Feedback-Aware Inference for Multiple Samples Generation

Anonymous ACL submission

Abstract

Generating multiple text sequences and refining them through feedback is essential for improving the quality of outputs in many NLP tasks. While Large Language Models can leverage iterative feedback during inference, smaller models often lack this capability due to limited capacity and the absence of suitable training paradigms. In this paper, we propose a novel Feedback-Aware inference approach that enables iterative sequence generation with integration of feedback signals. Our method allows models to generate multiple sequences, incorporate feedback from previous iterations, and refine outputs accordingly. This approach dynamically adjusts to different quality metrics, making it adaptable to various contexts and objectives. We evaluate our approach on two distinct tasks: Answer Selection for Question Generation and Keyword Generation, arguing for its generalizability and effectiveness. Results show that our method outperforms strong baselines, maintaining high performance across iterations and achieving superior results even with smaller, open-source models.

1 Introduction

Many NLP tasks extend beyond generating a single response from a model. Such tasks often benefit from producing a diverse pool of possible sequences rather than a single deterministic output. Generating multiple sequences allows for a more comprehensive exploration of possible solutions, improving the chances of obtaining a result that better aligns with the desired outcomes.

However, the value of generating multiple sequences is limited if the model cannot iteratively refine its outputs based on feedback. The ability to self-correct during inference by learning from previous iterations is essential for enhancing sequence quality. Large language chat models have demonstrated the capability to adjust their responses based on a conversational flow. Yet, this behavior remains largely absent in smaller models due to their limited generalization capacity and because they are not specifically trained for this. Despite their limitations, smaller models are needed for real-world applications due to their lower computational requirements and cost-effectiveness. Our work addresses this gap by proposing a method that enables models to generate sequences iteratively, integrate feedback signals from previous outputs, and refine their responses accordingly.

In this paper, we propose a novel approach called Feedback-Aware inference. This method iteratively generates sequences based on information from a text and allows for user or model feedback regarding the quality of the proposed sequence and resulting content. By incorporating this feedback from the previous sequences and providing information on the content resulting from this sequence, the model refines its process, leading to identifying spans that are more aligned with the desired ones. This approach offers several advantages. First, it enables the generation of multiple sequences, leading to a richer set of potential content. Second, it enables the integration of different quality measures, making the method adaptable to various generation goals. Finally, the iterative feedback process allows the model to use in-context learning during inference.

To argue for the effectiveness of our method, we evaluated it on two distinct tasks: Answer Selection for Question Generation and Keyword Generation. In the Answer Selection for Question Generation task, the objective is to identify multiple spans within a given context that can serve as answers for question generation. The goal of the second task is to generate relevant keywords for a paper based on its abstract. Notably, our method is generalizable to any task that involves generating multiple text sequences from a given context.

Our main contribution is the novel Feedback-Aware training and inference approach that enables the dynamic refinement of generated sequences. As such, our model has the opportunity to correct itself and learn from feedback and previously generated options. Our approach achieves superior results on two different generative tasks even with smaller, open-source models, highlighting the potential for cost-effective solutions without relying on massive, proprietary LLMs. We release our code and best models as open-source. [1]

2 Related Work

2.1 Feedback for Model Refinement

Few-shot prompting, a technique that involves providing a small number of examples as part of the prompt to guide the LLM's generation, is a frequently employed approach for controlling and refining LLM outputs at inference time. Early works by Gao et al. (2021) and Tam et al. (2021) proved the effectiveness of few-shot learning in improving LLM performance, even with limited training data. Subsequent works by Schick and Schütze (2022) and Perez et al. (2021) have further refined few-shot prompting techniques, making them more practical for real-world applications.

Alignment with human preferences is usually done through Reinforcement Learning from Human Feedback (Ouyang et al., 2022) or Direct Preference Optimization (Rafailov et al., 2024). While these solutions increase the quality of generated answers, they do not directly tackle the problem of generating multiple samples and do not take into account feedback signals at inference time.

Only recently, works have studied advanced techniques for feedback-based refinement. These approaches generally involve a cyclical process: an LLM is prompted to perform a task, its output is evaluated by another LLM (or itself), and the feedback is provided to the original LLM for refinement. For example, Fu et al. (2023) created a setting where chat models are prompted to negotiate a selling price for goods, acting as the buyer or the seller and improving their offer based on the conversation. Self-refinement, a technique in which LLMs rate, highlight errors, and give feedback to their own generated content, is studied in multiple contexts. Text summarization, code generation, machine translation, and Math reasoning are tasks improved by this approach (Xu et al., 2024; Chen et al., 2024; Madaan et al., 2024). Although the previous works highlighted very promising results in terms of feedback-based refinement, the experiments use proprietary LLMs with a huge number of parameters, such as GPT-4, Claude, or PaLM-2.

