System-Level Natural Language Feedback

Weizhe Yuan
New York University
wy885@nyu.edu

Kyunghyun Cho
New York University
Prescient Design, Genentech

Jason Weston
New York University

Abstract

Natural language (NL) feedback offers rich insights into user experience. While existing studies focus on an instance-level approach, where feedback is used to refine specific examples, we introduce a framework for system-level use of NL feedback. We show how to use feedback to formalize system-level design decisions in a human-in-the-loop-process – in order to produce better models. In particular this is done through: (i) metric design for tasks; and (ii) language model prompt design for refining model responses. We conduct two case studies of this approach for improving search query and dialog response generation, demonstrating the effectiveness of system-level feedback. We show the combination of system-level and instance-level feedback brings further gains, and that human written instance-level feedback results in more grounded refinements than GPT-3.5 written ones, underlying the importance of human feedback for building systems. We release our code and data at https://github.com/yyy-Apple/Sys-NL-Feedback.

1 Introduction

Users interacting with a machine learning system offer feedback, either actively or passively. The feedback can be binary ratings (Arora et al., 2022), preference feedback (Stiennon et al., 2020) and natural language (NL) feedback (Hancock et al., 2019; Scheurer et al., 2022a). Among them, NL feedback is the most general due to its free-form nature, as opposed to the limited choices in other feedback forms. Hence, it is crucial to harness the potential of NL feedback to improve a system.

Existing research on NL feedback typically adopts one of two strategies. The first uses feedback as an auxiliary target in addition to the original task, just like in multitask learning (Hancock et al., 2019; Xu et al., 2022b). The second modifies the original output based on per-instance feedback. The system can either be fine-tuned with the new output (Tandon et al., 2022; Scheurer et al., 2022b) or iteratively self-critique and self-refine at inference time (Madaan et al., 2023; Chen et al., 2023b). One common limitation of these studies is that they only focus on instance-level learning, where each feedback only serves the instance for which it was received. Furthermore, they often assume the availability of feedback for each and every example, which is not practical in real-world scenarios, where feedback is often sparse.

This paper asks the following question: *Can we aggregate instance-level NL feedback to make system-level design decisions that improve language generation systems?* We answer this question by proposing a general framework for aggregating instance-level NL feedback. A set of criteria (i.e., system-level feedback) are first derived from instance-level feedback through a human-in-the-loop process involving clustering and summarization. Those criteria then guide the design of instruction-following language model prompts to refine (i.e., correct) examples, and the development of metrics that align with users' needs. We conduct two case studies of the proposed framework on information-seeking dialog tasks where we improve both the query generator and the response generator of an Internet-augmented dialog system. The experimental results point to the effectiveness of system-level feedback. Our contributions are:

- We propose a new method that derives system-level feedback from instance-level feedback, which can guide text generation refinement.
- We show how human experts can use system-level feedback to design metrics for evaluating information-seeking dialog systems.
- We demonstrate that combining system-level and instance-level feedback for prompt design yields more helpful refinements for system training w.r.t. the designed metrics above.

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2773–2789

- We show the importance of human NL feedback by comparing it to GPT-3.5-generated feedback in response refinement. We find that human feedback leads to more grounded refinements that can better guide system learning.

2 Related Work

Dialog Systems The rapid development of large language models (LLMs) (Brown et al., 2020; Zhang et al., 2022) has advanced dialog systems, incorporating techniques like multi-session memory (Xu et al., 2022a), search engine support (Komeili et al., 2022), etc. Recently, ChatGPT's rise has captivated both the NLP community and the public at large. Nowadays, intelligent dialog agents have become an essential part of people's productivity, such as brainstorming (Zhang et al., 2023b), essay polishing (Buruk, 2023), code writing (Haensch et al., 2023), etc. However, LLMs also carry potential risks including misinformation (Chern et al., 2023), sycophancy (Sharma et al., 2023), etc., which calls for more thorough evaluations.

Learning from Human Feedback As language models increasingly integrate into people's daily life, aligning them with human needs becomes essential (Askell et al., 2021). As a result, researchers have been working on utilizing various human feedback, including preference feedback (Stiennon et al., 2020; Ouyang et al., 2022), binary feedback (Li et al., 2019; Arora et al., 2022; Adolphs et al., 2022), NL feedback (Weston, 2016; Li et al., 2017; Hancock et al., 2019; Saunders et al., 2022; Scheurer et al., 2022a), and so on. So far, the use of NL feedback is relatively less explored, with most studies focusing on instance-level feedback where each instance receives its own feedback (Scheurer et al., 2022a, 2023). In this work, we propose a general framework for deriving system-level feedback from instance-level feedback, and show the effectiveness of system-level feedback alone and its complementarity with instance-level feedback.

3 Methodology

3.1 Problem Formulation

Assume we have (1) a text generator $P_\theta(r|q)$ that generates a response r to a query q, (2) a text refiner $P_\phi(r'|r, q, c)$ that generates a refinement r' given the original response r, the query q, and criteria c that explains what makes a good response, (3) a quality checker $Q(q, r)$ that decides whether r is a satisfactory response given q. When deploying $P_\theta(r|q)$, for some unsatisfied responses $\mathcal{R}_n = \{r_1, \cdots, r_n\}$, we collect NL feedback for each of them $\mathcal{F}_n = \{f_1, \cdots, f_n\}$. We aim to use $\mathcal{F}_n$ to improve $P_\theta(r|q)$ by updating its parameters θ. In our setting, we take the text refiner and quality checker as given. They can either be based on large models like GPT-3 (Scheurer et al., 2022a) or specialized fine-tuned models (Shi et al., 2022).

3.2 Proposed Framework

Our proposed framework is shown in Figure 1. There are four steps within this framework.

Derive criteria from feedback When deploying the text generator $P_\theta(r|q)$, we collect feedback $\mathcal{F}_n$ for some responses $\mathcal{R}_n$. A clustering algorithm is then run (e.g., k-means clustering (Hartigan and Wong, 1979)) to identify common issues that can be potentially rectified. Next, a human-in-the-loop approach is used, where human experts derive a set of criteria c for what constitutes a good response from those clusters. These criteria, articulated in natural language, serve as part of the input (prompt) for the text refiner. This process relates to prompt engineering in large language models (Liu et al., 2023), where the NL feedback is used to help formalize the prompt engineering process. With these criteria, experts also design metrics $m_1(\cdot), \cdots, m_k(\cdot)$ to evaluate aspects of user interest.

Construct refinement training data To improve the text generator, we create a training dataset, $\mathcal{D}$, that reinforces positive behaviors and rectifies negative ones. If a sample (q_i, r_i) meets $Q(q_i, r_i) = 1$, it is added to $\mathcal{D}$ to reinforce good model behavior. Otherwise, the text refiner $P_\phi(r'|r, q, c)$ refines r_i to r_i' using prompts based on criteria c. If this refined sample (q_i, r_i') passes $Q(q_i, r_i') = 1$, it is added to $\mathcal{D}$ to modify bad behavior.

Fine-tune the model After collecting supervised data $\mathcal{D}$, we fine-tune the text generator $P_\theta(r|q)$. This data can be combined with existing data that was used to build the baseline deployed system (that did not use feedback).

