A Workbench for Autograding Retrieve/Generate Systems

Laura Dietz
dietz@cs.unh.edu
University of New Hampshire
Durham, NH, USA

ABSTRACT

This resource paper addresses the challenge of evaluating Information Retrieval (IR) systems in the era of autoregressive Large Language Models (LLMs). Traditional methods relying on passage-level judgments are no longer effective due to the diversity of responses generated by LLM-based systems. We provide a workbench to explore several alternative evaluation approaches to judge the relevance of a system's response that incorporate LLMs: 1. Asking an LLM whether the response is relevant; 2. Asking the LLM which set of nuggets (i.e., relevant key facts) is covered in the response; 3. Asking the LLM to answer a set of exam questions with the response.

This workbench aims to facilitate the development of new, reusable test collections. Researchers can manually refine sets of nuggets and exam questions, observing their impact on system evaluation and leaderboard rankings.[1]

CCS CONCEPTS

• **Information systems** → **Evaluation of retrieval results**.

KEYWORDS

Information Retrieval Evaluation, Large Language Mmodels

ACM Reference Format:
Laura Dietz. 2024. A Workbench for Autograding Retrieve/Generate Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3626772.3657871

1 INTRODUCTION

The evaluation of IR systems is traditionally based on passage-level relevance judgments. The assumption is that any good system should retrieve the set of as-relevant judged passages. Once collected, such judgments can be reused in the form of a test collection. However, with the advent of strong auto-regressive LLMs, such as ChatGPT, Llama2, Mistral, and FLAN-T5, the research community is developing new IR systems to which the traditional approach is no longer applicable. The problem is that even for the same search query, such systems will produce a plethora of different responses, each slightly different in linguistic style, level of detail, and tone. Even when asking the same system twice, two slightly different responses will be obtained. Hence, we do not expect to see the exact same system response more than once. While such system responses can still be evaluated by manually judging the relevance of the response, the resulting test collection will not be re-usable under the traditional evaluation paradigm.

There are some ways out of this conundrum:

(1) **Hire more judges or perform A/B tests.**—While this is possible in industry, it is too costly for academic use and does not support reproducible research.
(2) **Use a summarization evaluation metric**. Measure the similarity of system responses to the known correct responses.—While this has demonstrated success [16, 37] the match-quality deteriorates when longer system responses are generated.
(3) **Use Large Language Models (LLMs) to replace human judges**—While Faggioli et al. [9], Sun et al. [30], Thomas et al. [31] empirically demonstrated success, there are several concerns about the trustworthiness of LLMs, especially in a 100% automatic evaluation approach [8, 9].
(4) **Let human judges define which pieces of information are relevant, use automatic alignment methods.** This approach is used in nugget-based evaluation [19, 23, 25, 28] and the EXAM Answerability Metric [26]. Our resource builds on these ideas.

Contributions. In this resource paper, we provide a workbench that builds on three ideas for LLM-based "autograding" approaches:

Question: Relevant responses answer exam questions [11, 26],
Nugget: Relevant responses mention key nuggets [19, 23],
Direct: Relevance can be directly predicted [9, 18, 30, 31].

We are releasing the Rubric-Grading Workbench software, along with usage examples based on TREC DL 2020. (URL in abstract.)

2 RELATED WORK

Our work is unique in that it does not require any passage-level relevance judgments. We build on research ideas that we detail below.

2.1 Traditional IR Evaluation

The dominating approach to IR evaluation is based on (1) developing a collection of queries and a corpus, (2) collecting rankings from a range of retrieval systems, (3) building a judgment pool based on the top k of each ranking, (4) hiring human judges to assess whether each retrieved passage (or document) is relevant or not—optionally using a graded relevance scale.

The evaluation process is based on simplifying assumptions of the Cranfield paradigm [32], such as that the relevance of each passage can be judged independently of other entries in the ranking.

