Increasing the Difficulty of Automatically Generated Questions via Reinforcement Learning with Synthetic Preference

Anonymous ACL submission

Abstract

The demand for high-quality question-answering (QA) datasets has surged with the proliferation of language models and conversational agents in various emerging domains. As these models become ever more capable, the need for more challenging datasets for benchmarking and training is growing. Manual dataset annotation is costly and time-consuming, necessitating a more efficient approach. We propose a methodology to increase the difficulty of automatically generated questions using synthetic preference data, derived from SQuAD, to fine tune a question generation model using reinforcement learning. We empirically show an improvement in question difficulty and quality over a simple supervised-finetuned model and perform an extensive error analysis. We make all our code and results publicly available.

1 Introduction

Question-answering (QA) datasets serve diverse purposes, from providing educational materials for students (Das et al., 2021) to serving as crucial resources for model training and evaluation (Rajpurkar et al., 2016). As conversational assistants and the application of language models continue to expand into new domains, the demand for creating challenging, high-quality datasets for these tasks has become increasingly evident. Difficult datasets are crucial for advancing the capabilities of language models, pushing them to handle complex tasks and enhancing their performance in real-world, challenging scenarios. This growing need is underscored by the rapid proliferation of QA datasets, with over 80 new datasets emerging within the last two years alone (Rogers et al., 2023).

One major challenge faced in developing QA datasets is cost. Annotation cost for QA datasets is especially high because of the time and cognition required to write questions and validate them. To exemplify this, the popular question-answering dataset SQuAD (Rajpurkar et al., 2016) recommended workers to take 4 minutes for every 5 questions at a rate of $9/ hour. This amounts to roughly $12,000 just to write the dataset's 100,000 questions; moreover, the cost is likely much higher when considering answer validation, and discarded samples due to duplication or poor quality.

Automatic Question Generation (AQG) systems present a remedy to these challenges given their efficiency and scalability compared to human annotators. Even in a zero-shot setting, language models are able to generate coherent questions (Sachan et al., 2022; Wang et al., 2023b); as such, we argue that writing coherent questions is no longer the main goal of AQG systems. Controlling more abstract attributes such as question difficulty remains challenging, as the concept is somewhat subjective and hard to manipulate. However, recent innovations in reinforcement learning for language models now enable these human-like ideals to be injected into the model learning process (Ouyang et al., 2022).

Pinning down a definitive description of question difficulty is near impossible as it depends on many factors. Common syntactic measurements of question difficulty include: question length; the average frequency of question terms in the English language (AlKhuzaey et al., 2023; Beinborn et al., 2014); and the syntactic difference between the dependency parse trees of a question and answer sentence (Rajpurkar et al., 2016). Semantic measurements may consider the relatedness between an answer span and the surrounding context (Beinborn et al., 2015), or the cosine similarity between distractors and the correct answer (Hsu et al., 2018). Moreover, we argue that difficult questions also require: reasoning over long spans of text; disambiguation of entities; and the use of synonyms to distance the question from the source text. A combination of all of these features is incredibly challenging to directly incorporate into the model training process.

We initially attempted to define such a task to encourage Large Language Models (LLMs) to rank samples with respect to difficulty. However, we found that there were always exceptions to our criteria, and that current zero-shot approaches simply lack the ability to capture such nuance. Instead, we therefore investigate whether we can learn from empirical evidence of difficulty and allow models to extract the relevant features for themselves.

In this paper we present a methodology for increasing the difficulty of automatically generated questions using synthetic preference data. We derive this preference data from the ability of question-answering models to correctly identify answer spans in a subset of SQuAD, assigning to each question a score based on the number of models that incorrectly answered the question. We assume that more challenging questions are answered correctly less frequently, and use this as the basis for our comparisons. In doing so, we also model these QA models as agents, identifying their weaknesses and generating questions to target them specifically.

We summarise this paper's contributions as follows:

1. A methodology for increasing the difficulty of automatically generated questions;

2. Empirical evidence of the methodology's efficacy;

3. An in-depth error analysis and study of interesting phenomena that emerge as part of this approach.

We release all code to recreate our work on GitHub.

2 Related Work

A similar question generation approach to ours is employed by Zhang et al. (2022) who adopt a pipeline structure. However, their primary objective is to generate suitable questions rather than specifically focusing on difficulty. An important distinction lies in their extensive pre-processing applied to identify candidate answers before feeding them to the question generation model. We argue that pre-identifying answers may limit diversity and prevent the inclusion of potentially complex and intriguing answer types.

