61.1	39.4	28.0	20.8
Iteration-2	**34.9**	61.1	**39.8**	28.3	21.4
Iteration-3	**34.9**	**61.3**	39.7	**28.6**	**21.6**
Iteration-4	34.7	61.3	**39.8**	28.5	21.3

Table 4: BLEU scores on different methods: copying the input evidence as the answer (**copy**), finetuning Bart-large on training samples (**Bart-large**), Bart-large + conditional back-translation (**CBT**) and Bart-large + noisy self-training (**NST**).

```
(Inilialization) $G_e = G_a =$ Bart-large;
for $i$=1 to $N$ do
    finetune $G_e$ on $\{q - a - e\}_{real}$;
    Generate $e'$ with $G_e$ from $\{q - a\}_{AmazonQA}$;
    finetune $G_a$ on generated
      $\{q - e' - a\}_{AmazonQA}$;
    finetune $G_a$ on $\{q - e - a\}_{real}$;
    Noisy Self-training ($G_a$);
    Generate $a'$ with $G_a$ from $\{q' - e\}_{QG}$;
    finetune $G_e$ on generated $\{q' - a' - e\}_{QG}$;
end
```

Algorithm 1: Iterative Training Process. G_e is the evidence generator and G_a is the answer generator. $\{q - a - e\}_{real}$,$\{q - a\}_{AmazonQA}$ and $\{q' - e\}_{QG}$ indicate the data from the real annotation, AmazonQA and question generation respectively.

Evaluated	Faithfulness (%)	Naturalness (%)
copied evidence	-	15.44
our best	94.39	95.51
human reference	97.00	95.82

Table 5: Human evaluation results.

scenario, however, the unlabeled input data is not readily available since it requires positive question-evidence pairs. We first apply the same question generation model used for evidence ranking to create "noisy" $q' - e$ pairs. The current model then generates an answer a' based on the $q' - e$ pairs. We use beam search with beam size 5 to generate the answers as the generation quality is more important than diversity in self-training (He et al., 2020). A new model is then initialized from Bart-large, first finetuned on the $q' - e - a'$ triples, then finetuned on the real training data. We also experimented with adding noise to the input side when training on the $q' - e - a'$ triples, which has shown to be helpful for the model robustness (He et al., 2020) [3]. As shown in Table 4, NST improves the model performance by over 1 BLEU point. Adding the noise to the input further brings slight improvement.

Iterative Training We further investiated combining the proposed CBT and NST into an iterative training pipeline. The intuition is that CBT can improve the answer generator which then helps NST to generate higher-quality pseudo answers. The higher-quality triples from NST can in turn be used to 'warm up' the evidence generator for CBT. Algorithm 1 details the process. It can be considered a variant of iterative back-translation (Hoang et al., 2018) with an additional condition on the question and the noisy self-training process inserted in between. It essentially follows a generalized EM algorithm (Cotterell and Kreutzer, 2018; Graça et al., 2019) where the evidence generator and the answer generator are guaranteed to improve iteratively. We show the results after each iteration in Table 4. As can be seen, the iterative training pipeline further improves generation quality. Most gains are found in the first iteration and the model saturates at iteration 3 with a BLEU score of 34.9.

Human Evaluation We run a human evaluation to assess generation quality of our best generator (iteration-3 from Table 4), human reference and the copied evidence. We evaluate from two perspectives: (1) Faithfulness: A sentence is unfaithful to the evidence if it contains extra or contradictory information, and (2) Naturalness: A sentence is unnatural if it is not fluent; contains additional information that not relevant as an answer; or does not directly reply to the question.

We show the results in Table 5. We can observe that copying the evidence directly leads to a naturalness score of only 0.15, which further confirms that an answer generator is needed for a natural presentation. The generations from our best model improve the naturalness score to 0.9551 and are faithful to the evidence in 94.39% of the cases, only slightly lower than the human references.