[1] Resource available at https://github.com/TREMA-UNH/rubric-grading-workbench

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657871

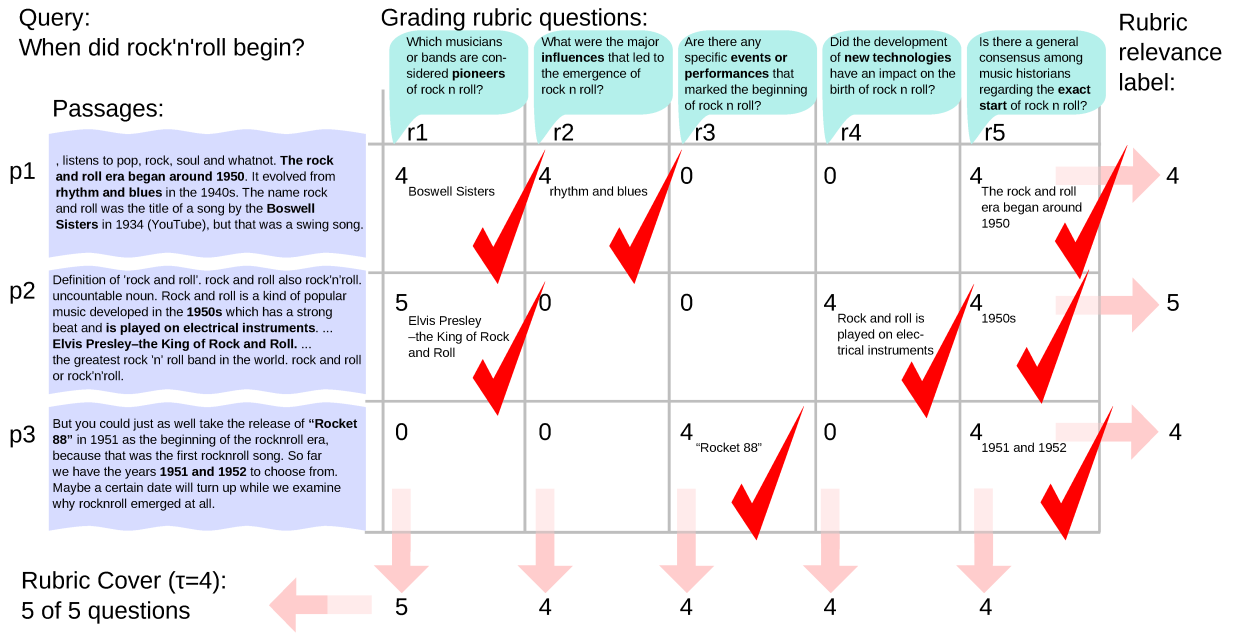

Figure 1: Our Rubric-Grading approach uses LLMs to grade how well a passage p addresses each rubric question r. For each passage and question, the grade on a scale from 0 (worst) to 5 (best) is depicted in each cell. Extracted answers for verification. Relevance labels and coverage scores for the RUBRIC evaluation are derived from these grades. This example is based on actual RUBRIC grades obtained with our system for TREC DL 2020 query `940547`.

This process is supported with several tools, such as `trec_eval`,[2] that determine the quality of a ranking based on the number of relevant passages (or documents) placed at high ranks. All these evaluation measures require that the relevance of each passage is judged, as "holes" can negatively affect the reliability of the evaluation [21].

This paradigm yields reusable test collections when passages from a provided corpus are ranked without modification to the text. In this situation, we can expect that all relevant passages have been manually inspected, and hence any unjudged passages can be safely assumed to be non-relevant.

However, when the assumption that unjudged passages are non-relevant no longer holds, the test collection is not reusable. Violations of this assumption can be due to (1) increased corpus size, (2) reduced variability of ranking methods contributed to the judgment pool, and (3) limited size of judgment pools. Issues arise already when passages are extracted from document fragments [1]. The assumption no longer holds when text is generated with natural language generation or abstractive summarization—especially when the same system may produce a (slightly) different response every time. *Hence, with the Rubric-Grading Workbench, we offer an alternative evaluation paradigm for IR for the era of Large Language Models (LLMs).*

2.2 Nugget-based Evaluation

In contrast to the Cranfield paradigm, work on max marginal relevance [2] and search result diversification [27] argue that redundancy should be avoided. This is captured by evaluation measures such as α-NDCG [3, 24], which define system quality by the coverage of a set of different query interpretations in the ranking.

Nugget-based evaluation is a concrete instance of this idea: Human judges identify important key facts (so-called "nuggets"). The system quality is quantified by the number of key facts mentioned in the system's ranking. This idea is explored in the IR community for a ranking of passages/documents and in the summarization community where a single text is to be assessed [19, 23]. The set of important nuggets is defined by a human judge during topic development and refined during the assessment process. The common definition of a nugget is "the smallest portion of text that constitutes relevant information in and of itself" [23].

Manually identifying which of the key facts/nuggets are mentioned in the systems' response requires additional manual work that poses a hurdle in the application of this evaluation approach. Unfortunately, this work needs to be repeated whenever a new system is to be evaluated.

Our Rubric-Grading Workbench supports evaluation with nuggets, while automatically matching nuggets mentioned in text with LLMs. The success of Chain-of-Thought prompting [35] leads to instruction-tuned LLMs that are trained to identify such nuggets.

[2]The `trec_eval` tool is available at https://github.com/usnistgov/trec_eval

2.3 Question-based Evaluation

While the technology of matching nuggets in text is still new, several decades of research have been invested in the development of question answering systems [17]. By designing a test question for each key fact, we can scan the IR system's response for possible answers. This evaluation is reminiscent of conducting an exam in school: the more exam questions can be answered based on information provided in the IR system's response, the higher the quality of the system.

Originally attempted with manual answer verification [6], it empirically showed success when an automatic question answering system is used for identifying answers to test questions in the system's response [11, 26]. The approach also demonstrated merit for the evaluation of summarization systems, when generating multiple-choice questions [15], free text questions with exact-match answer verification [5], or entity-based questions [7, 33] from a gold summary or source text.

Our Rubric-Grading approach leans on the ability of LLMs to generate and answer questions with context, as recently demonstrated [11]. Our workbench allows to analyze the quality of exam questions and refine the test bank.

2.4 LLM-based Relevance Assessment

Since the release of ChatGPT, several researchers suggested using LLMs to directly assess the relevance of retrieved/generated passages [9, 18, 21, 31]. This approach uses a simple LLM prompt such as "does the passage answer the query?" and uses the response as a relevance label. This approach mimics the passage-level relevance judgment as conducted with human judges, albeit with full automation. Empirically this approach is extremely successful, being able to reproduce the official leaderboard of the TREC Deep Learning track. Similar prompts can be used for passage ranking [30] or to evaluate the quality of LLMs [22].

BertScore [37] and GptScore [13] work similarly but measure quality by the embedding similarity against a known good passage.

We support many such methods for comparison.

2.5 LLM Evaluation with Human-in-the-loop

Despite the empirically successful approach, critiques have been raised about using LLM's for relevance labeling. Faggioli et al. [8, 9] elaborate a wide range of theoretical concerns, centered on questions of trustworthiness and reliability of LLMs especially as new model versions are released in the future. Wang et al. [34] empirically demonstrates that LLMs exhibit unfair positional bias. Liu et al. [20] confirm that when using a particular LLM model for evaluation, it will prefer systems that use the same LLMs model for retrieval and/or generation.

Fok and Weld [12] study general issues of human over-reliance and under-reliance on LLMs. They elaborate why rationales produced by LLMs for human verification do generally not lead to improvements. In the context of judging the veracity of statements with crowd workers, Xu et al. [36] find that LLMs can present wrong rationales convincingly, which can mislead inattentive human judges. An open research question is how to achieve ideal competence partitioning [14] between humans and LLMs to obtain an IR system evaluation paradigm that is reusable, cost-effective, and trustworthy [10].