Evaluate using designed metrics Finally, we use our designed metrics to assess system performance against user requirements. If successful, the updated system will exhibit improved metrics $m_1(\cdot)$, $\cdots$, $m_k(\cdot)$ compared to the baseline system.

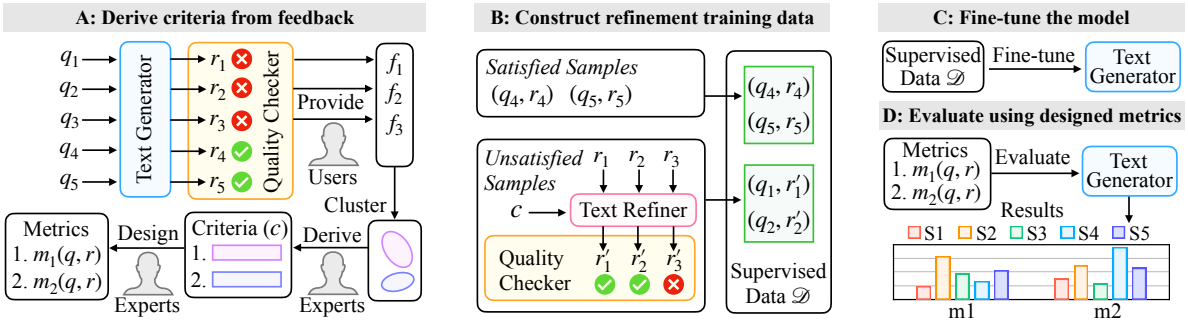

Figure 1: Our framework for incorporating NL feedback into system-level model design. Using a human-in-the-loop approach, criteria derived from NL feedback guide the creation of prompts for refining responses and metric design to evaluate the improvements. Notation: q: query, r: response, f: feedback, r': refinement, $m(\cdot)$: metric function. $S1 \cdots S5$ represent different systems one can compare using this framework.

4 Experimental Setup: Dialog Systems

We study our framework within dialogue system deployment, a context where users naturally offer NL feedback, such as "that's not correct" for incorrect responses (Shi et al., 2022). Our case studies focus on information-seeking dialogues, where users interact with dialog agents to obtain answers or relevant information (Glaese et al., 2022).

Dialog System Selection We choose the Blenderbot2 (BB2) dialog system (Komeili et al., 2022; Xu et al., 2022a) comprised of two modules: (1) Query Generator (QG) that generates an Internet search query from dialogue history. (2) Response Generator (RG) that generates a response using dialogue history and retrieved web documents.[1] We select BB2 because it allows us to study two scenarios: *query generation* and *response generation*.

Deployment Data We use the FITS dataset (Xu et al., 2022b) for experiments, which collects diverse feedback from user interactions with Internet-augmented dialogue systems like BB2 and SeeKeR (Shuster et al., 2022). Though the dataset includes binary, NL feedback, and gold corrections, we only use binary and NL feedback, given users are less inclined to provide gold corrections for mistakes.

Text Refiner Given no gold corrections, we turn to model-based refinement techniques. In this work, we use GPT-3.5[2] as the text refiner and apply greedy decoding during inference.

Quality Checker We train quality checkers for queries and final responses using collected binary feedback. Our classifier is based on FLAN-T5[3] (Chung et al., 2022) trained on 20% training data, using binary feedback following Shi et al. (2022). We select a threshold to ensure 80% precision for labels it predicts as positive on the validation set.

5 Case Study 1: Query Generation

5.1 Derive Criteria from Feedback

We collect all NL feedback from the FITS training split to understand human preferences and derive criteria. We first use SimCSE encoder[4] (Gao et al., 2021) to encode each feedback. Then, we use k-means clustering to group feedback related to query generation into five clusters. From inspecting these (see Appendix A.1 for detailed manual efforts), we summarize them into four groups (see Table 1) and derive that a successful search query should (i) rephrase the user's question while keeping important keywords, (ii) be relevant and specific, (iii) use common words for better search coverage, (iv) be concise. The criteria text for crafting the prompt c for the text refiner $P_\phi(r'|r,q,c)$ is in Table 2.

5.1.1 Criteria-guided Metric Design

Using feedback-derived criteria, we design metrics to mirror users' preferences.[5] Ideally, an effective query should score high across all these metrics.

Non-copy rate measures how much a search query rephrases the user's utterance by examining n-gram matching. We define it in Equation 1 based on BLEU-4 (Papineni et al., 2002) where s is the

[1]We use Google search (`https://www.google.com/`) to retrieve the top five relevant documents given a search query.

[2]We use the model `gpt-3.5-turbo` for our experiments.

[3]We use the `flan-t5-large` model.

[4]We use the `sup-simcse-roberta-large` model.

[5]When evaluating a set of queries, for a metric defined as a fraction with a constant numerator, we take the average of the denominators of all queries on that metric and take its reciprocal to multiply the numerator.

Group	Feedback type	Num.	%
1	User suggests a search query for Internet search directly.	2715	52.87%
2	Suggests specific edits, such as shortening the query or using common words, and so on.	996	19.40%
3	Points out that the search query should use keywords instead of copying the original question and should be specific.	995	19.38%
4	Points out that the search query is not relevant to the problem.	429	8.35%

Table 1: Case study 1 (query generation): 4 groups of system-level feedback derived from automatic clustering.

Type	Criteria (Abbreviated)	NCR	Spec.	Read.	Con.	Cov.	Sat.
(1): Baseline	None	4.06	79.40	19.46	14.87	29.80	61.50
(2): (1)+Rephrase	Rephrase the user's question and keep keywords.	4.98	83.20	19.54	15.04	26.50	62.10
(3): (2)+Specificity	Above + Be accurate and specific for user needs.	5.00	**84.20**	18.77	14.50	28.80	**63.30**
(4): (3)+Readability	Above + Use simple and common words for better results.	**5.08**	80.80	19.53	15.97	29.40	62.40
(5): (4)+Conciseness	Above + Be concise; focus on user's first question.	4.81	80.00	**19.70**	**16.63**	**35.30**	62.70

Table 2: Case study 1 (query generation): refinement quality via designed metrics when using different criteria to prompt GPT-3.5 for query refinement. Metrics measured: **NCR**: non-copy rate, **Spec.**: specificity, **Read.**: readability, **Con.**: conciseness, **Cov.**: coverage. **Sat.**: satisfaction. The full criteria texts can be found in the Appendix A.2.

search query and u is the user question.

$$\text{Non-copy Rate} = \frac{1}{\text{BLEU-4}(s, u)} \quad (1)$$

Specificity measures whether the search query sufficiently captures the necessary information to retrieve relevant documents. We use GPT-3.5 as the evaluator (Fu et al., 2023). Details are in the Appendix A.3.

Readability measures a search query's clarity based on the word frequency rank (WFR)[6] of its terms, as defined in Equation 2, where w is a word in s and C is a scaling constant. Ideally, a query should use common words to improve readability.

$$\text{Readability} = \frac{C}{\text{AVG}_{w \in s}(\text{WFR}(w))} \quad (2)$$

Conciseness measures the query's brevity by its word count, with its value being the query length's reciprocal, scaled by a constant 100.