2.2 Multi-sequence Generation Tasks

Question generation models usually take a context and a desired answer as input to generate a question based on them. Yao et al. (2022) specifically focused on deriving multiple answer-question pairs from a text by employing heuristic-based rules for answer selection. The authors extracted noun chunks, named entities, and event descriptions to serve as selected spans for directing the generation of questions. Zhao et al. (2022) learned to predict the possible distribution of different types of questions based on the context and extracted the relevant information from the text that would adhere to that distribution. Yoon and Bak (2023) approached the task by iteratively generating questions and appending the previously generated questions in the prompt to ensure a diverse set and direct the model's output. The fine-tuning considered the questions provided in the prompt and generated a query that diverged from the previous ones. Although this was a step forward regarding question diversity, the generation is not controllable and cannot consider other requirements. Across these papers, the common theme is supervised fine-tuning on specific QA datasets. This approach ensures that the generated questions and answers conform to the patterns found in these datasets, but the models cannot adapt based on user feedback.

Keyword generation is the task of identifying or generating terms or phrases that encapsulate the key topics of a given text, such as a paper abstract. Extensive research has been conducted on keyword extraction, where spans directly from the abstract are selected as keywords. A recent survey (Song et al., 2023) presents the latest advancements in keyphrase extraction. Advancements have been made with pre-trained language models, specifically encoder models. Specifically, for unsupervised keyphrase extraction, semantic similarity with the abstract of spans is ranked using cross-attention (Ding and Luo, 2021) or graph structures (Liang et al., 2021). In another approach, Song et al. (2021) proposed a system consisting in three modules (chunking, ranking, and matching) jointly trained. However, the extractive task has notable limitations. Specifically, paper authors frequently assign keywords that are not explicitly mentioned

[1] https://anonymous.4open.science/r/ACL-Feedback-Aware/

within the abstract but are instead derived from the broader context or inferred concepts.

3 Method

This section provides a structured overview of our proposed method. While we evaluate on two distinct tasks, each with specific characteristics, the following framework serves as a generalizable methodology applicable to all similar tasks that require generating multiple text sequences. Task-specific details are elaborated in Section 4.3. It is important to note that the objective of the paper is to propose a general method, not to focus on specific state-of-the-art solutions for each task.

3.1 Prerequisites: Labeling System

The proposed method incorporates a feedback-driven mechanism to iteratively enhance performance. This feedback signal may be derived from either human evaluation or an automated assessment procedure. The specific feedback mechanism utilized is not the primary focus of this study and can be adapted to different application domains and requirements.

The Labeling System, denoted as LS, takes a generated text sequence g as input and produces a label that signifies the quality of the generated content. In our experiments, the possible labels are $GOOD$ or BAD, indicating the suitability of the generated output ($LS(g) \in GOOD, BAD$). The labeling criteria depend on task-specific requirements (e.g., grammatical correctness, coherence, relevance, overall quality).

Optional Component. For complex tasks, such as Answer Selection in Question Generation, evaluating only the generated text sequence may be insufficient. In such cases, additional contextual information may be required for assessment. To accommodate this, our framework supports an optional component, referred to as the Resulting Content Generator (RCG), which automatically generates auxiliary content based on the generated sequence. Consequently, the Labeling System receives both the generated sequence (g) and the resulting content ($RCG(g)$) as input to determine the final label - i.e., $LS(g, RCG(g)) \in \{GOOD, BAD\}$.

3.2 Dataset Construction

The original datasets are augmented using the Labeling System (LS) to ensure that generated sequences are annotated as either $GOOD$ or BAD. To enable effective model fine-tuning, it is essential to include multiple examples of both $GOOD$ and BAD sequences for the same context.

Formally, let the initial dataset be defined as $D_{initial} = [C_1, C_2, ..., C_n]$, where each C_i represents a context. For each C_i, we construct a set of possible generated sequences $[g_{i,1}, g_{i,2}, ..., g_{i,m}]$ along with their corresponding labels $[LS(g_{i,1}), LS(g_{i,2}), ..., LS(g_{i,m})]$.

The assigned labels function as quality indicators, aiding in both model evaluation and comparative analysis against baseline approaches.

3.3 Feedback-Aware Inference

We want our model to have the opportunity to correct itself and learn from previously generated content as well as the feedback given for it. For that, the following algorithm was designed for inference.

Require: A Labeling System LS (Section 3.1)
Require: [optional] A Resulting Content Generator RCG (Section 3.1)
Require: The Feedback-Aware Model FAM
  Initialize the prompt
  P = [Task description, Context]
  while Still want to select **do**
    $g \leftarrow FAM(P + \text{"}GOOD\text{"})$
    $rc \leftarrow RCG(g)$
    $l \leftarrow LS(g, rc)$
    if $l = GOOD$ **then**
      Found a good generated sequence
    end if
    $P \leftarrow [P, l, g, rc]$
  end while

Model training is done following its purpose - i.e., selecting the information from the context and being capable of distinguishing between $GOOD$ and BAD sequences. The Feedback-Aware Model should be trained in a similar scenario to the one used during inference. Because of this, the prompt needs to include a diverse list of sequences with their corresponding quality label and generated questions. The prompt structure is shown in Figure 1, whereas Figures 2a and 2b introduce prompt examples for the evaluated tasks.

