RadQA-DPO: A Radiology Question Answering System with Encoder-Decoder Models Enhanced by Direct Preference Optimization

Md Sultan Al Nahian and Ramakanth Kavuluru
Division of Biomedical Informatics, Department of Internal Medicine
University of Kentucky, Lexington, KY USA
{mna245,ramakanth.kavuluru}@uky.edu

Abstract

Extractive question answering over clinical text is a crucial need to help deal with the deluge of clinical text generated in hospitals. While encoder models (e.g., BERT) have been popular for this reading comprehension–style question answering task, recently encoder-decoder models (e.g., T5) are on the rise. There is also the emergence of preference optimization techniques to align decoder-only LLMs with human preferences. In this paper, we combine encoder-decoder models with the direct preference optimization (DPO) method for the RadQA radiology question answering task. Our approach achieves a 12–15 F1 point improvement over previous state-of-the-art models. To the best of our knowledge, this effort is the first to show that DPO method also works for reading comprehension via novel heuristics to generate preference data without human inputs.

1 Introduction

Clinical domain is rich in text data, such as progress notes, discharge summaries, and radiology/pathology reports, which constitutes a significant portion of electronic medical records (EMRs). These documents contain essential patient information but are often lengthy and idiosyncratic to specific clinicians, making it difficult and inefficient for doctors to manually extract specific details during care transfers or follow-ups (Jin et al., 2022). From a natural language processing (NLP) perspective, machine reading comprehension (MRC) systems can address this challenge by extracting precise answers to specific queries directly from these documents, facilitating more efficient decision-making for physicians (Demner-Fushman et al., 2009). In this paper, we achieve state-of-the-art results for a MRC task in radiology, with encoder-decoder language models (LMs) enhanced by direct preference optimization (DPO). Before we proceed, we first trace the origins of DPO since it was first introduced for a very different purpose than reading comprehension.

Since mid 2020, large language models (LLMs) have become pivotal in NLP, showcasing remarkable performance across a variety of tasks. These models undergo an initial phase of unsupervised pretraining, acquiring a comprehensive language representation that equips them with robust and contextual generation capabilities, which can then be transferred to specific downstream tasks through supervised fine-tuning (Dai and Le, 2015; Radford et al.; Devlin et al., 2019; Khandelwal et al., 2019). However, while supervised fine-tuning has been proven effective in enhancing model performance, it struggles to align models with human preferences (Stiennon et al., 2020). The high-quality output achieved through supervised fine-tuning often poorly correlates with human judgment, as the maximum likelihood objective struggles to capture the nuances of human preferences (Chaganty et al., 2018; Dusek et al., 2017). To address this challenge, reinforcement learning from human feedback (RLHF) has recently emerged as a promising approach for aligning LLMs with human preferences (Ziegler et al., 2019; Stiennon et al., 2020). RLHF utilizes human feedback on the model's output to guide its learning process, resulting in enhanced performance and better correlation with human judgment across diverse NLP tasks (Ouyang et al., 2022; Glaese et al., 2022; Bai et al., 2022a).

Ability to evaluate the output of LLMs based on human preferences is a core part of RLHF. To acquire this ability, the RLHF technique involves building a reward model from human annotated preference data. The objective of the reward model is to assess the output of the language model based on human preferences and represent it in a scalar value, which is used to optimize the language model using RL algorithms, most commonly proximal policy optimization (PPO) (Schulman et al., 2017). Usually the reward models are built by fine-

Proceedings of the 24th Workshop on Biomedical Language Processing (BioNLP 2025), pages 101–113

tuning another LLM as it is expected that the reward model should have the similar language modeling capabilities to the original language model it is used to optimize. While RLHF demonstrates impressive performance across various NLP tasks (Chowdhery et al., 2023; Touvron et al., 2023), it is a complex and computationally expensive process that involves training multiple models, including a supervised fine-tuned model, a reward model, and the final RLHF model. To address this complexity, Rafailov et al. (Rafailov et al., 2024) introduced DPO, which directly learns human preferences from the preference dataset without requiring a reward model. By eliminating this step, DPO reduces computational costs while preserving the same optimization objectives as RLHF, making it a more efficient and dynamic alternative.

Thus far DPO has been primarily used to align decoder-only LLMs with human preferences; it has not been applied to encoder-decoder models used for the MRC task with a likelihood maximization objective. DPO inherently aims to increase the log probability of expected outputs over rejected outputs. A dataset of diverse instances of correct and incorrect output pairs can provide proper signals to the model about challenging examples that a supervised fine-tuned model struggles to predict accurately. Based on this observation, we hypothesize that DPO can be utilized to enhance the performance of a supervised fine-tuned encoder-decoder model in log-likelihood maximization. To test this, we experiment with a recent biomedical MRC dataset, Radiology Question Answering (RadQA) (Soni et al., 2022), resulting in the following contributions and findings:

- Compared with the encoder-only models used in prior efforts with RadQA, we show over 10% F-score improvement by shifting to encoder-decoder models, achieving a new state of the art (SoTA) score.

- We introduce two new methods to automatically generate paired preference data for the MRC task and use them to produce additional 1-3% F1 gains with DPO, leading to overall gains of 12–15% F1 points over SoTA.