Nugget- and question-based evaluation paradigms allow for a natural integration of a human-verifier-in-the-loop. We provide the Rubric-Grading Workbench as a proofing ground for developing novel approaches for integrating humans and LLMs into evaluation paradigms.

3 RUBRIC APPROACH

Task statement. Given a set of queries and IR systems, we implement a workbench that aids with the following **evaluation task**:

An information retrieval/generation system is given a search *query q* to produce a relevant *system response* comprised of passages *P*. The response can take the form of a passage ranking, a set of extractive summaries, or a single (generated) text.
Given system responses across queries from multiple systems, the task is to assign each system an *evaluation score* that represents the quality of the information content provided by the system.

Our Rubric-Grading approach proceeds as follows (cf. Figure 1).

3.1 Phase 1: Test Bank Generation

Human judges develop a test bank of nuggets and/or exam questions. To support this activity, our workbench can generate an initial set of test nuggets or exam questions with the help of ChatGPT, using one of the prompts given in Table 1. The human's role is to confirm and refine the test bank, by editing, adding, or removing nuggets or questions. Our process permits to refine the test bank at any time, for example by iterating the process based on outputs of Phase 3.

3.2 Phase 2: Grading

The LLM will grade all system responses, passage-by-passage, by either scanning the passage for mentions of each nugget or trying to answer each exam question based on the passage content.
Our workbench supports three grading modes:

Self-rating answerability: The LLM is prompted to rate to which extent the passage contains information pertaining to the nugget or question on a scale from 0 (worst) to 5 (best).

Answer extraction with verification: For the case of exam questions with a known correct answer, the LLM is trying to answer the exam question with the passage content. Subsequently the produced answer is verified against the known correct answer and counted as "correctly answered" if the answers match.

Informational answer extraction: To support human verification even without known correct answers, the LLM is instructed to extract the answer to an exam question or extract the text span that mentions a nugget.

3.3 Phase 3: Manual Oversight and Verification

To manually verify that the LLM is operating as intended, extracted answers and nugget mentions should be inspected and cross-correlated with derived passage-level grades. Grading prompts may need to be adjusted in response to incorrectly graded passages. Known correct answers to exam questions may be manually expanded whenever correct answers are missed.

Table 1: ChatGPT Prompts for test bank generation. The instruction to respond in valid JSON is appended.

Test Bank	Exam Question	Nugget / Key Fact
TREC DL	Break the query '{query_text}' into concise questions that must be answered. Generate 10 concise insightful questions that reveal whether information relevant for '{query_text}' was provided, showcasing a deep understanding of the subject matter. Avoid basic or introductory-level inquiries. Keep the questions short. {instruction}	Break the query '{query_text}' into concise nuggets that must be mentioned. Generate 10 concise insightful nuggets that reveal whether information relevant for '{query_text}' was provided, showcasing a deep understanding of the subject matter. Avoid basic or introductory-level nuggets. Keep nuggets to a maximum of 4 words. {instruction}
TREC CAR Y3	Explore the connection between '{query_title}' with a specific focus on the subtopic '{query_subtopic}'. Generate insightful questions that delve into advanced aspects of '{query_subtopic}', showcasing a deep understanding of the subject matter. Avoid basic or introductory-level inquiries. {instruction}	Explore the connection between '{query_title}' with a specific focus on the subtopic '{query_subtopic}'. Generate insightful nuggets (key facts) that delve into advanced aspects of '{query_subtopic}', showcasing a deep understanding of the subject matter. Avoid basic or introductory-level nuggets. Keep nuggets to a maximum of 4 words. {instruction}
Instruction	Give the question set in the following JSON format: ```json { "questions" : {question_text_1, question_text_2, ... } ```	Give the nugget set in the following JSON format: ```json { "nuggets" : {nugget_text_1, nugget_text_2, ... } ```

Table 2: FLAN-T5 grading prompts for grading system responses (suggested by ChatGPT 4). Prompt class given in small font.

Grading	Exam Question	Nugget / Key Fact
Self-Rated	QuestionSelfRatedUnanswerablePromptWithChoices: Can the question be answered based on the available context? choose one: - 5: The answer is highly relevant, complete, and accurate. - 4: The answer is mostly relevant and complete but may have minor gaps or inaccuracies. - 3: The answer is partially relevant and complete, with noticeable gaps or inaccuracies. - 2: The answer has limited relevance and completeness, with significant gaps or inaccuracies. - 1: The answer is minimally relevant or complete, with substantial shortcomings. - 0: The answer is not relevant or complete at all. Question: {question} Context: {context}	NuggetSelfRatedPrompt: Given the context, evaluate the coverage of the specified key fact (nugget). Use this scale: - 5: Detailed, clear coverage. - 4: Sufficient coverage, minor omissions. - 3: Mentioned, some inaccuracies or lacks detail. - 2: Briefly mentioned, significant omissions or inaccuracies. - 1: Minimally mentioned, largely inaccurate. - 0: Not mentioned at all. Key Fact: {nugget} Context: {context}
Answer Extraction	QuestionCompleteConciseUnanswerablePromptWithChoices: provide a complete and concise answer to the question based on the context. Question: {question} Context: {context}	NuggetExtractionPrompt: Extract the passage from the text that best relates to the key fact (nugget), ensuring relevance and clarity. Key Fact: {nugget} Context: {context}

We recommend analyzing the effect of individual test bank entries on system rankings (leaderboards). Whenever manual relevance assessments are available, the verification can be complemented with additional analyses such as inter-annotator agreement and rank correlations with official leaderboards.

We envision that the test bank created in Phase 1 will be refined based on the inspection of graded results. For example, additional nuggets and exam questions should be created whenever relevant passages are missed by our Rubric-Grading Workbench.

3.4 Phase 4: Evaluation

An evaluation score for each IR system is computed based on the following assumption: the more nuggets are mentioned in, or the more questions can be answered with a system's response, the higher the system's evaluation score. Our workbench supports two evaluation measures:

Rubric-Cover: A recall-based evaluation score computed as the fraction of test nuggets or exam questions covered with k passages of the system's response.