Coverage measures how specific vs. general a search query is by counting the number of Google search result pages. Considering the wide variation in page count, we employ a relative metric. For refined queries obtained using Table 2 with the same dialog context, the query with the most results gets a "Coverage" score of 1, and others receive 0.

Satisfaction measures whether the search query will satisfy the user. It is an overall metric, and we use our trained satisfaction classifier to determine the percentage of satisfied refinements.

[6]We use the Kaggle dataset for WFR: https://www.kaggle.com/rtatman/english-word-frequency

5.2 Construct refinement training data

We sample 1,000 satisfied queries from the FITS training set along with their contexts to add to our supervised training data $\mathcal{D}$. Then, based on Figure 1-(B), for each unsatisfied query r, we (1) use GPT-3.5 and criteria c derived from §5.1 to get a refinement r'. (2) Use a quality checker to check r''s satisfaction. (3) Add (q, r') to $\mathcal{D}$ if r' is satisfactory. We elaborate on step (1) in the next section.

5.2.1 Refinement Generation

We use GPT-3.5 with criteria-based prompts to refine 1,000 randomly sampled unsatisfied queries (details in Appendix A.2). To demonstrate the effectiveness of Figure 1-(A), we conduct ablation studies with different criteria for query refinement. Given our computational budget, for metrics relying on GPT-3.5, we sample 500 dialog contexts and compare the queries resulting from different criteria.

The results are in Table 2. Adding criteria in the prompt will shift GPT-3.5's generation, and the performance differences are interpretable using our designed metrics. Specifically, (i) The rephrase criterion increases the non-copy rate. (ii) The relevance criterion increases the relevance metric. (iii) The readability criterion increases the readability and coverage metrics. (iv) Using all the criteria, the refinements achieve reasonably good performance

	Valid						Test						Test Unseen					
	NCR	Spec.	Read.	Con.	Cov.	Sat.	NCR	Spec.	Read.	Con.	Cov.	Sat.	NCR	Spec.	Read.	Con.	Cov.	Sat.
BB2(QG)	**32.8**	40.5	**22.4**	**32.3**	**50.6**	4.8	**18.8**	34.9	14.0	**34.3**	**50.9**	8.8	**22.7**	37.7	15.4	**32.9**	**50.3**	3.2
SLT(QG(👍))	2.6	60.4	19.8	21.0	30.1	9.2	2.8	58.0	17.4	22.9	30.5	12.9	3.0	55.4	18.3	22.9	31.7	7.4
SLT(QG(👍+👎))	4.8	**73.5**	22.0	18.3	19.3	**29.6**	3.8	**74.5**	**21.7**	18.0	18.6	**29.0**	3.6	**73.5**	**19.4**	17.8	18.0	**17.2**

Table 3: Evaluate query generators on FITS using designed metrics. See Table 2 caption for abbreviation meanings.

	Valid		Test		Test Unseen	
	F1	PPL	F1	PPL	F1	PPL
BB2(QG)	9.74	16.09	14.28	9.61	16.09	10.15
SLT(QG(👍))	48.63	12.83	50.51	7.64	51.75	7.84
SLT(QG(👍+👎))	**51.19**	**10.34**	**52.99**	**7.23**	**52.21**	**7.73**

Table 4: Evaluate query generators on FITS using F1 and perplexity (PPL).

in all our designed perspectives and overall satisfaction. Thus, when collecting training data, we use the four criteria augmented prompt for refinement.

5.3 Fine-tuning the Model

We start from the 400M BB2 query generator and consider two fine-tuning settings: (1) using the satisfied data; and (2) using satisfied and refinement data. During training, we use the Adam optimizer (Kingma and Ba, 2015) with a batch size of 8 and learning rate of 7×10^{-6} for three epochs. The best checkpoint is chosen based on validation loss.

5.4 Evaluation using designed metrics

We evaluate the following query generators.

- **BB2(QG)** The original BB2 query generator.
- **SLT(QG(👍))** System-level trained query generator using only satisfied data.
- **SLT(QG(👍+👎))** System-level trained query generator using satisfied and refinement data.

Results on Standard Metrics Table 4 presents the results using standard metrics, as per Shi et al. (2022). Compared to the original BB2 query generator, training with domain-specific data (2nd row) significantly improves F1 word overlap and perplexity metrics. Adding refinement data (3rd row) further enhances these metrics.

Results on Our Designed Metrics We also report results on our designed metrics for different query generators in Table 3. It is clear that training on satisfied data produces more specific and satisfactory queries, with further improvements when incorporating refinement data. The original BB2 query generator often generates overly concise queries, hindering the retrieval of the most relevant documents. In other words, although it generates queries that perform well in terms of readability or coverage, it is still an inadequate query generator, as evidenced by the poor satisfaction of the queries it generates. Later, when we refer to "our trained query generator", we mean the one trained using both satisfied data and refinement data.

6 Case Study 2: Response Generation

6.1 Derive criteria from feedback

Following the approach in §5.1, we group all feedback related to response generation into ten clusters. Then, we summarize the following eight groups (see Table 5) of feedback types by merging some clusters. From Table 5, we derive that an improved response as indicated by users should (i) ground its answer on relevant search results, (ii) be concise and targeted, (iii) be confident in its answer. The criteria text for crafting the prompts c for the text refiner $P_\phi(r'|r, q, c)$ is given in Table 6.

6.1.1 Criteria-guided Metric Design

After deriving criteria for response generation from feedback, we design the following metrics to measure the quality of a response as indicated by users.[7]

Groundedness measures how much the response utilizes the search results by examining n-gram matching. We define it in Equation 3 based on ROUGE-2 (Lin, 2004). Here, r is the response, d is a document from the relevant search set $\mathcal{S}$.

$$\text{Groundedness} = \max_{d \in \mathcal{S}} \text{ROUGE-2}(r, d) \quad (3)$$

Factuality checks whether the information in the response is backed by search documents. We use

[7]When evaluating a set of responses using one of the following metrics, we take the average of all responses' scores on that metric.

Group	Feedback type	Num.	%
1	Clarify his/her demand again.	3702	26.54%
2	Complain that the bot (1) does not answer the question or (2) gives irrelevant information or (3) asks the user to find out the answer on his or her own.	2260	16.20%
3	Point out specific search results that can answer the question.	2255	16.17%
4	Suggest that the bot should use the search results.	2130	15.27%
5	States that the answer is (1) factually incorrect, or (2) not grounded on the search results.	1572	11.27%
6	Point out that the bot's answer is not specific/accurate/complete/detailed.	1309	9.39 %
7	Point out that the bot is not confident in its answers and always begins its responses with "I am not sure" or "I don't know".	582	4.17%
8	Complain about repetition/rudeness in bot responses.	137	0.99%

Table 5: Case study 2 (response generation): 8 groups of system-level feedback derived from automatic clustering.