The code and data from our experiments are available here: RadQADPO-code. If accepted, we will make them available on our lab's GitHub.

2 Related Work

2.1 Machine reading comprehension

MRC is a key research area within information extraction that focuses on enabling machines to extract answers from given texts. Specifically, an MRC model receives a passage (context) and a question as input and aims to answer the question by reasoning over both. Unlike general or open-domain question answering (QA) (Reddy et al., 2019; Karpukhin et al., 2020; Yasunaga et al., 2021), which typically involves retrieving answers from large corpora or knowledge bases, MRC operates in a more constrained setting where the relevant information is already provided in the input context. While MRC is important in and of itself, it also plays a crucial role in open ended QA where an initial retrieval model extracts relevant documents for a question from a search index. MRC is then applied to each of these documents and the answers are ranked using other heuristics. Prior efforts in deep learning for MRC focused on attention mechanisms, which helped models focus on relevant parts of the query and the context (Seo et al., 2016; Cui et al., 2017). More recently, approaches using transformer-based LMs, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019) have demonstrated superior performance on this task. These models leverage large-scale pre-training on diverse datasets followed by fine-tuning on specific MRC tasks, enhancing their ability to generate accurate answers. For example, ForceReader (Chen and Wu, 2020) is a BERT based method that addressed the attention deconcentration problem in MRC and introduced a few novel ideas including *attention separate representation*, *multi-mode reading*, and *conditional background attention* to improve MRC. Similarly, Lu et al. (Luo et al., 2020) proposed a novel approach that leverages BERT and BiDAF (Seo et al., 2016), extending probability vectors to probability matrices to predict the start and end positions of the answer span more accurately.

More recently, transformer-based decoder-only large language models (LLMs) (Yang et al., 2022; Singhal et al., 2023; Wu et al., 2024) have demonstrated strong or even state-of-the-art performance on a variety of machine reading comprehension (MRC) benchmarks across both general and biomedical domains, largely due to their powerful generalization capabilities. These models are typically evaluated on generative and multiple-choice

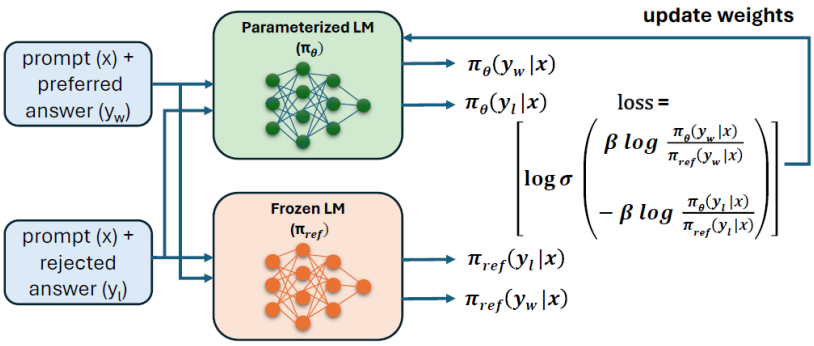

Figure 1: Pipeline of fine-tuning the language model using DPO. π_θ is the language model we want to fine-tune, and π_{ref} is the reference model, which is kept frozen during the fine-tuning process. Both models are initialized with the Supervised Fine-Tuned (SFT) model.

question-answering tasks that rely on given contexts, rather than on traditional span-based MRC tasks such as SQuAD (Rajpurkar et al., 2016) or RadQA, which require predicting exact answer spans within the context.

In our approach we also used transformer-based LMs. In contrast to the previously discussed methods, we have used an encoder-decoder transformer model (Raffel et al., 2020) as the base model and fine-tuned it by adopting the DPO method. Thus, the most closely related work to ours involves RL-based MRC methods. Although this domain is less explored compared to other deep learning approaches discussed above, several studies have applied RL techniques in question answering systems (Hu et al., 2018; Lee et al., 2021; Gharagozlou et al., 2022). These approaches typically design a reward function to optimize the model using RL algorithms. However, by leveraging the DPO technique in our method, we obviate the need of a reward function for training the model.

2.2 Reinforcement learning from human feedback (RLHF)

RLHF is an RL technique that optimizes models using human feedback instead of predefined reward functions. Initially explored for training RL agents (Akrour et al., 2012) where reward functions are difficult to specify, RLHF has more recently been widely used to fine-tune LLMs to better align with human preferences. This method has been successfully applied in various NLP tasks, including conversational agents (OpenAI, 2022), text and dialogue summarization (Chen et al., 2023), question-answering (Nakano et al., 2021), and recommendation systems, where aligning the responses with human judgment is crucial. However, RLHF is a multi-step process that can be computationally intensive. Direct Preference Optimization (DPO) (Rafailov et al., 2024) has emerged as a more efficient alternative, aiming to achieve similar objectives with reduced computational costs. While DPO is primarily used to align language models with human judgment (Tunstall et al., 2023; Zhao et al., 2023), we explore its application in likelihood maximization for MRC. By applying DPO to enhance supervised fine-tuned models, we aim to improve performance by optimizing responses to match ground truth answers more closely.