Rubric-Qrels: Exporting relevance labels, based on the grading level of at least one nugget or exam question. This metric is implemented by using the labels as a "qrels" file for `trec_eval`, and hence supports a wide range of measures.

3.5 Direct Grading

For comparison, we also implement direct grading prompts [9, 18, 30, 31]. These skip over phase 1 and yield passage-level relevance labels for evaluation with `trec_eval`.

4 RESOURCE: RUBRIC WORKBENCH

With this work we release the Rubric-Grading Workbench software, an implementation in python, along with example data. The different phases of the Rubric-Grading approach are implemented as individual commands so it is easy to build complex custom pipelines with a few lines in a python or bash script. Compressed JSON-lines is used as an interchange format, allowing to perform the grading phase on a high-performance GPU server, while human verification can be performed on a local computer with GUI support. Complex queries on data in compressed JSON-lines format can be performed with many available tools such as `jq`,[3] and `DuckDB`,[4] or directly in `python` via provided `pydantic`[5] bindings.

Installation and usage instructions of the Rubric-Grading Workbench is described in detail online.[6] All commands provide extensive line documentation when called with `--help`.

Below we focus on describing the data model and elaborate steps for manual oversight and verification. In Section 6, we provide a detailed walk-through using an example evaluation for the TREC DL 2020 track.

4.1 Phase 1: Test Bank

We envision that a test bank of nuggets or exam questions will be created with a semi-automatic approach. In the example of TREC CAR Y3, exam questions are directly available in the TQA dataset. Test collections with manual nuggets annotations [29] can be imported. Our Rubric-Grading Workbench ingests exam questions and nuggets in JSON format depicted in Figures 2 and 3.

```
{ "query_id": "940547",
  "query_text": "when did rock n roll begin?",
  "info": { "prompt_target": "questions" },
 "items": [{
   "query_id": "940547",
   "question_id": "940547/a4c82219840e6d197d185ed1eda27c61",
   "question_text": "Which musicians or bands are considered
                     pioneers of rock n roll?"
  },...
```

Figure 2: Example question generated for TREC DL 2020.

```
{ "query_id": "940547",
  "query_text": "when did rock n roll begin?",
  "info": { "prompt_target": "nuggets" },
  "items": [{
   "query_id": "940547",
   "nugget_id": "940547/3e9afdb8aeb54b6f496bb72040d7f212",
   "nugget_text": "Early 1950s innovation"
  },...
```

Figure 3: Example nugget generated for TREC DL 2020.

To help seed the test bank creation process, the Rubric-Grading Workbench can generate exam questions and nuggets with the help of ChatGPT. The prompts for test bank generation are given in Table 1. Examples of questions and nuggets generated for TREC DL 2020 query "940547" are provided in Figures 2 and 3. The question and nugget ID should be unique, as this key will be used when grading system responses. Our implementation uses an MD5-hash of the question text along with the query ID.

While we obtain strong empirical results with fully automatically generated questions and nuggets, the concerns raised by Faggioli et al. [8, 9] would still apply. We emphasize that a human-in-the-loop should manually verify the utility of the test bank, to ensure that test nuggets and questions are relevant for the query and are not missing subtle aspects that would indicate relevance. This manual refinement step can be performed before grading or during inspection of graded passages during manual oversight and verification in Phase 3.

A potential concern is that systems under evaluation might be trained to merely address test questions and nuggets, without intending to provide useful information to the user. To discourage systems that "study to the test", we suggest keeping parts of the test bank secret, and only sharing leaderboards and relevance labels.

4.2 Data Model for Autograding

Figure 4 depicts the JSON data model. To use the Rubric-Grading Workbench, system responses are converted into our format, providing at a minimum `Query ID`, `paragraph_id`, and `text` (depicted in light grey in Figure 4). Optionally, the `paragraph` can be provided in full markup, if helpful for manual verification support.

[3] Command line tool `jq` available at https://jqlang.github.io/jq/
[4] The data base DuckDB is available at https://duckdb.org
[5] Documentation for the `pydantic` package is available at https://docs.pydantic.dev
[6] Rubric software: https://github.com/TREMA-UNH/rubric-grading-workbench

```
[ "Query ID",
 [ {  "paragraph_id": "Unique Paragraph Identifier",
    "text": "Full Text of the Paragraph",
    "paragraph": ... // additional markup if available.
    "paragraph_data": {
      "judgments": [ {
         "paragraphId": "Same Paragraph Identifier",
         "query": "Associated Query/Subquery ID",
         "relevance": 2, // judgment grade
         "titleQuery": "Query ID"
       }],
      "rankings": [ {
         "method": "Ranking Method or System Name",
         "paragraphId": "Unique Paragraph Identifier",
         "queryId": "Associated Query/Subquery ID",
         "rank": 6, // retrieval rank
         "score": 17.560528 // retrieval score
       }]
    },
    "exam_grades": [ // grades populated in Phase 2
     {  "correctAnswered": ["Correctly Answered Question IDs"],
        "wrongAnswered": ["Incorrectly Answered Question IDs"],
        "self_ratings": [{
             "question_id": "Question ID",
           // alternatively: "nugget_id": "Nugget ID"
            "self_rating": 4 // self-rating grade
          },
        "answers": [ ["Question ID", "Answer Text"] ],
        "llm": "Huggingface Language Model Used",
        "prompt_info": {
           "prompt_class": "QuestionSelfRatedPrompt",
          "prompt_style": "Can the question be answered based on...",
          "context_first": false,
          "check_unanswerable": true,
          "check_answer_key": false,
          "is_self_rated": true
        },
        "exam_ratio": 0.25 // fraction of mentioned nuggets
     }],
    "grades": [ { // for direct relevance grading
        "correctAnswered": true, // if graded as relevant
        "self_ratings": 4 // grade
        "answers": "Answer Text"
        "llm": "Huggingface Language Model Used",
        "prompt_info" : ...,
     }]
  }]
]
```

Figure 4: Data Model. Query, passage text and ID must be provided. If available, manual judgment level and system information can be used for analysis and verification in Phase 3 and 4. Phase 2 adds the fields `exam_grades` and/or `grades` with information about correct questions/nuggets, self-ratings of answerability, and answers for manual verification. All phases support filtering based on fields `llm` and `prompt_class`.

