Machine Reading Comprehension: Generative or Extractive Reader?

Anonymous ACL submission

Abstract

While both extractive and generative readers have been successfully applied to the Question Answering (QA) task, little attention has been paid toward the comparison of these two readers. Which reader performs better? What are the reasons for the performance differences? In this paper, we aim to answer these questions in the setting of extractive QA tasks. We design multiple transformer-based models and different scenarios to systematically compare these two readers. Our findings characterize the difference of two readers and their pros and cons, which can instruct the optimal selection of the two readers, and open up new research avenues to improve each reader. Our major findings are: 1) generative readers perform better when the input context is long, whereas extractive readers are better when the context is short; 2) extractive readers generalize better as compared to the generative ones under out-of-domain settings, in both single- and multi-task learning scenarios. Our experiments also suggest that, although an encoder-only pre-trained language model (PrLM) is an intuitive choice for extractive readers, the encoder from encoder-decoder PrLM is a strong alternative that performs competitively.

1 Introduction

Question Answering (QA) is an important subtask of reading comprehension and can be directly used in real applications such as search engines (Kwiatkowski et al., 2019) and dialogue systems (Reddy et al., 2019; Choi et al., 2018). Extractive question answering is a specific type of QA; i.e., the answer to the question is a span in the context (Rajpurkar et al., 2016; Fisch et al., 2019), and this work focuses on such QA tasks. Extractive readers (Seo et al., 2017; Devlin et al., 2019) are widely used to effectively tackle such a task, where the goal is to classify start and end positions of the answer in the context. Generative readers (Raffel et al., 2020; Lewis et al., 2020b; Izacard and Grave, 2021) have also shown remarkable performance on QA tasks, where the goal is to generate answers by autoregressively predicting tokens.

While both extractive and generative readers have been successfully applied to the QA task, a natural question arises: which approach is better, extractive or generative? This question is appealing as (1) it can provide guidance on which reader should be applied in certain cases; (2) it reveals the pros and cons of the two reader approaches, and thus opens up opportunities to improve each reader. Motivated by the aforementioned, we design multiple transformer-based models and scenarios (e.g., single-tasks, multi-task learning, short and long context settings, etc.) to compare these two reader approaches.

Our first comparison is to examine which answer prediction approach is better (i.e. classifying the start and end positions v.s. generating sequential tokens) when we have two comparable readers (in terms of the model size and their training corpus). Previously, extractive and generative readers are usually based on different PrLMs, for example, XLNET (Yang et al., 2019) for extractive and T5 (Raffel et al., 2020) for generative readers. Under such circumstances, both the pre-training corpus and the pre-training objective are different. Thus, it is hard to attribute whether the performance difference is due to their differences in training objectives, pre-training corpus, or the way the answer is obtained. To address this concern, we use T5 PrLM for both the readers and make their model size comparable such that the only variation is in the way that a reader makes predictions. As such, the comparison focuses more on the answer prediction approaches and reduces the influence introduced by other factors. To the best of our knowledge, this is the first work comparing the two readers with minimal distinctions. In addition, our work is the first attempt to employ the encoder in a pre-trained encoder-decoder model for extractive readers.

Towards better generalizability and grounding of our conclusions, we also apply ELECTRA (Clark et al., 2020) for the extractive reader and compare it with the T5 extractive and generative readers. Together, we present a systematic evaluation for comparing three readers (T5 generative, T5 extractive, and ELECTRA extractive readers) on the MRQA task (Fisch et al., 2019), a collection of multiple extractive QA datasets. Our experiments reveal several interesting insights:

1. Extractive readers can generalize better in out-of-domain (OOD)[1] datasets as compared to the generative reader. Additionally, extractive readers are better at learning from different domains; i.e. extractive readers benefit more from multi-task learning.
2. Generative readers have an advantage when the input context is long, and the performance is not affected by variations in the context length. On the other hand, extractive readers are better at short context but show sensitivity to the context length.
3. Previous work used to apply encoder-only PrLM to the extractive reader, however, our experiments indicate that the encoder of encoder-decoder models (e.g. T5) is a viable alternative with strong performance. It can even outperform traditional encoder-only extractive readers when the model learns from different domains or when the context is long.

Along with the comparison, our experiments also reveal that the inference length affects the performance significantly – encoding the full context during inference leads to significantly better performance as compared to truncating and only encoding to its training context length.

While the focus of this paper is the comparison of different readers, we also explore two constrained generations for the generative reader. Traditionally, the decoder generates a token from the entire token vocabulary. A natural but under-explored question is whether it is beneficial to constrain the generation space with input context, especially for extractive QA tasks, where an answer is supposed to be extracted from a given context. To answer this question, we introduce two constrained generations conditioned on the context (§3.2). Surprisingly, we find that these additional conditions result in no improvements. Further analysis reveals that in most cases, the decoder, without any condition, already generates tokens from the context tokens and thus the extra constraints have no additional impact. In other cases, the generative reader generates the same surface form as in context, but with a different sequence of subword tokens,[2] and limiting the generation space results in the refusal of potentially correct answers.

To summarize, our contributions are threefold: 1) we systematically compare extractive reader and generative reader, and our findings can instruct the optimal selection of extractive and generative readers under certain conditions; 2) our experiments suggest that the encoder from encoder-decoder PrLM architecture is a strong candidate that can be used for extractive readers; 3) for the generative reader, we conclude that constrained generation does not lead to noticeable improvement because the decoder already generates tokens from the input context, in most cases.

2 Related Work

Pretrained Language Models Here, we mainly discuss two types of pre-trained models based on transformers architecture (Vaswani et al., 2017), Autoencoder and sequence-to-sequence models. Autoencoder only relies on the encoder part in the original transformer, and in the pretraining time, the input is a corrupted sentence, for example, a sentence with mask tokens, such as BERT (Devlin et al., 2019) and its variants (Liu et al., 2019; Lan et al., 2020). ELECTRA (Clark et al., 2020) also belongs to this family. ELECTRA adapts GAN-style training (Mirza and Osindero, 2014) where a language model is given a sentence with masked tokens and outputs a corrupted sentence, ELECTRA then aims to detect if a token is replaced or is from the original text. Sequence-to-sequence models keep both the encoder and decoder, for example, BART (Lewis et al., 2020a) and T5 (Raffel et al., 2020). While most PrML is pre-trained on Wikipedia, T5 is pre-trained on Colossal Clean Common Crawl Corpus[3].

Question Answering Systems We focus on QA systems that are built upon PrLMs. Extractive QA readers assume that answers can be found in the context and aim to predict the corresponding start and end tokens from the context (Fisch et al., 2019; Li et al., 2019; Clark et al., 2020; Karpukhin et al., 2020). Differently, generative QA readers are not

[1] In this paper, we use OOD to represent out-of-domain

[2] In this paper, a "token" corresponds to a "subword token."

[3] https://www.tensorflow.org/datasets/catalog/c4

restricted to the input context, where they can freely generate answers token by token using the entire vocabulary in an autoregressive manner (Raffel et al., 2020). Generative readers are more often used in open domain (Lewis et al., 2020b; Izacard and Grave, 2021; Xiong et al., 2021) and unified settings (Khashabi et al., 2020; Tafjord and Clark, 2021). Fajcik et al. (2021) combines extractive and generative readers by adding a classification module to decide which reader predicts answers. Cheng et al. (2021) proposes a unified system of extractive and generative readers, but different from (Fajcik et al., 2021), the output is computed by both extractive and generative readers.