3 Methods

We use the encoder-decoder model T5 (Raffel et al., 2020) as the backbone of our main method as opposed to the BERT based baselines reported earlier (Soni et al., 2022). We also experimented with the Flan-T5 model (Longpre et al., 2023) which have been *instruction tuned* on a variety of NLP datasets and tasks. Our DPO-based optimization consists of two steps: (1) training a supervised fine-tuned T5 model and (2) optimizing it using DPO.

3.1 Training supervised fine-tuned (SFT) model

In this step, we trained an initial model for MRC using the supervised fine-tuning approach with the original training data, which we refer to as the SFT model. We model MRC as a text to text task and opted to use a seq-2-seq model for training the SFT model. The model's input is the tokenized vectors of the concatenated context and question and the output is the answer span from the context or "no answer" if the answer is not available in the context. We formatted the input sequence before tokenization as follows: "context: the text of the

context <SEP>question: text of the question."

3.2 Optimizing using DPO

After training the SFT model, we further fine-tuned it using the DPO method. This requires a preference dataset consisting of tuples (x, y_w, y_l), where x is a prompt and y_w and y_l are the preferred and rejected responses for the prompt x, respectively. In standard RLHF/DPO techniques, the preference dataset is usually constructed using human annotators. For each input, multiple outputs are generated by the initial SFT model and human annotators are asked to rate them as preferred or rejected outputs. In contrast to the standard DPO, here we built the preference dataset automatically without human interventions. Our approaches to create the preference dataset are discussed in Section 4.2.

After generating the preference dataset, we applied DPO to optimize the SFT models. The DPO architecture employs two models simultaneously for fine-tuning: one is the reference model (π_{ref}), while the other is the active model, π_θ, which is being optimized. Both models are initialized with the SFT model trained in the previous step. The weights of the reference model (π_{ref}) are kept frozen throughout the training process, while the weights of the model π_θ are updated using the DPO loss (Eq. (4) of Appendix A.1). The reference model ensures that fine-tuning does not cause the policy of the model π_θ to deviate significantly from the initial SFT model. While the DPO loss aims to increase the difference between the policies for the preferred and rejected outputs, it also aims to minimize the difference between the policies of the SFT and the active model π_θ. Both models receive input in the form of the tuple (x, y_w, y_l). In our study, the prompt x consists of the concatenated string of the context and question, y_w is the correct answer span and y_l corresponds to one of the incorrect answers for the question, given the context. Given the prompt, both models provide the probability distribution of the tokens of the preferred and rejected answers, which are used to compute the loss and update the weights of the active model π_θ. Figure 1 depicts the process of DPO more elaborately.

4 Datasets

We need two datasets to build the models in the two phases of our method. The first is the original RadQA dataset, which was used for training and validating the SFT model. The second is a preference dataset created from RadQA, and used for further tuning of the SFT model via DPO.

Preference Dataset	F1 Threshold		
	0.9	0.7	0.5
Model-based-T5	3280	2865	2354
Model-based-Flan-T5	3089	2533	2036
Rule-based	3716	3501	3332

Table 1: #instances in the preference dataset created by each method applying different F1 threshold values.

4.1 RadQA

RadQA(Soni et al., 2022) is an MRC dataset created from radiology reports from the MIMIC III dataset (Johnson et al., 2016). The questions were manually created from the clinical referral sections to capture the actual information needs of ordering physicians, without being influenced by seeing the answer context. Answers were annotated in the Findings and Impressions sections and consist of complete, concise phrases that may span multiple lines and are not limited to named entities. The dataset also includes unanswerable questions, supporting the challenges of real-world clinical question answering.

The RadQA dataset comprises 6148 unique question-answer pairs sourced from 1009 radiology reports of 100 patients. The dataset was split at the patient level into training, development, and testing sets, with an 8:1:1 ratio, respectively. This resulted in 4878 questions in the training set, 863 questions in the development set, and 894 questions in the test set. We used the original format of training data of RadQA exclusively to train the SFT model, while the development and test data were used for evaluating both the SFT and DPO models to assess the effectiveness of our approach.

4.2 Preference dataset

Preference data is the main element for optimizing a language model through DPO. This consists of tuples that include examples of preferred and rejected outputs for a given prompt. Although preference data is typically collected from human annotators, we automatically generated it, eliminating the need for manual annotation. We used the original training corpus of RadQA for this purpose. Specifically, each prompt was formed by concatenating the context and question from the RadQA training dataset,

separated by a special token. The preferred output is the original gold answer span provided in the dataset. To generate the corresponding rejected output, we propose two automated approaches: a model-based approach and a rule-based approach.

4.2.1 Model based approach

In this approach, we used the SFT model itself to generate negative examples. The process began by training a model on 50% of the RadQA training data and then using it to predict answers for the entire training dataset, including the data it was trained on. The rationale behind training on half of the data was to equip the model with sufficient knowledge for effective performance. Thus, mistakes made during these predictions indicate the types of examples the model needs to focus on to improve its performance. Testing the model on both seen and unseen data helps identifying specific examples that remain challenging despite prior exposure. Our intuition behind this design is that by using the model's own incorrect predictions, we can better identify the types of examples where it struggles. These incorrect predictions highlight situations where the model needs improvement, making them valuable for training. Additionally, since the model is also tested on examples it was trained on, any errors it makes on these familiar examples indicate that they are particularly challenging. By focusing on these hard examples, we aim to improve the model's overall performance.