If applicable and available, manual judgments, system information, and information of a ranking can be provided (cyan). If not available, `judgments` and `rankings` can be set to an empty list. For systems that are generating a longer response instead of a ranking, the field `rank` can be used to indicate a sequence or priority (or be set to "1"). During Phase 3 (Evaluation) it is used as a cut-off criterion for Rubric-Cover.

Phase 2 (Grading) will populate the fields `exam_grades` with information about the coverage of nuggets and answerability of exam questions. Direct grading prompts will populate the field `grades` (`Sun` and `Sun_few` [30], `Fag` and `Fag_fewshot` [9], `Thom` [31], and `HELM` [18]). In both cases, `correctAnswered` will list the nuggets/questions correctly addressed (or true/false for direct grading). The field `wrongAnswered` lists nuggets/questions that the system response did not appropriately address. In particular, this applies to prompts that verify the correctness of the grader LLM's response, as in the case for question answering with a known correct answer or exact match of a nugget.

We suggest to identify addressed questions and nuggets with self-rated prompts given in Table 2. The grader-LLM will rate the extent to which the provided passage covers respective questions and nuggets on a scale from 0 to 5. This information is provided in field `self_ratings` along with the `nugget_id` or `question_id`.

For manual verification of the process in Phase 3, the raw answer of the grader-LLM is stored in the `answers` field, along with nugget or question ID.

Information about the Hugging face model used as grader-LLM is stored in the field `llm`. The grading prompt class and options are preserved in the field `prompt_info`, available for grade selection for the evaluation in Phase 4.

4.3 Phase 2: Grader-LLMs and Prompts

The grader-LLM can work with a wide range of models from Hugging face,[7] supporting the Hugging face pipelines `text2text-generation` (for the T5 family), `text-generation` (for OLMo, Llama-2, and gpt2 models), and `question-answering` (for models that are fine-tuned on SQuAD2).

While the grading prompts can be adjusted by the user of the Rubric-Grading Workbench, we provide a set of default prompts for grading which exam questions are answerable and which nuggets are mentioned. The prompts in Table 2 are designed to work well with models of the FLAN-T5 family. They were chosen based on suggestions from GPT-4.

4.4 Phase 3: Manual Verification and Oversight

A key concern about using LLMs, especially via self-rating prompts, is how to diagnose when the LLM fails to correctly assess whether a question/nugget is addressed in a passage. If many low-quality passages are associated with high grades, the grading prompts need to be adjusted, or a stronger grading LLM needs to be chosen.

To ensure that as-relevant-rated passages are indeed relevant, we recommend that human verification should inspect passage text with a high rating along with the corresponding question/nugget and extracted answer. To reduce the passage pool, emphasis can be placed on passages with at least m highly graded answers.

Our Rubric-Grading Workbench offers support to verify that grading levels correlate with the quality of question's answers and extraction of nuggets via the following analyses:

[7] Models available at https://huggingface.co/models

Verify grading: To confirm the quality of the grading-LLM, this analysis lists extracted answers and corresponding grades question-by-question. This test indicates when grading prompts need to be adjusted.

Grid display: To oversee the process, passages with grades per test question/nugget along with extracted answers are listed.

Missing in test bank: To grow the test bank as needed, this analysis lists relevant passages that do not address any of the current test nuggets or questions. The test bank needs to be extended to account for the relevant information in those passages.

Spurious test bank entries: To identify spurious test bank entries, which would yield false positive relevance labels, this analysis lists test nuggets or questions that are frequently covered by non-relevant passages. Those entries should be removed from the test bank or be reformulated.

Additional analyses are easy to derive by tools that handle JSON formatted files, such as `DuckDB` or `jq`.

4.5 Evaluation Metrics

After refining the test bank and verifying that the grading process works as expected, evaluation scores for each graded system can be exported via two mechanisms: Rubric-Qrels and Rubric-Cover.

For Rubric-Qrels, a `trec_eval`-compatible relevance file is exported, where grades are aggregated into a passage-level relevance label. Our current implementation can be configured to derive the passage relevance label as either (1) number of correctly covered test nuggets/questions, or (2) the highest grades across all test nuggets/questions. Additional measures are easy to implement in our code base. The exported relevance labels ("qrels") can be shared with system developers, so they can evaluate their systems with `trec_eval`. Furthermore, our implementation supports to automatically obtain evaluation measures from `trec_eval` on a collection of "run"-files, to produce a leaderboard of system evaluation scores.

For classes of information needs that rely on covering all relevant information, we provide the Rubric-Cover measure. Rather than obtaining a passage-level relevance label, we consider the total number of test nuggets/questions addressed across the first k passages p of a system's response P. This applies to system responses in the form of a passage ranking as well as to multi-passage responses produced with natural language generation methods. Our implementation will assign a quality score for each system based on the fraction of test nuggets or questions that are addressed. For grading prompts with self-rating, our implementation supports different grade thresholds, at which a test nugget/question is considered covered.

We imagine that several similar evaluation measures can be implemented on the basis of these ideas, including α-NDCG [3, 24].

5 CODE RELEASE

We release the Autograder Workbench as a python implementation along with some example data. The code is released under the BSD-3 open-source license. The workbench is implemented as a Python library which also offers a command-line interface for each phase when installed with `poetry`. To support reproducibility, we lean on the `nix` build system for releases of the Rubric-Grading Workbench software.

