LLMs as Factual Reasoners: Insights from Existing Benchmarks and Beyond

Philippe Laban Wojciech Kryściński Divyansh Agarwal Alexander R. Fabbri Caiming Xiong Shafiq Joty Chien-Sheng Wu
Salesforce AI
{plaban, wojciech.kryscinski, dagarwal, afabbri, cxiong, sjoty, wu.jason}@salesforce.com

Abstract

With the recent appearance of LLMs in practical settings, having methods that can effectively detect factual inconsistencies is crucial to reduce the propagation of misinformation and improve trust in model outputs. When testing on existing factual consistency benchmarks, we find that a few large language models (LLMs) perform competitively on classification benchmarks for factual inconsistency detection compared to traditional non-LLM methods. However, a closer analysis reveals that most LLMs fail on more complex formulations of the task and exposes issues with existing evaluation benchmarks, affecting the evaluation precision. To address this, we propose a new protocol for inconsistency detection benchmark creation and implement it in a 10-domain benchmark called SUMMEDITS. This new benchmark is 20 times more cost-effective per sample than previous benchmarks and highly reproducible, as we estimate inter-annotator agreement at about 0.9. Most LLMs struggle on SUMMEDITS, with performance close to random chance. The best-performing model, GPT-4, is still 8% below estimated human performance, highlighting the gaps in LLMs' ability to reason about facts and detect inconsistencies when they occur.

1 Introduction

With recent progress in generation capabilities of LLMs, automatic summarization is making its appearance in practical information consumption situations such as summarizing work meetings (Arabzadeh et al., 2022), health records (Jain et al., 2022), or scientific documents (Cachola et al., 2020). To ensure the safe and effective implementation of these applications, it is essential to limit the reach of factually inconsistent summaries, a known issue with generated summaries (Kryściński et al., 2019; Maynez et al., 2020).

Prior work (Kryściński et al., 2020; Fabbri et al., 2021; Gao and Wan, 2022) has annotated corpora of model summaries with labels of factual consistency, finding that most abstractive summarization systems produce a non-negligible portion of inconsistent summaries. In turn, such corpora are used to instantiate tasks such as *inconsistency detection* (ID) (Laban et al., 2022a; Tang et al., 2022), in which models are given (`document, summary`) pairs, and must identify whether the summary is consistent with the document.

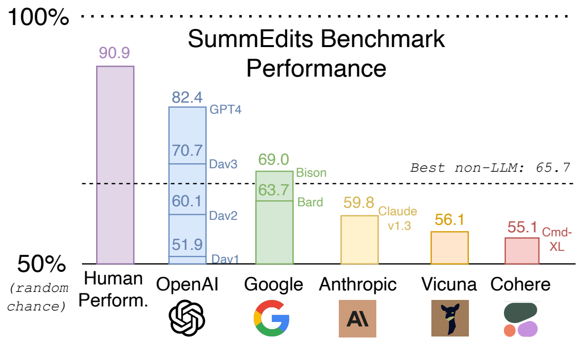


Figure 1: SUMMEDITS is a benchmark to evaluate the factual reasoning abilities of LLMs, measuring if models detect factual inconsistencies when they occur in summaries. Capable detection models can help build more reliable NLG systems.

Recent investigations of using LLMs for evaluation have shown promising results across different NLP tasks (Liu et al., 2023; Fu et al., 2023), including factual consistency (Luo et al., 2023). In this work we continue this line of research and explore applying LLMs as factuality evaluators in the context of text summarization. We first establish baseline performance for a suite of LLMs on three existing consistency benchmarks. Accuracy results confirm that some LLMs perform competitively with state-of-the-art specialized methods such as QAFactEval (Fabbri et al., 2022). However, a manual analysis of free-text explanations that LLMs generate reveals two key limitations of the accuracy-only analysis. Ideally, a model correctly predicting the consistency label of a summary should be capable of generating an explanation for its binary prediction. Yet, we find that most LLMs generate explanations that do not accurately pinpoint factual inaccuracies, with only three models – GPT4 (OpenAI, 2023), Claude V1.3 (Bai et al., 2022), and Bard (Thoppilan et al., 2022) – generating correct explanations for more than 50% of cases we annotated. Second, the manual analysis in the AggreFact consistency benchmark (Tang et al., 2022) of conflict cases – in which GPT4 predictions disagree with the dataset label – reveals a significant number of mislabeled samples (7+%) of factual inconsistencies undetected by annotators during dataset creation that the model explanation reveals. This lack of quality of benchmarks limits the precise evaluation of model performance at factual inconsistency detection.

To address this issue, we introduce a protocol designed to create challenging benchmarks while ensuring the reproducibility of the labels. The protocol involves manually verifying the consistency of a small set of seed summaries and subsequently generating numerous edited versions of these summaries. We discover that assessing the consistency of edited summaries is relatively straightforward and easy to scale for human annotators, thus guaranteeing low cost and high agreement among annotators, yet keeping the task challenging for models.