We identified all instances where the model generated incorrect answers. For each prompt and question pair where the model's prediction differed from the original answer, the incorrect prediction was recorded as the rejected output in our preference dataset. To refine the preference dataset, we filtered these incorrect answers based on their F1 scores. The F1 score was calculated by comparing word-level matches between each incorrect answer and its corresponding original answer. To filter the incorrect predictions, we applied three different thresholds for the F1 score: 0.9, 0.7, and 0.5. If the F1 score between the original and the predicted answer was less than the chosen threshold, the predicted answer was selected as the rejected output. To ensure comprehensive coverage, we repeated this process by training another model on the remaining 50% of the training data. This model was then used again to predict answers for the entire dataset, allowing us to identify additional incorrect predictions. We used two variants of SFT models (T5-3B and Flan-T5-3B) to create the negative examples. The total number of instances created by this process is shown in Table 1.

By iteratively training on different halves of the dataset and collecting incorrect predictions, we effectively created a robust set of negative examples without the need for manual annotation. This automated generation of preference data not only streamlined our process but also ensured a diverse range of negative examples, enhancing the quality of our preference dataset. Our assumption is that DPO will help the model improve on these challenging examples, enhancing overall performance.

4.2.2 Rule based approach

We generated negative examples from the training data by applying a set of predefined rules. These rules were formulated based on experimental findings regarding the types of errors that SFT model typically makes. For each tuple (context, question, gold answer) in the training data, we generated a number of incorrect answers applying the following rules (also shown with a few examples in Figure 3 of Appendix A.3):

- *Random text span:* Select a random span from the context that does not contain any part of the gold answer.
- *Text span containing part of the gold answer:* Here, a text span from the context that includes a part of the original answer is randomly chosen. This partial inclusion can occur in two ways: 1) choosing a segment starting a few words before the left side of the gold answer and continuing until it includes a partial span from the gold answer, or 2) selecting a partial segment from the right side of the answer and including a few words after the answer text. The lengths of these segments are chosen randomly (see Figure 3).
- *Longer answer:* This entails a text span that includes the entire gold answer as a part of it with ≥ 1 additional tokens.
- *Partial answer only:* This involves selecting a smaller segment (strict substring) from the original answer.
- *Answers of a different question:* Here, an answer text from another question in the same context is chosen, provided it is not the same as the original gold answer or a part of it. For

example in Figure 3 of Appendix A.3, "kidneys are normal in appearance" is an answer to a different question for the same context, but is not part of the ground truth answer.

- *No answer:* In this approach, we used empty string in place of the gold answers to create negative examples. For questions without available answers, we chose responses from other questions within the same context as negative examples. If there were no other questions within the same context that provided answers, we randomly selected a span from the context as the negative answer.

Following these rules provided us with a large number of examples of rejected answers for each (context, question, gold answer) tuple. From each set of rejected answers, we randomly chose a few examples to create the preference data. We did not include the entire set of rejected answers for generating the preference data to prevent the dataset from becoming intractably large. Finally, we included 4000 instances and further filtered them by applying F1 threshold (see Table 1).

5 Experimental Setup

5.1 Baselines

We compared our T5-based SFT models with the BERT-based models from Soni et al. (Soni et al., 2022), which offered SoTA results on the RadQA dataset. Thus, we selected all of their BERT-MIMIC-based models as our baselines. These models come in four variants, based on the datasets used for fine-tuning. The first variant, BERT-MIMIC-RadQA, was fine-tuned only on the RadQA dataset. The remaining three variants were additionally fine-tuned on external QA datasets such as SQuAD (Rajpurkar et al., 2016) and EmrQA (Pampari et al., 2018). For example, BERT-MIMIC-SQuAD-RadQA was trained on both RadQA and SQuAD, while BERT-MIMIC-EmrQA-RadQA was trained on both EmrQA and RadQA.

We also compared our DPO-based method with the T5 SFT models to assess the effectiveness of applying DPO on an already high-performing fine-tuned model.

5.2 Evaluation metrics

To evaluate our proposed method, we used the standard MRC metrics: Exact match (EM) and F1-Score. Exact Match is a strict metric that compares the predicted answer with the exact ground truth answer, ensuring they are identical. The F1-Score, on the other hand, is calculated by taking word-level matches between the predicted and ground truth answers. To maintain consistency and comparability in our evaluation, we used the evaluation code from SQuAD (Rajpurkar et al., 2016).

5.3 Network parameters and resources

The network parameters for each model in our experiments were chosen through hyperparameter tuning. We used the validation F1 score as an evaluation metric to select the optimal values of these parameters. For training both the SFT and DPO models, we employed the Adam optimizer. The learning rate for the SFT model was set to $5e^{-5}$, and for the DPO model, it was $5e^{-7}$. The weight decay was set to 0.01 for both models. The batch size was 16 for T5-Large models; however, to accommodate the 3 billion parameter models in memory, we used a batch size of 2 with gradient accumulation steps of 8. The maximum prompt length was set to 768, and the target length was 128. Early stopping was applied during the training of both the SFT and DPO models, by using the validation F1 score to select the best models. All our experiments were conducted on a single NVIDIA H100 GPU, equipped with 80 GB of memory.