3 Model

3.1 Extractive Reader

In extractive reader, an encoder firstly receives a question $\mathbf{q}:\{q_1,\ldots,q_t\}$ as well as a context $\mathbf{c}:\{c_1,\ldots,c_m\}$, where q_i and c_j are tokens in question and context, respectively. Then, it produces contextual representations $\{h_1,\ldots,h_m\}$, denoted by $\mathbf{h}$. Last, two linear layers predict the probability of each token in $\mathbf{h}$ of being start and end positions independently. More formally, given a tuple $(\mathbf{q},\mathbf{c},\mathbf{a})$, where $\mathbf{a}$ is an answer, the training objective is to minimize the following loss function

$$\mathcal{L}_{\text{Ext}} = -\log(\mathbf{P}_{\mathbf{start},\mathbf{s}}) - \log(\mathbf{P}_{\mathbf{end},\mathbf{e}}) \quad (1)$$

where $\mathbf{P}_{\mathbf{start}}$ and $\mathbf{P}_{\mathbf{end}}$ are defined by

$$\mathbf{P}_{\mathbf{start}} = \text{softmax}(\mathbf{w}_{\mathbf{start}}\mathbf{h}) \quad (2)$$

$$\mathbf{P}_{\mathbf{end}} = \text{softmax}(\mathbf{w}_{\mathbf{end}}\mathbf{h}) \quad (3)$$

where $\mathbf{P}_{\mathbf{start},\mathbf{s}}$ and $\mathbf{P}_{\mathbf{end},\mathbf{e}}$ denote the probability of the ground truth start and end tokens of answer $\mathbf{a}$, respectively. In testing time, the answer span is decoded by $\text{argmax}_{i,j}\{\mathbf{P}_{\mathbf{start},\mathbf{i}} \times \mathbf{P}_{\mathbf{end},\mathbf{j}}\}$. In this work, we apply two encoders, T5 and ELECTRA.

When T5 is applied to QA tasks, previous work tends to take it as a generative reader (§3.2). Differently, we also use the encoder from T5 in an extractive reader, where a classification layer is added on top of the encoder. In addition, we also tried to use the entire T5 model and add a classification layer on top of the decoder. The latter performed consistently worse as compared to the former (see Appendix E). Therefore, we use the former and denote it as T5-Ext (i.e. encoder+linear layer) as an extractive reader from T5 in later experiments.

ELECTRA is an encoder-only model that is pretrained on detecting replaced tokens in the input, which is similar to the QA downstream task since both are at the token level. Building an extractive reader on top of ELECTRA has obtained rather good performance on QA tasks (Xiong et al., 2021), thus, we also use it to compare with T5 PrML.

3.2 Generative Reader

In a generative reader, an encoder, similar to the extractive reader, takes a question $\mathbf{q}$ and a context $\mathbf{c}$ as input and outputs contextual representation $\mathbf{h}$. A decoder takes the previously generated answer tokens as input and performs attention over $\mathbf{h}$ and then generates a token. Formally, given a tuple $(\mathbf{q},\mathbf{c},\mathbf{a})$, the training objective is to minimize the following loss function

$$\mathcal{L}_{\text{Gen}} = \sum_{i=1}^{K} \log \mathbf{P}(a_i \mid \mathbf{h}, a_{:i}) \quad (4)$$

where K is the number of tokens in answer $\mathbf{a}$, a_i is the i^{th} token in $\mathbf{a}$, and a_0 corresponds to a special beginning of sequence (`BOS`) token. In the inference time, we use the greedy search method to autoregressively generate the answer.

Here, we further investigate variants for the generative reader from training target and answer generation aspects.

Training Targets T5 uses SentencePiece (Kudo and Richardson, 2018), a subword tokenization toolkit, such that the same word can be represented by different tokens if the surrounding contexts are different. For example, a single word *NASA* is represented by one token {'_NASA'}, while in context *"... the National Aeronautics and Space Administration (NASA),..."*, *NASA* is represented by two tokens { 'NAS', 'A'}. In conventional training, the targets are tokens of answer without any surrounding context (e.g. {'_NASA'}). Such kinds of targets are not assured to be the tokens in the given context. To better fit the purpose of extractive QA tasks, we extract the tokens of answer in the context as the target, (e.g. { 'NAS', 'A'}). We term the conventional one as A-Target (i.e. direct tokenization of answer), and the new one as C-Target (i.e. use tokens in the given context as answer targets).

Constrained generation Usually, the decoder generates tokens from the entire vocabulary. However, due to the nature of the extractive QA task,

Dataset	Training size	Avg. tokens in Q	Avg. tokens in C
In-domain datasets			
SQuAD	86,588	14.7	179.87
NewsQA	74,160	9.79	699.75
TriviaQA	61,688	20.49	1071.66
SearchQA	117,384	23.74	1081.63
HotpotQA	72,928	24.29	315.41
NQ	104,071	12.87	359.2
Out-of-domain datasets			
DROP	-	14.0	279.02
RACE	-	13.98	403.77
BioASQ	-	19.07	368.89
TextbookQA	-	13.76	792.493
RE	-	13.43	42.97
DuoRC	-	11.61	932.08

Table 1: Statistics of in-domain data of MRQA shared Task.T5 tokenizer is used to obtain the tokens.

only spans presented in the context are valid. Restricting the generating space by the valid tokens can prevent the model from generating invalid tokens and thus is likely to improve the accuracy. The same intuition had been mentioned in (Xue et al., 2021). Motivated by this, we explore two constrained generations. In *Context constrained generation (C-Con)*, the decoder can generate any token given in the context. In *Next Token constrained generation (NT-Con)*, the decoder can generate any token which are the next tokens of the current predicted one in the context. The decoder is allowed to generate *end token* anytime to denote the end of prediction. We only enforce these conditions in the testing time and the generation in training time is still across the entire vocabulary.

4 Experiments

4.1 Dataset

Table 1 shows the training size of in-domain (IID)[4] datasets and the number of average tokens in questions and context of IID and OOD datasets. We present the histogram of the context length of every dataset in Appendix C. It is easy to see that some datasets have longer context as compared to others, thus we further group them into long/short context datasets. The long context datasets include TriviaQA, SearchQA, TextbookQA, and DuoRC. The remaining datasets are in the short context category. We find that the context length has a strong impact on models' performance (§5).

4.2 Learning Strategy

Single Task Learning: we use each IID datasets to train extractive and generative readers. **Multi-Task Learning:** we consider training with all (six) IID datasets as multi-task learning for two reasons. As (Su et al., 2019) showed that different IID datasets share a low similarity, therefore, they may require different reasoning skills. In addition, Table 1 shows that different datasets have different question and context lengths, which may lead to different difficulties between datasets.

[4]In this paper, we use IID to represent in-domain

	T5-Ext	ELECTRA	T5-Gen	T5-Gen
Size	large	large	base	large
# Params	335M	334M	223M	737M

Table 2: Size and Number of parameters of each reader.

4.3 Experimental Setup

We use Huggingface (Wolf et al., 2020) and Pytorch (Paszke et al., 2019) implementation for training each model. Detailed information on hyperparameters and the type of hardware we use for our experiments is given Appendix A. **Model Size** To have a fair comparison, we make the number of parameters of extractive and generative readers as close as possible (see Table 2). To do this, we use T5-large for T5-encoder extractive reader, ELECTRA-large for ELECTRA extractive reader, and T5-base for generative readers. For the rest of the paper, we use T5-Ext, ELECTRA, and T5-Gen to represent each reader. For the purpose of comparison, using T5-base for generative reader is more suitable than T5-large since the advantage of T5-large generative over T5 extractive reader is a lot more significant than the advantage of T5 extractive reader over T5-base generative reader. **Input Format** Given a question **Q** and a context **C**, the input to extractive readers is {**Q** [SEP] **C**} and the input to generative readers is {*question:* **Q** [SEP] *context:* **C**}. We also tried other alternatives and results are given Appendix E.