For training, we format the prompt according to the template from Figure 1, but we only compute the loss and propagate gradients for the last sequence (depicted in blue). In this manner, the model is trained to recognize the template and gen-

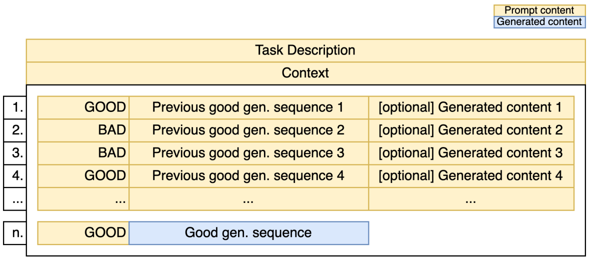


Figure 1: Prompt template for Feedback Aware Generation

erate the sequence (depicted in blue) by attending to the context, previous labels, previous generated sequences, and, optionally, previous resulting content. The model is trained to generate only *GOOD* sequences since the gradients and loss are computed only for the tokens of the last generated sequence. Moreover, we can edit the prompt using a decoder-only model - i.e., change a previously thought *GOOD* label in the actual prediction given by *LS* (the prerequisite Labeling System) and append the resulting content obtained from the *RCG* (the optional Resulting Content Generator).

4 Performance Evaluation

4.1 Baselines

Four strong baselines were selected to compare the performance of our method. Details on the full prompts (for both our model and the baselines) are provided in Appendix C.

Single Sequence Generation (SSG). This approach involves iteratively generating a possible sequence given the task and the context. This baseline is trained by supervised fine-tuning to generate the *GOOD* sequence. More formally, the general prompt format on which we fine-tune is the following: *"<task>. Text: <text>. Generated sequence: <sequence>"*. The loss and gradients are computed just for the tokens of *<sequence>*. At inference time, we over-sample 100 sequences, eliminate duplicates, and choose the top sequence in terms of log-likelihood.

All Sequences Generation (ASG). This approach simultaneously generates all *GOOD* sequences for a given text. This baseline is trained by supervised fine-tuning to generate all the *GOOD* sequences for a context. More formally, the fine-tuning prompt is the following: *"<task>. Text: <text>. Generated sequence: <seq_1>, <seq_2>... <seq_n>"*. The loss and gradients are computed just for the tokens of *<seq_i>*.

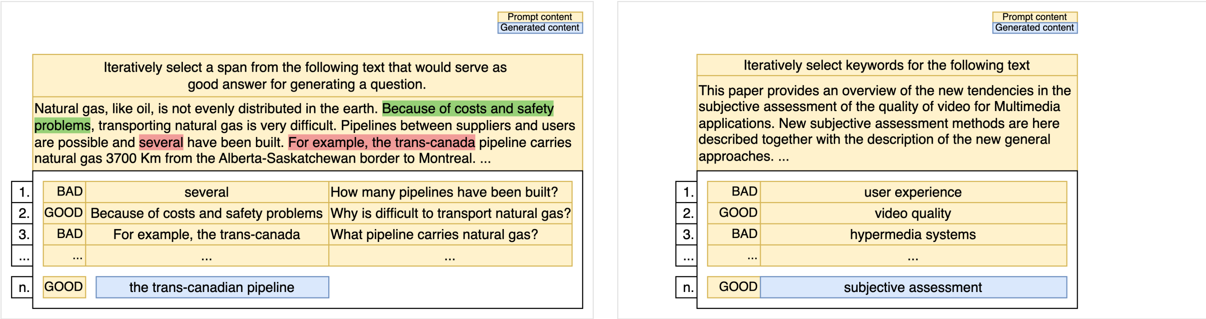
All Sequences Generation with Resulting Content (ASG-RC). In the optional case of using the Resulting Content Generator, this approach serves as a strong baseline to prove the efficacy of the feedback labels. It is trained similarly to the All Sequences Generation (ASG) approach, with the addition of the resulting content (RC) in order to inform the current iteration about the previously generated sequences and their corresponding resulting content. More formally, the fine-tuning prompt is the following: *"<task>. Text: <text>. Generated sequence: <seq_1>- <rc_1>, <seq_2>- <rc_2>, ... <seq_n>- <rc_n>"*. The loss and gradients are computed just for the tokens of *<seq_i>*. It is important to acknowledge that this baseline also functions as a control group, evaluating the impact of our design decision to train the Feedback-Aware model using *GOOD* / *BAD* labels.

GPT-4o. We use GPT-4o similarly to the previous baseline - namely, to generate sequences and (optionally) their resulting content. We use it in a 1-shot setting that includes the task and provides one text example from the training dataset with a set of GOOD sequences and their resulting content. The considered prompt is the following: *"<task>. Write your generated sequences together with the resulting content on separate lines in the following format: <generated_sequence>-<resulting_content>. Don't add any additional characters or numbering. Take into consideration the following example: <train_example>. Text: <context>. Response:"*.