We create the SUMMEDITS benchmark by implementing the protocol in ten diverse textual domains, including the legal, dialogue, academic, financial, and sales domains. Figure 1 summarizes experimental results on the benchmark, which indicate that SUMMEDITS presents a challenge for both specialized models and current LLMs, with only four models — GPT3-Davinci003, ChatGPT, PaLM2-Bison, and GPT4 — outperforming the specialized model QAFactEval. Our estimate of human performance of 90%+ is largely above all model performance, suggesting most current LLMs are not yet proficient at complex factual reasoning, and still cannot assess the factual validity of summaries with precision.

We believe SUMMEDITS can serve as a tool for evaluating LLMs' abilities to detect when factual inconsistencies (inevitably) occur and encourage LLM developers to report their performance on the benchmark. For practitioners requiring specific domain expertise, the protocol can be adapted to generate low-cost, in-domain benchmarks that can probe for model capabilities prior to production use. We release the code, protocol steps, and benchmark data publicly[1].

2 Related Work

Annotating Factuality of Summaries. With advances in language models and the increase in fluency and abstractiveness of summarizers, prior work showed that one of the key challenges in summarization was enforcing factual consistency (Kryściński et al., 2019), particularly with models trained on datasets with unfaithful references (Maynez et al., 2020). Several efforts – such as FactCC (Kryściński et al., 2020), SummEval (Fabbri et al., 2021), Polytope (Huang et al., 2020), FRANK (Pagnoni et al., 2021), and CLIFF (Cao and Wang, 2021) – annotated the generated summaries of tens of model, finding that most models produce a non-negligible portion of inconsistent summaries. Although most annotation effort has focused on the summarization of news, some prior work also looked at dialogue summarization (Gao and Wan, 2022), or the medical domain (Tang et al., 2023). In most work, scalable high-quality annotation is challenging, due to low inter-annotator agreement when relying on crowd-workers, with some work showing that 10+ annotators are required to achieve some level of consensus (Falke et al., 2019), and some work recommending solely relying on experts (Fabbri et al., 2021). At the heart of the issue, annotating the factual consistency of a summary is challenging: it requires careful reading of long documents and the detection and interpretation of nuanced facts. In this work, we propose a new protocol to annotate factual consistency resources and show that it lowers the cost and increases reproducibility by minimizing the amount of reasoning required for each annotation.

Detecting Factual Errors. Some work has taken an automated approach to the detection of inconsistencies, with approaches falling into two main categories: question and entailment-based. In question-based approaches, questions are generated with the expectation that paired documents and summaries should provide consistent answers. QAFactEval (Fabbri et al., 2022) unified prior work (Wang et al., 2020; Scialom et al., 2021) by systematically evaluating each element of the pipeline and proposing a best-performing combination. Entailment-based methods either rely on entailment on depen-

[1]`https://github.com/salesforce/factualNLG`

dency parses, such as with the DAE method (Goyal and Durrett, 2020), or directly leverage natural-language entailment models, such as SummaC (Laban et al., 2022a). We include these three representative models in our experiments and find that although they require several orders of magnitudes fewer parameters than LLMs, they can reach similar performances on challenging benchmarks.

3 LLM Aptitude In Controlled Setting

In this section, we present the initial set of experiments that were conducted on the FactCC benchmark (Kryściński et al., 2020). FactCC was created based on the XSum news summarization dataset (Narayan et al., 2018) and consists of news article-summary sentence pairs manually labeled based on their factuality. While simple in nature, the benchmark can serve as a test bed for exploring the basic understanding LLMs have of the task at hand. Furthermore, the data points come with manually annotated error types, allowing for experiments in fine-grained error detection.

In the following subsections, we define the experimental setup, i.e. prompts, models, and data, and present the experiment results along with a discussion.

3.1 Prompt Selection

As part of this initial study, we explore a wide range of prompts that have been shown to unlock some of the emergent abilities of LLMs. These prompts can be organized into four groups as follows:

Zero-Shot Prompts (Radford et al., 2019) Evaluate the zero-shot transfer abilities of models. These prompts are limited to a short task description and the input data based on which the models generate their output. In our study, we included three different zero-shot prompts offering varying levels of detail in the task description. The best-performing prompt was selected by a majority vote across models and used as the base for the prompts described in the following paragraphs.

Few-Shot Prompts (Brown et al., 2020) Enable the in-context learning abilities of LLMs. These prompts include a task description and one or more demonstrations of the task. The provided demonstrations condition the model for the actual input data that the model is expected to process. In this study we experiment with one-, two-, and three-shot prompts which build upon each other.

Chain-of-Thought Prompts (Wei et al., 2022) Explore LLM models' ability to generate step-by-step reasoning for answers and have been shown to improve performance on complex reasoning tasks. The models are given a task description and input data and are asked to generate a series of intermediate reasoning steps necessary to solve the task alongside the answer. We explore chain-of-thought prompts both in zero- and few-shot settings.

Generate-with-Evidence Prompts (Lampinen et al., 2022) Explore the models' ability to present evidence for the generated answers and has also been shown to improve performance on reasoning-intense tasks. Similar to chain-of-thought prompts, the models are given a task description and input data and are asked to answer the task, and then generate evidence for the chosen answer. In this work we explore generate-with-evidence prompts in zero- and few-shot settings.