5 Results and Analysis

In this section, we present the results and analysis of three readers. For generative readers, we use C-Target to train the model and the regular generation method. By our experimental result (Appendix D), C-Target and A-Target do not make a noticeable difference when regular generation is applied, but the C-Target model is more stable than A-Target regarding different generation methods. We use the maximum answer length to be 16 in the testing time for T5-Gen. One can find the performance of

other lengths (32 and 64) in Appendix H, which do not result in a noticeable difference.

5.1 Comparison of Readers with Same PrML

Single-Task Learning The first two rows in Single task learning in Table 3 present the performance of T5-Ext and T5-Gen readers. For IID datasets, we train a model on dataset A and evaluate it on the same dataset A. Comparing the results of two readers, T5-Ext achieves better performance than T5-Gen on four out of six datasets. On the other two datasets, TriviaQA and SearchQA, which are of long context, T5-Gen outperforms T5-Ext by more than 3%, which leads to a higher average F1 than T5-Ext. For OOD datasets, we evaluate each single-task model on every dataset and present the result which is the best among six single-task models[5]. We see that on one long context dataset, i.e. TextbookQA, T5-Gen achieves a much higher F1 score than T5-Ext. This is consistent with our previous observation and shows the advantage of T5-Gen in a long context setting. For the other five datasets, T5-Ext achieves better performance, especially on three datasets, DROP, RACE, and BioASQ, T5-Ext is much better, indicating that T5-Ext generalizes better on OOD datasets. We hypothesize that this is because there are more unseen (or less observed) tokens present in the answer for the OOD datasets, which presents a challenge for the generative reader. For generative readers, when a token is less observed during training, it is less likely to get predicted at test time.

Multi-Task Learning For multi-task models, we present the results using full data in training. Following (Fisch et al., 2019), we also compare full data training with down-sampling data and find that full data is slightly better than down-sampling (see Appendix G). From the first three rows in the multi-task learning block in Table 3, we find that the performance of the extractive reader improved significantly when trained on all datasets. Specifically, the average performance of T5-Ext on the IID and OOD datasets are both increased compared to the single-task model (11 out of 12 datasets are increased). On the other hand, though the generative reader benefits from multi-task learning on OOD datasets, the performance decreases on three IID datasets. We also notice that on the dataset when two models are both improved, T5-Ext always achieved larger improvement as compared to T5-Gen. For OOD datasets, T5-Ext still performs better than T5-Gen, similar to single-task learning, which indicates the potential advantage of extractive readers in OOD settings.

Long and Short Context We further investigate the short and long context within each dataset. Specifically, given a question and a context from a dataset, if the context exceeds 800 tokens[6], then we consider such pair as a long context question; otherwise, a short context question. We then obtain a mixed long context question set that contains questions and contexts from all datasets, similarly for a mixed short context question set. Table 4 shows how two readers perform on each set. In the mixed long context question set, T5-Gen is better than T5-Ext in both IID and OOD datasets while in the mixed short context question set, T5-Ext performs better. It is worth mentioning that although the average F1 score of OOD achieved by T5-Ext is higher than T5-Gen, the latter achieves higher F1 on the subset of long context questions. We also observe that T5-Gen performs more stable in terms of different context lengths, shown by the smaller gap of F1 score between the mix short and long context (e.g. for IID datasets, the gap of T5-Ext is 4.45%, and the gap of T5-Gen is 0.6%). We conjecture that T5-Gen is more stable w.r.t the context length because the decoder always generates tokens from the entire vocabulary that is unlikely to be affected by the input length. However, the situation is quite different for the T5-Ext setting. Because the classification layer needs to classify every token, the longer the input is, the larger the classification space becomes, in other words, the more difficult the prediction becomes.

Rare Tokens When looking into the answers of testing sets, we find that few answers include rare characters such as *ń* and *ł*, which raises a question *does rare token affect the performance of the model?* Driven by this question, we define a list of normal characters [7]. If an answer includes any character which is not in the normal character list, we consider it belongs to the rare answer set, otherwise, it belongs to the normal answer set. The percentage of rare cases for IID and OOD datasets is 1.4% and 2%, respectively. From Table 5, first

[5]The performance of each single-task model is given Appendix Table 12.

[6]We choose 800 based on the statistic shown in Table 1

[7]Normal characters are obtained by the printable characters in the string library of Python including lower and upper case alphabets, digits, punctuation, and white-space.

Model	In-domain Datasets							Out-domain Datasets						
	SQuAD	NewsQA	TQA	SQA	HQA	NQ	Avg.	DROP	RACE	BioASQ	TbQA	RE	DuoRC	Avg.
Single Task Learning														
T5-Ext	92.27	**72.76**	76.34	82.78	80.05	**80.55**	80.79	53.47	49.05	**70.29**	49.69	85.61	**62.37**	61.75
T5-Gen	91.33	71.6	**80.1**	**85.93**	79.88	78.17	**81.17**	48.09	46.99	66.33	**60.74**	85.34	61.56	61.51
ELECTRA	**93.08**	65.21	76.51	82.93	**81.21**	79.69	79.77	**58.74**	**50.86**	70.21	50.94	**87.36**	59.83	**62.99**
Multi-Task Learning														
T5-Ext	92.84$_{+0.57}$	**73.27**$_{+0.51}$	78.01$_{+1.67}$	83.66$_{+0.88}$	82.24$_{+2.19}$	**81.0**$_{+0.45}$	**81.84**$_{+1.05}$	60.96$_{+7.49}$	**53.48**$_{+4.43}$	69.52$_{-0.77}$	60.94$_{+11.25}$	86.71$_{+1.1}$	**64.37**$_{+2.0}$	**66.0**$_{+4.25}$
T5-Gen	91.54$_{+0.21}$	71.4$_{-0.2}$	**80.76**$_{+0.66}$	**85.83**$_{-0.1}$	78.98$_{-0.9}$	78.26$_{+0.09}$	81.13$_{-0.04}$	51.98$_{+3.9}$	49.51$_{+2.5}$	69.21$_{+2.9}$	**61.66**$_{+0.9}$	85.42$_{+0.1}$	63.02$_{+1.5}$	63.47$_{+1.96}$
ELECTRA	**93.92**$_{+0.84}$	65.33$_{+0.12}$	73.63$_{-2.88}$	81.94$_{-0.99}$	**83.36**$_{+2.15}$	79.98$_{+0.29}$	79.69$_{-0.08}$	**61.77**$_{+3.03}$	52.71$_{+1.85}$	**71.86**$_{+1.65}$	54.8$_{+3.86}$	**87.18**$_{-0.18}$	59.76$_{-0.07}$	64.68$_{+1.69}$
T5-Gen (large)	93.54	73.44	84.08	88.38	82.88	80.58	83.82	63.63	55.87	72.33	68.76	87.33	68.09	69.33

Table 3: Three readers trained by single and multi task learning and evaluated on in-domain and out-domain datasets by F1 Score. TQA: TriviaQA; SQA:SearchQA; HQA:HotpotQA; NQ: NaturalQuestions; TbQA:TextbookQA; RE:RelationExtraction; Avg.: the Macro Average of in-domain/out-domain datasets. **Bold** values are the best performance in a column for each block.

Context type	Model	IID	OOD
Mix Long	T5-Ext	80.73	61.32
	T5-Gen	**82.46**	**61.64**
Mix Short	T5-Ext	**85.18**	**72.98**
	T5-Gen	83.06	69.96

Table 4: Compare T5-Ext and T5-Gen on the mixed long/shot context questions sets.