6 Results

Table 2 presents the main results of our experiments, comparing the performance of BERT baselines, the T5-based supervised fine-tuned (SFT) models, and the DPO based models. The results are evaluated on the development and test sets of the RadQA dataset.

The SFT model type includes three T5 variants (T5-large, T5-3B, and Flan-T5-3B) trained on the RadQA training data. From Table 2, we can see that all the T5 variants outperform the baseline RadQA models on the test set, with Flan-T5-3B also performing better on the dev set. Specifically, the SFT Flan-T5-3B achieves an F1 score of 76.38 and an exact match (EM) score of 55.93 on the test set, marking improvements of 13 points in F1 score and 6.5 points in EM over the best baseline model. Although the three variants of BERT-MIMIC were trained on additional datasets (SQuAD and emrQA) along with RadQA, the T5 models still outperformed them, establishing a strong baseline for our DPO-based method. It is important to note

Model Type	Models	Dev		Test	
		EM	F1	EM	F1
Baseline (BERT-MIMIC) (340M)	RadQA	48.05	65.85	45.73	60.08
	emrQA-RadQA	50.65	67.97	47.71	61.60
	SQuAD-RadQA	52.28	69.42	**49.39**	**63.55**
	SQuAD-emrQA-RadQA	53.26	67.79	48.32	62.29
T5-large (770M)	SFT	47.86	66.22	49.89	71.10
	DPO-MB	47.74	66.25	**51.34**	**71.62**
	DPO-RB	48.20	66.59	51.00	71.36
	DPO-MRB	47.80	66.10	50.11	71.20
T5-3B	SFT	49.83	68.59	51.68	72.29
	DPO-MB	51.10	70.45	52.46	74.29
	DPO-RB	50.87	70.26	**52.57**	74.03
	DPO-MRB	50.40	70.13	52.01	**75.18**
Flan-T5-3B	SFT	54.35	72.62	55.93	76.38
	DPO-MB	53.77	73.68	55.15	**77.48**
	DPO-RB	52.49	72.55	**56.15**	77.40
	DPO-MRB	53.42	73.51	55.70	77.41

Table 2: Model performances on the RadQA development and test sets compared with the RadQA BERT-MIMIC model variants.

that, although BERT-MIMIC was fine-tuned on a large corpus of clinical notes (Si et al., 2019) (1.9 million notes comprising approximately 786 million tokens), our T5 models have more parameters than the 340M BERT-based models used in the RadQA paper and were pretrained on a much larger and more diverse dataset—the C4 corpus, which contains around 750GB of clean web text. This provides T5 with stronger language capabilities.

The DPO-based methods include three groups of models: DPO-Model Based (DPO-MB), trained on model-based preference data; DPO-Rule Based (DPO-RB), trained on rule-based preference data; and DPO-Model & Rule Based (DPO-MRB), trained on a combined dataset of model-based and rule-based preference data. For all the models, we selected the preference data generated by 0.9 F1 threshold. Additionally, for training the DPO-MB models, we used the model specific preference data. For instance, we applied model-based-T5 preference data for the T5 models and model-based-Flan-T5 preference data for the Flan-T5 based DPO models. From Table 2 we can see that both model and rule-based DPO models improved the performance of the corresponding SFT models. Although the T5-large SFT model did not see a significant improvement, the T5-3B and Flan-T5-3B improved their corresponding SFT models nontrivially, both in DPO-MB and DPO-RB settings. For instance, the F1 score of the DPO-MB T5-3B is 74.29, a 2-point improvement over its SFT counterpart and an 11-point increase compared to the best performing baseline model, BERT-MIMIC-SQuAD-RadQA, on the test F1 score. The combined dataset further improved the test F1 score of the T5-3B model by 1%, but it did not enhance the other variants, indicating saturation in the performance of the models.

7 Discussion

Our experimental results demonstrate that further fine-tuning an SFT model through DPO can enhance its performance between 1–3% F1 points. This is particularly important because these SFT models have already been optimized using the full training dataset, making further improvements challenging. From our experiments, we found several factors that influence the performance of the models trained with DPO, including the size of the SFT models, the method used to create negative examples in the preference data, the types of examples

included, and the quantity of preference data. In this section, we provide a detailed discussion on the observed performance improvements using DPO and the factors influencing these improvements.

7.1 Size of the model

From our results, we notice that a smaller model is less likely to benefit from additional fine-tuning with DPO. However, with larger models, notable improvements were observed. For instance, with DPO both T5-3B and Flan-T5-3B increased the test F1 score of their corresponding SFT models by 1-3%. This indicates the ability of larger encoder-decoder models to capture signals from examples of preferred and rejected outputs. However, among 3B models, the improvement is much better in the non-Flan model. Since the Flan model is instruction tuned on hundreds of datasets, its SFT performance (76.38 F1) is already over 1% better than the best DPO model of its non-Flan counterpart.

