Citation Benchmark

Ali Nazari, Ilia Hashemi Rad, Shayan Salehi,
Mehdi Lotfian, Seyed Mohammad Yousef Najafi, Amir Mohammad Fakhimi
`<ali.nazari.8102@gmail.com><iliahashemirad@gmail.com>`
`<s.salehi1381@gmail.com><mehdilotfian.meloent@gmail.com>`
`<najafim2002@gmail.com><fakhimi.amirmohamad@gmail.com>`
Computer Engineering Department, Sharif University of Technology, Tehran

August 16, 2024

Abstract

In this paper, we propose a comprehensive approach to enhance the evaluation of language models by focusing on the generation and utilization of reference citations. The evaluation of model outputs is crucial in determining their quality and relevance, yet current methodologies often rely heavily on human judgment, which can be time-consuming and unreliable. To address this issue, we have developed a standardized benchmark framework, including novel metrics that emphasize automated, correlation-based assessments of sentence fragments within documents. This framework includes optimized input processing techniques, structured input formats, and improved methods for extracting references from web pages. Leveraging diverse datasets such as ASQA, QAMPARI, ELI5, and Wikidata5m, our approach aims to create a robust and scalable evaluation system. By employing models such as Vicuna and LLaMA alongside established models like GPT-4, we study the effectiveness of these strategies in reducing human intervention and enhancing the accuracy of model evaluation.

1 Introduction

The rapid advancement of natural language processing (NLP) models has brought about significant improvements in the generation of human-like text, enabling applications across various domains, from automated customer service to content creation. However, evaluating these models remains a challenging task, particularly when assessing the relevance and accuracy of the generated content. Traditional evaluation methods often rely on human judgment, which can be subjective and resource-intensive. Furthermore, standardized benchmarks and metrics are needed to ensure the objective comparison of model performance.

This paper addresses these challenges by proposing a novel framework for evaluating language models, with a particular focus on the generation and use of reference citations. The importance of references in assessing the quality of model outputs cannot be overstated, as they provide an objective basis for evaluating the factual accuracy and relevance of generated text. Our approach aims to create a standardized benchmark that minimizes human intervention by introducing automated metrics that leverage correlation checks for sentence fragments within documents.

The proposed framework is built upon several key objectives:

- We aim to establish a standardized procedure for executing benchmarks, ensuring consistency and comparability across different models.
- We introduce new metrics designed to reduce human involvement in the evaluation process, particularly by implementing automated checks for the correlation of meaning within sentence fragments.
- We explore methods for optimizing input processing, such as reducing the token count by providing relevant snippets and summaries.
- We focus on developing structured input formats that improve the consistency and accu-

racy of model outputs, particularly in scenarios requiring multiple document references.

- We suggest a new metric for evaluating the quality in citation task.

To validate our approach, we utilize a diverse set of datasets, including ASQA, QAMPARI, ELI5, and Wikidata5m, which cover a wide range of question types and information needs. These datasets provide a comprehensive foundation for testing our evaluation framework across different content domains and question formats. Moreover, we explore the use of various language models, including Vicuna, LLaMA, and established models like GPT-4, to assess the effectiveness of our proposed methods.

In the following sections, we will detail the methodologies used in the development of our benchmark framework, the implementation of new evaluation metrics, and the results of our experiments with different models and datasets. Through this paper, we aim to contribute to the field of NLP by providing a more robust, scalable, and objective approach to the evaluation of language models, ultimately improving their reliability and applicability in real-world scenarios.

2 Related Work

Evaluating large language models (LLMs) has become increasingly complex as the sophistication and diversity of these models have grown. Traditional evaluation metrics are often inadequate for capturing the nuanced performance characteristics of LLMs, especially in open-ended tasks like question answering. Recent work has explored alternative approaches to overcome these challenges. For instance, [15][**?**] propose a peer rank and discussion-based evaluation framework that involves LLMs assessing each other to mitigate self-enhancement and positional biases. Their approach focuses on pairwise comparisons and utilizes peer discussions to refine evaluations, improving alignment with human judgments. Similarly, [6][13][17] offer a systematic analysis of risks in LLM systems, introducing a comprehensive taxonomy for evaluating and mitigating potential safety and security risks. Their work underscores the need for structured assessment methods that extend beyond performance metrics to include considerations of safety and reliability in real-world applications.

