Unsupervised Question Answering via Answer Diversifying

Yuxiang Nie[123], **Heyan Huang**[123]*, **Zewen Chi**[123], **Xian-Ling Mao**[123]
[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2]Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, Beijing, China
[3]Beijing Institute of Technology Southeast Academy of Information Technology, Fujian, China
{nieyx,hhy63,czw,maoxl}@bit.edu.cn

Abstract

Unsupervised question answering is an attractive task due to its independence on labeled data. Previous works usually make use of heuristic rules as well as pre-trained models to construct data and train QA models. However, most of these works regard named entity (NE) as the only answer type, which ignores the high diversity of answers in the real world. To tackle this problem, we propose a novel unsupervised method by diversifying answers, named **DiverseQA**. Specifically, the proposed method is composed of three modules: data construction, data augmentation and denoising filter. Firstly, the data construction module extends the extracted named entity into a longer sentence constituent as the new answer span to construct a QA dataset with diverse answers. Secondly, the data augmentation module adopts an answer-type dependent data augmentation process via adversarial training in the embedding level. Thirdly, the denoising filter module is designed to alleviate the noise in the constructed data. Extensive experiments show that the proposed method outperforms previous unsupervised models on five benchmark datasets, including SQuADv1.1, NewsQA, TriviaQA, BioASQ, and DuoRC. Besides, the proposed method shows strong performance in the few-shot learning setting.[1]

1 Introduction

Extractive question answering (extractive QA) aims to provide answer spans extracted from the context to answer questions. It can improve the interaction between humans and machines in applications such as search engines and dialog systems.

Existing extractive QA methods can be divided into two categories: supervised QA and unsupervised QA. Traditionally, for supervised QA (Seo et al., 2016), human-labeled context, questions and answers are given to train a QA model. Since the construction cost of labeled data is too high for supervised QA, recently, researchers pay more attention to the unsupervised setting, where QA pairs are unavailable. For example, Lewis et al. (2019) propose an unsupervised machine translation model to generate QA pairs. Li et al. (2020), Hong et al. (2020) and Lyu et al. (2021) try to alleviate the overlap between the context and the generated question. However, most existing works regard named entities as the only answer type, which ignores the high diversity of answers in the real world. For instance, as shown in Figure 1, in SQuADv1.1 (Rajpurkar et al., 2016) and NewsQA (Trischler et al., 2017), the answer type of named entities only account for 52.4% and 41.1% respectively.

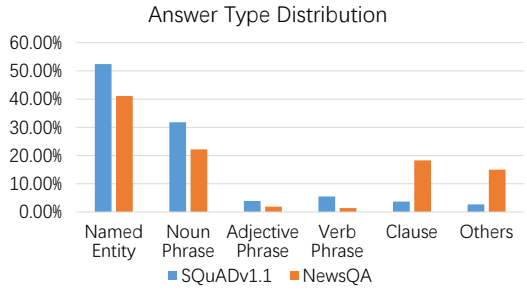

Figure 1: The answer type distributions of SQuADv1.1 and NewsQA, where we notice that named entities are just a fraction of each dataset.

To solve the problem, an intuitive way is to extract each type of answer spans independently in the raw text. Yet, it leads to two critical problems. Firstly, it's hard to determine the proportion of each answer type in the synthetic dataset. Secondly, extracting answers without any guidance could probably generate trivial and noisy question-answer pairs. To tackle the problem above, we propose **DiverseQA**, a novel unsupervised QA method. Specifically, the proposed method consists of three modules: data construction, data augmentation and denoising filter. Firstly, the data construction mod-

*Corresponding author

[1]We have released our codes and data in https://github.com/JerrryNie/DiverseQA.

Proceedings of the 29th International Conference on Computational Linguistics, pages 1732–1742
October 12–17, 2022.

ule employs a simple answer-extending rule to construct a dataset with diverse answers. As shown in Figure 2, a named entity (NE) is extracted and extended into a longer answer span, which should be a constituent of the sentence (e.g. noun phrase, verb phrase). In this way, the proportion of each type of answers can be obtained entirely from the original text. Besides, NE-based extension largely guarantees that extracted answers are meaningful. Secondly, an answer-type dependent data augmentation module is proposed. Concretely, an adjusting vector generator is designed to produce answer-type enhanced QA pairs in the embedding level. Then, a discriminator classifies the embeddings into the corresponding answer type while minimizing the KL divergence between the distribution and its prior to fool the discriminator in an adversarial way. Thirdly, the denoising filter is applied to alleviate the noise in the synthetic QA pairs.

Extensive experiments on six benchmarks, including SQuADv1.1 (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), NaturalQuestions (Kwiatkowski et al., 2019), NewsQA (Trischler et al., 2017), BioASQ (Tsatsaronis et al., 2015) and DuoRC (Saha et al., 2018) show that DiverseQA outperforms previous unsupervised methods on five datasets and obtains comparable results on one dataset among unsupervised QA models. Further analysis shows that DiverseQA can largely improve the question-answering ability of the model on diverse answer types. In addition, our method shows strong performance in the few-shot learning setting.

The contributions of our method are as follows:

- We propose DiverseQA, a novel method to improve an unsupervised QA model to handle answers beyond named entities.
- Our method outperforms previous unsupervised works on five benchmarks and reaches the comparable result on a benchmark.
- Further analysis shows that the drift of answer length distribution and the quality of extracted answers are important to the performance of the model.