7.2 Model- vs rule-based preference data

While DPO-MB and DPO-RB both enhanced the performance of the SFT models, our experiments showed that the model-based approach yielded comparatively better results than the rule-based approach. One potential reason for this could be the nature of the negative examples generated by each method. Rule-based examples are created using predefined rules. Although these rules are designed to generate plausible negative examples, they may not always reflect the same distribution as the original RadQA dataset. This can lead to less effective training, as the model might not encounter a representative range of challenging examples during the DPO training. In contrast, the model-based approach derives negative examples from the predictions of the SFT model itself. These examples are intrinsically linked to the specific weaknesses of the model. By focusing on these model-specific errors, the preference data reflects the instance spaces where the model is prone to generate incorrect outputs. Consequently, this approach may offer more targeted training, enabling the model to learn from its mistakes and improve its performance. However, one limitation of this method is that each new model requires the creation of a new preference dataset, as each model has different weaknesses and strengths. In contrast, the training examples created by the rule-based approach are model-agnostic.

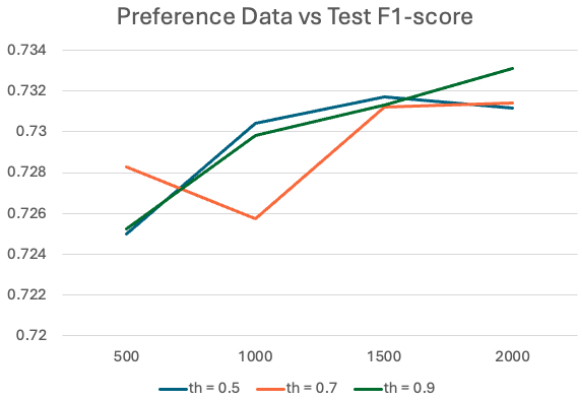

Figure 2: Performance comparison of DPO-T5-3B model with varying training examples and preference datasets generated using different thresholds. X-axis plots #training-examples, Y-axis is the F1 score, and the line colors represent different preference datasets created by applying three different F1 thresholds.

7.3 Diversity of training instances

Filtering the preference data based on different F1-score thresholds also influences the performance of DPO. Negative examples with higher F1 scores tend to be closer to the ground truth answers, while those with lower F1 scores present more dissimilarity with gold spans. Incorporating a broader range of negative examples from both ends of the F1-score spectrum provides a diverse and more informative training set for the model. A higher F1-score threshold facilitates a mix of examples that are both similar and dissimilar to the ground truth answers, offering a wide variety of training data. Conversely, a lower threshold focuses only on the examples that are very different from the ground truth, excluding those that are more similar. Therefore, preference data created using higher thresholds may enable the model to learn from a diverse set of examples, which can enhance its generalization and performance. Our experiments also support this hypothesis. Figure 2 illustrates the test F1-scores of DPO-T5-3B models trained with preference data filtered at different thresholds. The results show that the model trained with a threshold of 0.9 outperforms those trained with lower threshold data, demonstrating the benefits of using a more diverse set of training examples.

7.4 Number of training Instances

Besides diversity of training examples, the number of training examples also impacts the performance of DPO based models. We fine-tuned DPO-T5-3B with different numbers of training examples (500,

1000, 1500 and 2000) for each filtering threshold. As shown in Figure 2, an increase in the number of training examples generally leads to an increase in the test F1 score across all thresholds.

7.5 Other variants of DPO

DPO has evolved into several variants, each with different a loss function, designed to address specific issues. For instance, Identity Preference Optimization (IPO) (Azar et al., 2024) was developed to mitigate the overfitting problem identified in DPO by introducing a new loss function. We trained our model using three DPO variants: Identity Preference Optimization (IPO), Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024), and Statistical Rejection Sampling Optimization (RSO) (Liu et al., 2024). Our experimental results show that DPO outperforms other variants for both T5-3b and Flan-T5-3b models. Detailed results are provided in Table 3 of Appendix A.2.

8 Conclusion

In this paper, we proposed an approach that combines encoder-decoder models with DPO based optimization to achieve new SoTA performances on the MRC task for radiology using the RadQA dataset. Our study shows that encoder-decoder models, although computationally expensive due to large model capacities, can offer substantial gains in performance (by over 10% in F1 scores). Originally introduced for aligning LLMs with human preferences, our study demonstrated that DPO methods can also be effectively used for likelihood maximization for MRC tasks and can lead to further gains of up to 3% beyond the encoder-decoder based gains. By focusing on challenging examples (the model-based preference data setup), DPO can further improve large models already fully trained.

While effective, one key challenge in fine-tuning models using DPO is that its performance is highly dependent on the quality of the preference data. Collecting high-quality examples of preferred and rejected outputs is crucial for maximizing the model's performance through DPO. In this work, we introduced two techniques—the model-based and the rule-based approaches to generate preference data for the MRC task, which can be adopted in other tasks as well. In future, we will explore the applicability of our approach in other information extraction tasks such as named entity recognition and relation extraction.

Acknowledgment

This work is supported by the U.S. National Library of Medicine through grant R01LM013240. The content is solely the responsibility of the authors and does not necessarily represent the official views of the U.S. National Institutes of Health.

References

Riad Akrour, Marc Schoenauer, and Michèle Sebag. 2012. April: Active preference learning-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012. Proceedings, Part II 23*, pages 116–131.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the ACL*, pages 643–653.