4.2 Experimental Setup

All previous models above (our approach and the baselines) were fine-tuned from the foundational Llama-3 (8B) model (Dubey et al., 2024). All our experiments for fine-tuning and inference were done on a single A100 80GB GPU to ensure accessibility and cost-effectiveness. More details on the actual models, their training, and hyperparameters can be found in Appendix B.

We were interested in the most likely sequences for models with iterative generation (i.e., FA, ASG, ASG-RC). This is typically done using beam search, but this method is not well suited for the current task. Due to the high number of possible positive answers, beam search is too restrictive with the options for the first token. Because of this, we



(a) Answer Selection for Question Generation

(b) Keyword Generation

Figure 2: Prompt examples for different tasks.

opted for over-sampling by generating in each iteration 10 samples and choosing the one with the highest log-likelihood each time.

In the case of the SSG model, we opted for a similar over-sampling approach, sampling 100 sequences from which duplicates are removed, and the top 25 sequences are kept.

4.3 Evaluated Tasks

4.3.1 Answer Selection for Question Generation

For this task, the goal is to select several spans from a context to serve as the answer for generating a question. The aim is to generate a richer set of potential questions by identifying multiple answer spans from a given text. Our method improves the generated sequences through iterative feedback, making the model more adaptable.

Prerequisites: Labeling System. For these experiments, we employ an automated method to evaluate the quality of the selected answers. We measured the quality of a selected answer by generating a question starting from it and assessing how likely a model would answer the question given the selected span. To achieve this, we required two models: one to generate questions based on a given context and a piece of information identified as the answer ($QGen$), and another capable of providing accurate answers to questions based on a given context ($QAns$). By utilizing these models, we can label responses and proceed with the task of generating and evaluating questions based on proposed answers. Any instruction-tuned model capable of generating and answering questions can be used for these tasks. As the details of these models fall outside the scope of this experiment, further information is provided in Appendix A. To establish an automatic criterion for a $GOOD$ selected answer, we define it as a text span that can be used to generate a valid question. This question should be answerable by the $QAns$ model when provided with the given context. Furthermore, the context should be necessary for answering the question, avoiding excessively general inquiries. Additionally, the selected answers must exhibit diversity, avoiding redundancy in represented concepts.

Formally, the following procedure is employed to determine the suitability of a selected span:

1. A question is generated using the context and the selected answer: $q = QGen(ctx, a)$.

2. The probability of the $QAns$ model of returning the selected span for the given question must exceed an empirically chosen threshold of 3%: $P(QAns(ctx, q) = a) \geq 0.03$; this threshold was selected given the distribution of correct answers in our training dataset;

3. The context must be important for answering the question, so we impose that it should be easier for $QAns$ to answer the question when given the context compared to answering without it: $\frac{P(QAns("", q)=a)}{P(QAns(ctx, q)=a)} \leq 1$

Here, we denote $P(QAns(ctx, q) = a)$ as the probability of $QAns$ to generate the answer a, given the context ctx and question q. We use the sum of log-likelihood estimates for each token in the answer, which is then converted into a probability between 0 and 1 by exponentiation:

$$P(QAns(ctx, q) = a) = e^{\sum logP(a_i|ctx,q,a_{0:i-1})}$$

This formalization ensures that the selected answers are not only relevant to the context but also exhibit a strong dependency on the context for accurate question answering. Furthermore, by applying this evaluation, we can categorize the selected spans as either $GOOD$ or BAD based on their adherence to the established procedure. It is important to note that this metric can be altered for other purposes (e.g., question difficulty), and its form is not essential in our method.

Optional Component: Resulting Content. Evaluating a selected answer without knowledge of the resulting question might add an unnecessary abstraction layer to the task. Because of this, we include the question as additional information in the prompt. In this case, the Resulting Content Generator (RCG) will be the $QGen$ model that generates a question based on the selected answer. The question will be used in the pipeline as resulting content.

Dataset Construction. We consider two datasets for the specific task of Answer Selection for Question Generation. The *TASA* corpus (Ivens and Koslin, 1991) is a collection of text excerpts designed to represent the reading a college student might encounter throughout their academic career. It contains over 60k passages from textbooks, literature, and various nonfiction and fiction works. For our experiments, we sampled 10,000 texts for training, 1000 for validation, and 1000 for the test partition from the following domains: language and arts, health, science, industrial arts, economics, business, and social studies. *FairytaleQA* (Xu et al., 2022) is a specialized dataset focused on narrative comprehension for kindergarten to eighth-grade students. It addresses the scarcity of high-quality question-answering datasets devised for diverse reading skills. The dataset is constructed by educational experts from children-friendly stories. For our experiments, we used the already-established partitions of the dataset (8548 for train, 1025 for validation, 1007 for test).