4.4 Best Practices

To summarize the best practices, the attribute json strings can be directly flattened and all sources are mixed together and trained with a single unified encoder. The ranker is finetuned on AmazonQA, augmented data obtained by question generation and manually annotated training data in order. Source selection can be performed based solely on the

[3] We apply a similar noise function as in Edunov et al. (2018) that randomly deletes replaces a word by a filler token with probability 0.1, then swaps words up to the range of 3.

Subjectivity of Relevance
Question: what is the snow made of? **Predicted**: kind of cotton like.
Noise in Attribute Value
Question: is it dark green or brown? **Predicted**: color: { value:“as shown” }
Overfit to String Match
Q: what is the length of the solder? **Predicted**: coil of solder is about 1-3/4 long and ...
Uncertain Evidences
Q: can it be machine washed? **Predicted**: not sure if it is machine washable.

Table 6: Common errors of evidence relevance ranking.

Number accuracy
Q: what’s the size of the house? **E**: item_dimensions: {width:... height:{unit:centimeters, normalized_value:{unit:inches, value:15.35},value:39.}} **G**: The house is 10.24 × 30.31 × 39 inches
Hallucination when Inference needed
Q: are the dolls with it? **E**: included_components:{value:“my dreamy doll-house”} **G**: Yes, it comes with a doll.
Sensitivity to typographical errors
Q: for he many players? **E**: ages 7 and up, 2-12 players **G**: this game is for ages 7 and up.

Table 7: Common errors of answer generations.

model confidence score and the answer generator can be trained as in Algorithm 1.

5 Error analysis

Based on the human evaluation, we identified the following key problems that exist in the current system. For evidence ranking, the major problems are: (1) **subjectivity of relevance**: It can be subjective to define whether a piece of evidence is enough to answer a given question. The model will sometimes pick a somewhat relevant piece of evidence, even though there could be other, better options that support a more comprehensive answer. (2) **noise in attribute value**: When an attribute value contains uninformative data due to the noise of data sources, the model still may choose it based on its attribute name. (3) **overfitting to string match**: The model tends to select strings similar to the question while ignoring their fine semantics, a common problem from the bias to ‘shortcut learning’ of neural networks (Geirhos et al., 2020). (4) **uncertain evidence**: The model ranks evidence highly, even if this evidence is an uncertain expression. This can be viewed as a special case of over-fitting to string match. We show examples in Table 6. We can attempt to alleviate errors of type 1 by providing finer-grained labels in the training data instead of only binary signals (Gupta et al., 2019). Error types 2 and 4 could be mitigated by data augmentation, constructing negative samples by corrupting the attribute values or making evidences uncertain. Error type 3 is more challenging. One possible solution is to automatically detect spurious correlations and focus the model on minor examples (Tu et al., 2020). Nevertheless, a fundamental solution to fully avoid Error 3 is still an open question.

For answer generation, we identify the major problems as: (1) **Number accuracy**: The model cannot fully understand the roles of numbers from the limited training examples. (2) **Hallucination if inference is needed**: when it is not possible to generate an answer by simple rephrasing, the model can hallucinate false information. (3) **Sensitivity to typos**: The model is not robust to typos in the question. A tiny typo can easily break the system.

We provide examples of these errors in Table 7. Error types 1 and 3 could be alleviated through data augmentation. We can create new samples to let the model learn to copy numbers properly and learn to be robust to common typos. Another way to reduce number sensitivity could to delexicalize numbers in the inputs, a common strategy in data to text generation (Wen et al., 2015; Gardent et al., 2017). Error type 2 is a challenging open problem in neural text generation. Many techniques have been proposed such as learning latent alignment (Shen et al., 2020), data refinement with NLU (Nie et al., 2019), etc. These could potentially be applied to our task, which we leave for future work.