Analyzing and Controlling Question Difficulty Understanding and managing question difficulty holds significant importance, especially in tasks involving the creation of exams and assessments (AlKhuzaey et al., 2023). One approach, as presented by Loginova et al. (2021), involves modelling the difficulty of multiple-choice questions through the use of softmax scores obtained from a pre-trained QA model. The variance in these scores is then calculated, with higher variance indicating greater difficulty.

Lin et al. (2015) controls the difficulty of quiz questions through the selection of distractor answers based on semantic similarity between linked data items. This involves collecting both structured RDF data and unstructured text, computing similarity scores through K-means clustering, and generating questions and answers via template-based methods. Importantly, the semantic similarity plays a role in determining the difficulty level, with more challenging questions featuring distractors exhibiting higher semantic similarity.

Reinforcement Learning with Human Feedback RLHF is a machine learning paradigm that combines reinforcement learning with human-provided guidance to steer language models to meet the needs of users, finding frequent use in chatbot and AI assistant settings (Ouyang et al., 2022). The basis for most modern methods is the Proximal Policy Optimisation (PPO) algorithm (Schulman et al., 2017), which iteratively enhances the language model's policy to maximize cumulative rewards through interactions with a dataset or language simulation. It collects experiences, evaluates advantages, and updates the policy with a clipped surrogate objective to ensure stability, gradually improving the model's performance. PPO is renowned for being unstable to train, which is addressed by Direct Preference Optimisation (DPO) (Rafailov et al., 2023). DPO directly optimises the policy model, converting the reward modelling problem into a classification task over the preference data.

Automatic Question Generation Chen et al. (2019) introduce a cross-entropy loss with a reinforcement learning-based loss function when training a gated bi-directional neural network for question generation. In this context, the reward model is optimising the semantic and syntactic quality of the question. BLEU-4, as a reward function, optimises the model for the evaluation metrics and the negative Word Movers Distance component is used to ensure semantic quality by maximising the similarity between a generated sequence and a ground truth sequence. Although this method produces high quality questions, factors such as question difficulty are hard to control.

Self-critic sequence training (SCST) (Rennie et al., 2017) uses a classical policy gradient method, REINFORCE, which is a Monte Carlo method. SCST computes rewards with n-gram token overlap as sub-sentence level rewards. Since training sets often have limited questions, these training rewards are arguably sparse, hindering the question generation model from extrapolating beyond the training distribution.

Liu et al. (2019) adopt a two-component reward for refining ill-formed questions. Question wording is used as a measure of short-term reward, and alignment between the question and answer represents a long-term component.

3 Method

To challenge the high cost of manual annotation while maintaining quality and increasing difficulty, we design and implement a robust system capable of generating contextually relevant, coherent, and challenging question-answer pairs from textual input. The process follows the core methodology of RLHF, deviating only in the use of synthetic preference data to train a reward model. Rather than explicitly defining the characteristics of difficulty and risking failure to capture certain aspects, we exploit the ability of leading question-answer models to derive which questions are challenging, and allow a reward model to extract the component features of the task.

We task three models with answering all questions in our validation split of SQuAD. These questions are assigned a score based on the number of times they were answered incorrectly, which are in turn used to generate pairwise preference data. These pairwise samples enable the training of a reward model for use in fine-tuning a supervised model for the task of question generation.

We embed this synthetic RLHF process into a greater pipeline for generating samples, shown in Figure 1. This ensures the quality of the final dataset. The pipeline consists of a set of rule-based critics which are used to exclude samples that are malformed and those with non-unique answers in the source text. Samples are then deduplicated using exact string matching.

The remainder of this section discusses each of the relevant components of the pipeline and the RLHF process.

3.1 Supervised Fine-Tuning

In our training process for question generation and response formatting, we begin by employing a reformatted version of the SQuAD v1 training split (see Table 1). The reformatting converts SQuAD to the task of question-answer pair generation, as shown in Figure 2. We select the "correct" answer as the one that appears most frequently in the list of answers for each question in the dataset, selecting randomly among the most common if there is no victor. To ensure model robustness without overfitting, the model undergoes a single epoch of training, enabling it to effectively capture the nuances of the task.

3.2 Reward Modelling

To control the difficulty of our model, we leverage the intrinsic properties present in challenging questions from SQuAD. To extract these attributes, we employ three question answering models that almost match or exceed human performance on SQuAD v2 to evaluate our development split: a RoBERTa-large model[1], a DeBERTa-large model[2] and RetroReader (Zhang et al., 2020). Each question is assigned a score based on the number of models that failed to correctly answer the question. These scores are used to place questions into a pairwise ranking setup against other questions for the same input context. Where a question's scores are equal, they are considered ties, and no pairwise sample is created. We also record the margin, defined as the difference in score between the better and worse sample, to experiment with the marginal ranking loss, as defined in Touvron et al. (2023b).