Moreover, recent studies have highlighted the importance of factual accuracy in LLM-generated content. [31][27] emphasize the growing reliance on LLMs for daily tasks, particularly in providing factually accurate information. They introduce new evaluation benchmarks to assess factuality and explore strategies to reduce misinformation, such as leveraging external knowledge retrieval and self-reflection mechanisms. Additionally, [3][20] focus on multi-hop reasoning tasks, where LLMs must integrate multiple pieces of evidence to answer complex questions. Their work highlights the limitations of current benchmarks due to potential data contamination during pre-training and introduces the Inherent Reasoning Evaluation (IRE) method to more accurately assess reasoning chains. Finally, [14][19] explore the attribution of generated answers by leveraging knowledge graphs, developing a benchmark that categorizes attributions into supportive, insufficient, contradictory, and irrelevant. This fine-grained approach provides a more detailed analysis of LLMs' citation behaviors and helps improve the accuracy and reliability of automatic attribution evaluation.

In addition to attribution and factuality concerns, novel frameworks have emerged to enhance LLM outputs' reliability and verification processes. [33] introduce the Hierarchical Graph of Thoughts (HGOT) framework for retrieval-augmented in-context learning, which improves the selection of pertinent passages and emphasizes citation quality in answer generation. HGOT leverages a structured graph approach to enhance factual consistency, utilizing a weighted voting system that prioritizes citations with higher credibility. Similarly, [21][30][32] propose a method for integrating symbolic references into generated text, allowing for easier manual verification by linking different spans of generated text to their source data. Further, [34][4] present `AttributionBench`, a comprehensive benchmark designed to evaluate the accuracy of attribution in generative models, revealing the persistent challenges even state-of-the-art models face in processing nuanced information. Finally, [10] propose the UFO framework, which unifies and extends fact verification by utilizing various fact sources in a plug-and-play manner, demonstrating flexibility across different text generation tasks[16]. These recent innovations provide critical advancements toward improving LLM reliability in both factuality and citation accuracy[8].

3 Implementation

3.1 Base Code

Our implementation is structured into four primary components: **Utils**, **Searcher**, **Run**, and **Eval**. Each of these components plays a crucial role in the workflow, from data preprocessing to evaluation of the final output. Below is a detailed explanation of each part.

Utils: The *Utils* section contains essential utilities that streamline the overall process. These utilities include functions like normalizer, which standardizes input data by removing noise and ensuring consistency, and *load_model*, which facilitates the loading of pre-trained models. These tools are fundamental for setting up the environment and preparing data before it's passed through the main stages of the pipeline.

- **Normalizer:** This function standardizes the input string by performing several text preprocessing steps. It involves:
  - **Lowercasing:** Converts all characters in the string to lowercase.
  - **Punctuation Removal:** Removes all punctuation marks from the string.
  - **Article Removal:** Removes common English articles like "a," "an," and "the."
  - **Whitespace Fix:** Collapses multiple spaces into a single space and trims leading/trailing spaces.

  This method is typically used to normalize text before further processing, ensuring consistency and reducing variability caused by differences in case, punctuation, or articles.

- **Removing citations:** This function removes citation markers from a given sentence, which are typically denoted by numbers in square brackets (e.g., "[1]"). It does this by using regular expressions to find and replace these patterns. The cleaned sentence is then returned without citation markers, making it easier to read and process.

- **Getting maximum memory:** This method calculates the maximum memory available for the current GPU(s) to load models. It:
  - Retrieves the free memory in GB using PyTorch's CUDA utilities.
  - Subtracts a buffer (6 GB) from the available memory to avoid over-allocation.
  - Creates a dictionary that maps each GPU to the computed maximum memory.

  This function is useful for ensuring that the model loading process stays within the available memory limits, preventing out-of-memory errors.

- **Creating formatted document prompt:** This function generates a formatted document prompt by filling in placeholders within a template string. It involves:
  - Replacing ID with the document ID (incremented by 1).
  - Replacing T with the document title.
  - Replacing P with the document text or a shortened version, if specified by *use_shorter* attribute.

  This is typically used to prepare documents for input into a model, allowing the use of either full text or more concise representations like summaries or extractions.

- **Getting summary of a document:** This method attempts to retrieve a shorter version of the document text based on a specified key (e.g., "summary"). It iterates through the documents, and if a shorter version isn't found for any document, it logs a warning and defaults to using the full text. The function ensures that at least one document is always provided, even if the shortened version is unavailable.

- **Creating a prompt:** The *make_demo* function is designed to generate a formatted prompt that can be used to demonstrate or test the model's capabilities. It prepares the input text by filling in placeholders within a template based on provided data.