6 Conclusion

To the best of our knowledge, this work is the first comprehensive study of product answer generation from heterogeneous sources including both semi-structured attributes and unstructured text. We collect a benchmark dataset with annotations for both evidence ranking and answer generation. It will be released to benefit relevant study. We find that the best practice is to leverage a unified approach to handle all sources of evidence together and further experimented with a set of data augmentation techniques to improve the model performance. Error analysis is provided to illustrate common errors, which we hope will lead to inspire future work.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic qa corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021. Neural data-to-text generation with lm-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Ryan Cotterell and Julia Kreutzer. 2018. Explaining and generalizing back-translation through wake-sleep. *arXiv preprint arXiv:1806.04402*.

Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. Superagent: A customer service chatbot for e-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Yue Feng, Zhaochun Ren, Weijie Zhao, Mingming Sun, and Ping Li. 2021. Multi-type textual reasoning for product-aware answer generation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1135–1145.

Shen Gao, Xiuying Chen, Zhaochun Ren, Dongyan Zhao, and Rui Yan. 2021. Meaningful answer generation of e-commerce question-answering. *ACM Transactions on Information Systems (TOIS)*, 39(2):1–26.

Shen Gao, Zhaochun Ren, Yihong Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019. Product-aware answer generation in e-commerce question-answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 429–437.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2020. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *AAAI*.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Miguel Graça, Yunsu Kim, Julian Schamper, Shahram Khadivi, and Hermann Ney. 2019. Generalizing back-translation in neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 45–52.

Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C. Lipton. 2019. Amazonqa: A review-based question answering task. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4996–5002. International Joint Conferences on Artificial Intelligence Organization.

Timothy J Hazen, Shehzaad Dhuliawala, and Daniel Boies. 2019. Towards domain adaptation from limited data for question answering using deep neural networks. *arXiv preprint arXiv:1911.02655*.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Tuan Lai, Trung Bui, Sheng Li, and Nedim Lipka. 2018a. A simple end-to-end question answering model for product information. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 38–43.

Tuan Lai, Trung Bui, and Nedim Lipka. 2020. Isa: An intelligent shopping assistant. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 14–19.

Tuan Manh Lai, Trung Bui, Nedim Lipka, and Sheng Li. 2018b. Supervised transfer learning for product information question answering. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1109–1114. IEEE.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Julian 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*, pages 625–635.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unified open-domain question answering with structured and unstructured knowledge. *arXiv preprint arXiv:2012.14610*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Henry Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165.

Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. 2020. On the importance of diversity in question generation for qa. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Review conversational reading comprehension. *arXiv preprint arXiv:1902.00821*.

Qian Yu, Wai Lam, and Zihao Wang. 2018. Responding e-commerce product questions via exploiting qa collections and reviews. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2192–2203.

Shiwei Zhang, Jey Han Lau, Xiuzhen Zhang, Jeffrey Chan, and Cecile Paris. 2019. Discovering relevant reviews for answering product-related queries. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1468–1473. IEEE.

Wenxuan Zhang, Qian Yu, and Wai Lam. 2020. Answering product-related questions with heterogeneous information. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 696–705.

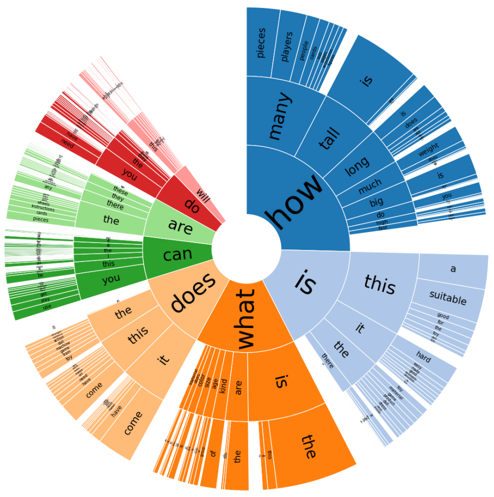

Figure 4: The ngram distribution of prefixes of questions.

A Collected Data

All our collected data have also been manually verified to remove sample with private or offensive information.