Type	Criteria (Abbreviated)	GRD	Fact.	Help.	Rel.	Conf.	Sat.
(1): Baseline	Use a conversational tone; no more than 20 words.	34.68	86.60	81.40	89.40	99.60	74.10
(2): (1)+Groundedness	Above + Use search results to give answers.	36.81	86.60	85.00	89.00	**99.90**	75.80
(3): (2)+Relevance	Above + Be concise and targeted, no irrelevant information.	36.77	**88.80**	85.60	89.40	**99.90**	74.90
(4): (3)+Confidence	Above + Don't start with "I'm not sure" or "I don't know".	**39.02**	87.20	**86.60**	**90.60**	**99.90**	**77.00**

Table 6: Case study 2 (response generation): refinement quality via designed metrics when using different criteria to prompt GPT-3.5 for response refinement. Metrics measured: **GRD**: groundedness, **Fact.**: factuality, **Help.**: helpfulness, **Rel.**: relevance, **Conf.**: confidence. **Sat.**: satisfaction. The full criteria texts can be found in the Appendix A.2.

GPT-3.5 with chain-of-thought to measure factuality (Luo et al., 2023). See Appendix A.3 for details.

Helpfulness measures whether the response directly answers the user's question. We use GPT-3.5 to measure helpfulness. See Appendix A.3 for details.

Relevance measures whether the response remains on topic and offers pertinent information. We again use GPT-3.5, with further details in the Appendix A.3.

Confidence measures whether the response is in a certain and confident tone. We use simple heuristics to gauge confidence, counting the occurrences of "I'm not sure" and "I don't know." If either phrase appears, we consider the response unconfident; otherwise, it's considered confident.

Satisfaction measures whether the response satisfies the user, similar to "satisfaction" in §5.1.1.

6.2 Construct refinement training data

As in §5.2, we first randomly sample 1,000 satisfied responses together with their contexts to add to our training data $\mathcal{D}$. Then, we go through the following three steps: (1) refinement generation, (2) quality check and (3) collection of filtered data. We will describe (1) in detail in the following section.

6.2.1 Refinement Generation

We use GPT-3.5 with criteria-based prompts to refine 1,000 sampled unsatisfied responses (details in Appendix A.2). As in §5.2.1, we conduct ablation studies to demonstrate the effectiveness of derived criteria. The results in Table 6 highlight: (i) Adding the groundedness criterion improves the groundedness metric. (ii) Adding the relevance criterion increases helpfulness and relevance. (iii) GPT-3.5 refinements are confident and rarely include phrases like "I'm not sure" or "I don't know". (iv) In terms of satisfaction, the best performance is achieved by the prompt with all criteria added. Therefore, when collecting training data, we use the three criteria-augmented prompt for response refinement.

6.3 Fine-tuning the Model

We use the 400M BB2 main model as the baseline response generator and consider two fine-tuning settings: (1) using only satisfied data; and (2) using both satisfied and refinement data, following §5.3.

	Valid						Test						Test Unseen					
	GRD	Fact.	Help.	Rel.	Conf.	Sat.	GRD	Fact.	Help.	Rel.	Conf.	Sat.	GRD	Fact.	Help.	Rel.	Conf.	Sat.
BB2(QG+RG)	34.1	50.0	19.0	68.2	66.8	27.1	32.4	58.3	22.0	67.8	73.7	34.9	32.9	58.4	21.8	69.0	65.7	32.1
SLT(QG)+BB2(RG)	39.0	66.4	26.8	74.2	80.6	33.3	35.2	58.4	29.8	71.4	83.4	40.9	37.5	59.1	30.2	73.8	**77.5**	37.8
SLT(QG+RG(👍))	30.6	59.1	29.2	75.6	76.4	35.3	27.8	53.7	31.5	69.6	80.6	41.7	29.7	60.5	31.3	73.4	72.6	39.3
SLT(QG+RG(👍+👎))	**48.2**	**69.1**	**41.3**	**81.6**	**81.1**	**50.7**	**43.2**	**66.7**	**44.5**	**76.4**	**83.6**	**55.7**	**45.3**	**71.6**	**43.9**	**79.6**	76.3	**51.4**

Table 7: Evaluate dialog systems on FITS using designed metrics. See Table 6 caption for abbreviation meanings.

	Valid		Test		Test Unseen	
	F1	PPL	F1	PPL	F1	PPL
BB2(QG+RG)	25.78	9.40	28.30	7.41	22.99	7.75
SLT(QG)+BB2(RG)	26.69	8.24	28.66	6.66	24.88	7.03
SLT(QG+RG(👍))	**28.20**	**7.41**	**29.73**	**6.04**	**25.54**	**6.43**
SLT(QG+RG(👍+👎))	25.57	7.62	26.90	6.15	24.34	6.58

Table 8: Evaluate dialog systems on FITS via F1 & PPL.

6.4 Evaluation using designed metrics

We evaluate the following systems:

- **BB2(QG+RG)** Original BB2 response generator paired with the original BB2 query generator.
- **SLT(QG)+BB2(RG)** Original BB2 response model paired with our system level trained query generator.
- **SLT(QG+RG(👍))** Our system-level trained response generator using satisfied data only, paired with our system level trained query generator.
- **SLT(QG+RG(👍+👎))** Our system-level trained response generator using satisfied and refinement data, paired with our system level trained query generator.

Results on Standard Metrics Standard metrics are shown in Table 8. Key takeaways include: (i) When using the BB2 response generator, our trained query generator improves the final response quality compared to the BB2 query generator. (ii) Training the response generator on satisfied data leads to further improvements when using our best query generator. (iii) However, training with additional refinement data does not surpass using satisfied data alone. The reason behind (iii) relates to FITS's gold response collection. Often, the gold response is a user-guided, BB2-generated reply. This biases reference-based metrics towards the original BB2 outputs. Moreover, low-quality references may underestimate model performance when using reference-based metrics (Zhang et al., 2023a) and we confirmed this with a human evaluation of response quality (see Appendix A.4 for details).

Results on Our Designed Metrics Table 7 shows the results when using our designed metrics. Notably, (i) when using the BB2 response generator, our trained query generator improves the final response quality from all perspectives compared to the BB2 query generator. (ii) When equipped with our trained query generator, training the response generator on satisfied data leads to consistent improvements in helpfulness compared to the BB2 response generator, indicating the importance of domain-adapted training. (iii) Training the response generator on both satisfied and refinement data improves the final response quality from all perspectives compared to training on satisfied data only, highlighting refinement data's utility in rectifying model errors. (iv) In terms of satisfaction, the best-performing system employs our query and response generators, both trained on satisfied and refinement data. Additionally, as a further baseline, we gathered the first 200 unsatisfied responses into a sparse refinement training set, refined via instance-level feedback. A model trained on this set alongside satisfied data, fell short compared to our system-level trained response generator, as measured by our designed metrics, see Appendix A.5 for details.