3.2.1 Format Critics

To ensure the quality of the final dataset, we utilise a collection of rule-based critics which we call *Format Critics*. These critics have three main functions: (i) they remove questions that don't adhere to the desired format of *Q? (answer: A)*; (ii) they filter out questions that contain the entire input text in the question - a sign of degeneration during inference; and (iii) they ensure the provided answer

[1] https://huggingface.co/deepset/roberta-large-squad2

[2] deepset/deberta-v3-large-squad2

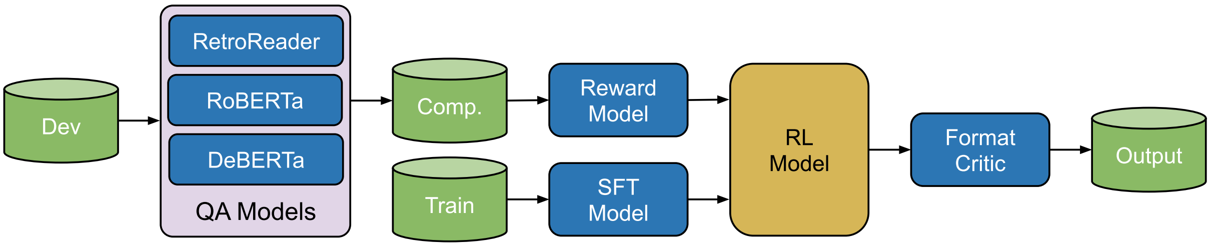

Figure 1: Depiction of our dataset generation pipeline. Question-Answering models are first used to create pairwise comparison data to train a reward model. An SFT model is trained on the train split of SQuAD and then fine-tuned using the reward model, producing the RL model. When generating question-answer pairs for the final dataset, generations are passed through the format critics to ensure data quality.

Instruction Write 1 answerable span extraction question and provide the correct answer based on the text.

Input ... Upon its arrival in Canberra, the Olympic flame was presented by Chinese officials to local Aboriginal elder Agnes Shea, of the Ngunnawal people. She, in turn, offered them a message stick ...

Response Who received the flame from Chinese officials in Canberra? (answer: Agnes Shea)

Figure 2: Example training sample from the reformatted SQuAD dataset for use in supervised fine-tuning.

is unique in the text, minimising the number of ambiguous or impossible questions. Samples that pass these critics are then deduplicated using exact matching.

3.3 Reinforcement Training

We use Proximal Policy Optimisation (Schulman et al., 2017) with multiple sets of adapters to reduce the memory overhead during training, implemented using the Transformers Reinforcement Learning library (von Werra et al., 2020). A single base model is used with separate LoRA adapters for the policy, value and reward model components; each is switched to perform the relevant components of the reinforcement training process.

4 Experimental Setup

4.1 Models

We conduct our experiments with the leading open-source language model, LLaMA2-7B-chat - the successor to LLaMA-7B (Touvron et al., 2023a) with instruction tuning applied on release. We apply LoRA adapters to improve training times and reduce memory usage to enable training on a single A100 80GB GPU. All LoRA adapters share the same hyperparameters: LoRA rank of 16, α of 32, dropout of 0.05, and no bias.

For all training procedures, we use Flash Attention 2 (Dao, 2023) in the BrainFloat (BF16) datatype to improve training efficiency. For supervised fine-tuning (SFT), we leverage sample packing to reduce training times further.

4.2 Generation Settings

During generation, the model is tasked with producing a single output for each question in the training set using nucleus sampling (Holtzman et al., 2020). We maintain the original configurations of a repetition penalty of 1.1, top P of 0.7, and top K of 0 are used but increase the temperature from 0.6 to 0.9 to increase the diversity of generations. The model is provided with the input data in the original LLaMa-2 prompt format, using the same instructions as during supervised fine-tuning.

4.3 Data Splits

We base our splits off the original SQuAD v2 to minimise the risk of data leakage. We maintain the full train split unchanged as any model previously trained on SQuAD will have seen the full train split. We extract a test split of 500 contexts from the dev split, ensuring no contexts appear in both the dev and test splits. In all cases, context-question pairs were only kept if they fit into the context length of LLaMa2 when formatted in the correct prompt format. All samples were formatted into the three instruction components: *instruction*, *input*, *response* as shown in Figure 2.