In Figure 4, we show the ngram distribution of question prefixes i our collected data. As can be seen, a large proportion of questions are boolean questions starting with “is”, “does”, “can”, “are”, “do” and “will”. The rest are mostly factual questions like “how many/tall/long ...” and “what ...”. Most of them should be able to answer with a short span since there are not many opinion questions like “how is ...”, “why ...”.

B Instruction for Human Annotation

All annotators are based on the US. We first perform in-house annotation and then estimate the time needed for each annotation. We then set the payment to be roughly 15 USD per hour. The payment is decided based on the average payment level in the US. All annotators are informed that their collection will be made public for scientific research according to the Amazon Mechanical Turk code of rules. The data collection protocol has been approved by an ethics review board.

B.1 Question Collection

Read the given product name and image, imagine you are a customer and are recommended this product. Write one question about it to decide whether or not to purchase this product.

Examples of questions: is it energy efficient? does it require a hub? can I watch sports on this TV? will the plug work with an extension cord?

B.2 Evidence Selection

At the start of each task, the workflow application will present a product, a question about the product and a set of candidates which describe the product. Your annotation task is to mark the proper candidate that contains information to answer the question from the attribute set. If none of the provided candidates contain the information, select ”None of the above”.

B.3 Answer Generation

Read the raised product question and provided information, write a natural, informative, complete sentence to answer this question. If the provided information cannot address the question, write ”none”. Make sure the answer is a natural, informative and complete sentence. Do not write short answers like ”Yes”, ”Right”, ”It is good”, etc. Provide enough information to help the asker understand more about the question. If the provided information can only partially answer the question, only reply to the answerable part.

Good Examples:

question: what age range is this product designed for?

Provided information: age_range_description: value:”3 - 8 years

Answer: It is designed for the age range of 3 - 8 years old.

question: how many people can play at one time?

provided information: number_of_players: value:”8

answer: It is designed for 8 players at one time.

Bad Examples:

question: what age range is this product designed for?

Provided information: age_range_description: value:”3 - 8 years

Answer: 3 - 8 years.

question: how many people can play at one time?

provided information: number_of_players: value:”8

answer: 8.

C Template Data-to-Text System

When designing the template system, we aim to capture general rules across different attribute types so that one template can be reusable to other similar attributes. We define each template should contain (1) a precondition specializing when to apply the template, (2) one or several corresponding text with gaps to fill, and (3) a set of rules defining which text to select and how to fill in the gaps. For example, the following is a template defined from the attribute type *ARE_BATTERIES_REQUIRED*:

```
Precondition: applies if the POS
    tag of the attribute
name follows the pattern of
    be_NOUN_VERBed.
Rule: (1) If the value is ''Y''
    or ''yes'' or ''True'':
output ''It VERBs the NOUN''.
(2) Otherwise: output ''It does
    not VERB the NOUN''.
```

where VERBs and VERBed mean the third person singular and past particle form of the verb. For the attribute *ARE_BATTERIES_REQUIRED*, VERBs would be "requires" and VERBed is "required". It can also apply to other attribute types following the same pattern like "is_assembly_required", "is_software_included", etc.

During template construction, we maintain a template bank starting from empty. As we see more attribute types, we check if any template from the bank can be applied, and if so, whether it generates the correct text or whether we need to manually update the template. Otherwise, we create a new template for this attribute type. This process is repeated until we go over all the 320 attribute types three times, to refine, merge and fix the template bank and rules. After these rounds, we end up with a total of 23 distinct templates.

Nevertheless, during the construction process, we realize it is nearly impossible to devise a template system to cover all cases well, even for the limited 320 attribute types that we focus on. The difficulty lies in the following two diversities in the data:

1. linguistic diversity: The attribute values do not follow any strict rule. They can be free text as long as it conveys the meaning, which makes it hard to design general rules even for a single attribute type.

2. structural diversity: The json format is a loose structure. The same semantic meaning can be organized in different ways and hierarchies. Applying one rule for different structures can easily lead to parsing errors.