7 Combining System-level Feedback and Instance-level Feedback

Previous studies (Scheurer et al., 2022b; Shi et al., 2022; Chen et al., 2023a) have shown the effectiveness of instance-level feedback in the refinement process. To take a step further, we explore the synergy of system-level and instance-level feedback on dialogue systems. Using response generation as a case study, we collect both human and GPT-3.5 feedback (prompt in Appendix A.6) for the 1,000 unsatisfied responses from §6.2.1. We then design a refinement prompt integrating both system-level

	Valid						Test						Test Unseen					
	GRD	Fact.	Help.	Rel.	Conf.	Sat.	GRD	Fact.	Help.	Rel.	Conf.	Sat.	GRD	Fact.	Help.	Rel.	Conf.	Sat.
SLT(QG+RG(👍+👎))	48.2	**69.1**	41.3	**81.6**	81.1	50.7	43.2	66.7	44.5	76.4	83.6	55.7	**45.3**	71.6	43.9	79.6	76.3	51.4
SLT(QG+RG(👍+HFB👎))	**48.8**	68.1	**43.3**	81.4	**91.9**	**57.3**	**43.8**	**68.5**	**47.8**	**79.4**	**93.5**	**61.2**	45.0	**72.2**	**45.4**	**81.2**	**88.0**	**57.5**
SLT(QG+RG(👍+GPT3.5FB👎))	44.0	66.3	39.4	78.6	80.2	49.4	38.9	66.7	45.6	78.6	81.7	54.7	40.9	69.9	45.2	80.6	75.3	53.1

Table 9: Case study for combining system-level and instance-level feedback: performance of different dialog systems on FITS datasets, evaluated using our designed metrics. See Table 6 for the meaning of the abbreviations.

and instance-level feedback, i.e. both the desired criteria and the specific example-based feedback (see Appendix A.2). We introduce three systems for comparison.

- **SLT(QG+RG(👍+👎))** Our system-level trained response generator using satisfied and refinement data, paired with our trained query generator. The system does not use instance-level feedback.
- **SLT(QG+RG(👍+HFB👎))** Our system-level trained response generator paired with trained query generator. The response generator is trained on satisfied and refinement data (where we incorporate human-written instance-level feedback (HFB) into the response refinement prompt).
- **SLT(QG+RG(👍+GPT3.5FB👎))** Our system-level trained response and query generators, where the response generator is trained on satisfied and refinement data. We incorporate GPT-3.5, rather than human, generated instance-level feedback (GPT3.5FB) into the response refinement prompt.

7.1 Results of Adding Instance-level Feedback

Results using our designed metrics are in Table 9. We observe that adding human-written feedback to the response refinement part brings improvements in the five criteria-based metrics most of the time, and increases the overall satisfaction consistently. However, adding GPT-3.5 feedback results in degraded performance in groundedness, factuality and confidence. Those observations raise two questions: (1) How does GPT-3.5 feedback differ from human feedback? (2) How does human/GPT-3.5 feedback impact response refinement? We address these questions in subsequent sections.

7.2 Human vs. GPT-3.5 Feedback Metrics

To understand why adding human feedback is more beneficial than GPT-3.5 feedback, we analyze their differences through the following perspectives. (1)

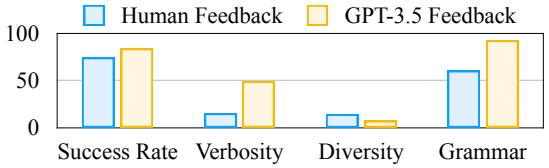

Figure 2: Comparison of human and GPT-3.5 feedback.

Refinement	GRD	Fact.	Help.	Rel.	Conf.	Sat.
No feedback	39.16	**90.35**	83.48	98.10	**100.00**	76.50
Human FB	**40.11**	87.50	81.10	97.80	99.84	74.60
GPT-3.5 FB	32.77	81.50	**90.20**	**98.40**	99.84	**79.50**

Table 10: Quality of refinements with no/human/GPT-3.5 feedback. See Table 6 for abbreviation meanings.

Refinement Success Rate: Percentage of satisfactory feedback-driven refinements. (2) **Verbosity**: Average word count of feedback. (3) **Diversity**: Percentage of unique words. (4) **Grammar**: Percentage of grammatical feedback sentences.[8]

In Figure 2, we show characteristics of human and GPT-3.5 feedback. Though GPT-3.5 feedback is lengthier and grammatically sound, it lacks the language diversity of human feedback. Upon manual examination, GPT-3.5 feedback is often general, whereas human feedback is direct and specific. See the Appendix A.7 for feedback examples.

7.3 Feedback Impact on Refinements

While GPT-3.5 feedback leads to a higher refinement success rate (see Figure 2), the performance of the resulting dialog system trained with these refinements falls short w.r.t. all our designed metrics compared to the system trained using human feedback-driven refinements as shown in Table 9. Therefore, to understand this further we also evaluate the refinement quality via designed metrics from §6.1.1, with results in Table 10. Refinements obtained using human feedback mainly stand out

[8]We use Gramformer for grammar error checking: https://github.com/PrithivirajDamodaran/Gramformer.

in groundedness and factuality. This aligns with the feedback clusters in Table 5 where over 40% of the feedback suggests the bot focus more on the search results; that is, focusing more on the search results will make the refinements more grounded, leading to a more grounded final system (see Table 9). Since language models are known to hallucinate regardless of their size (Ji et al., 2023; Li et al., 2023), grounding their generations to the documents is important to ensure factuality. Hence, groundedness of refinements plays an essential role in the performance of trained models.

7.4 Advantages of Human Feedback

We find that human feedback pinpoints issues more effectively than GPT-3.5 feedback. For example, when a response does not answer a question, GPT-3.5 will say that the response is unhelpful because it does not contain the information the user wants. In contrast, human feedback often provides specific hints from the search results, guiding the model towards a better response. Thus, despite GPT-3.5 producing seemingly informative feedback, it currently can't match the nuance of human annotators.

8 Conclusion

In this paper, we present a framework that harnesses system-level NL feedback. By using a set of instance-level feedback, we derive system-level feedback for refinement prompt engineering and metric design. We show the effectiveness of system-level feedback through two case studies: generating queries and formulating dialogue responses. We further combine system-level and instance-level feedback in the refinement data construction process, and observe that the resulting trained response generator makes considerable improvements versus either alone. Finally, we explore the possibility of substituting instance-level human feedback with GPT-3.5 feedback. We find that human feedback stands out in capturing main issues, while GPT-3.5 feedback is lengthy and less focused.

9 Limitations

Due to the lack of publicly available natural language feedback datasets, our experiments were limited to the small-scale dialog system BB2, which does not represent the current state-of-the-art. We recognize that integrating more advanced models such as ChatGPT could yield further insights, presenting a promising direction for future research. As relevant datasets become more accessible, we look forward to exploring these possibilities.

10 Acknowledgement

The work was done as part of the Meta–NYU mentorship program and partly supported by the National Science Foundation (under NSF Award 1922658). Kyunghyun Cho is supported by the Samsung Advanced Institute of Technology (under the project Next Generation Deep Learning: From Pattern Recognition to AI).

References

Leonard Adolphs, Tianyu Gao, Jing Xu, Kurt Shuster, Sainbayar Sukhbaatar, and Jason Weston. 2022. The cringe loss: Learning what language not to model. *arXiv preprint arXiv:2211.05826*.

Kushal Arora, Kurt Shuster, Sainbayar Sukhbaatar, and Jason Weston. 2022. Director: Generator-classifiers for supervised language modeling.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33:*

Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Oğuz 'Oz' Buruk. 2023. Academic writing with gpt-3.5: Reflections on practices, efficacy and transparency.