- **Loading the model:** This function loads a Hugging Face model and tokenizer, preparing them for inference or training. This function provides flexibility in terms of model precision and memory management.

These methods collectively provide essential utilities for text processing, memory management, and prompt preparation. They are modular, allowing them to be easily reused across different parts of the project.

Searcher: This module is designed to search for relevant documents within a given set of documents using either the TF-IDF (Term Frequency-Inverse Document Frequency) method or a dense retrieval model (like GTR). The module provides a class $SearcherWithinDocs$ that can be initialized with a set of documents and a chosen retrieval method ($tfidf$ or gtr). It then allows you to search for the most relevant document based on a given query. The key functions and classes are:

- $doc\_to\_text\_tfidf$: This function takes a document in the form of a dictionary and concatenates its title and text fields into a single string. This string serves as the input text for TF-IDF vectorization.

- $doc\_to\_text\_dense$: Similar to $doc\_to\_text\_tfidf$, this function concatenates the title and text of a document but separates them with a period (.) and space. This format is used for dense retrieval models.

- $SearcherWithinDocsclass$: This class is the core of the module and provides methods for initializing a document retriever and performing searches within the documents. Its functionality is specified by $retriever$ attribute.
  - If $tfidf$ is chosen, it initializes a $TfidfVectorizer$ and fits it on the concatenated text of the documents using $doc\_to\_text\_tfidf$. Then cosine similarity between the query vector and each document's vector is calculated.
  - If gtr (or a similar dense retrieval method) is chosen, it encodes the documents into embeddings using the provided model and stores these embeddings. Then cosine similarity (or dot product) between the query embedding and the document embeddings is calculated
  - If an unsupported method is passed to retriever, the method raises a $NotImplementedError$.

Run: This module orchestrates the overall process of generating language model outputs for evaluation or interaction, utilizing a variety of models and APIs (such as OpenAI's API). It includes configurations, data loading, prompt generation, model invocation, and result handling. The key components are:

- $remove\_citations$: Removes citation-like patterns from text using regular expressions.

- LLM **class:** First it sets up the model based on the provided arguments, either by using the OpenAI API or by loading a local/Hugging Face's model. Then it generates text using $generate$ function.

- **Main execution:**
  - **Argument parsing:** Uses $argparse$ to manage various configurations like model selection, evaluation files, prompt files, and decoding strategies.
  - **Configuration Loading:** Loads configuration from a YAML file if specified, otherwise defaults to the command-line arguments.
  - **Prompt and Evaluation Data Loading:** Loads prompt and evaluation data from JSON files then it generates in-context learning (ICL) demonstrations based on the configuration.
  - **Interactive Mode:** Implements an interactive querying process where the model can search within documents or check documents before producing a final output.
  - **Result Saving:** Handles saving of the generated outputs and associated data to disk. It includes logic for naming the output files based on the various configurations used during execution.

  - **Token Usage Calculation:** If the OpenAI API is used, it calculates the total token usage and estimates costs based on the model selected.

A more detailed workflow:

- **Initialization and Setup:** The script starts by configuring logging, parsing arguments, and loading configurations.
- **Model Loading:** Depending on the *openai_api* flag, either a local/Hugging Face's model or the OpenAI API is used for text generation. The script is capable of handling different models, including those accessed via Azure.
- **Prompt Generation:** Based on the prompts and evaluation data, it constructs input prompts with ICL demonstrations. This is crucial for tasks requiring context-sensitive completions.
- **Text Generation:** Text is generated using the model. The script supports both batch processing and an interactive mode where the model can refine its answers based on user feedback or additional document checks.
- **Final Output Handling:** Results are compiled and saved, including model outputs, prompts, and any relevant metadata like token usage. This ensures that the output can be analyzed and reproduced if needed.

Eval: This module is a comprehensive evaluation script for assessing various metrics related to natural language processing tasks, particularly those involving QA systems, ROUGE scoring, and other metrics for language generation models. Here's a breakdown of the key components and functionality:

- *compute_f*1 : Computes the F1 score between two strings, *a_gold* (reference answer) and *a_pred* (predicted answer). It tokenizes both strings and calculates the precision, recall, and F1 score based on overlapping tokens.
- *compute_exact* : Checks if the two strings are exactly equal after normalization.
- *exact_presence* : Checks if any of the short answers are present in the given context.
- *compute_rouge* : Calculates the ROUGE-L score [5] between generated text and reference texts. If two reference texts are provided, it picks the best score.
- *compute_str_em* : Calculates the STR-EM (Exact Match) metric for short answer questions.
- *compute_len* : Computes the average length of predictions in terms of word count.
- *compute_qa* : Uses a pre-trained RoBERTa [18] model fine-tuned on SQuAD [23] to calculate QA-based metrics such as QA-EM (Exact Match), QA-F1, and QA-Hit. This involves using a QA pipeline to assess the quality of the generated answers.
- *compute_mauve* : Calculates the MAUVE score [22], which is a metric for comparing the distribution of generated text with human text.
- *_run_nli_autoais* : Runs an NLI (Natural Language Inference) model to assess the alignment between a passage and a claim, used for evaluating attribution in generated text. The NLI model used in the function is *google/t*5*_xxl_true_nli_mixture* [9]. This is a variant of the T5 model that has been fine-tuned specifically for Natural Language Inference (NLI) tasks.
- *compute_claims* : Computes the entailment between claims and model-generated outputs using the AutoAIS model.
- *compute_autoais* : Evaluates generated outputs using the AutoAIS model, considering citations, decontextualization, and QA-extracted documents. This function is complex and includes handling multiple citations and checking precision and recall of references.
- *compute_qampari_f*1 : Computes the QAMPARI F1 metric [2], which evaluates answer predictions against a list of possible answers, considering both precision and recall.

Totally this module is a powerful tool for evaluating NLP models, particularly for tasks involving text generation, QA systems, and attribution. It integrates various evaluation metrics and models to provide a comprehensive assessment of the model's performance.

3.2 Suggested Metric

The evaluation of citation quality has traditionally relied on basic metrics like precision, recall, or manual assessment, which often fail to capture the nuanced requirements of citation tasks. We've countered this issue by using some methods combined together. Copilot and Perplexity.AI were used as the models that cite their text while responding to a query. We aim to check the correctness of each cited part of the response, using atomic facts of that part and comparing them with the source that's provided for the part. Here's how it is approached; a query is given to a model (Either Copilot or Perplexity.AI), then the response is divided into different cited parts. For each cited part, atomic facts of the text are first extracted using GPT-4o-mini by specific prompting, then we collect the webpage's content by a web scraper, then we ask GPT-4o-mini to validate each atomic fact, regarding the webpage content provided. Then we'll have a vector for each cited part, that consists of validity of each atomic fact checked and saved by either 0s(as invalid) and 1s(as valid) in each element of the vector. The quality of citation in a particular cited part is evaluated by this: Consider we have a validation vector, named V:

$$score = \frac{\text{number of 1's in V}}{\text{number of V's elements}}$$

The final task of this part is to determine the citation quality of the whole text generated by the model. So we need to aggregate the scores of each cited part of the model's response. In order to do that, we choose to aggregate the scores using weighted average, by dedicating a decaying weight to scores of cited parts. In this way, the first parts of the response matters the most for us and as we approach the end of the text, the weight and the importance decreases in the final score.
Assuming we have $score_1$, $score_2$,..., $score_n$ for n cited parts, we aggregate them in the following method outlined below. Here's the response validation score.

$$\frac{n \times score_1 + (n-1) \times score_2 + ... + 1 \times score_n}{\text{Sum } \sum_1^n}$$

3.3 Reconstruct Summary

One known limitation of current models is their token length, which typically ranges from 2,000 to 8,000 tokens depending on the model. This constraint can impact the number of documents that can be processed by the model during inference. To address this, one approach is to use summarization techniques to condense each document, allowing more documents to be passed to the model. We extended this idea by reconstructing sentences with their named entities.

In the first step, we aimed to extract named entities from the sentences. We tested various approaches, including classic methods like spaCy and deep learning models such as fine-tuned BERT, RoBERTa Large NER, and Microsoft's DeBERTa. However, we found that the classic spaCy model performed best in terms of accuracy. Additionally, we recognized that other parts of the text, such as positive or negative verbs, are also important in some documents. To account for this, we introduced a hyperparameter that allows for the inclusion of other sentence components alongside the named entity tokens. The default value for this hyperparameter is set to 0.25, meaning that one-fourth of the non-NE parts are added to the NE tokens.

In the second step, we needed to reconstruct the sentence using the extracted tokens in their original order. We employed the Llama3 model and utilized a prompt similar to this one to generate the reconstructed sentence:

Generate a sentence using the following entities:

As a result, the quality of the documents improved, leading to better metrics, which are presented in the Table 1.

We evaluated other models, such as GPT-2 and PaLM, but found that Llama3 produced the best-generated sentences, so we chose it for our approach.