We show an example in Table 8. Even for this single attribute, it requires many verbalizing rules to handle different structures and attribute values, let alone extending the template rules to multiple attribute types.

Attribute value	Text
{ value:"gas-powered"}	The product is gas-powered.
{ value:"batteries"}	It runs on batteries.
{ value:"Manual" }	This doesn't have power.
{ value:"NA" }	This doesn't run on any power.

Table 8: Different instances of the attribute type "power_source_type" and human annotated text.

D Neural Data-to-Text

To avoid pre-defined rules and to generalise to unseen attributes, we train a neural generator model initialized with T5-large (Raffel et al., 2020). As input, we feed the linearized json-formatted data and the output is the annotated text.

To get the training data, we obtain the semi-structured attributes of product information from a product database. These attributes are aggregated from different providers with varied schema. We select 320 unique attribute types from it [4], filter out information only for internal use and indicator tags containing no actual information like "language_tag", "attribute_id" etc. For each of the 320 attribute types, we randomly sample 20 products containing such attribute from 5M products sold in the US market (The 5M products are randomly sampled from different categories), then extract their attribute instances. After removing duplicate ones, we get 3,316 unique attribute instances in the end. We then preprocess them to lower-case all characters, remove emojis and normalize all floats to contain at most 2 decimals, since customers will barely need overly precise decimal numbers.

We hire annotators from Amazon Mechanical Turk [5] to write a natural sentence for each attribute instance. We restrict to US-based annotators who completed >500 tasks, out of which more than 97% had been accepted. Before the formal annotation, we did a pilot study with 100 samples. Without extra information, we find 16% of attributes are not understandable to humans, which indicates proper context is necessary to understand the meanings of

[4]We manually check the product attributes, select 320 types of them which contain meaningful information about the products and exclude those only used for internal management, metadata, etc.

[5]https://www.mturk.com/

#Annotated attributes	3191
#unique attribute types	320
#Tokens per attribute	7.93
#Tokens per text	7.52
Vocab size of attribute	4567
Vocab size of text	4334

Table 9: Dataset Statistics. Each annotated attribute has one text describing its content.

attributes. Therefore, we also provide the product image and title in the second round of pilot study. By adding the extra information, only 4% of them are not understandable. We then continue with this setting and get all attributes annotated. We also remove all attributes that are not understandable to annotators (usually those that rely on other information to interpret), and end up with 3,191 attribute instances annotated with their description text. Table 9 lists the statistics of the dataset.

We further normalize the numbers in both the attribute and text to keep them in a consistent form, to help the model learn their correspondence in the generation task. For example, we turn forms like "1.", "1.0" and "1.00" into 1, and normalize words to numeric values ("one" $\rightarrow$ "1" etc).

We consider two test scenarios, one containing only seen attributes with unseen values, and the other containing only unseen attributes to test the model generalization capability. For the unseen scenario, we randomly sample 30 attribute types from all 320 types. We sample 58 instances from them and add into the dev set, while the rest are used as test set. For the seen scenario, we randomly sample 440 instances from the remaining 290 attribute types. 220 of them are added into the dev set and the rest serve as the test set. We use one fixed dev set containing both seen (220) attributes and unseen (58) attributes. All remaining instances serve as training set. Due to the small data size, we perform cross validation to get more reliable results. We repeat the above process ten times with different seeds to get 10 different splits, then train/evaluate on them and average the results.

E Human Evaluation on Data-to-Text

We conduct a human evaluation of the generated texts, focusing the following three dimensions:

1. **Faithfulness**, whether the text is faithful to the attribute. 1 for faithful and 0 if the text contains any wrong information that does not exist in the attribute.
2. **Coverage**, whether the text covers all contents in the attribute. 1 if covers all and 0 if it misses any information contained in the attribute.
3. **Naturalness**, whether the text is a natural sentence rather than a machine-generated rigid one. 4-ary score from 1(rigid), 2(slightly rigid), 3(slightly natural) to 4(natural)

On seen attributes, we evaluate the T5-large, the template system and the annotated reference. On unseen attributes, we only evaluate T5-large and the reference since handcrafted templates cannot be applied to unseen attributes at all.