Jiaao Chen, Mohan Dodda, and Diyi Yang. 2023. Human-in-the-loop abstractive dialogue summarization. In *Findings of the ACL: ACL 2023*, pages 9176–9190.

Zheng Chen and Kangjian Wu. 2020. ForceReader: a BERT-based interactive machine reading comprehension model with attention separation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2676–2686.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 593–602.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, volume 28.

Dina Demner-Fushman, Wendy W. Chapman, and Clement J. McDonald. 2009. What can natural language processing do for clinical decision support? *Journal of Biomedical Informatics*, 42(5):760–772.

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 ACL: Human Language Technologies, Volume 1*, pages 4171–4186.

Ondrej Dusek, Jekaterina Novikova, and Verena Rieser. 2017. Referenceless quality estimation for natural language generation. In *1st Workshop on Learning to Generate Natural Language*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Hamid Gharagozlou, Javad Mohammadzadeh, Azam Bastanfard, Saeed Shiry Ghidary, et al. 2022. Rlasbiabc: A reinforcement learning-based answer selection using the bert model boosted by an improved abc algorithm. *Computational Intelligence and Neuroscience*, 2022.

Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4099–4106.

Qiao Jin, Zheng Yuan, Guangzhi Xiong, Qianlan Yu, Huaiyuan Ying, Chuanqi Tan, Mosha Chen, Songfang Huang, Xiaozhong Liu, and Sheng Yu. 2022. Biomedical question answering: A survey of approaches and challenges. *ACM Comput. Surv.*, 55(2).

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

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.

Urvashi Khandelwal, Kevin Clark, Dan Jurafsky, and Lukasz Kaiser. 2019. Sample efficient text summarization using a single pre-trained transformer. *arXiv preprint arXiv:1905.08836*.

Hyeon-Gu Lee, Youngjin Jang, and Harksoo Kim. 2021. Machine reading comprehension framework based on self-training for domain adaptation. *IEEE Access*, 9:21279–21285.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2024. Statistical rejection sampling improves preference optimization. In *The Twelfth International Conf. on Learning Representations*.

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

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648.

Huaishao Luo, Yu Shi, Ming Gong, Linjun Shou, and Tianrui Li. 2020. MaP: A matrix-based prediction approach to improve span extraction in machine reading comprehension. In *Proceedings of the 1st Conf. of the Asia-Pacific Chapter of the ACL and the 10th International Joint Conf. on NLP*, pages 687–695.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2022. ChatGPT: Optimizing language models for dialogue.

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.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrQA: A large corpus for question answering on electronic medical records. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2357–2368.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

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.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *International Conference on Learning Representations 2023*.

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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. 2019. Enhancing clinical concept extraction with contextual embeddings. *Journal of the American Medical Informatics Association*, 26(11):1297–1304.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Sarvesh Soni, Meghana Gudala, Atieh Pajouhi, and Kirk Roberts. 2022. RadQA: A question answering dataset to improve comprehension of radiology reports. In *Proceedings of the 13th Language Resources and Evaluation Conf.*, pages 6250–6259.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. 2024. Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, 31(9):1833–1843.

Xi Yang, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Anthony B Costa, Mona G Flores, et al. 2022. A large language model for electronic health records. *NPJ digital medicine*, 5(1):194.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546.

Zhiyuan Zhao, Bin Wang, Linke Ouyang, Xiaoyi Dong, Jiaqi Wang, and Conghui He. 2023. Beyond hallucinations: Enhancing lvlms through hallucination-aware direct preference optimization. *arXiv preprint arXiv:2311.16839*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Appendix

A.1 Background for RLHF and DPO

Fine-tuning LLMs for downstream tasks using RLHF technique involves three main phases (Stiennon et al., 2020; Bai et al., 2022b): 1. supervised fine-tuning, 2. constructing reward model, and 3. fine-tuning the language model using RL methods.

A.1.1 Supervised fine-tuning

This is the initial step of RLHF technique, where the language model undergoes supervised fine-tuning on downstream tasks. During this phase, the model is trained on specific task-related training datasets, allowing it to adapt its pre-trained knowledge to the particular downstream task. The model trained in this phase is commonly referred to as supervised fine-tuning (SFT) model, denoted as π_{sft}.

A.1.2 Constructing reward model

After training the SFT model, the next step is to develop a reward model that evaluates the SFT model's outputs based on human preferences and represent it as scalar values. This reward model can be built using pre-trained models capable of assessing outputs according to human judgment (Bai et al., 2022b), or by training it on human preference data collected from annotators.

To construct human preference data, multiple responses are first generated for each prompt by the SFT model, using different variants of the model or sampling methods (Stiennon et al., 2020; Bai et al., 2022a). The collection of prompts and their generated responses are then formatted into a batch of tuples $(x, y1, y2)$, where x is the prompt and y_1 and y_2 are pair of responses sampled from the set of generated responses of the prompt x. Human labelers are then instructed to choose their preferred response between the two. This process creates a preference dataset consisting of tuples (x, y_w, y_l), where y_w represents the preferred output and y_l represents the rejected output.