Answer type	Model	IID	OOD
Rare	T5-Ext	**78.77***	**83.5***
	T5-Gen	69.11	63.56
Normal	T5-Ext	**83.14**	**69.91**
	T5-Gen	82.97	68.05

Table 5: Compare T5-Ext and T5-Gen reader on rare and normal answers, * denote significantly better than others on the same answer type and same domain.

we see that the extractive reader performs better in both rare and normal cases, but more importantly, the performance of T5-Gen on rare answer sets decreases significantly, indicating that the generative reader is more sensitive to rare characters. A similar reason for relatively poor performance on OOD dataset can also be considered here – in the training time, the frequency of these rare characters is much less, and hence are less likely to be predicted during test time. Table 6 shows examples when T5-Gen drops the rare characters in answers.

To summarize, our findings are the following: 1) in single task, for IID data, two readers achieve comparable performance. T5-Ext performs better on OOD data as compared to T5-Gen; 2) in multi-task setting, T5-Ext is better at utilizing data from different domains, thus performs better than T5-Gen in both IID and OOD datasets; 3) in multi-task setting, T5-Ext is better at utilizing data from different domains, thus performs better than T5-Gen in both IID and OOD datasets, and 4) the extractive reader is more stable than the generative one when answer contains rare characters.

Question	Answer	Prediction
Who was one of the most famous people born in Warsaw?	Maria Skłodowskacurie	Maria Skodowska-Curie
What museum preserves the memory of the crime?	Katyń Museum	Katy Museum

Table 6: Examples of questions with answers containing rare characters and the prediction of T5-Gen.

5.2 Comparison of Different Pre-trained Models

In this section, we compare different readers based on different pre-training models, T5 and ELECTRA. We would like to note that the pre-trained model of T5 is already trained on SQuAD, while ELECTRA does not. However, based on the results on SQuAD that ELECTRA is even better than T5, we assume that the effect of the downstream training task dominates the effect of the pretraining.

T5 Generative Reader and ELECTRA Extractive Reader. Table 3 shows that in both single and multi-task learning, T5 generative reader is consistently better in long context datasets (e.g. for TextbookQA, in single-task learning, T5-Gen is better than ELECTRA by 3.59%; in multi-task learning, T5-Gen is better by 7.13%); in short context datasets, ELECTRA extractive reader is better (e.g. for SQuAD, in single-task learning, ELECTRA is better than T5-Gen by 2.75%; in multi-task learning, ELECTRA is better by 2.84%). **Two Extractive Readers.** In single-task learning, two readers are comparable in general except for the NewsQA on which T5-Ext is better than ELECTRA by 7.5%. In multi-task learning, T5 significantly outperforms

ELECTRA in long context datasets, and in short context datasets, the performance of two readers is comparable (see table 3). This suggests that the encoder in the encoder-decoder pre-trained model is also suitable for the extractive reader.

To summary, 1) T5 PrLM has an advantage over ELECTRA PrML if the context is long in both single- and multi-task learning scenarios; 2) in multi-task setting, the average performance of T5-Ext is the best on both IID and OOD datasets.

5.3 T5 Large Generative Reader

We also present the result of the T5 large generative reader as the reference of the upper bound of the T5 generative reader (last row in Table 3). For IID datasets, in short context datasets, the performance is comparable to other models, while in long context datasets, T5 large performs much better than the others (e.g. at least 3.33 % and 2.55% better than other models). For OOD datasets, T5-large performs consistently the best, demonstrating that the large model has better generalization capacity. We would like to note that the winner of the MRQA 2019 competition, D-NET (Li et al., 2019), achieves a 69.67 average F1 score on OOD datasets by the ensemble of XLNET (Yang et al., 2019) and ERNIE (Zhang et al., 2019). T5-large achieves comparable performance.

5.4 Constrained generation

Table 7 presents the F1 score of the generative reader using three different generation methods discussed in §3.2. The difference between the three methods is marginal and the main reason is that our generative reader already generates the tokens from the context in most cases, in which the predictions of the three methods are always the same. The last row in Table 7 shows the number of novel cases of each dataset, which are all less than 1% of the testing data. Table 8 presents some examples when the regular prediction is wrong but our constrained generation is correct. In Example1, by regular generation, T5-Gen predicts the synonym of the answer, which is not present in the context, and by our constrained generation, it predicts the correct answers. In Example2, by regular generation, T5-Gen predicts a partially correct answer but injects some words, and by our constrained generation, these words are removed from the prediction and thus match with the correct answer. In example3, by regular generation, T5-Gen predicts a completely wrong answer, and by our generation method, it predicts the correct answer. There are also few cases that regular prediction is correct but the constrained generation is wrong. In such cases, answers are surrounded by punctuation in the context, T5-Gen tends to predict novel tokens as shown in Table 9. In the first example, by regular generation, T5-Gen predicts a novel token '_3', while in the context the corresponding token is '_(3'. If this happens and the reader fails to recognize the exact tokens from the context, it predicts wrongly.

6 Additional Findings

6.1 Input Length in Training and Testing

T5 uses relative position embeddings (Ruder et al., 2019) which allows the input to be any length. We discover that input length dramatically affects the performance of T5-Ext. Specifically, we examine two types of input length in testing time: the length used in training time and the maximum length in a batch. In the former setting, the extractive reader uses the window-stride strategy to slice a long document into multiple short inputs; while in the latter setting, the given document can be represented by a single and complete input. In addition, we train different multi-task T5-Ext by three values of input length, 384, 512, and 1024.

Table 10 shows the F1 score of three models. For three models, using longer input length in testing time consistently leads to better performance, especially for long context datasets. When diagnosing the predictions of using short input length, we observe that T5-Ext can predict the correct answer in the top 5 but mistakenly predicts higher scores for distracting answers. In contrast, when the full input is encoded directly, T5-Ext can predict the correct answer with the highest score. This indicates that directly encoding the full input provides more complete information, which in turn helps T5-Ext to predict more accurate answers. From Table 10, we also see that for IID long context datasets, the longer the input length used in training time, the better performance the model has. Our findings lead to the two suggestions: 1) when computation budget is allowed, it is better to train a model with longer input length limit; 2) irrespective of the input length limit at training, one should do inference with full input encoded directly.

6.2 Re-rank Answers from Two Readers

By the comparison between two readers, a natural question is that can we utilize two readers to predict

Generation Method	IID Datasets						OOD Datasets					
	SQuAD	NewsQA	TQA	SQA	HQA	NQ	DROP	RACE	BioASQ	TbQA	RE	DuoRC
Regular	91.54	71.4	80.76	**85.83**	**78.98**	78.26	**51.98**	49.51	69.21	61.66	85.42	63.02
C-Con	**91.57**	**71.44**	80.76	85.82	78.96	78.26	51.78	**49.7**	**69.29**	61.66	**85.56**	63.02
NT-Con	91.52	71.42	80.73	85.75	78.93	**78.27**	51.76	49.66	69.23	61.66	85.49	63.02
# Novel tokens	21	8	28	33	9	18	5	10	3	4	9	5

Table 7: Test Multi task T5-Gen with three different generation methods. Regular: the reader can choose one of the tokens from the entire vocabulary. # Novel tokens:number of novel tokens predicted by regular generation.

Question	**Answer**	**Regular**	**C-Con**	**NT-Con**
How fast is pharmacy informatics growing?	quickly	rapidly	quickly	quickly
What did Tesla work on in 1888?	alternating current system	creating an alternating current system	an alternating current system	an alternating current system
Money put in a collection plate at church	offering	coin	offering	offering

Table 8: Answers predicted by T5-Gen with different generation methods.