2 Related Work

Data Augmentation. Data augmentation methods can be regarded as regularizers to make the model robust and reduce dependence on the training data. In computer vision domains (Krizhevsky et al., 2012), geometric transformation and color space transformation are effective. In natural language processing, word removing, synonym replacing and back-translation can enlarge the diversity of examples (Xie et al., 2020). Lee et al. (2021) proposes an embedding-level data augmentation method to improve the performance of QA on out-of-distribution data. This work relies on supervised (low noise) training instances, while in the unsupervised (high noise) scenario, embedding-level data augmentation methods have not been explored yet.

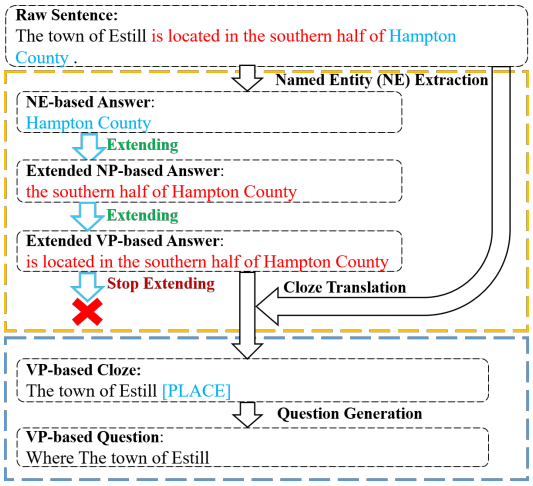

Figure 2: An example of our data constructing module. We extract the named entity (NE) in the *Raw Sentence* and extend it into a longer sentence constituent until meeting the stop extending condition. The final extended answer span is a VP in this example, which is used in cloze translation for question generation.

Extractive Question Answering. Extractive question answering (extractive QA) aims to output an answer span given the context (containing the answer) and the question. It has gained much progress with the help of large labeled datasets such as SQuAD, NewsQA, and TriviaQA. To better handle these datasets, strong extractive QA models are proposed, including BiDAF (Seo et al., 2016), BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). However, because they largely depend on human-annotated data, the lack of labeled data for some specific domains constrains the capacity of the supervised extractive QA model.

Unsupervised Question Answering. Unsupervised question answering (unsupervised QA) becomes attractive among researchers recently (Lewis et al., 2019). Like supervised extractive QA, given context and question, the model needs to extract a text span from the context to answer the question. The difference is that in the unsupervised setting, models need to learn to answer the question without any human-labeled ⟨context, question, answer⟩

triples, which is a more challenging task than the supervised counterpart. Lewis et al. (2019) extracts named entity from the context, and then trains an unsupervised neural machine translation model to convert the cloze question into a natural question. Li et al. (2020) makes use of cited documents and a refine phase to alleviate serious question-context overlapping and improve answer diversity. Hong et al. (2020) uses paraphrasing and trimming to remove unanswerable questions as well as alleviate the context-question similarity problem. Lyu et al. (2021) takes advantage of a supervised summarization dataset to tackle the context-question overlapping problem. However, most of the existing works regard the named entity (NE) as the only answer type, while other types (e.g. noun phrases, adjective phrases, verb phrases, sub-clauses) are nearly ignored. Although in Lewis et al. (2019), the inclusion of noun phrases (NPs) was discussed, the reported poor performance eventually led the author to use the NE-based synthetic QA pairs. In Li et al. (2020), even though generating more answers is considered, most of the model-generated answers have strong relationships with the named entity and we will discuss it later in Section 4.3.4.

3 Method

To explore answer types beyond named entities, we propose an unsupervised method **DiverseQA**, which can be divided into three modules, data augmentation, data construction and denoising filter. Firstly, a simple but effective span extending rule is applied in the data construction module to do answer extraction and natural question generation to construct a synthetic QA dataset with diverse answers. Secondly, an answer-type dependent data augmentation module via adversarial training is designed to produce high-quality augmented QA pairs. Thirdly, a denoising filter is proposed to alleviate high noise in the dataset. In the next few subsections, we will introduce the three modules in detail.

3.1 Synthetic QA Data Construction via Span Extension

In this section, we illustrate our QA data constructing module, which can be divided into two steps. Firstly, we extract answers in the text through span extension. Secondly, we take advantage of pseudo-NER label to generate questions.

3.1.1 Answer Generation via Span Extension

As shown in Figure 2, given a *Raw Sentence*, first of all, we extract the named entity span *Hampton County*. Then, we extract all the constituents containing *Hampton County*. We extend the span into a longer constituent iteratively. When the next extending span makes up more than $\omega\%$ of the *Raw sentence*, we stop extending and regard the extended span as the final answer span. In this example, the NE is firstly extended into a longer NP. Again, the NP is extended into the VP. As the VP '*is located in the southern half of Hampton County*' is the longest constituent while making up less than $\omega\%$ (called *Span Extending Threshold*) of the *Raw Sentence*, it becomes the answer span of the QA pair. The intuition is that even if many constituents are trivial, the constituent containing the named entity and making up a proper portion of the *Raw Sentence* might be meaningful to create a high-quality QA pair.