For the task of Answer Selection for Question Generation, we identify possible text spans that could serve as an answer. These text spans for training are the nodes from the constituency tree of each sentence from the context. By leveraging these nodes, we systematically cover a wide range of possible answers. For each possible answer a_i, we generate a question given the context and a_i, as $q_i = QGen(ctx, a_i)$. With the method described above, we label a sample of possible spans as $GOOD$ and BAD.

4.3.2 Keyword Generation

Keyword generation involves automatically generating relevant terms that summarize the core topics of a paper.

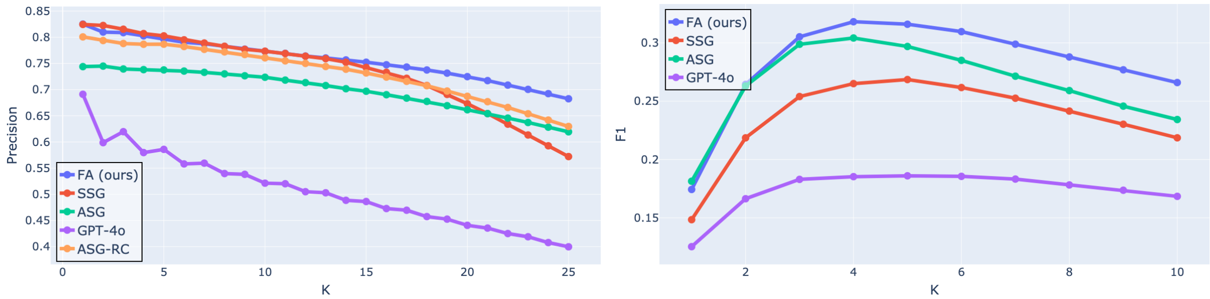
Prerequisites: Labeling System. For these experiments, we rely on the human-chosen keywords from the dataset to serve as $GOOD$ labels. Any other option not included in the set for the given context would be considered BAD. In this manner, we cover a different possibility for Feedback-Aware: the feedback signal is provided by humans rather than an automatic system.

Dataset Construction. We use the *KP20K* dataset introduced by Meng et al. (2017). This is an extensive dataset containing scientific article abstracts and their corresponding keywords, as chosen by the authors. It contains 500k entries, from which we randomly selected 2000 from their testing partition to evaluate the models, 2000 for validation, and 50k for training. As this dataset only has positive examples annotated (the keywords selected by the authors, labeled as $GOOD$), we created negative examples (BAD) by selecting keywords from other entries with high similarity to the abstract that are not in the subset of $GOOD$ keywords of that specific entry. More formally, having the dataset $D = [(abs_1, KL_1), (abs_2, KL_2), ..., (abs_n, KL_n)]$ where $KL_i = [k_{i,1}, ..., k_{i,m})]$ is the list of the $GOOD$ keywords for the abstract abs_i, we select the negative (BAD) keywords for abs_i as being from $(\bigcup_j KL_j) \setminus KL_i$. From this set, we select m negative keywords that are most similar to the abstract, based on embeddings computed with an encoder model[2]. This will be the set of BAD keyword examples for the abstract abs_i.

5 Results

We evaluate the models and baselines using slightly different metrics based on the task. For *Answer Selection for Question Generation*, we evaluated in terms of Precision@K for the $GOOD$ sequences. More formally, for each text, $P@K = \frac{\text{no. } GOOD \text{ in the first K}}{\text{K}}$. Figure 3a showcases the P@K aggregated results for the test partitions of both TASA (Ivens and Koslin, 1991) and FairytaleQA (Xu et al., 2022). We evaluated using precision

[2]https://huggingface.co/sentence-transformers/all-mpnet-base-v2



(a) Answer Selection for Question Generation

(b) Keyword Generation

Figure 3: Results for different tasks.

since the number of *GOOD* answers for a context can be large and recall would be less meaningful.

For *Keyword Generation*, we followed the approach stated in the literature and evaluated in terms of F1@K for the *GOOD* keywords. As there is a varied number of *GOOD* keywords per text but still within a limited range, we computed the F1@K until K=10, based on the observation that the 95th percentile of the number of *GOOD* keywords per text is 10 on KP20K (Meng et al., 2017). Figure 3b showcases the F1@K results.

6 Discussion

The results presented in Figures 3 argue that our proposed method consistently achieves high scores across multiple generated sequences. It outperforms all other models by a significant margin. The observed decline in metrics as K increases is primarily due to the diminishing pool of accessible answers for further generation.