	Context	**Answer**	**Regular**	**C-Con**	**NT-Con**
Text	...(325 mi)...	325	325	(325 mi)	(325 mi)
Tokens	_(3, 25, _mi,)	_3, 25	_3, 25	_(3, 25, _mi,)	_(3, 25, _mi,)
Text	...all-Gemini...	Gemini	Gemini	G mission	G mission
Token	G, e, mini	_Ge, mini	_Ge, mini	_G, _mission	_G, _mission

Table 9: Tokens of answer and the predictions of T5-Gen using three generation methods.

an answer. Furthermore, from Table 11, we find that both models predict high-quality answers in the top5 which provides enough room for improvement. Thus, we design a re-ranking paradigm. First, we use T5-Ext to generate five answers, then for each answer, we concatenate it with the question and the context and use T5-Gen to predict a score. The final score of an answer is computed by the sum of the T5-Ext and T5-Gen scores. We select the answer with the highest score as the final answer (see Appendix I for more details). As shown by Table 11[8], for IID data, the re-ranking method yields a small improvement, while for OOD data, the re-ranking method is slightly worse than T5-Ext. The inferior performance on OOD might be due to the relatively poor performance of T5-Gen.

[8]More detailed results are given Appendix I Table 21.

Train Len	Test Len	IID		OOD	
		Long	Short	Long	Short
384	384	67.02	83.59	52.64	65.0
	Max	76.96	85.1	**63.3**	67.72
512	512	70.0	84.17	53.53	67.1
	Max	77.54	85.21	62.52	**67.74**
1024	1024	74.56	84.9	60.98	67.67
	Max	**78.31**	**85.36**	62.66	67.67

Table 10: Three T5-Ext readers trained with different input lengths. For each one, use two input lengths in the testing time. Max: Maximum length in a batch.

	T5-Ext		T5-Gen		T5-Ext-Gen	
	in	out	in	out	in	out
top1	81.84	66.0	81.13	63.47	82.07	65.88
top5	91.65	79.68	89.81	79.74	-	-

Table 11: The top1 performance for T5-Ext-Gen is selecting the top1 after re-ranking.

7 Conclusion

In this paper, we aim to systematically compare the extractive and generative reader for QA tasks. To minimize the effect other than the reader, we design a fair comparison by using comparable pre-trained models for these two types of readers. We conduct comprehensive experiments to understand the pros and cons of two readers. Our findings provide guidelines on how to choose extractive or generative readers under certain conditions and open new avenues for improving each reader. Furthermore, our experiments reveal that encoder-only models are not always the best options for extractive readers, rather, encoder-decoder models also fit. In addition, the input length is a key factor for applying encoder-decoder models to extractive readers. Last but not least, we find that both extractive and generative readers can predict high-quality answers at top5, which suggests that a good answer re-ranking method has the potential to achieve significant improvement.

References

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021. UnitedQA: A hybrid approach for open domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3080–3090, Online. Association for Computational Linguistics.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

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

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Martin Fajcik, Martin Docekal, Karel Ondrej, and P. Smrz. 2021. Pruning the index contents for memory efficient open-domain qa. *ArXiv*, abs/2102.10697.

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, Hong Kong, China. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. 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, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Hongyu Li, Xiyuan Zhang, Yibing Liu, Yiming Zhang, Quan Wang, Xiangyang Zhou, Jing Liu, Hua Wu, and Haifeng Wang. 2019. D-NET: A pre-training and fine-tuning framework for improving the generalization of machine reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 212–219, Hong Kong, China. Association for Computational Linguistics.

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

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

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, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603.

Dan Su, Yan Xu, Genta Indra Winata, Peng Xu, Hyeondey Kim, Zihan Liu, and Pascale Fung. 2019. Generalizing question answering system with pre-trained language model fine-tuning. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 203–211, Hong Kong, China. Association for Computational Linguistics.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with macaw.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

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

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Ouguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. *ArXiv*, abs/2009.12756.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *ArXiv*, abs/2105.13626.

Zhilin Yang, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

A Training Setup

We use Huggingface implementation and Pytorch to train each model. For all experiments, we use 4 A-100 GPUs, and all models are trained in 4 epochs with a learning rate of 1e-4, batch size of 128. We set the maximum length in training to be 1024 for T5 and 512 for ELECTRA. We use 1 GPU to test models with a batch size of 16, and use the maximum length in a batch to be the input length for T5 and 512 for ELECTRA. Using the maximum length guarantees that every question and context pair can be encoded by one single and complete sequence of tokens.

B F1 and Exact Match

Here, we present the detailed F1 (Table 13) and Exact Match (Table 12) score for each reader trained with single domain or multi-domain data. We provide more analysis on single task learning and the observation on multi-task learning is discussed in the main paper.

We find that there is some correlation between the source of the dataset and the performance of different PrLM. Table 14 shows the source of each dataset. In IID datasets, ELECTRA achieves the best performances on SQuAD and HQA. It might because the context of SQuAD and HQA are from Wikipedia, which is part of the pretraining corpus of ELECTRA but not of T5. Thus, ELECTRA has pre-owned domain knowledge of these datasets and achieves the best performance. For NewsQA, TQA and SQA, the context are from News articles, trivia and quiz-league websites, and Jeopardy! TV show, respectively, ELECTRA is not pretrained on such domains and does not show advantage on these datasets, and T5 PrLM presents better performances. Although NQ is also based on Wikipedia, T5 PrLM is better than ELECTRA. The reason for such a result is that T5 has advantage when the context is long and the NQ dataset contains a crucial portion of long context questions. In OOD datasets, again, ELECTRA has advantage of pre-owned knowledge about Wikipedia and thus performs the best on DROP and RE, whose contexts are from Wikipedia. Furthermore, on DROP, ELECTRA is better than T5-Ext by 5.37% and T5-Gen by 10.65%. These improvement is much larger than IID dataset (e.g. on SQuAD, ELECTRA is better than T5-Ext only by 0.91% and T5-Gen only by 1.75%.), indicating the positive effect of pretraining corpus on the performance of OOD dataset.

C Distribution of Context Length

Figure 1 and 2 show the histogram of the context length of IID and OOD dataset. We see that the distribution of each dataset is quite different from each other. Some are mainly short, some are mainly long, and others are the combination of short and long.

D Two Training Targets of Generative Reader

As we mentioned in §3.2, we have two training strategies, one is to use the target tokens of answers without any context, termed as A-Target, the other is to use the target tokens of answers in given context, termed as C-Target. We obtain two readers by each training strategy and count the number of questions such that the reader generates answers with novel tokens (see Table 16). As expected, the A-Target reader can generate more novel tokens (tokens) than the C-Target reader. Meanwhile, we also compare the Macro-average F1 of IID and OOD using three generation methods by two generative readers as shown in Table 15. Although two readers achieve similar performance and the difference between three generation methods is marginal, for A-Target generative reader, the constrained generation methods decrease the performance more compared to C-Target, especially on datasets on which the regular generation produces more novel tokens. For example, on HotpotQA, T-Target model predicts 303 novel answers, and compared to regular generation, the F1 of C-Con generation decreases by 0.93% and NT-Con decreases by 1.54% . For the same dataset and for the C-Target model, compared to regular generation, the F1 of C-Con generation decreases by 0.02% and NT-Con decreases by 0.05%. The same trend can be observed on SQuAD and TextbookQA.