3.1.2 Question Generation with Pseudo-NER Label

To construct a NE-based QA pair, the NER label of the answer span is mapped to a question word (Lewis et al., 2019) for generating the natural question. However, it cannot be directly applied to other answer types. To tackle the problem, we regard the original NER label of the named entity as the *pseudo-NER label* of the extended answer span. After that, we replace the answer span with the high-level NER mask token (Lewis et al., 2019). The intuition is that the semantic information of the extracted named entity can be probably consistent with the extended constituent.

Followed Lewis et al. (2019) and Li et al. (2020), we do the mask token mapping and cloze to natural question conversion.

3.2 Answer-type Dependent Data Augmentation

We firstly introduce the background of extractive QA, and then illustrate how we design our answer-type dependent data augmentation modules.

3.2.1 Backgrounds of Extractive Question Answering

Given question $\mathbf{q} = (q_1, q_2, ..., q_m)$ and context $\mathbf{c} = (c_1, c_2, ..., c_n)$, extractive QA models aim to predict the start and end token of the answer span $\mathbf{a} = (a_1, a_2)$ from the context. Assuming that there are N observations: $\{\mathbf{c}^{(i)}, \mathbf{q}^{(i)}, \mathbf{a}^{(i)}\}_{i=1}^{N}$, we

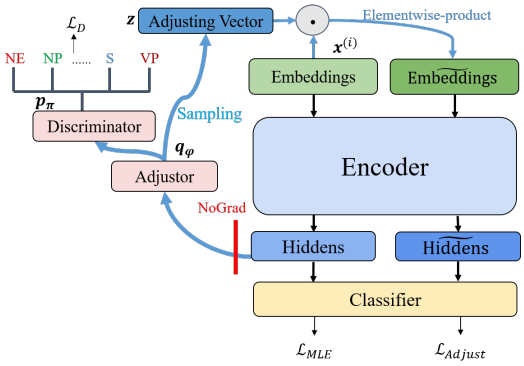

Figure 3: Our data augmentation module. The Embeddings of the input sequence are firstly encoded into the Hiddens. Then, the Adjustor module receives the Hiddens and produces an Adjusting Vector to adjust the Embeddings by the element-wise product. Meanwhile, the Discriminator classifies the produced vector into an answer type to make the vector answer-type dependent.

estimate the model parameter θ by maximizing the following function:

$$\mathcal{L}_{\text{MLE}}(\theta) = \sum_{i=1}^{N} p_\theta(\mathbf{a}^{(i)}|\mathbf{c}^{(i)}, \mathbf{q}^{(i)}) \quad (1)$$

3.2.2 Answer-type Dependent Embedding Adjustment

As QA instances might have specific feature related to answer types, motivated by Lee et al. (2021), we design an answer-type dependent embedding-level *Adjusting Vector* to create high quality instances for model training.

The proposed adjusting vector $\mathbf{z} \in \mathbb{R}^{d\times(m+n+r)}$ is sampled from the distributions $q_\phi(\mathbf{z}|\mathbf{x}, l)$ to augment the input sequence $\mathbf{x} \in \mathbb{R}^{d\times(m+n+r)}$, where r is the special token length, d is the size of a word embedding vector, $\mathbf{x}$ is the embeddings, l is an answer type. We use the element-wise production between the embedding and the Adjusting Vector as extra data to train a QA model by maximizing the log-likelihood function of $p_\theta(\mathbf{a}|\mathbf{x}, \mathbf{z})$:

$$\mathcal{L}_{\text{Adjust}}(\theta, \phi) = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},l)} \left[\log p_\theta(\mathbf{a}^{(i)}|\mathbf{x}^{(i)}, \mathbf{z})\right] + \beta \text{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, l)||p_\psi(\mathbf{z})\right) \quad (2)$$

where $p_\psi(\mathbf{z})$ is the prior of the distribution $q_\phi(\mathbf{z}|\mathbf{x}, l)$, ψ is the predetermined parameter of the distribution. We assume it obeys multivariate Gaussian distributions $\mathcal{N}(\mathbf{1}, \gamma\mathbf{I_d})$.

To make the embeddings answer-type dependent, we train a neural network as the *discriminator* to classify the adjusting vector into the corresponding answer type. The discriminator receives the adjusting vector $\mathbf{z}_j \in \mathbb{R}^d$ and classifies the vector into a corresponding answer type by using the predicting distribution $p_\pi(l|\mathbf{z}_j) = \frac{e^{f_l(\mathbf{z}_j)}}{\sum_{i=1}^{L} e^{f_i(\mathbf{z}_j)}}$, where $f_i(\mathbf{z}_j)$ denotes the logit of answer type label i given $\mathbf{z}_j$. Due to imbalanced label distribution among different answer types, followed Menon et al. (2020), we modify the distribution as: $p'_\pi(l|\mathbf{z}_j) = \frac{e^{f_l(\mathbf{z}_j)+\log\ p_l}}{\sum_{i=1}^{L} e^{f_i(\mathbf{z}_j)+\log\ p_i}}$, where p_i is the frequency of the answer type i and π is the parameters of the discriminator. The discriminator is trained via optimizing the following loss:

$$\mathcal{L}_D(\pi|\mathbf{z}) = \sum_{i=1}^{N}\sum_{j=1}^{P}\sum_{l=1}^{L} y_{il} p'_\pi(l|z_j^{(i)}) \quad (3)$$

where P is the length of the input sequence, L is the number of answer types. $y_{il} = 1$ when the i-th observation is related to the answer type l, otherwise $y_{il} = 0$. To fool the discriminator, the Adjustor needs to minimize the KL divergence between its distribution q_ϕ and the prior p_ψ in Eqn. 2.