Mode	rougeLsum	citation rec	citation prec
Base	25.82	2.76	5.8
Our	29.92	4.2	5.67

Table 1: Reconstruction Results

4 Conclusions

In conclusion, to efficiently run the model and utilize as many documents as possible, we recognized the need to employ techniques to reduce the size of the documents. While other studies have used summarization methods for this purpose, we introduced a novel approach by leveraging named entities and LLaMA 3 to reconstruct the documents. This strategy enabled us to achieve higher reference-finding metrics. We also proposed a new method for evaluating citation task that works with Perplexity.AI and Copilot models.

References

[1] S. Agarwal, H. Zeng, and D. Chakrabarti. Citation: A key to building responsible and accountable large language models, 2023.

[2] S. J. Amouyal, T. Wolfson, O. Rubin, O. Yoran, J. Herzig, and J. Berant. Qampari: An open-domain question answering benchmark for questions with many answers from multiple paragraphs, 2023.

[3] W. Chen et al. Evaluating llms' inherent multi-hop reasoning ability, 2024.

[4] Z. Chen et al. Towards verifiable generation: A benchmark for knowledge-aware language model attribution, 2023.

[5] K. Ganesan. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks, 2018.

[6] S. Ghosh et al. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems, 2024.

[7] M. Gupta et al. Uda: A benchmark suite for retrieval augmented generation in real-world document analysis, 2024.

[8] B. He et al. This reference does not exist: An exploration of llm citation accuracy and relevance, 2024.

[9] O. Honovich, R. Aharoni, J. Herzig, H. Taitelbaum, D. Kukliansy, V. Cohen, T. Scialom, I. Szpektor, A. Hassidim, and Y. Matias. TRUE: Re-evaluating factual consistency evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3905–3920, Seattle, United States, July 2022. Association for Computational Linguistics.

[10] L. Huang et al. Ufo: a unified and flexible framework for evaluating factuality of large language models, 2024.

[11] K. Jain, D. R. Gangwar, R. Gupta, A. Dixit, and A. Srinivasan. Towards reliable and fluent large language models: Incorporating feedback learning loops in qa systems, 2023.

[12] D. Khashabi et al. Ask, assess, and refine: Rectifying factual consistency and hallucination in llms with metric-guided feedback learning, 2024.

[13] H. Kim, G. Han, J. Lee, J. Kang, and M. Seo. Hagrid: A human-llm collaborative dataset for generative information-seeking with attribution, 2023.

[14] M. Li et al. Benchmarking large language models in complex question answering attribution using knowledge graphs, 2024.

[15] R. Li, T. Patel, and X. Du. Prd: Peer rank and discussion improve large language model based evaluations, 2023.

[16] H. Liu et al. Citation: A key to building responsible and accountable large language models, 2023.

[17] N. F. Liu, K. R. C. Burns, M. Gardner, and Y. Tsvetkov. Rarr: Researching and revising what language models say, using language models, 2022.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[19] S. Min et al. Attributed question answering: Evaluation and modeling for attributed large language models, 2022.

[20] F. Nan et al. Automatic evaluation of attribution by large language models, 2023.

[21] T. M. Nguyen et al. Towards verifiable text generation with symbolic references, 2023.

[22] K. Pillutla, S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers, 2021.

[23] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[24] N. Saleh et al. Fables: Evaluating faithfulness and content selection in book-length summarization, 2024.

[25] S. Scialom, M. Gallé, and C. Mazurek. This reference does not exist: An exploration of llm citation accuracy and relevance, 2024.

[26] Y. Shen et al. Aliice: Evaluating positional fine-grained citation generation, 2024.

[27] Y. Shen, P. Lewis, G. Izacard, R. Nogueira, W.-t. Yih, et al. Evaluating verifiability in generative search engines, 2023.

[28] W. Shi et al. Improving language models via plug-and-play retrieval feedback, 2023.

[29] M. Wang et al. Paperqa: Retrieval-augmented generative agent for scientific research, 2023.

[30] S. Zhang et al. Towards reliable and fluent large language models: Incorporating feedback learning loops in qa systems, 2023.

[31] T. Zhang et al. Factuality of large language models in the year 2024, 2024.

[32] W. Zhao et al. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation, 2023.

[33] J. Zhou et al. Hgot: Hierarchical graph of thoughts for retrieval-augmented in-context learning in factuality evaluation, 2024.

[34] Y. Zhu et al. Attributionbench: How hard is automatic attribution evaluation?, 2024.