Persona-based Prompts (White et al., 2023) Extract certain points of view from LLMs or focus them on a set of abilities. Shown to work best with chat-tuned LLMs, models are assigned a role, or "persona", and next prompted to complete a given task. The assigned personas condition the models on the task at hand. In this work, models were assigned the persona of a “journalist” who is fact-checking a text before publication. Three prompts were tested, where the persona-based prompt was used in zero- and few-shot settings, and in combination with a chain-of-thought prompt.

All prompt templates described in this section and used in the study are presented in the associated code repository.

3.2 Model Selection

Similar to the prompt selection, we begin with evaluating a wide range of methods that can be applied to the task of factual consistency evaluation. Selected models span different architectures and training procedures, and can be categorized into the following groups:

Non-LLM Models that were designed and trained specifically for the task of factual consistency evaluation in text summarization. Those models include two NLI-based approaches, DAE (Goyal and Durrett, 2020) and SummaC (Laban et al., 2022a), and a QA-based method QAFactEval (Fabbri et al., 2022). In this work, we treat the Non-LLM models as baselines and points of comparison with LLM-based factuality evaluators.

Foundation Models Large-scale models that have been pre-trained on web-scale corpora, but have not been fine-tuned on task-specific or instruction-following datasets. Such models have shown emergent abilities, such as zero- and few-shot in-context learning. Models in this group include Meta's LLaMa-13b (Touvron et al., 2023), and OpenAI's Ada001, Babbage001, Curie001, and DaVinci-001.

Instruction-tuned LLMs Foundation models which were further tuned on instruction-following data either through supervised learning or RL-based methods. Such models show enhanced capabilities of following natural language instructions, including zero- and few-shot prompts as well as chain-of-thought approaches. Models in this group include Databrick's Dolly, Stanford's Alpaca (Taori et al., 2023), Anthropic's Claude V1.3, Cohere's Command-XL, Google's PaLM2-bison, and OpenAI's DaVinci-002, and DaVinci-003 models.

Chat-based LLMs Foundation models tuned on conversational and instruction-following datasets. The fine-tuning procedure aims to enhance the model capabilities of engaging in multi-turn dialog with end-users while being able to solve a wide range of complex tasks. This group includes Google's Bard, Mosaic's MPT-7b-chat (Team, 2023), Vicuna-13b (Chiang et al., 2023), and OpenAI's GPT3.5-turbo (ChatGPT), and GPT-4.

For each model, model cards and method of access are provided in Appendix A, model architecture and training details are described in the associated literature.

3.3 Experimental Setup

Experiments described in the following subsections were conducted on the synthetic part of the FactCC dataset. We select a total of 150 samples to conduct experiments, by including 25 examples for each of the 5 error types in the dataset, i.e. date-, entity-, negation-, number-, and pronoun-related errors, and 25 factually correct samples. Considering that the original data was generated using heuristics, all examples were selected manually to ensure high-quality data.

	Model (↓)	Non-LLM Models				
✓	DAE	67.2				
✓	SummaC	96.8				
✓	QAFactEval	93.6				
	Prompt Group →	ZS	FS	Pers	CoT	GwE
	LLaMa-13B	50.0	51.6	52.4	-	-
	Alpaca-13B	54.8	48.4	57.2	-	-
	Dolly-v2-12B	50.4	50.8	50.8	-	-
	MPT-7B-Chat	58.7	54.0	54.4	-	-
✓	Vicuna-13B	65.5	68.0	63.2	-	-
✓	Cohere-CMD-XL	61.3	50.0	53.3	64.7	54.8
✓	Claude-v1.3	76.4	83.9	72.0	79.7	77.2
✓	Bard	79.3	72.3	73.7	82.0	71.9
✓	PaLM2-Bison	82.3	75.5	63.7	73.1	71.3
	Ada001	46.4	47.7	49.6	52.0	50.0
	Bab001	51.9	57.1	49.5	49.1	53.1
	Cur001	53.5	53.3	51.1	57.5	56.3
✓	Dav001	61.2	56.8	52.9	61.6	58.1
✓	Dav002	74.5	81.3	57.9	78.5	73.2
✓	Dav003	82.3	78.4	62.4	85.5	76.8
✓	GPT3.5-turbo	84.3	82.9	75.1	84.0	86.3
✓	GPT4	91.3	90.1	66.3	85.7	78.0

Table 1: Balanced accuracy on the synthetic FactCC benchmark per prompt group (averaged across prompts in each group). Specialized non-LLMs, (top) Foundation Models, Instruction-tuned LLMs, and Chat-based LLMs (bottom). For LLMs, performance is evaluated with Zero-shot (`ZS`), Few-Shot (`FS`), Persona (`Pers`), Chain-of-Thought (`CoT`), and Generate-with-Evidence (`GwE`) prompts when sequence length allows.

3.4 Inconsistency Detection

We first evaluate the models' ability to detect factual inconsistencies in a binary classification setting with `Yes-No` labels. Non-LLM models return a continuous score attributed to a label class using a tuned threshold, while LLM-based approaches generate free-form text where the final output is retrieved using regular expressions. Due to input length restrictions, certain models could not be tested on more complex (and longer) prompts. Table 1 presents the balanced accuracy scores averaged across three prompts within each prompt category. The results provide the following insights:

Two non-LLM models achieve near-perfect accuracy and substantially outperform LLM-based evaluators. We speculate that this might be caused by non-LLM models being over-optimized to the idiosyncrasies of this simple error detection task and might not hold for more involved detection examples. We investigate this question further in later sections of the paper.