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. 2023a. Improving code generation by training with natural language feedback.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023b. Teaching large language models to self-debug.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. 2022. Improving alignment of dialogue agents via targeted human judgements.

Anna-Carolina Haensch, Sarah Ball, Markus Herklotz, and Frauke Kreuter. 2023. Seeing chatgpt through students' eyes: An analysis of tiktok data.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

J. A. Hartigan and M. A. Wong. 1979. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8460–8478, Dublin, Ireland. Association for Computational Linguistics.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Dialogue learning with human-in-the-loop.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2019. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *arXiv preprint arXiv:1911.03860*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for text summarization.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators.

Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2022a. Training language models with language feedback.

Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2022b. Training language models with language feedback.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training language models with language feedback at scale.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. 2023. Towards understanding sycophancy in language models.

Weiyan Shi, Emily Dinan, Kurt Shuster, Jason Weston, and Jing Xu. 2022. When life gives you lemons, make cherryade: Converting feedback from bad responses into good labels.

Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. 2022. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 373–393, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352, Seattle, United States. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jason E Weston. 2016. Dialog-based language learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Jing Xu, Arthur Szlam, and Jason Weston. 2022a. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.

Jing Xu, Megan Ung, Mojtaba Komeili, Kushal Arora, Y-Lan Boureau, and Jason Weston. 2022b. Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback.

Weizhe Yuan, Ethan Chern, Steffi Chern, Chunting Zhou, Chunpu Xu, Binjie Wang, and Pengfei Liu. 2023. chateval.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2023a. Benchmarking large language models for news summarization.

Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023b. Visar: A human-ai argumentative writing assistant with visual programming and rapid draft prototyping.

A Appendix

A.1 Manual Efforts Required to Derive System-level Criteria

In our approach, the feedback grouping was a semi-automated process. Initially, we employed k-means clustering, utilizing the SimCSE encoder to categorize the feedback sentences. This clustering process was conducted once without human

intervention. Then we employed a streamlined, non-iterative manual approach for cluster curation. Specifically, two domain experts independently reviewed 50 samples (each requires around 30s to read) from each of the 15 clusters (including clusters for queries and responses), requiring approximately 375 minutes per person for this phase. This was followed by a collaborative discussion to merge insights and remove duplicate clusters, amounting to an additional 60 minutes per expert. Thus, the total human effort amounted to approximately 14.5 person-hours.

A.2 Refinement with GPT-3.5

We instruct GPT-3.5 to generate query refinements and response refinements using carefully crafted prompts, as shown in Table 11 and Table 12. The corresponding ablation studies with full criteria text for query refinement and response refinement are shown in Table 13 and Table 14. The prompt for using both system-level feedback and instance-level feedback for response refinement is shown in Table 15.

A.3 Evaluation with GPT-3.5

Since previous studies have demonstrated GPT-3's capability in the evaluation of aspects such as factuality (Luo et al., 2023), helpfulness (Fu et al., 2023), relevance (Fu et al., 2023), etc. We use GPT-3.5 to evaluate the following perspectives with the help of `ChatEval` (Yuan et al., 2023).

Query Specificity We use GPT-3.5 to measure specificity (Fu et al., 2023) where we concatenate the dialog context and search query together and ask GPT-3.5 to judge whether the search query is specific using the chain-of-thought technique (Wei et al., 2022). In particular, we use the prompt as shown in Table 16. Before applying it to measure the quality of query refinements. We manually labeled 50 search queries from the FITS training split, and each one's specificity label was decided by three annotators through majority vote. We calculate the agreement between GPT-3.5 and human annotators, and the result is 80%.

Response Factuality We concatenate the search documents and response, and ask GPT-3.5 to judge if all information in the response is supported by the search documents. The prompt we use is shown in Table 17. We conducted a meta-evaluation where we asked three NLP PhD students at the same university as the first author to manually label 50 responses from the FITS training split. The annotation guideline we showed them is the same as the prompt designed for GPT-3.5 evaluation. Then, the three annotators decided on each one's factuality label through a majority vote. The agreement between GPT-3.5 and human annotators is 88%.

Response Helpfulness The prompt we use is shown in Table 18. We conducted a meta-evaluation where we manually labeled 30 responses from the FITS training split and 20 responses from the Red Team dataset (Bai et al., 2022). Three annotators decided on each response's helpfulness label through majority vote. The agreement between GPT-3.5 and human annotators is 84%.

Response Relevance The prompt we use is shown in Table 19. We conducted a meta-evaluation where we manually labeled 30 responses from the FITS training split and 20 responses from the Red Team dataset (Bai et al., 2022). Three annotators decided on each response's relevance label through majority vote. The agreement between GPT-3.5 and human annotators is 84%.

A.4 Human Evaluation on Response Outputs

In §6.4, our analysis revealed that training the response generator with both satisfied data and refinement data does not yield superior performance over using satisfied data alone, as evidenced by the F1 score and Perplexity (PPL) metrics. We hypothesized that this outcome might be attributed to a data bias in the FITS dataset, wherein the gold standard references are frequently produced by the BB2 model. Consequently, standard reference-based metrics, such as F1, tend to favor responses that closely resemble BB2 outputs. This bias potentially results in the underestimation of performance for models generating responses deviating from the BB2 distribution.

To address this limitation, we expanded our evaluation methodology beyond model-based metrics. We conducted an additional human evaluation to compare 100 responses generated by SLT(QG+RG(👍)) and SLT(QG+RG(👍+👎)) against the same queries. In this evaluation, two human annotators were asked to select their preferred response from the two provided, with the options including a "tie". If both annotators agreed that one response was superior, the corresponding model was awarded a "win". In cases of disagreement or agreement on ties, the outcome was recorded as a tie. The results of this human

Prompt for query refinement with GPT-3.5

Given the dialog history, your task is to refine the original search query used to search the Internet so that the modified search query will search for documents that better match the user's needs. You should follow the following requirements:
[Criteria]
Below is the dialog context.
[Dialog Context]
Below is the bot's unsatisfactory query.
[Original Query]
You should modify the original search query into the following:

Table 11: Case study 1: prompt for query refinement with GPT-3.5. **[Criteria]**, **[Dialog Context]** and **[Original Query]** are placeholders to be filled. The underlined sentence is removed when **[Criteria]** is None.

Prompt for response refinement with GPT-3.5

Given the dialog history and the unsatisfactory last response the bot gave, your task is to modify the response appropriately to keep the conversation fluent and consistent. You should follow the following requirements:
[Criteria]
Below is the dialog context.
[Dialog Context]
Below is the bot's unsatisfactory response.
[Original Response]
Below are some useful search results that you could use.
[Search Documents]
You should modify the original response into the following:

Table 12: Case study 2: prompt for response refinement with GPT-3.5. **[Criteria]**, **[Dialog Context]**, **[Original Response]** and **[Search Documents]** are placeholders to be filled.