Only the dev set of our SQuAD dataset was used to derive difficulty comparison data, to ensure the reward model never sees the samples used for evaluation. To evaluate the reward model, we extract

Split	# Contexts	# Questions
Train	18,891	87,599
Dev	1,567	8,038
Test	500	2,532
Train comp.	1,107	8,394
Dev comp.	123	950

Table 1: Split of contexts and questions from SQuAD. The *comp.* splits are derived from the dev split, used to evaluate the performance of the reward model during training.

10% of the comparison contexts. Full dataset statistics can be found in Table 1.

4.4 Evaluation Metrics

As our goal is to evaluate the difficulty of answerable questions, we develop an evaluation task which we supply to GPT-4-turbo[3]. In this task, we query whether the source text and generated question entail the generated answer. GPT-based evaluations have demonstrated a robust alignment with human preferences across various complex tasks in reference-free settings (Fu et al., 2023; Liu et al., 2023). However, as in Dettmers et al. (2023), we validate GPT's alignment with human preference by manually annotating a subset of 50 samples extracted from SFT and RLHF. These samples are uniformly distributed with respect to GPT's predictions, and alignment is calculated using Cohen's κ.

To assess the quality of generated questions relative to our SQuAD test split, we *intentionally avoid* n-gram based metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and more modern alternatives such as Q-Metrics (Nema and Khapra, 2018), as we believe they restrict diversity of generation, constraining the model to reference questions and answers. We instead adopt the following reference-free metrics:

Syntactic Divergence was introduced in SQuAD v1 and provides a distance measure between two dependency paths. Word-lemma anchors, common to both the question and answer sentence, are first detected. A dependency path from the anchor to the interrogative word (who, what, etc.) in the question is compared to the dependency path between the anchor and the answer span in the answer sentence using Levenshtein distance (Levenshtein et al., 1966). Insertions and deletions have an edit score of 1 and substitution operations have a score of 2. This acts as a measure of difficulty, with harder questions having a higher syntactic divergence.

RQUGE is a reference-free metric which calculates an *acceptability-score* by generating an answer for the candidate question and predicting the semantic similarity between the predicted answer and the gold answer provided by the user. In our setup, this metric acts as an assessment of both the question and the answer quality, as the acceptability-score is dependent on both aspects (Mohammadshahi et al., 2023).

QAScore is a reference-free metric which attempts to align AQG evaluation to human judgements by using the log-probabilities returned by RoBERTa. For a given question-answer pair, the metric provides RoBERTa with a set of inputs, each prefaced with the context and question, where each input masks a word in the answer. The log-probabilities of the correct token in each of the masked token positions for each input are summed to arrive at the final statistic. QAScore claims to show strong correlation with human judgement (Spearman $r = 0.864$) (Ji et al., 2022).

Self-BLEU assesses how similar questions are to other questions generated for a given context. Each question is taken as a hypothesis and the others as a reference for the BLEU calculation. The self-BLEU is taken as the average BLEU for the question collection. We adopt this metric at a document level to measure the diversity of model generations for each context (Zhu et al., 2018).

5 Results

To evaluate the number of answerable samples for each set of generations, we employed our entailment task on GPT-4-Turbo. We show that our RLHF model increases the proportion of answerable questions from 83.7% for the SFT model to 87.3%. Based on our manual annotation of a uniformly selected set of samples from both SFT and RLHF, we observer a substantial Cohen's κ agreement of 0.62. The GPT-4 predictions can be found in in Table 3.

We apply each metric to our SQuAD test split, generating 1 sample for every question in the test set. We report the number of samples that were considered valid: i.e. that met the formatting requirements and were not duplicates. We conduct our testing after all models have been trained, to