Regarding the prompt design, we notice that for most models (8/12), providing a few examples of the task (zero- $\rightarrow$ few-shot) improves the per-

	Error Type					
Model (↓)	POS	DS	ES	NSent	NS	PS
DAE	96.0	12.0	44.0	28.0	52.0	44.0
SummaC	96.0	100.0	100.0	100.0	100.0	80.0
QAFactEval	96.0	84.0	92.0	96.0	96.0	84.0
LLaMa-13B	88.8	10.4	13.6	14.4	12.8	12.8
Alpaca-13B	80.0	30.4	20.0	36.0	25.6	28.0
Dolly-v2-12B	93.6	3.2	11.2	10.4	7.2	5.6
MPT-7B	72.0	36.0	41.6	52.8	38.4	40.0
Vicuna-13B	68.8	59.2	63.2	74.4	65.6	48.8
Cohere-CMD-XL	85.1	32.0	31.5	36.3	26.1	17.1
Claude v1.3	71.7	82.4	80.3	89.1	89.9	78.1
Bard	80.0	68.8	69.3	77.3	83.7	59.2
Palm2	96.5	47.7	45.3	65.1	52.0	38.9
Ada001	58.7	36.5	40.3	45.9	39.2	36.3
Bab001	70.1	33.9	29.6	41.6	30.7	34.7
Cur001	88.0	12.0	17.3	45.1	16.5	12.3
Dav001	88.0	21.9	28.5	50.4	27.5	13.1
Dav002	80.3	66.4	61.3	74.9	71.5	55.5
Dav003	93.1	66.1	58.4	71.2	69.9	39.7
GPT3.5-turbo	87.2	82.4	63.5	87.5	89.1	66.7
GPT4	86.1	74.9	77.3	81.6	84.3	74.1
LLM Avg.	81.64	44.95	44.25	56.10	48.81	38.87

Table 2: Accuracy on the synthetic FactCC benchmark per error type (averaged across all prompts). Specialized non-LLMs (top) Foundation Models, Instruction-tuned LLMs, and Chat-based LLMs (bottom). Performance is assessed individually for positive examples (`POS`) and each of the error types: Date Swap (`DS`), Entity Swap (`ES`), Negated Sentences (`NSent`), Number Swap (`NS`), Pronoun Swap (`PS`).

formance by an average of 2.7 percentage points. However, for two models, GPT4 and PaLM2, the performance in the same setting dropped substantially (-6.2 pp). Considering those the two models achieve strong performance across all prompt types, we conclude that few-shot examples can help models but are not necessary for top-performing models.

In the majority of cases (8/12) Generate-with-Evidence prompts outperform Chain-of-Thought prompts corroborating prior findings of Ye and Durrett (2022) that models perform better when generating an answer followed by the evidence, rather than generating reasoning followed by an answer as in CoT. An in-depth evaluation of the factual reasoning capabilities of models is presented in the following section.

Persona-based prompts improve the performance of GPT3.5-Turbo; however, they lower the performance of all other models, including the remaining chat-based LLMs. This finding suggests that conditioning the generation on a specific persona might be a feature exclusive to ChatGPT, possibly linked to the data used for tuning, rather than a broader trait of chat-tuned LLMs.

Input Document
Prue Leith, who takes over from **Mary Berry** on the show, insists “baking is the best way to get people into cooking”. The latest series returns this autumn on Channel 4 after it outbid the BBC. The previous hosts announced they “were not going with the dough” when the show switched channels. The pair had presented all seven series of the show alongside judges Berry and Hollywood. Similarly, Berry also announced **she** would not remain with the show on Channel 4, citing "loyalty" to the BBC. [...]
Inconsistent Summary
Berry also announced that **he** would not be staying on the show on Channel 4, citing "loyalty" to BBC.
● **Correct Explanation**
No. The summary incorrectly refers to Berry as "he" instead of "she". – `GPT-4`
● **Partially Correct Explanation**
The summary incorrectly states that Mary Berry is a he and that Berry cited loyalty to the BBC as a reason for not remaining on the show. – `Claude V1.3`
● **No Explanation**
No – `Bab001`
● **Unrelated Explanation**
For me the answer is No! The document says "The new BBC series returns this autumn on Channel 4 after it outbid the BBC". – `LlaMa-13b`
● **Incorrect Explanation**
The document does not mention Mary Berry announcing that she would not be staying on the show on Channel 4. – `Alpaca-13b`

Table 3: Examples of explanation types used to evaluate model ability at explaining inconsistencies they detect.

We study model performance for each error type, averaging the accuracy score across all prompts for LLM models. Results are presented in Table 2. We notice that the majority of LLM-based evaluators achieve satisfactory accuracy (> 80%) in classifying positive (factually correct) examples. The results also highlight that with a few exceptions (Claude, ChatGPT, and GPT4) LLMs struggle to consistently detect factual inconsistencies, in many cases achieving scores below random chance. Pronoun swap detection seems to be the most complicated of error types averaging 38.87% across all LLM-based evaluators and also scoring lowest for specialized, non-LLM models.