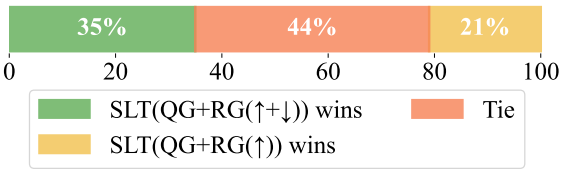

Figure 3: Win rates for system trained with both satisfied and refinement data and system trained with satisfied data only.

evaluation, presented in Figure 3, indicate that SLT(QG+RG(👍+👎)) achieved a higher win rate compared to SLT(QG+RG(👍)). This finding confirms our hypothesis that reference-based metrics alone are insufficient for evaluating this task, highlighting the need for more robust metrics in system assessment.

A.5 Instance-level Feedback vs. System-level Feedback

We argue that one of the drawbacks of instance-level approaches that utilize NL feedback is that they typically assume that every instance receives a feedback text, which is not practical in the real world where feedback tends to be sparse. Therefore, we also conducted a comparison experiment that assumes sparse instance-level feedback. Specifically, we collected the first 200 unsatisfied responses into a sparse refinement training set, refined via instance-level feedback only. We then train the response generator on this set alongside the satisfied data and compare its performance to our system-level trained model. Table 20 shows the performance of the two models as measured by our designed metrics. The system-level trained response generator outperforms the sparse instance-level trained response generator by a large margin on all metrics, demonstrating the importance of system-level feedback in a sparse instance-level feedback setting.

A.6 Feedback Generation with GPT-3.5

Previous studies have demonstrated the capability of large language models to generate informative and useful feedback (Madaan et al., 2023; Chen et al., 2023b). Therefore, we also investigate using GPT-3.5 to generate instance-level feedback for each unsatisfied response. The prompt we use for feedback generation is in Table 21.

A.7 Examples of GPT-3.5 and Human Instance-level Feedback

We list examples of instance-level feedback written by humans and GPT-3.5 in Table 22.

Type	Criteria	Non-copy	Specificity	Readability	Conciseness	Coverage	Satisfaction
Baseline	None	4.06	79.40	19.46	14.87	29.80	61.50
Baseline +Rephrase	(1) To better adapt to search engines, it is best not to copy the user's original words directly. You can rephrase the user's question, use some keywords for the search, and if the user mentions some abbreviations, restore them to their full names.	4.98	83.20	19.54	15.04	26.50	62.10
Baseline +Rephrase +Specificity	(1) To better adapt to search engines, it is best not to copy the user's original words directly. You can rephrase the user's question, use some keywords for the search, and if the user mentions some abbreviations, restore them to their full names. (2) Be accurate and specific enough to reflect the user's needs.	5.00	**84.20**	18.77	14.50	28.80	**63.30**
Baseline +Rephrase +Specificity +Readability	(1) To better adapt to search engines, it is best not to copy the user's original words directly. You can rephrase the user's question, use some keywords for the search, and if the user mentions some abbreviations, restore them to their full names. (2) Be accurate and specific enough to reflect the user's needs. (3) To be able to search for more results, you should use more simple and commonly used words.	**5.08**	80.80	19.53	15.97	29.40	62.40
Baseline +Rephrase +Specificity +Readability +Conciseness	(1) To better adapt to search engines, it is best not to copy the user's original words directly. You can rephrase the user's question, use some keywords for the search, and if the user mentions some abbreviations, restore them to their full names. (2) Be accurate and specific enough to reflect the user's needs. (3) To be able to search for more results, you should use more simple and commonly used words. (4) Your search query should be concise. If the user asks multiple questions, you should focus on his/her first question.	4.81	80.00	**19.70**	**16.63**	**35.30**	62.70

Table 13: Case study 1 (query generation): refinement quality via designed metrics when using different criteria to prompt GPT-3.5 for query refinement.

Type	Criteria	Groundedness	Factuality	Helpfulness	Relevance	Confidence	Satisfaction
Baseline	(1) The modified response should be conversational in tone and no more than twenty words.	34.68	86.60	81.40	89.40	99.60	74.10
Baseline +Groundedness	(1) The modified response should be conversational in tone and no more than twenty words. (2) If the user asks a question, you should use relevant search results to answer the user's question correctly. Please do not let the user check out some resources on his or her own.	36.81	86.60	85.00	89.00	**99.90**	75.80
Baseline +Groundedness +Relevance	(1) The modified response should be conversational in tone and no more than twenty words. (2) If the user asks a question, you should use relevant search results to answer the user's question correctly. Please do not let the user check out some resources on his or her own. (3) Your modified response should be as concise and targeted as possible, and not include additional information the user has not asked for.	36.77	**88.80**	85.60	89.40	**99.90**	74.90
Baseline +Groundedness +Relevance +Confidence	(1) The modified response should be conversational in tone and no more than twenty words. (2) If the user asks a question, you should use relevant search results to answer the user's question correctly. Please do not let the user check out some resources on his or her own. (3) Your modified response should be as concise and targeted as possible, and not include additional information the user has not asked for. (4) Please be confident in your response, and don't start your response with "I'm not sure" or "I don't know".	**39.02**	87.20	**86.60**	**90.60**	**99.90**	**77.00**

Table 14: Case study 2 (response generation): refinement quality via designed metrics when using different criteria to prompt GPT-3.5 for response refinement.

Prompt for response refinement with GPT-3.5 (with instance-level feedback)

Given the dialog history and the unsatisfactory last response the bot gave, your task is to modify the response appropriately to keep the conversation fluent and consistent. You should follow the following requirements:
[Criteria]
Below is the dialog context.
[Dialog Context]
Below is the bot's unsatisfactory response.
[Original Response]
Below is the feedback for the bot's unsatisfactory response.
[Feedback]
Below are some useful search results that you could use.
[Search Documents]
You should modify the original response into the following:

Table 15: Prompt for response refinement with GPT-3.5 (with instance-level feedback). **[Criteria]**, **[Dialog Context]**, **[Original Response]**, **[Feedback]** and **[Search Documents]** are placeholders to be filled.

Prompt for query specificity evaluation with GPT-3.5

You are evaluating a search query for a dialog using a specific set of standards. Below is the dialog context.
[Dialog Context]
Below is the search query.
[Query]
Below are the criteria.
Decide whether the search query is accurate and specific enough to enable retrieval of the most relevant documents on the Internet that are sufficient to answer the user's question.

Does the search query meet the criterion? First, write out in a step-by-step manner your reasoning about the criterion to be sure that your conclusion is correct. Avoid simply stating the correct answers at the outset. Then print only the single character "Y" or "N" (without quotes or punctuation) on its own line corresponding to the correct answer. At the end, repeat just the letter again by itself on a new line.
Reasoning:

Table 16: Prompt used to let GPT-3.5 evaluate query specificity. **[Dialog Context]** and **[Query]** are placeholders to be filled.

Prompt for response factuality evaluation with GPT-3.5

You are evaluating a response for a dialog using a specific set of standards. Below is the dialog context.
[Dialog Context]
Below are some search documents that may help continue this dialog.
[Search Documents]
Below is the response.
[Response]
Below is the criteria.
Determine if the information in the response can be found in one or more search documents.