[3] gpt-4-1106-preview as of 13th Dec. 2023

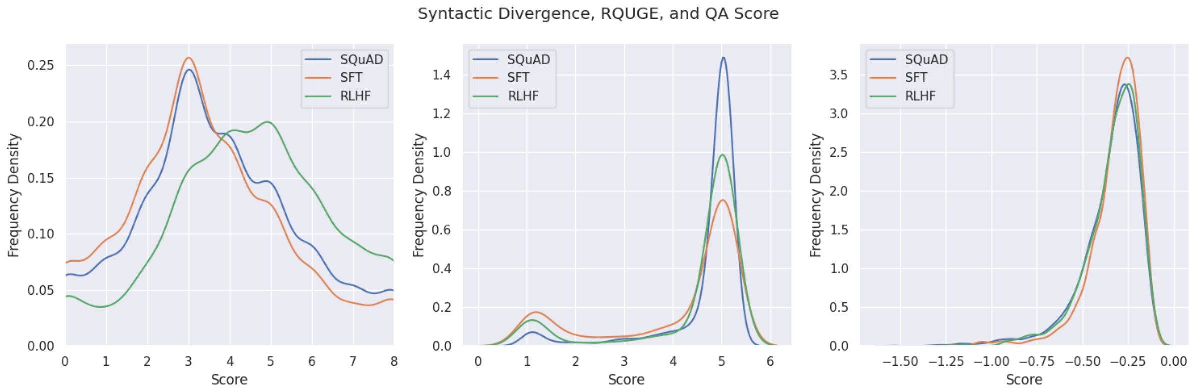

Figure 3: Distribution of metrics results for each set containing all samples on our SQuAD test set.

Model	Valid (↑)	Syn. Div. (↑)	RQUGE (↑)	QAScore (↑)	Self-BLEU (↓)
SQuAD	2,532 (-)	3.69 ± 2.00	4.67 ± 0.94	-0.35 ± 0.17	0.26 ± 0.11
SFT (all)	**1,867 (0.74)**	3.41 ± 1.96	4.10 ± 1.45	**-0.32 ± 0.14**	**0.53 ± 0.22**
RLHF (all)	1,354 (0.53)	**4.21 ± 2.29**	**4.43 ± 1.27**	-0.33 ± 0.15	0.75 ± 0.18
SFT (ans)	**1563 (0.61)**	3.09 ± 2.08	4.39 ± 1.22	**-0.31 ± 0.14**	**0.53 ± 0.22**
RLHF (ans)	1,182 (0.43)	**4.21 ± 2.21**	**4.70 ± 0.91**	-0.34 ± 0.15	0.75 ± 0.23

Table 2: Results of each approach on our SQuAD test split. We report number of unique, valid samples used in the calculations for each set, the proportion of samples that were unique and valid, and the mean and standard deviation for each metric. For syntactic divergence, we do not include samples for which a lexical path between question and answer sentence could not be found in our calculations. Self-BLEU is calculated including all exact duplicates. *ans* samples are calculated only across the samples deemed answerable by GPT4-Turbo.

ensure we are not optimising for the test data. The results of each set are displayed in Table 2 and the distributions are shown in Figure 3.

We show that our RLHF model outperforms the SFT model across syntactic divergence and RQUGE and is close to SFT on QAScore. We also see that our RLHF model outperforms the SQuAD baseline on syntactic divergence, suggesting that the reward model was effective in identifying long-range dependencies and complex parse trees as a signifier of difficulty.

6 Discussion

We show significance in surpassing the scores of the SFT model with our RLHF model for syntactic divergence and RQUGE but not for QAScore or self-BLEU Using a Mann-Whitney U-test (Mann and Whitney, 1947), we disprove the null hypotheses that the RLHF samples score lower than or equal to the SFT samples, both with significance $p \ll 0.001$ for Syntactic Divergence. For RQUGE we disprove that RLHF samples score lower with significance $p < 0.002$ and find $p < 0.003$ for the case that they are equal. However, we do not find this same significance for QAScore and

Model	Valid	Invalid	Proportion
SFT	1,563	305	0.19
RLHF	1,182	172	0.15

Table 3: GPT-4-Turbo predictions of question answerability based on entailment of source text and question to answer.

self-BLEU. QAScore favours the SFT results with significance $p < 0.002$ and $p < 0.001$ for equal to and greater than respectively and self-BLEU $p \ll 0.001$ for both. The results tell much the same story for the subset of answerable only questions.

QAScore proves less informative when comparing AQG model outputs We find that the change in score across syntactic divergence and RQUGE is greater, providing a better insight into the variance in performance of each model. We also find that models perform better on QAScore than the human written questions. This may be a result of using the log-probabilities of one model to evaluate the generations of another model. The model generations are likely to adopt a more similar distribution to the evaluation model than those of human origin.