E Two Input Format

When fine-tuning T5 on down-stream tasks like question answering and natural language inference, some special words are added before the real input to denote the type of task. In an extractive reader, usually, there are no special words added. Inspired by these input formats, Given a question **Q** and a context **C**, we examine two formats of inputs. One is to add the "question:" and "context:" in front of the real question and context such that the input is {*question:* **Q** [SEP] *context:* **C**}. The other one

Model	Train \ Test	In-domain Datasets						Out-domain Datasets					
		SQuAD	NewsQA	TQA	SQA	HQA	NQ	DROP	RACE	BioASQ	TbQA	RE	DuoRC
Single Task Learning													
T5-Ext	SQuAD	92.27	64.36	65.55	17.76	71.42	60.7	53.47	49.05	68.71	23.7	85.56	60.89
	NewsQA	86.89	**72.76**	67.99	27.57	67.65	65.62	31.79	48.64	66.76	33.68	79.47	**62.37**
	TQA	73.31	48.15	76.34	58.86	56.4	50.28	38.74	38.28	66.17	31.2	79.14	51.11
	SQA	34.56	22.94	70.73	82.78	37.59	40.51	20.52	18.7	50.14	38.31	42.57	30.05
	HQA	84.92	58.93	61.23	23.03	80.05	63.9	47.18	43.11	68.46	27.51	85.61	56.08
	NQ	82.35	58.03	66.72	42.79	65.48	**80.55**	46.99	43.6	**70.29**	49.69	82.95	54.94
T5-Gen	SQuAD	91.33	61.55	69.44	29.01	67.92	61.39	42.58	47.8	66.05	49.08	84.32	60.12
	NewsQA	86.51	71.6	69.74	43.07	64.59	63.53	26.01	47.83	61.11	54.38	77.77	61.56
	TQA	76.84	51.41	**80.1**	63.36	58.64	56.21	34.05	42.48	57.94	52.43	81.91	54.32
	SQA	74.63	48.09	78.13	**85.93**	58.65	54.59	31.68	39.67	59.31	54.21	79.93	53.35
	HQA	86.07	59.96	70.61	52.21	79.88	63.59	43.73	45.1	66.16	42.6	85.34	58.24
	NQ	85.4	62.15	71.68	58.01	67.41	78.17	48.09	46.99	66.33	**60.74**	83.84	59.17
ELECTRA	SQuAD	**93.08**	58.31	66.49	46.16	71.75	66.36	**58.74**	**50.86**	70.21	39.89	**87.36**	59.83
	NewsQA	86.16	65.21	63.88	49.72	61.83	67.96	32.05	49.08	64.87	48.74	78.3	57.26
	TQA	70.69	42.46	76.51	67.79	60.61	57.7	42.19	36.66	62.27	44.06	83.28	49.01
	SQA	51.54	28.59	72.12	82.93	43.6	41.11	30.72	21.76	53.28	42.67	70.66	35.77
	HQA	84.97	53.72	61.94	36.9	**81.21**	64.93	48.55	37.39	65.2	26.55	84.66	53.19
	NQ	85.75	55.12	67.88	62.14	64.06	79.67	53.7	50.69	67.76	50.94	83.83	56.54
Multi-Task Learning													
T5-Ext	Multi	92.84	73.27	78.01	83.66	82.24	**81.0**	60.96	53.48	69.52	60.94	86.71	64.37
T5-Gen (base)	Multi	91.54	71.4	80.76	85.83	78.98	78.26	51.98	49.51	69.21	61.66	85.42	63.02
T5-Gen (large)	Multi	93.54	**73.44**	**84.08**	**88.38**	**82.88**	80.58	**63.63**	**55.87**	**72.33**	**68.76**	**87.33**	**68.09**
ELECTRA	Multi	**93.92**	65.33	73.63	81.94	83.36	79.98	61.77	52.71	71.86	54.8	87.18	59.76

Table 12: Evaluation by F1 score. TQA: TriviaQA; SQA:SearchQA; HQA:HotpotQA; NQ: NaturalQuestions; TbQA:TextbookQA; RE:RelationExtraction. **Bold** values are the best performance in a column for single-task and multi-task learning.

is without these special words such that the input is {**Q** [SEP] **C**}. Table 17 shows no noticeable difference between these two formats.

F T5 Encoder-Decoder Extractive Reader

When we apply T5 to an extractive reader, we have two options, one is to add a linear layer on top of the encoder, the other one is to add a linear layer on top of the decoder. Specifically, in the later model, the input to the encoder is the concatenation of question and context, and the input to the decoder is the same as encoder. Notice that the initial input to the decoder in such a model is different from the input to the decoder in generative reader whose input is the start token in the vocabulary. Then we apply a linear layer on top of the decoder to classify the start and end token. To have a more reasonable comparison with the T5-large extractive reader, we use the T5-base model instead of T5-large model. We find that on two datasets, T5-Ext is consistently better (see Table 18), thus we choose T5-Ext as the extractive reader to compare with the generative reader in the main result.

G Training Data Size

Table 1 shows that the training samples from each dataset are imbalanced (e.g. the size of SearchQA is roughly twice as TriviaQA). To see the effect of imbalanced data, we apply two training strategies. In *full data training*, we use all samples from IID datasets; in *down-sampling training*, we sample up to 75K data points from each IID datasets following the setting in (Fisch et al., 2019). Specifically, for SQuAD, SearchQA, and NaturalQuestions, we sample 75K data points in the training set, for the other three datasets, we use all given training data. Table 19 shows that using full data to train T5-Ext yields higher average F1 score than using down-sampling data in both IID and OOD dataset. For T5-Gen, using full data leads to higher average F1 than using down-sampling data on IID datasets, and down-sampling leads to higher average F1 on OOD datasets. But such differences are marginal, and to align with T5-Ext, we present the results of a model that is trained on a full dataset.

H Answer Length of Generative Reader

For the generative reader, we tried different maximum lengths of the generated answer: 16, 32, and 64. Table 20 shows that increasing the length of

Model	Train \ Test	In-domain Datasets						Out-domain Datasets					
		SQuAD	NewsQA	TQA	SQA	HQA	NQ	DROP	RACE	BioASQ	TbQA	RE	DuoRC
Single Task Learning													
T5-Ext	SQuAD	85.02	45.99	55.86	12.01	55.14	45.71	41.78	35.76	53.46	16.7	74.83	50.23
	NewsQA	74.97	56.46	57.75	20.92	47.87	50.68	22.49	32.79	44.81	23.35	61.4	49.37
	TQA	61.35	33.5	70.92	50.81	40.69	37.57	27.94	27.6	51.53	25.15	66.82	39.37
	SQA	27.64	15.57	64.91	77.01	26.55	32.25	13.44	12.91	36.5	32.34	34.12	23.58
	HQA	75.56	41.71	51.92	16.57	63.8	50.19	33.2	28.93	52.06	19.69	74.83	45.04
	NQ	70.58	40.81	56.56	34.4	48.6	68.61	33.0	31.6	52.13	39.39	69.27	42.5
T5-Gen	SQuAD	83.62	43.61	61.27	21.58	52.5	46.34	33.8	35.16	52.13	37.99	73.68	49.7
	NewsQA	73.94	55.03	60.51	35.58	45.21	48.27	18.83	33.98	40.43	40.85	59.06	49.77
	TQA	65.52	34.9	75.74	55.74	41.98	40.36	25.68	30.56	45.81	43.65	69.37	42.84
	SQA	63.65	33.43	73.42	80.84	43.57	39.81	24.82	28.34	47.67	46.11	68.89	43.5
	HQA	77.65	42.55	62.61	43.0	63.79	49.6	34.66	32.94	52.53	34.8	74.93	47.63
	NQ	73.53	42.55	62.67	48.63	50.33	65.02	38.26	35.01	46.81	50.77	70.93	47.5
ELECTRA	SQuAD	85.75	41.48	58.19	37.17	55.97	50.2	47.5	37.69	55.85	28.34	77.48	50.23
	NewsQA	71.64	51.38	54.8	40.08	42.33	51.51	23.29	33.98	44.02	34.26	58.18	45.97
	TQA	59.52	29.01	71.84	59.49	43.48	42.28	32.4	27.0	48.87	36.06	72.05	38.64
	SQA	42.06	19.9	67.66	77.7	31.5	30.61	23.49	16.02	38.96	35.53	57.36	28.91
	HQA	74.77	38.08	54.18	28.5	65.51	50.57	36.46	24.33	49.2	19.03	72.56	43.5
	NQ	73.76	38.08	59.97	52.89	46.59	67.93	40.45	36.5	45.74	41.05	70.15	47.17
Multi-Task Learning													
T5-Ext	Multi	86.11	57.1	72.59	78.01	66.34	69.06	51.03	39.02	53.66	49.83	76.73	52.5
T5-Gen (base)	Multi	84.12	55.03	76.53	80.73	62.92	65.5	42.91	37.54	55.65	52.69	75.17	52.43
T5-Gen (large)	Multi	86.74	56.91	79.85	83.52	67.43	67.47	54.22	42.43	60.04	59.08	77.1	57.69
ELECTRA	Multi	87.83	52.02	69.02	76.64	68.02	68.34	52.3	36.94	57.18	44.78	77.07	49.83