3.5 Factual Reasoning

To gain further insights into LLM ability to reason about factual consistency, we performed a manual analysis of more than 3,600 explanations generated for sixteen of the seventeen LLMs included in our experiments[2].

In our analysis, we focus on cases the model classifies as inconsistent, as there is a clear explanation a model should provide to pinpoint elements in the summary it identifies as inconsistent.

For each known inconsistent `(document, summary)` sample in FactCC, and each model output explaining why the sample is inconsistent, we hired an annotator to label the explanation with one of five labels: ● entirely correct: the model's full explanation must accurately describe a factual inaccuracy in the summary, ● partially correct: the model correctly describes at least one factual inconsistency in the summary, but also describes an incorrect element or facts unrelated to factual consistency, ● no explanation: the model provides a classification label (Yes/No) without providing the requested explanation, ● unrelated: the model's output addresses aspects other than factual consistency or is not an explanation (e.g., the model writing a new summary), and ● incorrect: the model's explanation is invalid and does not correctly identify an element in the summary which is factually incorrect. Table 3 gives an example of each explanation type from the annotation, and Appendix B provides further information on the hiring and onboarding of the two annotators that completed the task. During annotation, the annotator samples were presented in a shuffled order, and the annotator was not aware of the models that had generated any particular sample.

We analyze annotation reproducibility by collecting multiple annotations for roughly 200 annotations and computing Cohen's Kappa. We find a moderate agreement amongst annotators of 0.72 on the five-way annotation.

Figure 2 summarizes the results by providing the distribution of types of explanations generated by each LLM.

First, we find that all models struggle to provide correct explanations pinpointing the inconsistencies in summaries, with nine of the sixteen models providing correct explanations less than 10% of the time and only three models – Bard, Claude V1.3, and GPT4 – providing correct explanations more than 50% of the time.

We also notice that better performance at the binary classification task (Table 1) does not necessarily lead to more accurate model explanations. For example, GPT3.5-turbo performs 5-10% better in terms of binary accuracy than Claude V1.3, yet it generates almost half as many correct explanations. This finding suggests accuracy metrics sometimes overlook models that are *right for the wrong reasons* (McCoy et al., 2019), which might negatively affect user trust.

Analyzing the failure behavior of models reveals differences amongst models. The first group of models – including Dav001, Dav002, and Cur001 – fails by not providing an explanation for the inconsistency, even though they were explicitly prompted to accompany their answer with an explanation. A second group – including Ada001, LLaMa-13B, and Cohere-cmd-XL – most often generates "unrelated explanations", which do not explain the nature of factual inconsistency, but might instead quote other facts omitted from the summary, propose a new summary, or other tangential text. A third group – with models such as MPT-7B-Chat and Dolly-v2-12B – generates plausible explanations that are factually incorrect, or present invalid logic. We argue that although all models should strive to produce only correct explanations, some failure cases are preferable to others. For example, it might be preferable for a model to provide no explanation than a plausible-looking but incorrect one that might mislead a user. For example, MPT-7B-Chat and Dav003 both generate roughly the same proportion of correct explanations (21 vs. 24%), yet when they fail, Dav003 is much more likely to abstain from providing an explanation, whereas MPT-7B-Chat is more likely to provide an explanation with incorrect reasoning, which could prove more harmful.

3.6 Fine-grained Inconsistency Detection

To explore the LLMs' fine-grained understanding of factual evaluation we designed a set of experiments prompting the models to evaluate each `(document, sentence)` pair with respect to individual error types. For each of the error types present in the data models were expected to overlook any other factual error, thus evaluating model ability to selectively focus on one error type while ignoring others. In the experiments, we filter out

[2]PaLM-Bison was released shortly before the publication of this work and could not be included in the manual analysis.

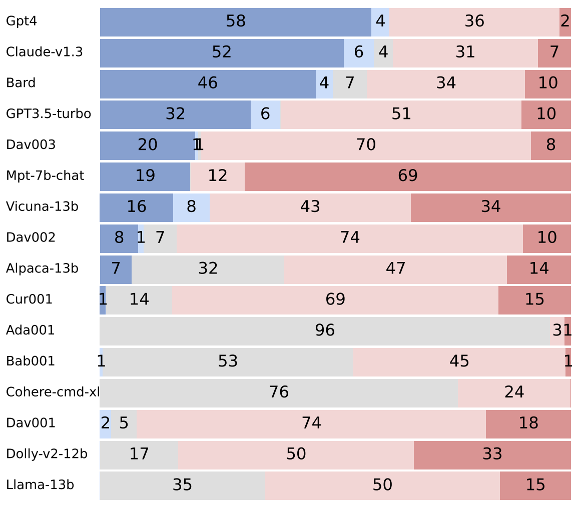


Figure 2: Percentage distribution of the types of explanations each LLM provides in its output when predicting a FactCC summary is inconsistent. Each model explanation is manually annotated as ● entirely correct, ● partially correct, ● no explanation provided, ● unrelated to factuality, ● or incorrect.

consistent summaries, and examples containing the error type associated with the prompt were considered "positive", while other error types are considered as negative, and measured performance in terms of Precision and Recall. We use individual prompts for each of the error types and also conduct experiments where individual prompts are combined with few-shot examples to aid the model in understanding the task. Results are presented in Table 4.