6 WALK-THROUGH ON TREC DL 2020

We provide a resource walk-though on an example application to TREC DL 2020 [4]. Data is provided as a separate archive,[8] to be unpacked into directory `./data/dl20/`.

For question generation, we use `gpt-3.5-turbo-instruct`; for grading we use `google/FLAN-T5-large` with the `text2text-generation` pipeline from Hugging Face.[9] In general, `FLAN-T5-large` provides a good trade-off between quality in responses and computational cost. We are able to run all computational steps of this walk-through in 8 hours, using a API access to OpenAI for Phase 1 and a single NVIDIA A40 GPU for Phase 2.

Table 3 gives an overview of all files used as input/output for this walkthrough of the Rubric process.

External input: Query IDs and query text are converted to a JSON dictionary. We select all passages in official judgments and the top 20 of all run submissions—a total of 11,386 passages. For each passage we provide Query ID, paragraph_id, text, paragraph_data as gzip compressed JSON-lines following the format described in Figure 4.

To compare to a generative system, we include several GPT-based systems that did not contribute to the judgment pool. We segment the response into paragraph-size passages of about 400 words.

Phase 1: With our test bank generation prompts, we obtain roughly ten exam questions and test nuggets for each of the 54 queries in TREC DL 2020. Examples are depicted in Figures 2 and 3. A description of file names is given in Table 3.

Phase 2: We grade all passages according to each test bank entry using the grading prompts in Table 2. Since our generated test bank is comprised of open-ended questions, the answer extraction prompts are merely used for manual verification.

Phase 3: To support a human supervising the process,[10] we first **verify the grading**. Below examples for the question "Which musicians or bands are considered pioneers of rock n roll?"

- (rating: 5) Elvis Presley–the King of Rock and Roll ✓(true pos.)
- (rating: 0) rhythm and blues ✓(true neg.)
- (rating: 0) the 1930s ✓(true neg.)

Next, we identify **spurious test bank entries**, i.e., questions and nuggets, that are often covered by non-relevant passages. For the example query `940547` on rock and roll (frequency in parentheses):

- (116) Did rock n roll start as a distinct genre or did it evolve from existing music styles?
- (102) Is there a general consensus among music historians regarding the exact start of rock n roll?
- (60) Were there any notable recordings or songs that played a significant role in popularizing rock n roll?

To detect when the **test bank is missing important entries** and needs to be augmented, we list relevant passages that do not have any high grades. Such passages do not exist for this example query.

[8] Rubric files for TREC DL 2020 available in the online appendix.
[9] https://huggingface.co/google/flan-t5-large
[10] Detailed data available in the online appendix.

Table 3: Description of filenames for TREC DL 2020 walk through, available in the online appendix.

Phase	Filename	Description
External Input	`dl20-queries.json`	JSON dictionary mapping query ID to query text.
	`trecDL2020-qrels-runs-with-text.jsonl.gz`	Passages selected for grading, providing Query ID, paragraph_id, text, paragraph_data as described in Figure 4.
Phase 1: Test Bank	`dl20-questions.jsonl.gz`	Generated exam questions
	`dl20-nuggets.jsonl.gz`	Generated test nuggets
Phase 2: Grading	`questions-explain--questions-rate--nuggets-...trecDL2020-qrels-runs-with-text.jsonl.gz`	Passages graded with nuggets and questions using the self-rating prompt (for formal grades) and the answer extraction prompt for manual verification. Grade information is provided in the field exam_grades.
Phase 3: Manual Verification	`dl20-verify-grading.txt`	All LLM responses are grouped by test question/nugget. by question/nugget ID.
	`dl20-bad-question.txt`	Questions/Nuggets that frequently obtain a grade above 4, judgment below 1.
	`dl20-uncovered-passages.txt`	Passages judged relevant without any grade above 4.
	`dl20-grid-display.txt`	Passages judged relevant without any grade above 4.
Phase 4: Evaluation	`dl20-rubric-qrels-$promptclass-minrating-4.solo.qrels`	Exported Qrel file treating passages with grades ≥ 4 as relevant.
	`dl20-rubric-qrels-leaderboard-$promptclass-minrating-4.solo.mrr.tsv`	Leaderboard produced with `trec_eval` using the Qrel file above and metric MRR.
	`dl20-rubric-cover-leaderboard-$promptclass-minrating-4.solo.tsv`	Leaderboads produced with Rubric-Cover treating test nuggets/questions as answered when any passage obtains a grade ≥ 4.
Analyses Input:	`official_dl20_leaderboard.json`	JSON dictionary with method names mapped to leaderboard ranks (top rank = 1).
Output:	`dl20-rubric-qrels-leaderboard-$promptclass-minlevel-4.correlation.tsv`	Rubric-Qrels leaderboard as before, but including rank correlation with the official DL20 leaderboard, using Spearman's rank and Kendall's tau correlations. (*)
	`dl20-rubric-cover-leaderboard-$promptclass-minlevel-4.correlation.tsv`	Rubric-Cover leaderboard as before with rank correlation with the official leaderboard. (*)
	`dl20-rubric-inter-annotator-$promptclass.tex`	LaTeX tables with graded and binarized inter-annotator statistics with Cohen's kappa agreement. "Min-answers" refers to the number of correct answers obtained above a grading threshold by a passage. (*)
	(*) For TREC DL `--min-relevant-judgment 2` must be set.	

However, we find such passages for query `1108651` "what is the best way to get clothes white":

> Use bleach. If you are only washing white clothes, add a capful of bleach to the load when you plan to wash it. If you are concerned about the powerful effects of standard bleach, try a non-chlorine bleach or a slow-acting bleach, like a 3-percent peroxide solution.

This passage does not answer the related rubric question "How does soaking clothes in bleach affect their whiteness?" It should be reformulated as "How to use bleach to wash white clothes?"