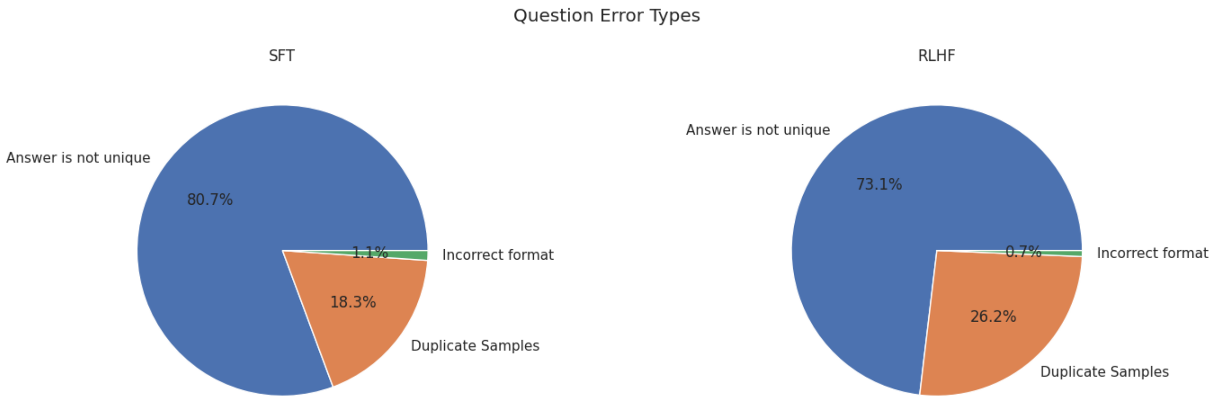

Figure 4: Error distribution of questions for SFT and RLHF versions of LLaMa2-7b-chat.

Allowing models to learn the semantics of difficulty is more effective than enforcing a definition In a previous conception of this project, we attempted to use zero-shot Large Language Models (LLM) to define our pairwise samples for training a reward model (Qin et al., 2023; Sun et al., 2023; Fu et al., 2023). This required us to fully define the meaning of difficulty in a manner that required no human intuition. Ultimately this approach was flawed for two reasons: first, we always found counter-examples to our criteria based on human preference; second, it seems from our results that language models do not yet possess the ability to reason over such complex requirements in the way that humans can. By leveraging the feature extraction abilities of transformer-based models, we relieve ourselves of this burden and show an effective increase in difficulty across the provided metrics.

6.1 Error Analysis

To understand our system's deficiencies and identify areas for improvement in the future, we critically analyse the reasons for our critics rejecting samples, and analyse interesting patterns in the questions generated by each iteration of the model.

The Issue of Non-Unique Answers As shown in Figure 4, the main cause for exclusion is that answers are not unique. This is because SQuAD does not require that an answer span appears only once in the source text. Where SQuAD has workers highlight the correct span, we use a language model to generate a question answer pair and thus cannot reliably expect the model to generate accurate start and end positions for the answer span. Future work could seek to remedy this with a classification model which predicts which instance of an answer span is most likely, although this may increase accumulated error. Another approach may be to apply a fixed negative reward signal to penalise a model for generating answer spans that appear multiple times; however, this may limit the types and quality of answers generated by the model.

Duplication Rate The other area for improvement is reducing the rate of duplication among generations. In identifying difficult questions, the reward model has likely learned that particular semantic structures are considered challenging from the comparison data. The objective of question generation is thus refined from simply generating a question, to generating a question with those particular semantics. As such, the pool of optimal generations is reduced and particular question-answer pairs become more probable. This is clearly seen in increase in the number of duplicate samples, shown in Figure 4. During generation, we opted for a high temperature of 0.9 to try to combat this constriction; however, future work could seek to address this issue more holistically through augmentation of training data or other such methods.

Positional Bias One interesting phenomenon is the positional bias in where the model chooses to generate answers. As seen in Figure 5, for SQuAD, there is a subtle bias toward the beginning of the text, which decays slightly as the text progresses. This is expected as answers must become shorter toward the end of the text as there are fewer characters remaining. In the case of SFT, the model selects nearly half of its answer spans from the first half of the text. This is less severe for RLHF but the bias remains evident. We can reasonably assume that the positional bias in SQuAD has some impact on the generative models, but given that positional

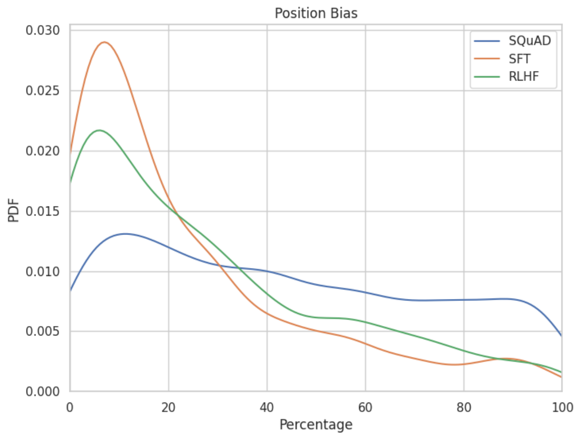

Figure 5: Position of the start index of answers for each dataset. SQuAD positions are selected from our test split and answers are chosen to be the most common from the list of suitable answers. Percentage is measurement of how far through the source text the start index is. Exact duplicate questions are not considered.

bias has been shown in LLM ranking (Wang et al., 2023a; Li et al., 2023) and introductory content is favoured in summarisation tasks (Ravaut et al., 2023), we can deduce this is an issue also present in this task. A potential remedy is to supply the model with a sliding window of sentences across the context paragraph to force the model to generate questions throughout the text. While this would improve the diversity of a final dataset, it may have the adverse effect of limiting the range of dependencies, restricting potentially challenging questions across the whole text.