The Final Objective Function. The final objective function is as follows:

$$\mathcal{L}(\theta, \phi, \pi) = \mathcal{L}_{\text{MLE}}(\theta) + \mathcal{L}_{\text{Adjust}}(\theta, \phi) + \alpha\mathcal{L}_D(\pi) \quad (4)$$

where α is the hyperparameter weighting between the question-answering predicting loss and the answer-type discriminating loss.

3.3 Denoising Filter

We design a denoising filter to further alleviate the negative effect of the noise in the data. The filter is composed of the *Top-K Filter* and the *Substring Filter*. To apply them, we firstly use the QA model to do inference on the unseen synthetic data. Then, if the synthetic answer falls in the K answers with the highest probability the model predicts (Top-K Filter) or the predicted instance is a substring of a NE-based answer span with predicting probability higher than γ (Substring Filter), we keep it. Otherwise, we remove the instance. Then, we use the filtered data to train our fine-tuned model. In this process, Top-K Filter aims to choose QA pairs with high confidence as low noise instances. Substring Filter keeps extra NE-based QA pairs with high predicting probabilities for training. The idea comes from an observation that a substring of a NE can probably represent the NE. For example, in the sentence *Apple CEO Tim Cook introduces*

Models	SQuADv1.1 EM/F1	TriviaQA EM/F1	NQ EM/F1	NewsQA EM/F1	BioASQ EM/F1	DuoRC EM/F1
Supervised Models						
Match-LSTM	64.1/73.9	-/-	-/-	-/-	-/-	-/-
BiDAF	66.7/77.3	-/-	-/-	-/-	-/-	-/-
BERT-base	81.2/88.5	69.4/74.3◁	66.1/77.9◁	49.4/64.4◁	-/-	-/-
BERT-large	84.2/91.1	75.7/80.2◁	69.0/80.8◁	56.0/71.0◁	-/-	-/-
Trained with Supervised Summarization Dataset						
Lyu et al. (2021)	65.6/74.5	36.7/43.0	46.0/53.5	37.5/50.1	32.0/43.2	38.8/46.5
Unsupervised Models						
Lewis et al. (2019)	44.2/54.7	19.1/23.8†	27.5/35.1†	19.6/28.5‡	18.9/27.0†	26.0/32.6†
RefQA	62.5/72.6	48.6/58.2‡	**43.4**/55.7‡	33.6/46.3	42.5/58.9‡	38.0/49.4‡
DiverseQA	**67.6/76.9**	**52.5/60.8**	41.3/**56.3**	**37.5/51.3**	**47.2/61.4**	**46.9/56.3**

Table 1: Results (EM/F1) of our method and various baselines on six different datasets. '†' denotes results taken from Lyu et al. (2021). '‡' denotes results from our reimplementation of RefQA(Li et al., 2020). '◁' denotes our fine-tuned BERT on supervised data. Because the pre-processed training sets of BioASQ and DuoRC in MRQA are not released, we don't fine-tune BERT on them.

two new products, "Tim Cook" is a NE referred to a specific person. Besides, "Tim" or "Cook" can also refer to him. Therefore, when the model predicts a substring of a NE with a high probability, it might refer to the original NE as the answer.

4 Experiments

4.1 Experiment Setup

Unsupervised QA Dataset Construction. We use Wikiref (Li et al., 2020) as the original text to construct QA pairs.

To extract answer spans, firstly, we use Spacy[2] to extract all of the named entities and their NER labels in the passage. Then, we apply Berkeley Neural Parser (Kitaev and Klein, 2018) to parse each sentence and extract a longer constituent containing a named entity with the constraint of $\omega\%$ sentence length as the final answer span. Here, we set $\omega = 80$. In our experiment, we consider named entity (NE), noun phrase (NP), adjective phrase (ADJP), verb phrase (VP), and sub-clause (S) as the candidate answer types. The dataset consists of 908,511 QA pairs. We randomly sample 300,000 to initially train a QA model, 600,000 to split them into N =6 parts (followed the empirical results in Li et al. (2020)) for the filtering phase.

Question Answering Model Settings. We use BERT as the backbone of our QA model. We use Adam (Kingma and Ba, 2014) as the optimizer. The learning rate is 3e-5 and the batch size is 24. The max sequence length is 384 and the doc stride is 128. The discriminator is set as a one-layer network. We set $L = 5, \alpha = 1, \beta = 1$. We use BERT-large-uncased-whole-word-masking, train the model for 2 epochs, save the checkpoint every 1,000 training steps and use the dev set to evaluate them for early stopping. Then, we continuously train the model with filtered data via the denoising filter, where $K = 1$ and the substring threshold $\gamma = 0.1$.