Phase 4: Table 4 displays the leaderboard generated with Rubric-Qrels, using different grade cutoffs. Our Rubric-Grading Workbench exports relevance labels as a "qrels" file, to evaluate with `trec_eval`. Here, we use mean reciprocal rank (`MRR`), but any evaluation measure can be used with this approach.

The leaderboard obtained with RUBRIC Question-4 correlates best with the official leaderboard, with comparable results achieved by `Fag` [9]. Question-based methods work well across different grading thresholds and `trec_eval` metrics. Nuggets are matched too frequently to discriminate between systems.

When the qrels file is shared with system developers, new systems are likely to retrieve new passages which need to be graded with our workbench to avoid bias against ungraded passages [21].

Alternatively, the Rubric-Grading Workbench provides an implementation of Rubric-Cover`@20`, which measures the fraction of the test bank correctly addressed with the overall system response.[11] The leaderboard offers Rubric-Cover evaluation scores

[11] Results provided in the online appendix.

Table 4: Leaderboard for TREC DL 2020 using Rubric-Qrels to score by reciprocal rank on exam questions and test nuggets with different grade thresholds, sorted by question-4. Additional text generation baselines are included by us (in italics, denoted with *). Bottom: Leaderboard correlation and direct grading prompts from Sun [30], Fag + fewshot [9], Thom [31], and HELM [18].

System	Question			Nuggets			Official
grade ≥	3	4	5	3	4	5	rank
*GPT4-question**	0.827	0.747	0.658	1.000	1.000	0.982	≈ 1
*GPT3.5-question**	0.812	0.743	0.650	1.000	1.000	0.991	≈ 1
pash_f3	0.856	0.738	0.579	0.991	0.982	0.978	3
...							...
bigIR-T5xp-T5-F	0.723	0.630	0.531	0.991	0.982	0.969	27
*GPT3.5-wiki **	0.723	0.627	0.497	1.000	1.000	1.000	≈ 27
TUW-TK-2Layer	0.779	0.622	0.511	0.991	0.968	0.956	34
...							...
terrier-InL2	0.675	0.541	0.442	1.000	0.975	0.963	44
*GPT4-wiki **	0.663	0.528	0.418	1.000	1.000	0.991	≈ 44
terrier-BM25	0.672	0.528	0.410	0.991	0.977	0.965	45
...							...
*GPT3.5-web **	0.597	0.506	0.466	0.932	0.932	0.932	≈ 49
...							...
*GPT4-web **	0.630	0.473	0.361	0.907	0.889	0.889	≈ 47
...							...
DoRA_Large	0.186	0.113	0.104	0.707	0.666	0.628	59
Spearman	0.937	**0.941**	0.845	0.275	0.609	0.838	
Kendall	0.800	**0.810**	0.656	0.203	0.449	0.656	
"direct"	Sun	Thom	Fag	Fag fewshot		HELM	
Spearman	0.924	0.751	**0.940**	0.918		0.930	
Kendall	0.785	0.623	**0.815**	0.786		0.785	

Table 5: Inter-annotator agreement with manual judgments.

	Grade	Judgments		Total	Cohen's κ
		2–3	0–1		
Question	4–5	**998**	2377	3375	0.25
	0–3	668	**7343**	8011	0.25
Nugget	4–5	**1211**	4095	5306	0.16
	0–3	455	**5625**	6080	0.16

with standard errors, as well as a normalized Rubric-Cover score as suggested in earlier work [26].

Evaluating text generation. We demonstrate that our approach can be applied to text generation approaches which were not submitted as systems to the TREC DL 2020 track. We ask GPT-4 and 3.5 to generate a system response that imitates a Wikipedia article, a Web page, and gives direct answer for each query. The results are integrated in Table 4, marked with "*", with approximate official ranks if sorted by Question-4.

We demonstrate that Rubric-Qrels is able place these methods on the leaderboard. For example, GPT*-question approaches rank above all submitted systems, GPT4-wiki in the middle and GPT*-web ranks near the bottom of the leaderboard. We remark that since questions and nuggets are generated with GPT 3.5, the scores of GPT3.5 generative methods may be inflated.

Post-hoc analyses. For TREC DL, an official leaderboard and manual judgments are available. These can be used to study the usefulness of the Rubric-Grading approach. We provide abridged results in Table 4. While our focus is to provide a resource to explore the merits of the Rubric evaluation paradigm, the high correlation with the official leaderboard demonstrates its usefulness.

Using the official judgments created by TREC assessors, we can study the inter-annotator agreement between the generated questions/nuggets and human judges. Example results are displayed in Table 5, noting that judgment 1 indicates non-relevance in TREC DL. While our goal was not to imitate the passage-level judgment process, we find a reasonable agreement with TREC judges (without tuning the system). This agreement would likely improve when a human is integrated into the development of the test bank. Direct grading approaches can obtain a higher κ (especially with GPT 4). However, our emphasis is to identify a worthwhile alternative that avoids passage-level relevance assessment altogether and that lends itself to integrating a human-in-the-loop.

We envision that manual effort is focused on passages where manual judgments and Rubric's relevance labels disagree. For example, many relevant passages only obtain self-rated grades of zero. This implies that additional test nuggets or questions need to be added to the test bank. Manual answer verification efforts should focus on passages that obtain a high grade but were explicitly judged as non-relevant.

7 CONCLUSION

We release the software for the Rubric-Grading Workbench (URL in abstract), a toolkit that allows to experiment with different approaches to incorporate LLMs into the evaluation process along with a human-in-the-loop. We focus on ideas that avoid passage-level relevance assessments, as these are difficult for humans to verify without being prone to over-reliance on LLMs or costly manual judgments. Concretely, we incorporate two ideas that were studied in IR literature before: coverage of test nuggets [19, 23] and exam questions [11, 26]. In both cases, a grader-LLM is used to automate the cost-intensive parts of scanning system responses for mentions of key facts and answers. However, addressing concerns about the trustworthiness of LLMs [9], we offer a workbench that supports the integration of manual supervision in the Rubric process. In particular, we are concerned about verifying the grading process (without performing passage-level relevance judgments manually) and providing an analysis framework to diagnose the coverage of the test bank. To instill trust in the process, we provide analyses to study correlation with official leaderboards and agreement with official manual judgments.