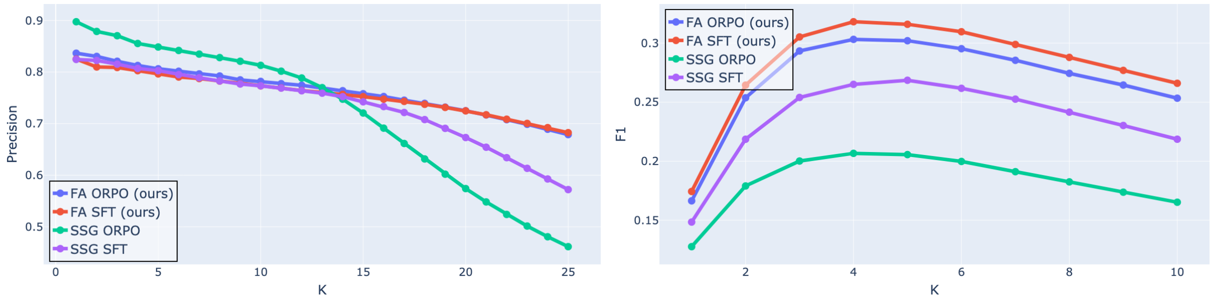
For the task of Answer Selection for Question Generation, the Single Sequence Generation (SSG) baseline initially exhibits high precision but rapidly declines as the number of generated sequences increases. This decline stems from the model's lack of awareness of previously selected samples, impairing its ability to generate additional high-quality answers, a limitation our approach effectively overcomes. The All Sequences Generation (ASG) baseline shows a more gradual decline in performance over multiple generated sequences, as its prompts incorporate previously selected answers, which aids in generating better outputs. However, despite this benefit, it starts from a lower precision point and experiences a slight decrease as the number of available high-quality options reduces. The strongest baseline is the All Sequences Generation with Resulting Content (ASG-RC). Incorporating the generated questions derived from previously selected answers helps the model better understand the task, resulting in higher precision. However, it still underperforms compared to our method and experiences a steeper decline in precision over time. This highlights the value of Feedback-Aware inference, as our model's superior performance can be attributed to leveraging feedback labels more effectively. Moreover, the results indicate that the GPT-4o model yields the poorest performance. Despite being a highly capable assistant, it underperforms relative to smaller, open-source, fine-tuned models for specific tasks, underscoring the importance of task-specific finetuning and open research.

For the Keyword Generation task, the performance of our proposed model remains the highest. In this case, the All Sequences Generation (ASG) has a good starting point, but its sequences decline in quality after a few iterations. Here, GPT-4o also performs poorly since choosing certain keywords requires finetuning on extensive datasets to learn to mimic human behavior and reasoning.

Recent studies that use the *KP20K* dataset focus on the extractive task and rely on encoder models. The best results in this case are around 34.5% F1@10 (Song et al., 2023), but are not directly comparable with our scores, since they discard keywords that do not appear in the abstract, which makes the task much simpler.

6.1 ORPO Alignment Experiments

Categorizing generated sequences as *GOOD* and *BAD* also leads us to consider an approach involving preference fine-tuning. Multiple alignment techniques (e.g., Ouyang et al., 2022, Rafailov et al., 2024, Hong et al., 2024) leverage positive and negative examples to train models to generate content close to the positive choice and diverge from the negative one. We experimented with the



(a) Answer Selection for Question Generation

(b) Keyword Generation

Figure 4: Results for different tasks (ORPO vs. SFT).

ORPO method (Hong et al., 2024) to fine-tune our Feedback-Aware model and the Single Sequence Generation baseline on positive ($GOOD$ answers) and negative (BAD answers) sequences. ORPO incorporates an odds ratio-based penalty for differentiating between chosen and rejected responses in the conventional loss computation. We chose ORPO since it outperformed the SFT+DPO setup (Hong et al., 2024), fine-tunes only on positive/negative examples, and does not require previous training. More formally, for a prompt in the form of [Prompt, $GOOD$ sequence], we diverge from the negative [Prompt, BAD sequence].

The ORPO alignment is done with the same setup as supervised fine-tuning (SFT), with an additional specific hyperparameter, $\beta = 0.3$ that defines the weight given to the odds ratio-based penalty in regards to the classical negative log-likelihood loss.

Figure 4 highlights the results of this experiment. While ORPO yields similar results with supervised fine-tuning in our case, a different case is made for the Single Sequence Generation baseline. In the case of Answer Selection for Question Generation, ORPO helps at first while starting from a high point in precision; however, the decrease is abrupt as more sequences are generated, mainly because ORPO tends to polarize the pool of samples, given its positive/negative alignment. This hinders the capability of the model to generate diverse answers. Our method is not affected by this polarization since it uses feedback signals that help the generation to be grounded on previous facts. For the Keyword Generation task, this behavior begins from the first sequence, ORPO having a poor performance from the start. One explanation is that ORPO specifically penalizes keywords that are not included in the initial list, but not all of them are necessarily unsuitable.

7 Conclusions and Future Work

In this work, we introduced a Feedback-Aware generation model that consistently outperforms existing baselines and proprietary models in iteratively generating high-quality sequences for different tasks. The results argue that our approach maintains high scores across multiple iterations, significantly surpassing baselines that lack feedback awareness. Our approach's superior performance highlights the effectiveness of leveraging feedback signals during training and inference. Moreover, our framing of the training and inference steps has the potential to be adapted to other tasks and feedback signals that can either come from proxy models or even human preference.