4.2 Results

We evaluate our model on SQuAD v1.1, NewsQA, TriviaQA, NaturalQuestions (NQ), BioASQ and DuoRC. We compare DiverseQA with supervised approaches (Wang and Jiang, 2016; Seo et al., 2016; Devlin et al., 2018), unsupervised approaches (Lewis et al., 2019; Li et al., 2020) and the approach using a supervised summarization dataset (Lyu et al., 2021). We use Exact Match (EM) and F1 score as our metrics. We use the pre-processed data provided in MRQA (Fisch et al., 2019).

The experimental results on six different benchmarks are shown in Table 1. The model trained on the synthetic QA data created by our DiverseQA reaches the state-of-the-art on five benchmarks, which shows the competitive performance of the proposed method in a wide range of domains. However, we find that our model underperforms on NaturalQuestions (NQ) dataset on exact match (EM). It is because the answers in this dataset are all entities, which means the model that only learns information from named entity[3] can have good performance. After all, the model only needs to choose a proper entity from an entity set rather than extract

[2] https://spacy.io

[3] Although 'named entity' and 'entity' are different, they share the common features in most aspects.

a possible span from the whole context, which finally leads to a good EM value. But as we know, named entities (or entities) are not the only answer type in the real world. Therefore, it's unfair for our method. Because Lyu et al. (2021) (the line under '*Trained with Supervised Summarization Dataset*' in Table 1) makes use of supervised summarization dataset XSUM (Narayan et al., 2018) to train the model, which is not a purely unsupervised method, we don't compare our method with it.

4.3 Analysis

We conduct experiments in this section to further understand our method. BERT-base-uncased is used to complete each experiment.

4.3.1 Effects of Different Components of DiverseQA

We conduct experiments on different components of DiverseQA. A brief illustration is as follows:

NeAnsQA Only extract NE as the answer type.

DiverseAnsQA Take the data-extending strategy proposed to build a dataset with diverse answers.

RandomAnsQA Each answer span in *RandomAnsQA* dataset has the same length with that of DiverseAnsQA while the answer span is randomly extended from the original NE-based answer.

Adjusting Vector It imposes the adjusting vector produced by the 'Adjustor' module on the embeddings of input tokens as an augmentation instance.

Answer-type Discriminator The module is to classify the adjusting vector into an answer type.

Top-K Filter Apply the Top-K Filter described in Section 3.3.

Substring Filter Apply the Substring Filter described in Section 3.3.

As shown in Table 2, the result illustrates that each component can improve the performance of our model. The difference between *NeAnsQA* and *DiverseAnsQA* shows that the use of our data construction strategy can greatly improve the performance of the model. It also demonstrates the importance of diverse answer types in unsupervised QA. Because answer length distribution is changed from NeAnsQA to DiverseAnsQA, we design *RandomAnsQA* to get rid of this extra factor. It shows that DiverseAnsQA still outperform RandomAnsQA by a large gap, which demonstrates the effectiveness of the proposed QA data construction strategy. The result of *Adjusting Vector* and *Answer-type Discriminator* shows that although adding the adjusting vector can slightly improve the performance, answer-type discriminator can continuously improve the performance, showing that adding the answer-type constraint to the distribution of the adjusting vector can benefit the performance of the model. To verify the necessity of the Discriminator, we also use a simple 'Answer-type Classifier' to classify input sequences into different answer types. The result shows that a simple classifier is not adequate to improve the method. In addition, the gains by adding the *Top-K Filter* reveal the importance of the filtering phase in the training process. What's more, *Substring Filter* can further gain the performance of the model. What's more, the results in the last row of Table 2 show that the Substring Filter is useful.

	EM	F1
NeAnsQA	49.2	59.3
RandomAnsQA	48.7	62.0
DiverseAnsQA	52.1	64.0
+ Answer-type Classifier*	51.3	63.6
+ Adjusting Vector	52.3	64.0
+ Answer-type Discriminator	52.9	64.6
+ Top-K Filter	54.7	65.6
+ Substring Filter	**55.0**	**66.2**

Table 2: Ablations on each component of DiverseQA method on the SQuADv1.1 development set. Components below the component with '*' are not added upon the '*' component.

ω%	20%	40%	60%	80%	100%
F1	63.0	63.6	65.4	**66.2**	62.2

Table 3: F1 scores of different Span Extending Thresold evaluated on SQuADv1.1 dev set.

4.3.2 Effects of Span Extending Threshold

We experiment with several Span Extending Threshold to construct the synthetic dataset. Table 3 shows that the optimal value is 80%. It illustrates that neither a too strict nor a too loose span extending condition can produce a high quality dataset.

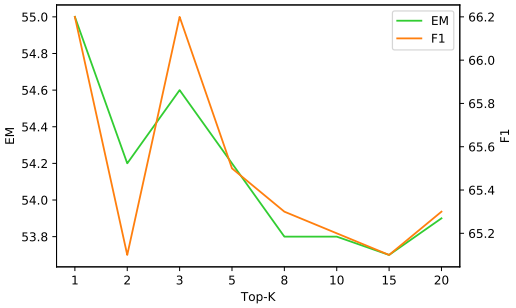

Figure 4: Results of using K answers with the highest predicting probabilities in the denoising filtering phase.