To ease the rate of adoption, we (1) support to initialize a test bank with automatically generated nuggets and questions and (2) offer an easy way to integrate with the evaluation tool `trec_eval`.

We hope this resource will help develop novel evaluation approaches for information retrieval and retrieval-augmented generation systems.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1846017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] James Allan. 2004. HARD track overview in TREC 2004 high accuracy retrieval from documents. *Computer Science Department Faculty Publication Series* (2004), 117.

[2] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[3] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 659–666.

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *arXiv preprint arXiv:2102.07662* (2021).

[5] Daniel Deutsch, Tania Bedrax-Weiss, and Dan Roth. 2020. Towards Question-Answering as an Automatic Metric for Evaluating the Content Quality of a Summary. *arXiv preprint arXiv:2010.00490* (2020).

[6] George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*. 138–145.

[7] Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question Answering as an Automatic Evaluation Metric for News Article Summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3938–3948. https://doi.org/10.18653/v1/N19-1395

[8] Guglielmo Faggioli, Laura Dietz, Charles Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, et al. 2024. Who Determines What Is Relevant? Humans or AI? Why Not Both? A spectrum of human-AI collaboration in assessing relevance. *Commun. ACM* (2024).

[9] Guglielmo Faggioli, Laura Dietz, Charles LA Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, et al. 2023. Perspectives on large language models for relevance judgment. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*. 39–50.

[10] Guglielmo Faggioli, Laura Dietz, Charles LA Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, et al. 2024. Who determines what is relevant? Humans or AI? Why not both! *Commun. ACM* 67, 4 (2024). Accepted for publication.

[11] Naghmeh Farzi and Laura Dietz. 2024. An Exam-based Evaluation Approach Beyond Traditional Relevance Judgments. *arXiv preprint arXiv:2402.00309* (2024).

[12] Raymond Fok and Daniel S Weld. 2023. In Search of Verifiability: Explanations Rarely Enable Complementary Performance in AI-Advised Decision Making. *arXiv preprint arXiv:2305.07722* (2023).

[13] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166* (2023).

[14] P.A. Hancock. 2013. Task partitioning effects in semi-automated human–machine system performance. *Ergonomics* 56, 9 (2013), 1387–1399. https://doi.org/10.1080/00140139.2013.816374

[15] Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge Graph-Augmented Abstractive Summarization with Semantic-Driven Cloze Reward. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020). https://doi.org/10.18653/v1/2020.acl-main.457

[16] Mostafa Keikha, Jae Hyun Park, and W Bruce Croft. 2014. Evaluating answer passages using summarization measures. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 963–966.

[17] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700* (2020).

[18] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[19] Jimmy Lin and Dina Demner-Fushman. 2006. Will pyramids built of nuggets topple over?. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. 383–390.

[20] Yiqi Liu, Nafise Sadat Moosavi, and Chenghua Lin. 2023. Llms as narcissistic evaluators: When ego inflates evaluation scores. *arXiv preprint arXiv:2311.09766* (2023).

[21] Sean MacAvaney and Luca Soldaini. 2023. One-Shot Labeling for Automatic Relevance Estimation. *arXiv preprint arXiv:2302.11266* (2023).

[22] Lidiya Murakhovs'ka, Chien-Sheng Wu, Philippe Laban, Tong Niu, Wenhao Liu, and Caiming Xiong. 2022. MixQG: Neural Question Generation with Mixed Answer Types. In *Findings of the Association for Computational Linguistics: NAACL 2022*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 1486–1497. https://doi.org/10.18653/v1/2022.findings-naacl.111

[23] Virgil Pavlu, Shahzad Rajput, Peter B Golbus, and Javed A Aslam. 2012. IR system evaluation using nugget-based test collections. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 393–402.

[24] Tetsuya Sakai. 2017. *α-nDCG*. Springer New York, New York, NY, 1–1. https://doi.org/10.1007/978-1-4899-7993-3_80619-1

[25] Tetsuya Sakai, Makoto P Kato, and Young-In Song. 2011. Click the search button and be happy: Evaluating direct and immediate information access. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 621–630.

[26] David P Sander and Laura Dietz. 2021. EXAM: How to Evaluate Retrieve-and-Generate Systems for Users Who Do Not (Yet) Know What They Want.. In *DESIRES*. 136–146.

[27] Rodrygo LT Santos, Craig Macdonald, Iadh Ounis, et al. 2015. Search result diversification. *Foundations and Trends® in Information Retrieval* 9, 1 (2015), 1–90.

[28] Mark Smucker, James Allan, and Blagovest Dachev. 2008. Human Question Answering Performance using an Interactive Information Retrieval System. (01 2008).

[29] Ian Soboroff, Kira Griffitt, and Stephanie Strassel. 2016. The BOLT IR test collections of multilingual passage retrieval from discussion forums. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 713–716.

[30] Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *arXiv e-prints* (2023), arXiv–2304.

[31] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2023. Large language models can accurately predict searcher preferences. arXiv:2309.10621 [cs.IR]

[32] Ellen M Voorhees. 2009. I come not to bury Cranfield, but to praise it. *HCIR'09* (2009), 13–16.

[33] Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and Answering Questions to Evaluate the Factual Consistency of Summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 5008–5020. https://doi.org/10.18653/v1/2020.acl-main.450

[34] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926* (2023).

[35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[36] Jiechen Xu, Lei Han, Shazia Sadiq, and Gianluca Demartini. 2024. On the Impact of Showing Evidence from Peers in Crowdsourced Truthfulness Assessments. *ACM Trans. Inf. Syst.* 42, 3, Article 87 (jan 2024), 26 pages. https://doi.org/10.1145/3637872

[37] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).