The results show a consistent pattern across all considered LLMs and error types, where the models achieve a low precision and high recall score. This indicates that the models are not able to follow the fine-grained instructions and distinguish different error types. Instead, they simply detect factual inconsistencies (to the best of their abilities) on a general level and assign a negative label. Providing the models with examples of the error to be detected (few-shot setting) does improve the performance of a subset of models; however, no general performance improvement patterns emerged.

In short, none of the models we experiment with can perform the task of fine-grained factual inconsistency detection, in which they are tasked with focusing on a single type of factual error.

Additionally, we carried out an experiment where the per-error-type prompts were combined into a single instruction with multiple tasks that the model was expected to complete in a sequence. Qualitative analysis of the results showed that most of the models could not follow the instructions and consistently provide answers in the expected format, thus the results were not included in the final results table.

4 Limits of Crowd-Based Benchmarks

In this section we analyze two popular benchmarks for factual consistency detection in summarization: AggreFact (Tang et al., 2022) and DialSummEval (Gao and Wan, 2022) and uncover limitations that guide the design principles of the SUMMEDITS benchmark we build.

4.1 Experimental Setup

In the subsequent sections of the paper, we incorporate the insights gained from the experiments on FactCC to inform our experiment design in terms of model and prompt selection.

First, we filter out all models that did not achieve a balanced accuracy above 60% on FactCC, as such models are unlikely to significantly outperform random chance on more challenging benchmarks. Checkmarks (✓) in Table 1 indicate models that are retained in the experiments of Sections 4.1-6.

Second, to minimize the computational cost of experiments, we select a single Zero-Shot (ZS) prompt that is used for all LLM models. We make this choice instead of using multiple prompts per model or selecting each model's best-performing prompt on FactCC results for three reasons: (1) there's no guarantee that prompt quality will transfer across benchmarks, and using a single common prompt removes variance from prompt optimization that does not measure underlying model ability, (2) top-performing LLMs such as GPT4 achieve their best performance on FactCC with ZS prompts, indicating that high performance with a simple ZS prompt is achievable, and (3) more complex prompts would require adaptation to each domain (e.g. domain-specific few-shot examples), and restrict the evaluation of models with shorter maximum sequence lengths due to longer prompts.

4.2 AggreFact

The AggreFact-SOTA (Tang et al., 2022) benchmark is a factual consistency benchmark focused on the news domain, modified from the SummaC benchmark (Laban et al., 2022a) focused on summaries generated by *SOTA* models (i.e., models based on pre-trained Transformers), as analysis

	Zero-Shot										Few-Shot									
	DS		ES		NSent		NS		PS		DS		ES		NSent		NS		PS	
Model (↓)	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
LLaMa-13B	12.0	12.0	18.8	12.0	29.4	40.0	20.8	20.0	12.5	8.0	-	-	-	-	-	-	-	-	-	-
Alpaca-13B	20.5	32.0	12.9	16.0	36.6	60.0	20.9	36.0	23.1	36.0	-	-	-	-	-	-	-	-	-	-
Dolly-v2-12B	26.7	16.0	18.8	12.0	14.3	16.0	22.2	8.0	33.3	12.0	-	-	-	-	-	-	-	-	-	-
MPT-7B-Chat	19.1	52.0	19.7	60.0	23.6	68.0	18.1	52.0	18.8	48.0	-	-	-	-	-	-	-	-	-	-
Vicuna-13B	20.8	100.0	19.1	84.0	23.4	100.0	21.4	96.0	18.3	80.0	-	-	-	-	-	-	-	-	-	-
Cohere-CMD-XL	22.1	92.0	19.4	84.0	26.9	100.0	22.1	100.0	17.3	76.0	17.6	72.0	21.2	72.0	30.8	96.0	20.2	84.0	21.1	80.0
Claude-v1.3	20.3	96.0	19.7	96.0	19.2	92.0	20.8	100.0	20.3	100.0	21.9	92.0	22.0	96.0	22.9	96.0	26.0	100.0	21.0	100.0
PaLM2-Bison	21.1	64.0	21.3	76.0	26.1	92.0	23.6	84.0	20.2	80.0	21.7	60.0	17.5	56.0	28.8	84.0	25.3	80.0	20.2	68.0
Ada001	19.7	92.0	20.0	96.0	19.3	92.0	19.3	92.0	18.4	84.0	33.3	4.0	0.0	0.0	0.0	0.0	22.8	92.0	20.0	4.0
Bab001	4.8	4.0	19.0	16.0	18.2	24.0	10.3	12.0	11.5	12.0	27.3	24.0	20.0	24.0	23.5	32.0	12.5	16.0	19.4	28.0
Cur001	13.5	28.0	22.0	44.0	32.8	84.0	17.9	28.0	15.4	24.0	16.4	44.0	15.9	44.0	26.6	84.0	14.3	32.0	16.5	52.0
Dav001	13.7	28.0	26.3	60.0	39.5	68.0	14.3	28.0	13.0	12.0	18.6	52.0	18.8	36.0	37.5	72.0	17.3	36.0	10.9	24.0
Dav002	20.2	92.0	22.4	96.0	23.4	88.0	21.9	100.0	20.5	92.0	18.3	84.0	19.8	92.0	21.1	96.0	20.3	100.0	19.7	96.0
Dav003	20.4	92.0	20.2	96.0	24.7	96.0	21.7	100.0	20.2	96.0	25.6	92.0	21.6	96.0	24.0	92.0	27.5	100.0	19.3	92.0
GPT3.5-turbo	20.6	88.0	18.0	80.0	24.0	100.0	22.9	100.0	21.1	96.0	20.2	92.0	16.7	76.0	21.8	96.0	22.7	100.0	20.7	96.0
GPT4	31.0	88.0	20.5	96.0	22.0	96.0	24.3	100.0	19.8	96.0	26.4	92.0	20.2	96.0	22.4	96.0	21.9	100.0	20.5	96.0