γ	0.0	0.1	0.2	0.4	0.6	0.8
F1	65.2	**66.2**	65.2	65.2	64.9	65.0

Table 4: F1 scores of different Substring Filter threshold evaluated on SQuADv1.1 dev set.

4.3.3 Effects of Denoising Filter

We conduct experiments on the K answers with the highest predicting probabilities as well as Substring Filter threshold γ, which will be used in the continuing training. Figure 4 shows that when K=1, the performance reaches the best. It means that when the model-predicted answer with high probability matches the span extracted via our extending strategy, the corresponding QA pair is probably of high quality and can be further reused to improve the performance of the model. Table 4 shows that when γ=0.1, the produced NE-based QA pairs with predictions of substring can be beneficial.

Furthermore, we make comparison between the proposed Denoising Filter and the Refining Phase (Li et al., 2020). Table 5 shows that firstly, DiverseQA largely outperforms RefQA without any filter. Secondly, the proposed Denoising Filter can be effectively used in the second phase of both RefQA and DiverseQA, showing its strong adaptation to different methods, while Refining Phase can only make improvement in RefQA. Thirdly, although the F1 score of RefQA with Denoising Filter is slightly lower than that of Refining Phase, it makes large improvement on the EM metric, demonstating the advantage of the proposed Denoising Filter.

4.3.4 Effects of Diverse Answer Types

We experiment on the SQuADv1.1 dev set partitioned by answer types. For a fair comparison, the re-implement of RefQA shares the raw text (from which we generate QA pairs and train the model) and the random seed with ours.

	No Filter	Refine	Denoise
RefQA	49.2/59.3	52.0/62.6	53.3/62.2
DiverseQA	**52.1/64.0**	**52.1/64.0***	**55.0/66.2**

Table 5: EM/F1 scores of RefQA and DiverseQA with different continuously training strategy. "Refine" denotes the Refining phase in RefQA. "Denoise" denotes the proposed Denoising Filter. '*' means that the model cannot make any improvement on the specific continuous training strategy.

Although Li et al. (2020) claims that their refine phase can generate diverse answers, the result in Table 6 shows that DiverseQA outperforms the performance of RefQA on nearly every kind of answer type, especially on VP and S by large margins. It is because the refine phase proposed in RefQA heavily relies on the QA model training with purely NE-based QA pairs. Therefore, the trained model could probably generate certain variants of NEs, which are short and used to continuously train the model, leaving relatively long answer spans like VPs and Ss out. Besides, we observe that the performance of DiverseQA is lower than RefQA on ADJP in the exact match (EM). It is because the ADJP merely accounts for 0.3% (in Appendix A) in the whole synthetic dataset generated by DiverseQA, while the refining phase in RefQA may probably generate many variants of NEs, which could be ADJPs[4]. Consequently, the model in RefQA might train with too many ADJP-like QA instances and prefer to choose the ADJPs as answer types, leading to the result in Table 6. Since in our method, we only consider NE, NP, ADJP, VP, S as our answer spans, it's important to know how well our method performs on other types of answers out of consideration. The result in the '*Others*' column of Table 6 demonstrates that the proposed model can generalize to other unseen answer types and outperform RefQA by a large gap.

In addition, we ablate the proposed span extending strategy and Answer-type Discriminator separately to explore the performance of the two components. The result in the last two rows of Table 6 shows that the two components can both contribute to the performance of model on most of answer

[4]For instance, in '*he was still being paid more than $ 10,000 as a legal advisor to the Chicago*', span '*10,000*' is a named entity with NER label 'MONEY' while '*$ 10,000*' is a variant of the named entity as well as an ADJP.

Models	NE EM/F1	NP EM/F1	ADJP EM/F1	VP EM/F1	S EM/F1	Others EM/F1	Overall EM/F1
RefQA	67.0/75.4	47.9/60.4	**31.9**/41.4	4.1/18.3	22.7/35.0	36.7/52.2	52.0/62.6
RandomAnsQA	58.8/70.7	43.2/58.8	25.0/39.7	9.4/25.6	21.4/39.0	36.8/53.4	48.7/62.0
DiverseAnsQA	62.3/72.4	47.2/61.3	26.5/39.3	13.2/29.3	22.9/41.0	38.5/54.6	52.1/64.0
DiverseQA	67.5/75.7	**49.1/62.8**	27.9/**42.5**	**13.3/30.4**	**29.1/42.6**	**41.4/57.2**	**55.0/66.2**
w/o Extension	**71.5/77.3**	48.5/60.2	24.5/35.1	1.6/12.1	21.9/30.6	34.4/47.0	52.0/60.6
w/o Discriminator	66.9/74.7	47.6/60.1	28.4/40.5	9.7/28.0	26.3/40.9	40.4/54.7	53.6/63.9

Table 6: EM/F1 scores of different models on the SQuADv1.1 dev set. Columns are each dev set associated with the type of answer. 'Others' means other types of answers out of our consideration. 'Extension' represents our answer span extending module. 'Discriminator' denotes the proposed Answer-type Discriminator.

Context: The Town of Estill is located in the southern half of Hampton County .	
Question: Where of the southern half in The town of Estill is located	**Question:** Where of the town Estill
The answer in RefQA: Hampton County	**Our answer:** is located in the southern half of Hampton County

Table 7: An example of RefQA (left half) and our QA pairs (right half).