Table 13: Evaluation by Exact Match(EM). TQA: TriviaQA; SQA:SearchQA; HQA:HotpotQA; NQ: NaturalQuestions; TbQA:TextbookQA; RE:RelationExtraction

Dataset	Source
SQuAD	Wikipedia
NewsQA	News article
TQA	Trivia and quiz-league websites
SQA	Jeopardy! TV show
HQA	Wikipedia
NQ	Wikipedia
DROP	Wikipedia
RACE	English reading comprehension exams for middle and high school
BioASQ	Science (PubMed) articles
TbQA	Lessons from middle school Life Science, Earth Science, and Physical Science textbooks
RE	Wikiread
DuoRC	wikipedia

Table 14: The source of each dataset

the target does not make improvement, this might be because the answer in the testing data is usually short and thus length of 16 is sufficient.

I Re-rank Answers from Two Readers

We observe that both models can predict high quality answers in top5 as shown by the large improvement on every dataset. Based on this observation, a good re-ranking method can potentially yield good performance, thus we propose a re-ranking method. One model is employed to produce candidate answers, termed as producing model, then the other model re-rank the candidate answers, termed as re-ranking model. Specifically, we use a production model to predict five answers. If the generative reader is the re-ranking model, then for each answer, we concatenate it with the question and the context and use T5-Gen to predict a score. If the extractive reader is the re-ranking model, then we feed the question and the context to the extractive reader, and take each candidate answer as a label to get a score. The final score of an answer is computed by the sum of the T5-Ext and T5-Gen score. We select the answer with the highest score as the final answer. When T5-Ext is the producing model and T5-Gen is the re-ranking model, the final score is computed by the following equations,

$$P_{m,i} = \frac{\exp(z_{m,i})}{\sum_j^K \exp(z_{m,j})} \quad (5)$$

$$S_{Ext}(Ans) = \log(P_{Ext,start} \times P_{\text{Ext},end}) \quad (6)$$

$$S_{\text{Gen}}(Ans) = \frac{1}{N} \sum_{t \in Ans} \log(P_{\text{Gen},t}) \quad (7)$$

$$Score(Ans) = S_{\text{Ext}}(Ans) + S_{\text{Gen}}(Ans), \quad (8)$$

where $P_{m,i}$ is the logit of i^{th} token given by model m, $\log(P_{Ext,start}$ and $\log(P_{Ext,end}$ mean the logits of ground truth start and end token, respectively.

When the T5-Gen is the producing model, if an

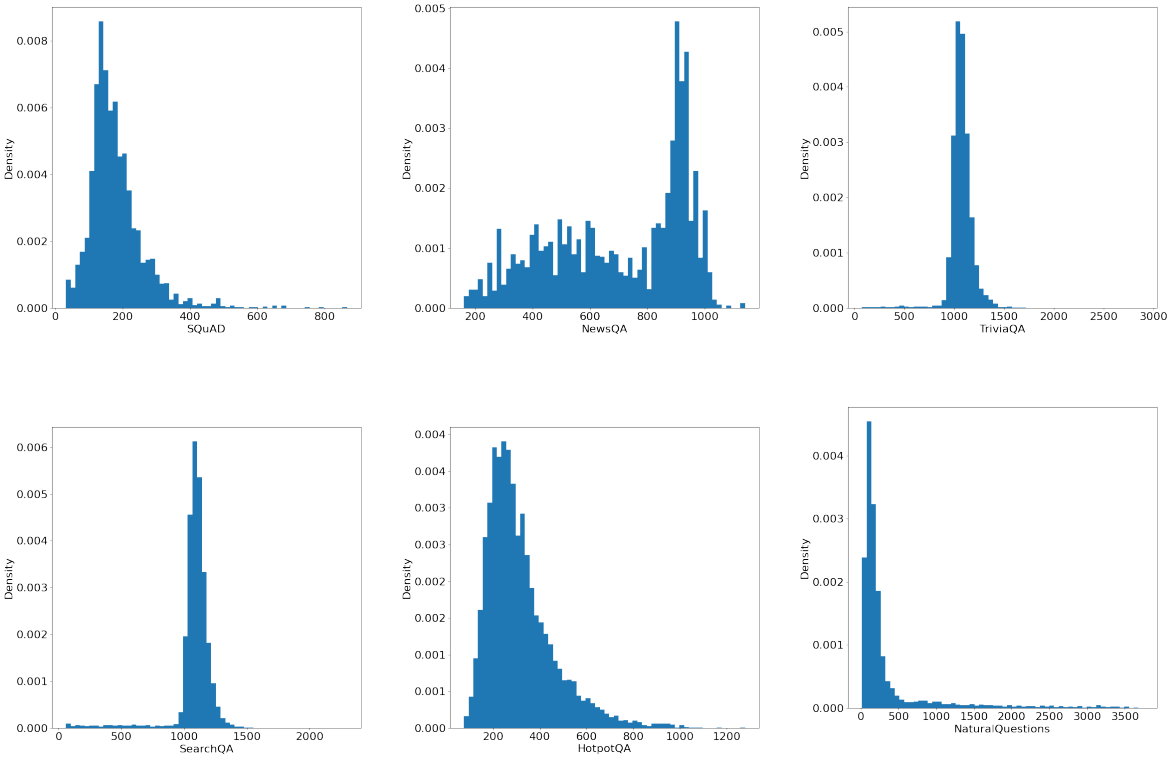

Figure 1: Context Length Histogram of In-domain dataset

answer is not an exact span in the context, then we ignore this answer. In a few cases, none of the answers in Top5 is actually a span coming from the context. In such cases, we do not use the re-ranking model and take the Top1 answer as the final prediction.