Does the response meet the criterion? First, write out in a step-by-step manner your reasoning about the criterion to be sure that your conclusion is correct. Avoid simply stating the correct answers at the outset. Then print only the single character "Y" or "N" (without quotes or punctuation) on its own line corresponding to the correct answer. In the end, repeat just the letter again by itself on a new line.
Reasoning:

Table 17: Prompt used to let GPT-3.5 evaluate response factuality. **[Dialog Context]**, **[Search Documents]** and **[Reponse]** are placeholders to be filled.

Prompt for response helpfulness evaluation with GPT-3.5

You are evaluating a response for a dialog using a specific set of standards. Below is the dialog context.
[Dialog Context]
Below is the response.
[Response]
Below are the criteria.
Does the answer directly solve the question?

Does the response meet the criterion? First, write out in a step-by-step manner your reasoning about the criterion to be sure that your conclusion is correct. Avoid simply stating the correct answers at the outset. Then print only the single character "Y" or "N" (without quotes or punctuation) on its own line corresponding to the correct answer. In the end, repeat just the letter again by itself on a new line.
Reasoning:

Table 18: Prompt used to let GPT-3.5 evaluate response helpfulness. **[Dialog Context]** and **[Reponse]** are placeholders to be filled.

Prompt for response relevance evaluation with GPT-3.5

You are evaluating a response for a dialog using a specific set of standards. Below is the dialog context.
[Dialog Context]
Below is the response.
[Response]
Below are the criteria.
Is the response relevant to the topic at hand? It's essential to recognize that the response does not need to be highly specific to the preceding question. As long as it remains focused on the topic at hand, it is considered relevant.

Does the response meet the criterion? First, write out in a step-by-step manner your reasoning about the criterion to be sure that your conclusion is correct. Avoid simply stating the correct answers at the outset. Then print only the single character "Y" or "N" (without quotes or punctuation) on its own line corresponding to the correct answer. In the end, repeat just the letter again by itself on a new line.
Reasoning:

Table 19: Prompt used to let GPT-3.5 evaluate response relevance. **[Dialog Context]** and **[Response]** are placeholders to be filled.

	Valid						Test						Test Unseen					
	GRD	Fact.	Help.	Rel.	Conf.	Sat.	GRD	Fact.	Help.	Rel.	Conf.	Sat.	GRD	Fact.	Help.	Rel.	Conf.	Sat.
SLT(QG+RG(👍+👎))	**48.2**	**69.1**	**41.3**	**81.6**	**81.1**	**50.7**	**43.2**	**66.7**	**44.5**	**76.4**	**83.6**	**55.7**	**45.3**	**71.6**	**43.9**	**79.6**	**76.3**	**51.4**
ILT(QG+RG(👍+👎))	35.5	59.9	30.2	76.2	69.1	38.2	32.4	53.2	36.2	73.2	74.4	44.5	34.7	57.9	35.0	76.2	66.6	42.4

Table 20: Evaluate dialog systems on FITS using designed metrics. **SLT(QG+RG(👍+👎))**: Our system-level trained response generator using satisfied and refinement data, paired with our trained query generator. **ILT(QG+RG(👍+👎))**: Our instance-level trained response generator using satisfied and sparse refinement data, paired with our trained query generator.

Prompt for feedback generation with GPT-3.5

Given the dialog history, the unsatisfactory last response the bot gave, and the requirements for a good response, your task is to write detailed and constructive feedback to improve the unsatisfactory response. The requirements for a good response include the following:
(1) The response should be conversational in tone and no more than twenty words.
(2) If the user asks a question, the response should use relevant search results to answer the user's question correctly. It should not leave the user to check out some resources on his or her own.
(3) The response should be as concise and targeted as possible, and not include additional information the user has not asked for.
(4) The response should be in a confident tone and not start with "I'm not sure" or "I don't know".

Below is the dialog context.
[Dialog Context]
Below is the bot's unsatisfactory response.
[Original Response]
The feedback for the unsatisfactory response is the following:

Table 21: Prompt for feedback generation with GPT-3.5. **[Dialog Context]** and **[Original Response]** are placeholders to be filled.

Human written instance-level feedback	GPT-3.5 written instance-level feedback
Good. You just need to include the reason and his age at the time of his death in addition to this for a complete answer.	The response is not relevant to the user's question and includes additional information that the user did not ask for. A good response should focus on answering the user's question about how Fitzgerald died, using relevant search results. It should be concise, confident, and conversational in tone.
You could tell me to take my cat to the vet when they get sick.	The response is not helpful. Please provide information on treatment options for cat rhinovirus. Use relevant search results to answer the user's question correctly and be concise in your response. Avoid providing additional information the user has not asked for.
The bot did not list Azerbaijan or Estonia, which were included in the search results. The bot should include these two other Soviet Republics in the response.	The response is informative but could be improved. Instead of listing the republics in a long sentence, try breaking them down into bullet points for easier reading. Also, it would be helpful to mention that there were 15 republics in total, as the user requested.
Again, the bot's "answer" doesn't really make sense. It claims that a character, not an actor, played the part of Celie.	The response is confusing and irrelevant. Please provide a clear and concise answer to the user's question using relevant search results. Avoid adding unnecessary information or asking unrelated questions.
The answer is right there in the results, and Charles de Gaulle isn't correct.	The response is good, but it could be improved by being more conversational and concise. Instead of just stating the answer, try something like "Charles de Gaulle was the first French president in 1848. Is there anything else I can help you with?" This makes the response more engaging and shows confidence in the answer.
Interesting. We need to know which premier specifically, with data to back up the claim. Can we be more specific?	The response is not helpful as it does not answer the user's question. Please provide a specific answer with relevant search results to support it. Avoid guessing or providing irrelevant information. Be confident and concise in your response.
Search about living creatures specifically.	The response is not relevant to the user's question. Please provide a direct answer to the user's question using relevant search results. Avoid providing additional information that the user has not asked for. Be confident in your response and avoid starting with "I'm not sure" or "I don't know".
Actually it looks like that happened way later. Let's go back to the Revolutionary War and how that affected Britain.	The response is not helpful as it does not answer the user's question. A better response would be to use relevant search results to explain the reasons for the decline of the British empire, such as economic challenges, decolonization movements, and global power shifts. The response should be confident and concise, and not include additional information the user has not asked for.
Use the search results provided to formulate some answer. Mention how there are wave patterns that one meteorologist uses and got 2/3 correct, mention a tool that they use like the doppler radar which looks for things like airborne tornado debris.	The response is too general and does not answer the user's question. Instead, use relevant search results to provide specific weather conditions that meteorologists look for to predict tornado formation. Keep the response concise and confident in tone.
One of the search results specifies that Paul Tillich was Rollo May's mentor and friend, confirming that they did know each other.	The response should have used relevant search results to answer the user's question. It should have been more targeted and concise, and not included additional information the user did not ask for. A better response would have been: "I'm not sure if they knew each other, but according to my search results, Tillich was a major influence on Rollo May's work."
Why didn't you mention CrossFit or Tabata?	The response should have used relevant search results to answer the user's question correctly. Instead of saying "I'm not sure," the bot should have confidently provided the correct answer. Additionally, the response should have been more concise and targeted, only providing the specific exercise type that burns the most calories.

Table 22: Examples of human written and GPT-3.5 written instance-level feedback. For each row, both human feedback and GPT-3.5 feedback are written to the same unsatisfied response.