The framework introduced in this work can be adapted beyond sequence generation to tasks that require structured reasoning. A future work extension can be on tasks like mathematical reasoning, where generating coherent reasoning chains is essential. Instead of producing multiple independent sequences, the proposed approach can be modified to generate structured reasoning steps, ensuring logical consistency throughout the inference process. A key adaptation involves incorporating incorrect reasoning chains into the prompt to improve the generation of correct ones. By explicitly conditioning the model on incorrect solutions, it may better learn to differentiate between valid and invalid reasoning paths, improving performance in tasks requiring step-by-step logical deductions. Future work should investigate the impact of feedback-aware generation in reasoning tasks, including how different types of feedback—whether model-generated or human-annotated—affect performance.

Limitations

While our study presents significant findings, it is important to acknowledge certain limitations. One such limitation lies in the extent of our hyperparameter tuning. Due to the computational expense associated with exhaustive hyperparameter searches, we opted for a less intensive approach. This decision was made to align with this paper's scope and ensure a manageable workload. It is worth noting that we maintained consistent hyperparameters across baseline models and our approach. This standardization helps to ensure a fair comparison.

Another limitation of our proposed method is that it requires negative samples for training. In cases where datasets do not provide such samples, they must be generated. Generating informative negative samples is non-trivial, as they should align with the task objectives and carry meaningful contrastive information. Basic or poorly constructed negative samples may fail to contribute to effective model learning.

Ethics Statement

In this research, we prioritize transparency, reproducibility, and sustainability. Our approach leverages publicly available, open-source datasets and models, ensuring our work is grounded in widely accessible resources. We aim to promote collaboration and innovation within the research community by using these open-source tools. We release all project elements, including the code, fine-tuned models, and labeled datasets. Moreover, we have carefully managed our computational resources by maintaining a low GPU budget, which not only makes our experiments more accessible and reproducible but also minimizes the environmental impact associated with high-energy computation.

References

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Haoran Ding and Xiao Luo. 2021. Attentionrank: Unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1928.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yao Fu, Hao Peng, Tushar Khot, and Mirella Lapata. 2023. Improving language model negotiation with self-play and in-context learning from ai feedback. *arXiv preprint arXiv:2305.10142*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *Preprint*, arXiv:2403.07691.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Stephen H Ivens and Bertram L Koslin. 1991. *Demands for reading literacy require new accountability methods*. Touchstone Applied Science Associates.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592.

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.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2022. True few-shot learning with Prompts—A real-world perspective. *Transactions of the Association for Computational Linguistics*, 10:716–731.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Mingyang Song, Yi Feng, and Liping Jing. 2023. A survey on recent advances in keyphrase extraction from pre-trained language models. *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2153–2164.

Mingyang Song, Liping Jing, and Lin Xiao. 2021. Importance estimation from multiple perspectives for keyphrase extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2726–2736.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wenda Xu, Daniel Deutsch, Mara Finkelstein, Juraj Juraska, Biao Zhang, Zhongtao Liu, William Yang Wang, Lei Li, and Markus Freitag. 2024. LLMRefine: Pinpointing and refining large language models via fine-grained actionable feedback. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1429–1445, Mexico City, Mexico. Association for Computational Linguistics.

Ying Xu, Dakuo Wang, Mo Yu, Daniel Ritchie, Bingsheng Yao, Tongshuang Wu, Zheng Zhang, Toby Li, Nora Bradford, Branda Sun, Tran Hoang, Yisi Sang, Yufang Hou, Xiaojuan Ma, Diyi Yang, Nanyun Peng, Zhou Yu, and Mark Warschauer. 2022. Fantastic questions and where to find them: FairytaleQA – an authentic dataset for narrative comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 447–460, Dublin, Ireland. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Bingsheng Yao, Dakuo Wang, Tongshuang Wu, Zheng Zhang, Toby Li, Mo Yu, and Ying Xu. 2022. It is ai's turn to ask humans a question: Question-answer pair generation for children's story books. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–744.

Hokeun Yoon and JinYeong Bak. 2023. Diversity enhanced narrative question generation for storybooks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 465–482.

Zhenjie Zhao, Yufang Hou, Dakuo Wang, Mo Yu, Chengzhong Liu, and Xiaojuan Ma. 2022. Educational question generation of children storybooks via question type distribution learning and event-centric summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5073–5085.

A Appendix: *QAns* and *QGen* models

As stated before, prerequisite models for the labeling system and content generation are required to evaluate and train both our approach and baselines. In our case, $QGen$ is a language model fine-tuned for question generation and $QAns$ is fine-tuned for question answering.

We fine-tune these models on three established datasets: a) SQuAD (Rajpurkar et al., 2016) - one of the most widely used resources for question answering and generation, SQuAD consists of over 100K question-answer pairs derived from a pool of 5K Wikipedia articles; b) HotpotQA (Yang et al., 2018) - designed to test a model's ability to answer questions that require reasoning across multiple paragraphs, HotpotQA contains questions that should be answered by bridging information from two different Wikipedia articles; and c) NarrativeQA (Kočiský et al., 2018) - designed to assess reading comprehension, particularly for lengthy texts, NarrativeQA consists of stories, along with corresponding questions and answers.