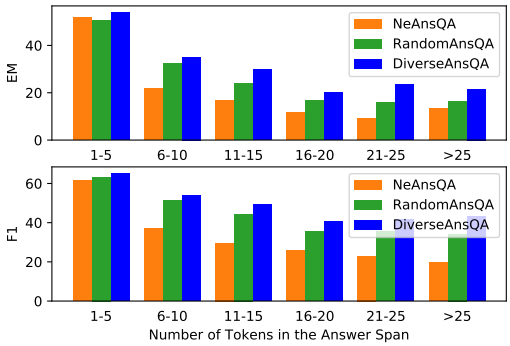

Figure 5: Comparison among *NeAnsQA*, *RandomAnsQA* and *DiverseAnsQA* (Section 4.3.1). SQuADv1.1 dev set is partitioned by answer length for evaluation.

types. Besides, we notice that the performance of "w/o Extension" on the NE obtains the best. It is because that under this setting, the model will only learn from NE-based QA pairs. Therefore, it prefers to choose a NE from the context, which can gain its performance on NE-based QA pairs.

What's more, we also show an example to make the comparison between the QA pair in DiverseQA and that of RefQA in Table 7. In the example, the question 'Where of the town Estill' needs more reasoning process to be answered correctly than the question generated in RefQA. As in the proposed method, when the generated answer becomes longer, the generated question becomes shorter. Therefore, it's also necessary to keep the question and answer length unchanged while exploring the effectiveness of answer diversity. In Table 6, the results of RandomAnsQA and DiverseAnsQA demonstrate the usefulness of the proposed answer extension method when both the question and answer length distributions are unchanged. More examples are shown in Appendix D.

4.3.5 Effects of Answer Length Distribution

In the proposed method, the first module is to extend the span into a longer constituent. Although it indeed introduces new answer types and gains performance, answer length distribution also changes simultaneously. Therefore, we further explore how the drift of answer length distribution affects the model's performance.

As shown in Figure 5, the performance of NeAnsQA is lower than RandomAnsQA and DiverseAnsQA. When the answer length becomes larger (>10 tokens), the gap is even larger. This demonstrates that model trained only with NE-based QA pairs lacks the ability to handle QA pairs with long answers. Besides, it can be found that the performance of RandomAnsQA is lower than DiverseAnsQA on each answer length. It means that without the influence of answer distribution, the proposed span extending strategy can generate higher quality instances than that of randomly extending strategy.

4.4 Few-Shot Learning

We conduct experiments in the few-shot learning setting. For a fair comparison, we use our best-trained model without the data augmentation component, our re-implementation of RefQA, and a BERT model, all of which use the BERT-large-whole-word-masking as the backbone.

Figure 6 shows that our model reaches the best

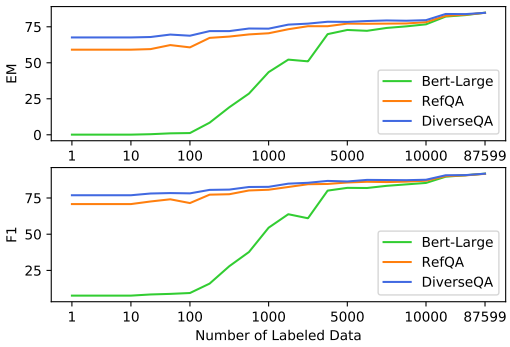

Figure 6: Few-shot results (EM and F1) using different sizes of SQuADv1.1 training data, comparing among DiverseQA, RefQA (Li et al., 2020) and BERT-large-uncased-whole-word-masking.

result trained on various sizes of supervised data ranging from 1 to 50,000, demonstrating the strong ability of our method in the low-resource scenario.

5 Conclusion

We propose DiverseQA, an unsupervised QA method comprising a synthetic QA dataset with diverse answers, an answer-type-dependent data augmentation process via adversarial training, and a denoising filter to improve the performance of a QA model. Our method reaches the state-of-the-art on five benchmarks and shows strong performance in the few-shot learning setting.

Acknowledgements

The work is supported by National Key R&D Plan (No. 2020AAA0106600), National Natural Science Foundation of China (No. U21B2009, 62172039 and L1924068). We would like to acknowledge Yuming Shang for the helpful discussions.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13.

Giwon Hong, Junmo Kang, Doyeon Lim, and Sung-Hyon Myaeng. 2020. Handling anomalies of synthetic questions in unsupervised question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3441–3448.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Seanie Lee, Minki Kang, Juho Lee, and Sung Ju Hwang. 2021. Learning to perturb word embeddings for out-of-distribution qa. *arXiv preprint arXiv:2105.02692*.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910.

Zhongli Li, Wenhui Wang, Li Dong, Furu Wei, and Ke Xu. 2020. Harvesting and refining question-answer pairs for unsupervised qa. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6719–6728.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Chenyang Lyu, Lifeng Shang, Yvette Graham, Jennifer Foster, Xin Jiang, and Qun Liu. 2021. Improving unsupervised question answering via summarization-informed question generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4134–4148.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2020. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

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*, pages 2383–2392.

Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. 2018. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1693.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1–28.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Appendix

A The Distribution of Extracted Answers

The distribution of our extracted answer types (NE, NP, ADJP, VP, S) is shown in Table 8. The frequencies of each answer type are used in the modified predicting distribution described in Section 3.2.2. Since previous works consider named entities as the only answer type (i.e. NE acounts for 100% of all the answers), we don't show the answer type distribution of them. Besides, as the answer span generated by the trained QA model in the refining phase of Li et al. (2020) fails to follow the rule of constituent parsing, we cannot obtain its answer type distribution in terms of sentence constituents.

	#Instances	%Frequency
NE	716,716	78.9%
NP	161,178	17.7%
ADJP	2,563	0.3%
VP	23,420	2.6%
S	4,634	0.5%

Table 8: The statistics of extracted answer types in the synthetic dataset constructed using DiverseQA.

B Algorithms

We describe the whole training procedure as follows:

Algorithm 1: Training Procedure

Data: Synthetic QA dataset $\mathcal{D}$, a BERT model $\mathcal{M}$ with answer-type dependent data augmentation module, a denoising filter composed of a Top-K filter and a Substring Filter with threshold γ.

Result: A fine-tuned model $\mathcal{M}'$.

Split $\mathcal{D}$ into $\mathcal{D}_I$ and $\mathcal{D}_F$;
Fine-tune $\mathcal{M}$ with $\mathcal{D}_I$;
Split $\mathcal{D}_F$ equally into $\{\mathcal{D}_{F_i}\}_{i=1}^{N}$;
for $i \leftarrow 1$ **to** N **do**
    $\mathcal{S} \leftarrow \emptyset$;
    foreach element e in $\mathcal{D}_{F_i}$ **do**
        Obtain the probability p_e using $\mathcal{M}$;
        if p_e is in the Top-K highest probabilities **then**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{e\}$
        end
        if the answer of e is a subtring of the extracted one **and** $p_e \geq \gamma$ **then**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{e\}$
        end
    end
    Fine-tune model M with dataset $\mathcal{S}$;
end

C Answer Length Distribution

We randomly sample 10,000 instances from SQuADv1.1, the dataset of RefQA, and the dataset of DiverseQA respectively, and count the answer length distributions, which are shown in Figure 7. It

Context: Joss Whedon has endorsed Mitt Romney (in a way) and now " The Simpsons " Mr. Burns , owner of Springfield 's nuclear power plant , titan of corporate capitalism and honcho in the Springfield Republican Party , has come out with his own backhanded endorsement of the Republican nominee .	
Question: Who As the chief of "Springfield Republican Party" endorsed Mitt Romney in 2012 US Presidential Election . **Answer:** Burns	**Question:** Who As the chief of endorsed Mitt Romney in 2012 US Presidential Election . **Answer:** Mr. Burns
Context: In 1931 , the Singers gave the Museum of Fine Arts to the community along with a substantial collection of American and European art .	
Question: Who American and art of a substantial collection with along the Singers gave the Museum of Fine Arts to the community . **Answer:** European	**Question:** Who art of a substantial collection with along the Singers gave the Museum of Fine Arts to the community . **Answer:** American and European
Context: Why do the new prequels sometimes contradict the history set forth in THE DUNE ENCYCLOPEDIA compiled by, Dr. Willis E. McNelly ?, THE DUNE ENCYCLOPEDIA reflects an alternate " DUNE universe " which did not necessarily represent the " canon " created by Frank Herbert ."	
Question: Who by written to accompany the "Dune" books **Answer:** Willis E. McNelly	**Question:** Who by written to accompany the "Dune" books **Answer:** Dr. Willis E. McNelly
Context:GDP per capita in the city increased by 2.4 per cent and employment by 4.7 per cent compared to the previous year .	
Question: How much by GDP per capita in the city increased **Answer:** 2.4 per cent	**Question:** How much in GDP per capita **Answer:** increased by 2.4 per cent

Table 9: Examples of the QA pairs of RefQA (left half) and that of DiverseQA (right half).

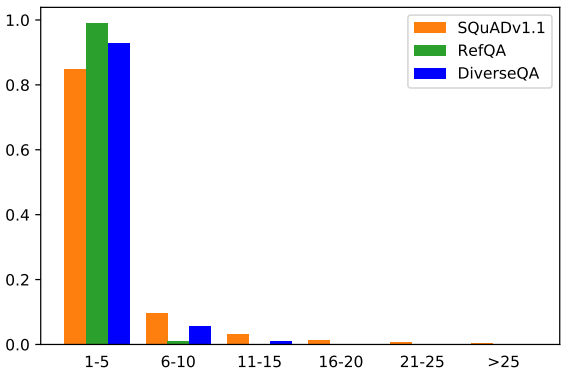

Figure 7: Answer length distributions.

shows that firstly, the proposed DiverseQA method can construct dataset with a similar answer length distribution to the annotated dataset SQuADv1.1. Secondly, the dataset only with NE-based QA pairs (like RefQA) is not able to cover long form answers (it accounts for 0% QA pairs in the answer length ranges '11-15', '21-25' and '>25')[5], which is harmful to the performance of model on long answers.

D Generated QA Instances

Examples of generated QA instances are shown in Table 9.

[5]The dataset of DiverseQA accounts for 0.16%, 0.13% and 0.03% in the ranges '16-20', '21-25' and '>25'. And it is not displayed properly in Figure 7.