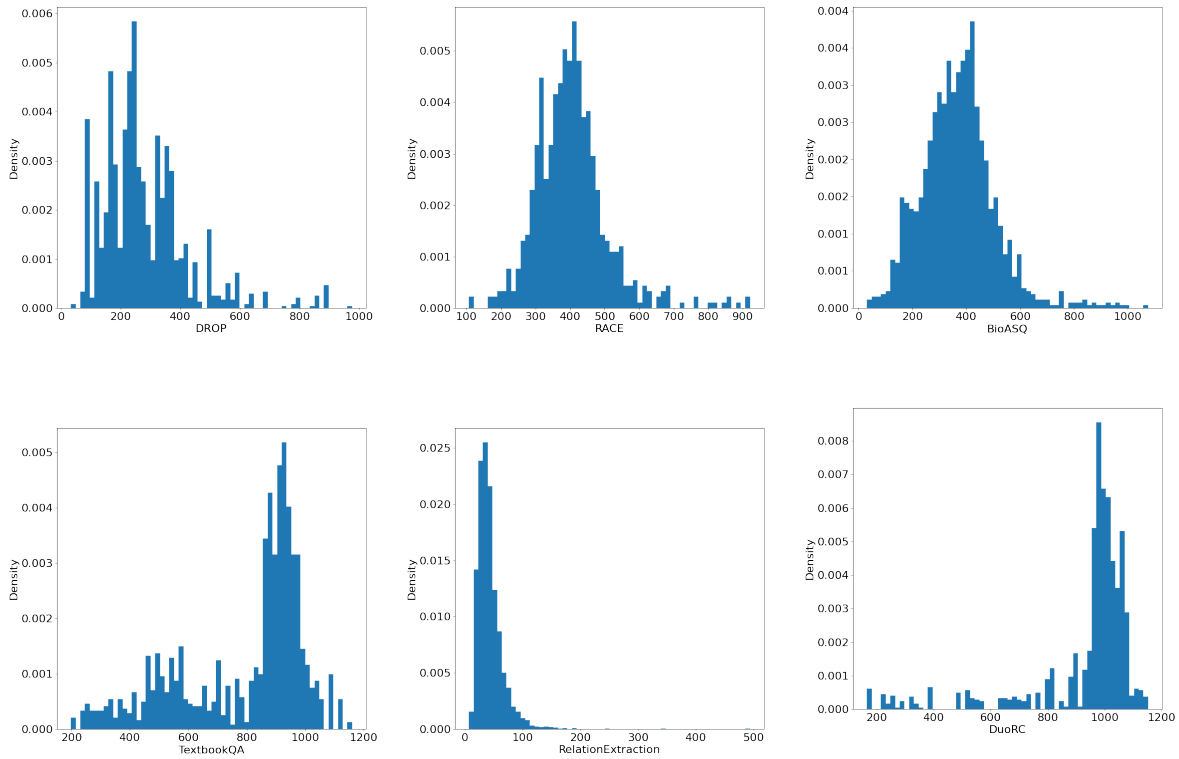

Figure 2: Context Length Histogram of out-domain dataset

Target	Inference Method	In-domain Datasets						Out-domain Datasets					
		SQuAD	NewsQA	TQA	SQA	HQA	NQ	DROP	RACE	BioASQ	TbQA	RE	DuoRC
A-Target	Regular	91.56	71.26	**81.01**	**86.62**	**79.65**	77.89	50.88	49.95	69.03	**63.13**	**85.88**	**63.04**
A-Target	C-Con	91.32	71.25	80.76	86.6	78.72	77.89	50.81	50.27	68.99	63.08	85.33	62.86
A-Target	NT-Con	90.69	71.22	80.57	86.47	78.11	77.87	50.79	49.63	69.0	63.11	85.12	62.31
C-Target	Regular	91.54	71.4	80.76	**85.83**	**78.98**	78.26	**51.98**	49.51	69.21	61.66	85.42	63.02
C-Target	C-Con	**91.57**	**71.44**	80.76	85.82	78.96	78.26	51.78	**49.7**	**69.29**	61.66	**85.56**	63.02
C-Target	NT-Con	91.52	71.42	80.73	85.75	78.93	**78.27**	51.76	49.66	69.23	61.66	85.49	63.02

Table 15: Generative model trained by multi-tasks and evaluated by F1 score on each datasets. Regular: the reader can choose one of the token from the entire vocabulary; C-Con: the reader can choose one of the token from the context tokens and the end token(</s>); NT-Con: the reader can choose one of the token from the the next token of current predicted tokens in the context and the end token.

	A-Target	C-Target
SQuAD	345	21
NewsQA	16	8
TQA	114	28
SQA	86	33
HQA	303	9
NQ	11	18
DROP	9	5
RACE	20	10
BioASQ	27	3
TbQA	8	4
RE	64	9
DuoRC	84	5

Table 16: Number of Novel tokens generated by A-Target and C-Target Reader using regular generation method.

Model	Format	SQuAD		NQ	
		EM	F1	EM	F1
T5-Ext	1	85.36	92.38	68.60	80.55
	2	85.02	92.27	68.61	80.55
T5-Gen	1	83.62	91.33	68.61	78.17
	2	83.02	90.75	63.87	77.08

Table 17: Comparison between different input format on two datasets. Format1 means input with "question:" and "context:" as format1, and format2 means without.

Model	SQuAD		NQ	
	EM	F1	EM	F1
T5-Ext	**85.02**	**92.27**	**68.61**	**80.55**
T5-EnDe-Ext	83.73	90.68	66.34	78.58

Table 18: Two style of extractive reader, T5-Ext consists of encoder and a linear layer, T5-EnDe-ext consists of encoder, decoder and a linear layer.

Tr Data	IID Datasets							OOD Datasets						
	SQuAD	NewsQA	TQA	SQA	HQA	NQ	Avg.	DROP	RACE	BioASQ	TbQA	RE	DuoRC	Avg.
T5-Ext Reader														
Full	92.84	73.27	78.01	83.66	82.24	81.0	81.84	60.96	53.48	69.52	60.94	86.71	64.37	66.0
Downsampling	92.62	73.54	78.58	82.95	81.77	80.71	81.69	58.66	51.86	69.27	59.4	87.5	64.02	65.12
T5-Gen Reader														
Full	91.54	71.4	80.76	85.82	78.98	78.25	81.12	51.84	49.47	69.14	61.66	85.42	63.02	63.42
Downsampling	91.19	71.13	80.64	84.86	79.24	77.87	80.82	49.78	51.49	69.63	61.31	85.99	63.32	63.59

Table 19: Training T5-Ext and T5-Gen with full or down-sampling data.

Length	IID Datasets							OOD Datasets						
	SQuAD	NewsQA	TQA	SQA	HQA	NQ	Avg.	DROP	RACE	BioASQ	TbQA	RE	DuoRC	Avg.
16	91.54	71.4	80.76	85.83	78.98	78.26	81.13	51.98	49.51	69.21	61.66	85.42	63.02	63.47
32	91.54	71.4	80.76	85.82	78.98	78.25	81.12	51.84	49.47	69.14	61.66	85.42	63.02	63.42
64	91.54	71.4	80.76	85.82	78.98	78.25	81.12	51.84	49.47	69.14	61.66	85.42	63.02	63.42

Table 20: Performance of using different Answer length for generative reader

Model	In-domain Datasets							Out-domain Datasets						
	SQuAD	NewsQA	TQA	SQA	HQA	NQ	Avg.	DROP	RACE	BioASQ	TbQA	RE	DuoRC	Avg.
Top1														
T5-Ext	92.84	73.27	78.01	83.66	82.24	81.0	81.84	60.96	53.48	69.52	60.94	86.71	64.37	66.0
T5-Gen	91.54	71.4	80.76	85.83	78.98	78.26	81.13	51.98	49.51	69.21	61.66	85.42	63.02	63.47
Top5														
T5-Ext	97.31	86.9	88.72	91.14	93.88	91.93	91.65	82.03	68.36	84.51	74.2	93.83	75.18	79.68
T5-Gen	95.67	81.17	90.68	94.24	90.17	86.96	89.81	80.73	67.61	79.75	79.35	93.06	77.96	79.74
T5-Gen as Re-ranking Model														
Re-rank	93.14	73.46	78.34	83.84	82.45	81.19	82.07	60.53	53.65	70.67	60.87	86.18	63.36	65.88
T5-Ext as Re-rankering Model														
Re-rank	91.03	68.61	81.11	86.28	80.03	76.34	80.57	57.85	52.42	66.32	63.47	86.5	65.35	65.32

Table 21: Top5 Answers