Tentative Language One emergence in the RLHF model is the use of 'approximately' when asking for details which are provided exactly in the text. There are no instances of this happening in the SFT set and only 10 in the SQuAD test split; however, in the RLHF samples, there are 333 instances. This goes against intuition as there are 14 instances of *approximately* appearing in the chosen split of the difficulty comparisons set but 25 appearing in rejected. One potential explanation is that the combination of *approximately* with other features exhibits high levels of difficulty in *chosen* set and the RLHF model is leveraging those features. Further work could seek to definitively explore this phenomenon.

Misuse of Articles Creates Ambiguous Questions In some instances, both the SFT model and the RLHF model misuse articles. One example is for the question *What is the main bus company that operates in Newcastle?* However, the source text clearly states *There are 3 main bus companies providing services in the city; Arriva North East, Go North East and Stagecoach North East.* This is a challenging problem to solve since it requires comprehension of the source text, which is beyond the limits of rule-based penalties. This could be resolved by using a classification model which is trained to understand when an ambiguous question-answer pair is generated and apply a fixed negative reward. Alternatively, a second reward model could be used to apply a scalar reward based on the degree of ambiguity. The risk of accumulated error is again present in these solutions.

Self-Answering Questions Another example of a failure mode is questions which anwer themselves. One such question from the RLHF model is *What is the term that refers to Turing machines that are not deterministic?* where the answer is *non-deterministic Turing machines*. An imperfect estimate of this occurrence across each set can be obtained by identifying the number of questions which contain the answer span. Our SQuAD test split contains 30 such examples, SFT 18 and the RLHF split the most with 39. More investigation is necessary but removing these samples from the training data could limit this effect downstream.

7 Conclusion

In this paper we have introduced a robust approach for automatically generating question-answer pairs from textual input. Using existing, high-performing question answer models, we are able to determine which questions are most challenging, and use them to develop synthetic pairwise data for training a reward model. Rather than explicitly defining the characteristics of question difficulty, we allow the reward model to extract these features, leading to a significant increase in question difficulty when used to fine-tune the SFT model.

Furthermore, we have conducted an extensive analysis of the current issues with this approach and provide potential remedies which may be explored in future work.

We believe this technique may be extended to address further abstract properties of question generation such as ambiguity, completeness and relevance. This method may also be adapted to tackle multiple aspects at once through the use of multi-reward model setups as in Wu et al. (2023).

Limitations

This project only shows the suitability of the method on a single model. In future work, we seek to address this by performing a more comprehensive review of the approach across a range of model sizes and architectures. We also acknowledge that this method currently only addresses answerable questions while most contemporary QA datasets utilise both answerable and unanswerable questions. Finally, despite using LoRA and multi-adapter training, we still required approximately 15 GPU hours on an A100 80GB which restricts the potential audience for this approach. Evaluating smaller models or quantisation will enable greater access to this project's benefits.

Ethics Statement

This project has been approved by the relevant institution's ethics committee. We use LLaMa2 in accordance with Meta's license[4].

References

Samah AlKhuzaey, Floriana Grasso, Terry R. Payne, and Valentina Tamma. 2023. Text-based Question Difficulty Prediction: A Systematic Review of Automatic Approaches. *International Journal of Artificial Intelligence in Education*.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the Difficulty of Language Proficiency Tests. *Transactions of the Association for Computational Linguistics*, 2:517–530.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2015. Candidate evaluation strategies for improved difficulty prediction of language tests. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–11, Denver, Colorado. Association for Computational Linguistics.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Reinforcement learning based graph-to-sequence model for natural question generation. *CoRR*, abs/1908.04942.

Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. ArXiv:2307.08691 [cs].

Bidyut Das, Mukta Majumder, Santanu Phadikar, and Arif Ahmed Sekh. 2021. Automatic question generation and answer assessment: a survey. *Research and Practice in Technology Enhanced Learning*, 16(1):5.