Table 4: Precision (P) and Recall (R) scores of error detection with fine-grained prompts for individual error types. Experiments run in Zero- and Few-shot settings for each of the error types: Date Swap (`DS`), Entity Swap (`ES`), Negated Sentences (`NSent`), Number Swap (`NS`), Pronoun Swap (`PS`).

	AggreFact	**DialSummEval**	
Model Name	**%BAcc.**	**%BAcc.**	**Corr.**
DAE	**76.0**	56.2	0.44
SummaC	71.6	62.7	0.35
QAFactEval	73.9	64.4	**0.59**
Cohere-cmd-XL	63.1	56.6	0.36
Claude V1.3	50.6	56.8	0.30
Bard	62.7	59.5	0.26
PaLM2-Bison	57.0	55.6	0.57
Dav001	53.3	52.9	0.11
Dav002	54.3	59.2	0.49
Vicuna-13b	60.3	58.6	0.36
Dav003	64.8	60.9	0.51
GPT3.5-turbo	70.2	62.0	0.56
GPT-4	73.6	**68.4**	0.58

Table 5: Performance of models on the AggreFact, DialSummEval consistency benchmarks reported in balanced accuracy (**%Bacc.**) and correlation (**corr.**).

showed that summaries from older models were less relevant to the field of consistency detection.

Table 1 reports the balanced accuracy of specialized models and LLMs on AggreFact. At first glance, the specialized models still outperform LLMs, even though increasing LLM size leads to performance improvements and helps close the gap, with GPT-4 performing within 2.4% points of the specialized DAE. However, all models perform relatively poorly, with no model reaching a balanced accuracy of 80% on a binary classification task.

To inspect performance on the AggreFact benchmark further, we conducted a manual annotation similar to the one conducted in FactCC Section 3.5 but focused on cases where GPT4 disagrees with the label of AggreFact. More precisely, we manually inspected the explanations provided by GPT4 for the 101 summaries it judged were inconsistent but labeled as consistent in the dataset.

Of the 101 samples, 80 were labeled by the annotator as correct or partially correct explanations that identify and explain a factual inconsistency in the summary. In other words, this manual analysis of a subset of AggreFact reveals that **a minimum of 6% of the samples in AggreFact are mislabeled.** The low reliability of labels in crowd-sourced benchmarks like AggreFact is a known issue (Pagnoni et al., 2021) stemming from task complexity that requires the annotator to carefully read and understand an entire document and accompanying summary, leading to low repeatability and inter-annotator agreement.

This methodology reveals the potential for LLMs as part of dataset creation. In some cases, an LLM explanation that is verifiable – such as an explanation for an identified factual inconsistency – can accelerate and improve the quality of annotation. We note however that LLM explanations are only valuable for a subset of the samples. For example, in cases where the model asserts a summary is consistent, manual verification is still required to assure quality. In Section 6, we explore a new protocol for factual consistency benchmark creation which can involve an LLM.

Based on the low reliability of labels in AggreFact, we note that a key requirement for future benchmarks is to improve label reliability, which can be demonstrated with high annotator agreement when multiple annotators are involved.

	Average Annotator Likert Score							
Model	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Dav001	68.1	78.4	84.6	90.2	83.6	84.9	86.0	88.9
Cohere-cmd-XL	46.2	51.0	70.3	83.6	88.6	89.2	91.7	96.3
DAE	30.8	56.9	63.7	83.6	86.8	94.3	90.3	94.2
PaLM2-bison	25.3	35.3	56.0	78.7	93.6	97.2	98.4	95.8
Dav002	13.2	29.4	47.3	62.3	77.7	83.0	88.3	90.0
Dav003	4.4	17.6	28.6	31.1	63.2	69.3	84.9	81.6
GPT3.5-turbo	8.8	15.7	29.7	45.9	73.6	76.4	88.5	90.0
GPT4	2.2	5.9	6.6	24.6	45.9	54.2	80.9	87.9
QAFactEval	3.3	5.9	17.6	24.6	44.5	54.7	70.3	74.7
Vicuna-13b	8.8	15.7	17.6	37.7	50.9	54.2	65.5	66.8
SummaC	4.4	5.9	20.9	21.3	27.7	40.1	43.7	58.9
Claude V1.3	1.1	9.8	11.0	13.1	33.6	37.3	47.1	45.8
Bard	9.9	7.8	5.5	9.8	18.2	21.2	36.5	42.6

Table 6: Percent of summaries classified as consistent in DialSummEval, bucketed by average Likert consistency score. Models are more uncertain in mid-range borderline buckets ([2.0, 4.0]).