The models were independently fine-tuned in a supervised manner using the prompt *"Generate a question based on the context and the answer. Context: <context>. Answer: <answer>. Question: <generated_question>"* for question genera-

	QGen	QAns
SQuAD	0.58	0.75
HotpotQA	0.50	0.50
NarrativeQA	0.54	0.60
FairytaleQA	0.51	0.45

Table 1: BLEURT score (Sellam et al., 2020) for the prerequisites models on their tasks

tion ($QGen$) and *"Answer the following question based on the context. Context: <context>. Question: <question>. Answer: <answer>"* for question answering ($QAns$). The loss is computed only on the *<generated_question>* and *<answer>* tokens, respectively.

These models are not proposed as state-of-the-art for these tasks but rather as plug-and-play modules independent of our proposed method or baselines. This means that any model can be a prerequisite as the proposed method is not dependent on the choice.

In order to assess the performance of these models and validate their usage, we computed the BLEURT score (Sellam et al., 2020) with the ground-truth for the test partition of the SQuAD, HotpotQA, and NarrativeQA, and for the test partition of FairytaleQA (Xu et al., 2022) (a dataset used for our method and baselines, but on which we did not train $QGen$ and $QAns$).

Table 1 showcases the BLEURT scores for the prerequisites models. Performance is acceptable as the models are capable of answering and generating questions with accuracy, considering the context. Moreover, the models generalize well on an unseen dataset (FairytaleQA), highlighting their capability to serve as suitable prerequisites for our tasks with a diverse range of texts from different domains.

B Appendix: Hyperparameter Details

For training, all models (the prerequisites $QGen$ and $QAns$, our model and the baselines, including the ORPO variants) are fine-tuned from the foundational Llama-3 (8B) model (Dubey et al., 2024). The setup for *training* considered: LoRA (Hu et al., 2022) with projection matrices for the attention layers; final batch size of 64 (resulted from gradient accumulation); half-precision (FP16) training; learning rate of 1e-5, AdamW-8bit optimizer.

The configuration for *inference* was: nucleus decoding with over-sampling and selecting the top generations (top_k=20, top_p=0.8, seed=42) for our approach and the baseline models; default settings and seed=42 for GPT-4o; mixed-precision (bf16, seed=42) computations for prerequisites models ($QGen$ and $QAns$).

C Appendix: Prompts

This section contains the prompts used for different models and tasks. In **bold** we denoted the expected generated text by the model.

C.1 Answer Selection for Question Generation

Feedback-Aware Model

Iteratively select a span from the following text that would serve as a good answer for generating a question.
Text: {{Text}}
Response:
GOOD: {{Previous selected answer}} - {{Previous resulting question}}
BAD: {{Previous selected answer}} - {{Previous resulting question}}
...
GOOD: {{Previous selected answer}} - {{Previous resulting question}}
GOOD: **{{Selected answer}}**

Single Sequence Generation

Select a span from the following text that would serve as a good answer for generating a question.
Text: {{Text}}
Response:
{{Selected answer}}

All Sequences Generation

Iteratively select a span from the following text that would serve as a good answer for generating a question.
Text: {{Text}}
Response:
{{Selected answer}}
{{Selected answer}}
...
{{Selected answer}}

All Sequences Generation with Resulting Content

Iteratively select a span from the following text that would serve as a good answer for generating a question.
Text: {{Text}}
Response:
{{Previous selected answer}} - {{Previous resulting question}}
{{Previous selected answer}} - {{Previous resulting question}}
...
{{Previous selected answer}} - {{Previous resulting question}}
{{Selected answer}}

GPT-4o

Select 25 spans from the following text that would serve as good answers for generating questions. Write your selected answers together with the corresponding question on separate lines, in the following format: <answer> -> <question>
Don't add any additional characters or numbering. Take into consideration the following example:
{{Example text and response}}
Text: {{Text}}
{{Selected answer}} - {{Generated question}}
{{Selected answer}} - {{Generated question}}
...
{{Selected answer}} - {{Generated question}}

C.2 Keyword Generation

Feedback-Aware Model

Iteratively select keywords for the following text.
Text: {{Text}}
Response:
GOOD: {{Previous selected keyword}}
BAD: {{Previous selected keyword}}
...
GOOD: {{Previous selected keyword}}
GOOD: **{{Selected keyword}}**

Single Sequence Generation

Select keywords for the following text.
Text: {{Text}}
Response:
{{Selected keyword}}

All Sequences Generation

Select keywords for the following text.
Text: {{Text}}
Response:
{{Selected keyword}}
{{Selected keyword}}
...
{{Selected keyword}}

GPT-4o

Generate 15 keywords for the following abstract. Write your selected answers on separate lines. Don't add any additional characters or numbering. Take into consideration the following example:
{{Example text and response}}
Text: {{Text}}
{{Selected keyword}}
{{Selected keyword}}
...
{{Selected keyword}}