[4] https://ai.meta.com/llama/license/

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. GPTScore: Evaluate as You Desire. ArXiv:2302.04166 [cs].

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. ArXiv:1904.09751 [cs].

Fu-Yuan Hsu, Hahn-Ming Lee, Tao-Hsing Chang, and Yao-Ting Sung. 2018. Automated estimation of item difficulty for multiple-choice tests: An application of word embedding techniques. *Information Processing & Management*, 54(6):969–984.

Tianbo Ji, Chenyang Lyu, Gareth Jones, Liting Zhou, and Yvette Graham. 2022. QAScore—An Unsupervised Unreferenced Metric for the Question Generation Evaluation. *Entropy*, 24(11):1514.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Shuai Wang, Cuiyun Gao, and Yang Liu. 2023. Split and Merge: Aligning Position Biases in Large Language Model based Evaluators. ArXiv:2310.01432 [cs].

Chenghua Lin, Dong Liu, Wei Pang, and Edward Apeh. 2015. Automatically Predicting Quiz Difficulty Level Using Similarity Measures. In *Proceedings of the 8th International Conference on Knowledge Capture*, K-CAP 2015, pages 1–8, New York, NY, USA. Association for Computing Machinery.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. ArXiv:2303.16634 [cs].

Ye Liu, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2019. Generative question refinement with deep reinforcement learning in retrieval-based QA system. *CoRR*, abs/1908.05604.

Ekaterina Loginova, Luca Benedetto, Dries Benoit, and Paolo Cremonesi. 2021. Towards the Application of Calibrated Transformers to the Unsupervised Estimation of Question Difficulty from Text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 846–855, Held Online. INCOMA Ltd.

Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

Alireza Mohammadshahi, Thomas Scialom, Majid Yazdani, Pouya Yanki, Angela Fan, James Henderson, and Marzieh Saeidi. 2023. RQUGE: Reference-Free Metric for Evaluating Question Generation by Answering the Question. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6845–6867, Toronto, Canada. Association for Computational Linguistics.

Preksha Nema and Mitesh M. Khapra. 2018. Towards a Better Metric for Evaluating Question Generation Systems.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

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

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. ArXiv:2306.17563 [cs].

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. ArXiv:2305.18290 [cs].

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. ArXiv:1606.05250 [cs].

Mathieu Ravaut, Shafiq Joty, Aixin Sun, and Nancy F. Chen. 2023. On Context Utilization in Summarization with Large Language Models. ArXiv:2310.10570 [cs].

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. QA Dataset Explosion: A Taxonomy of NLP Resources for Question Answering and Reading Comprehension. *ACM Computing Surveys*, 55(10):197:1–197:45.

Devendra Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving Passage Retrieval with Zero-Shot Question Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3781–3797, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. ArXiv:1707.06347 [cs].

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. ArXiv:2304.09542 [cs].

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and Efficient Foundation Language Models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models. ArXiv:2307.09288 [cs].

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large Language Models are not Fair Evaluators. ArXiv:2305.17926 [cs].

Zhenduo Wang, Yuancheng Tu, Corby Rosset, Nick Craswell, Ming Wu, and Qingyao Ai. 2023b. Zero-shot Clarifying Question Generation for Conversational Search. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pages 3288–3298, New

System: Determine if the question is answered or not based on the premise. Use your entailment reasoning knowledge to classify entailed by as yes or not entailed by as no. Make sure to response Label: yes or no. Your response should only be in the set yes, no.
Premise: ... Upon its arrival in Canberra, the Olympic flame was presented by Chinese officials to local Aboriginal elder Agnes Shea, of the Ngunnawal people. She, in turn, offered them a message stick ...
Question: Who received the flame from Chinese officials in Canberra?
Answer: Agnes Shea

Label: yes

Figure 6: Example prompt and response to GPT-4-Turbo (gpt-4-1106-preview as of 13th Dec. 2023)

York, NY, USA. Association for Computing Machinery.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-Grained Human Feedback Gives Better Rewards for Language Model Training. ArXiv:2306.01693 [cs].

Cheng Zhang, Hao Zhang, Yicheng Sun, and Jie Wang. 2022. Downstream transformer generation of question-answer pairs with preprocessing and postprocessing pipelines. In *Proceedings of the 22nd ACM Symposium on Document Engineering*, DocEng '22, pages 1–8, New York, NY, USA. Association for Computing Machinery.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2020. Retrospective reader for machine reading comprehension. *CoRR*, abs/2001.09694.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A Benchmarking Platform for Text Generation Models. ArXiv:1802.01886 [cs].

A GPT4-Turbo Entailment Task