From the generated preference dataset D, the probability distribution of human preference can be formulated as

$$p(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l)) \tag{1}$$

using Bradley-Terry model (Bradley and Terry, 1952) given an optimal reward model r, where σ is the logistic function.

With the preference dataset $D = \{(x^i, y_w^i, y_l^i)\}_{i=1}^N$, we parameterize the reward model r_σ and optimize it by maximizing the log likelihood of the difference between the reward of preferred response and rejected response (as in Eq. (1)) and hence minimize the loss

$$\mathcal{L}(r_\sigma) = E_{(x,y_w,y_l)\sim D}[-\log(p(y_w > y_l|x))]. \tag{2}$$

A.1.3 Fine-tuning Using RL method

Finally, in this step, the trained reward model r_σ is used to provide feedback on the output of the parameterized language model π_θ and optimize it by the objective of maximizing the expected reward

$$r(x, y) = r_\sigma(x, y) - \beta(\log(\pi_\theta(y|x)) - \log(\pi_{ref}(y|x))) \tag{3}$$

where π_θ denotes the policy of the language model we are optimizing and π_{ref} is the initial SFT model. During the RL training phase, the parameters of the SFT model π_{ref} remain fixed. π_θ is initialized with π_{ref} and optimized using an RL algorithm, most commonly PPO (Schulman et al., 2017) and other variants of actor-critic (Ramamurthy et al., 2023) algorithms. The parameter β ensures that the trained policy π_θ will not deviate significantly from the initial SFT model π_{ref}.

While RLHF is effective, it requires training a separate reward model, which makes the overall process costly. DPO eliminates the need for a reward model by directly optimizing the language model π_θ using the policies of both the reference model π_{ref} and π_θ itself. The objective function of DPO is to maximize the policy difference between the preferred output y_w and the rejected output y_l as in

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -E_{(x,y_w,y_l)\sim D}\left[\log \sigma(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)})\right]. \tag{4}$$

A.2 Additional results

Loss	T5-3B				Flan-T5-3B			
	Dev		Test		Dev		Test	
	EM	F1	EM	F1	EM	F1	EM	F1
DPO	**51.10**	**70.45**	**52.46**	74.29	53.77	73.68	55.15	**77.48**
IPO	50.64	69.41	51.57	73.41	53.53	73.06	53.36	76.79
RSO	49.83	69.55	50.90	**74.31**	53.88	73.50	**55.48**	77.24
KTO	47.74	68.21	51.12	74.24	**54.11**	**73.76**	53.36	77.20

Table 3: Results on the variants of DPO.

Table 3 shows the performance of the models for different variants of DPO. Although different DPO variants achieve better performance on different metrics, overall, DPO outperforms others for T5-3B in most cases, except for the test F1 score, where RSO achieves an F1 score of 74.31. For Flan-T5-3B, DPO outperforms others in the test F1 score and performs comparably to the others on the remaining metrics.

A.3 Examples of negative outputs created by rules

Figure 3 shows the example of negative samples created by the rule–based method.

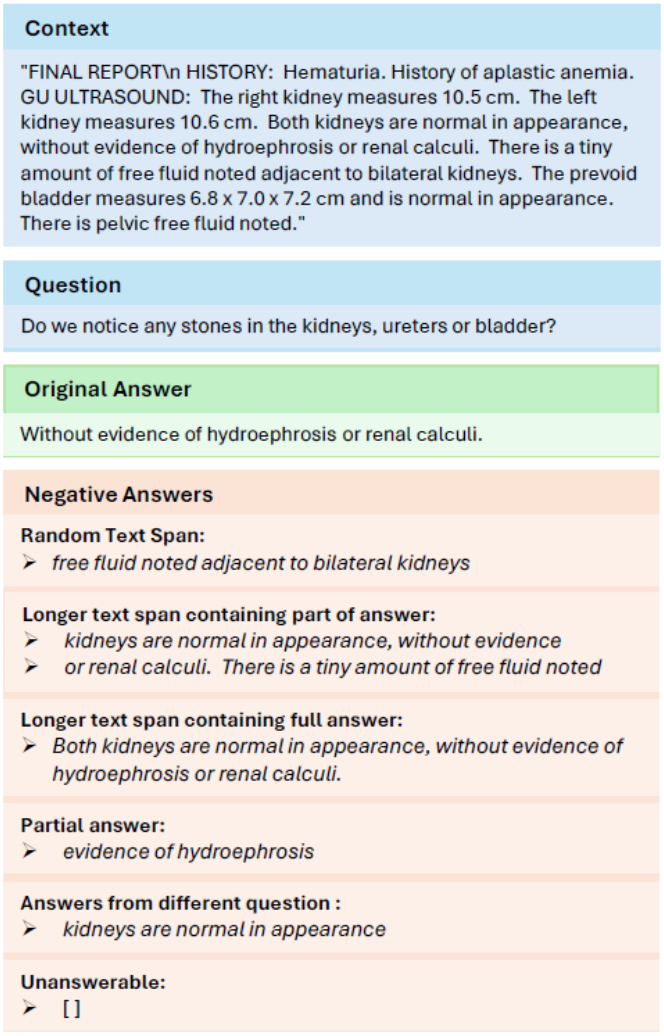

Figure 3: Examples of negative (rejected) outputs created by rules.