From each of 10 data splits, we randomly sample 50 attributes from it such that each model has 500 attribute-text pairs being evaluated. Each pair is evaluated by three annotators. The final scores are averaged over the 500 pairs for each model. We show the results and the agreement score among annotators in Table 10 and Table 11 respectively.

Overall, the evaluation has a rather high agreement score. Naturalness has the lowest agreement since it is 4-ary. We also calculate the binary score for naturalness by combining natural and slightly natural into one bucket, and combining rigid and slightly rigid into the other bucket. The agreement score grows to over 0.92 by this means. We then manually checked and corrected all attribute-text pairs that do not have an agreement score of 1 for faithfulness and coverage.

Overall all models have high scores on both faithfulness and coverage, and differences are small. **For naturalness, as expected, templates have the lowest score.**

Model	Faithfulness	Coverage	Naturalness
Performance on Seen Attributes			
Template	0.9612	0.9546	3.20266
T5-large	**0.9731**	**0.97761**	**3.65672**
Performance on Unseen Attributes			
T5-large	0.9125	0.9231	**3.6103**
Reference	**0.9401**	**0.9513**	3.5203

Table 10: Human Evaluation Results for Answer Presentation

Faithful	Coverage	Natural-4nary	Natural-binary
0.97762	0.97402	0.80499	0.92569

Table 11: Agreement Score for Answer Presentation Evaluation

Source	MAP	MRR	NDCG	P@1	HIT@5
Attribute	0.965	0.966	0.974	0.943	0.996
Bullet	0.935	0.935	0.952	0.890	0.993
Description	0.648	0.708	0.747	0.611	0.822
OSP	0.667	0.708	0.763	0.579	0.873
Review	0.796	0.860	0.875	0.778	0.966
CQA	0.643	0.750	0.766	0.636	0.897

Table 12: Performance of our best ranker on different sources.

F Full Results of Ranker

We show the full results of our best-performed ranker in Table 12. As can be seen, different sources have different accuracy score. The attribute and bullet point source have the highest accuracy score because the former is more structured, and the latter has a consistent writing style with only a few sentences. User reviews also have a high accuracy score. This might be because the candidates of reviews are already the top ones selected by our pretrained ranker. Many of them are already relevant and the negative-positive ratio is low. The model does not have extreme difficulty in handling the user reviews. The model performs worst on the description, OSP and CQA answer source. This might result from the diversity of their writing styles and the high negative-positive ratio, which increase the difficulty. Moreover, these two sources usually depend more on the context to interpret the evidence than other sources. The text description is extracted from the multi-media web page. Simply extracting the text part might lose richer context to interpret the extracted text. Similarly, the CQA usually depends on the community question. If we only extract a sentence from the answer, it might contains references that is not self-contained.

G Training details

For both the generative Bart-large model and the discriminative Electra-base model, we we truncate the total input length to 128 subword tokens and select the learing rate from $[5e-6, 1e-5, 3e-5, 5e-5, 1e-4]$. The warm-up step is selected from $[5\%, 10\%, 20\%, 50\%]$ of the whole training steps. For the discriminative model, we choose the best configuration based on the F1 score on the validation set. For the generative model, we choose the best configuration based on the perplexity on the validation set. In the end, we set the learning rate of Electra-base as $3e-5$ and that of Bart-large as $1e-5$. The warm-up step is set as 20% for Electra-base and 10% for Bart-large. The batch size is set as 64 for Electra-base and 16 for Bart-large. For Electra-base, we measure the validation F1 score after finishing every 1% of the whole training steps and stop the model when the valitaion F1 score does not increase for 30% of the whole training steps. For Bart-large, we measure the validation loss every 200 steps and stop the model when the validation loss stops decreasing for 1000 steps. All models are trained once on 8 Nvidia V100 GPUs and the random seed is set as 42.