4.3 DialSummEval

The DialSummEval (Gao and Wan, 2022) benchmark is a summarization evaluation benchmark created following the format of SummEval (Fabbri et al., 2021) for the domain of dialogue summarization. In DialSummEval, each `(dialogue, summary)` tuple is evaluated by three annotators, each assigning a Likert score (1-5) assessing the consistency of the summary. The authors of the benchmark report an agreement level of 0.67 Krippendorff's alpha on the labels, indicating a moderate amount of agreement among annotators.

We evaluate model performance in two ways: (1) direct correlation between model predictions and the average annotator score, and (2) we follow Laban et al. (2022a)'s procedure to transform the benchmark into a binary classification task, amenable to the balanced accuracy metric. Results are summarized in Table 5.

Echoing results on AggreFact, increasing model size leads to a minor improvement in performance both in balanced accuracy and correlation, but most LLMs still underperform specialized methods. In absolute terms, all methods struggle to achieve strong performance on the benchmark, with accuracies all below 70%.

In Figure 6, we aggregate model predictions into 0.5-width buckets on the Likert scale. We find that most models achieve strong performance on non-borderline buckets ([1.0, 1.5), [1.5, 2.0], [4.0, 4.5], [4.5, 5.0]), assigning a vast majority of samples to the correct class (inconsistent for low buckets, consistent for high buckets). The borderline buckets ([2.0, 4.0]) however are less clear-cut: most models assign large proportions of samples from each bucket into consistent and inconsistent classes.

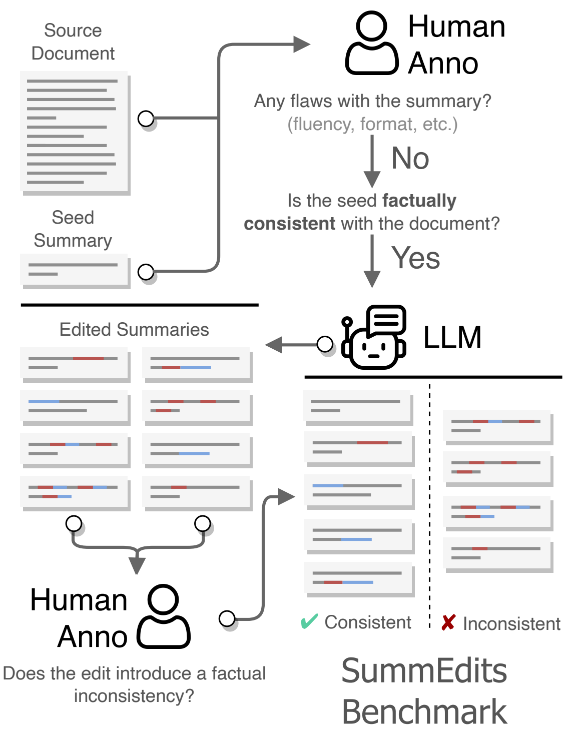

Source Document
Human Anno
Any flaws with the summary? (fluency, format, etc.)
No
Is the seed factually consistent with the document?
Yes
Seed Summary
Edited Summaries
LLM
Human Anno
Does the edit introduce a factual inconsistency?
✓ Consistent
✘ Inconsistent
SummEdits Benchmark

Figure 3: SUMMEDITS protocol diagram, a three-step protocol to create summarization ID benchmarks. See Table 7 for example samples produced by the protocol.

We argue that **annotating the consistency of summaries using a Likert scale limits the quality and interpretability of the benchmark**, as it is not evident to interpret the differences between scores, limiting reproducibility, which is reflected in the moderate Kripendorff's alpha. Instead, we favor framing factual consistency benchmarks as a *detection task*. In the detection task, identifying any factual inconsistency between the document and summary leads to an overall assessment of the summary being *inconsistent*. If no inconsistency is detected, the summary is *consistent*. The detection framing also allows for models to provide natural language explanations when identifying a summary as inconsistent, which can be manually verified to confirm model reasoning ability, and model failure modes, as done in Section 3.5.

In the next section, we propose a novel protocol to create factual consistency benchmarks, incorporating lessons learned from existing benchmarks.

Edited Summary Labeled As **Consistent**	Edited Summary Labeled As **Inconsistent**
The characters discuss ponder the consequences of banishing Marcius, with Cominius warning that his alliance collaboration with the Volscians will bring great danger to Rome.	The characters discuss the consequences of banishing Marcius, with Cominius warning that his alliance with the Volscians Romans will bring great danger to Rome the Volscians. – `Entity Manipulation`
We introduced a novel new, simple, and efficient data augmentation method that boosts improves the performances of existing GANs when training data is limited and diverse.	We introduced a novel, simple, and efficient data augmentation method that boosts the performances of existing GANs when training data is limited abundant and diverse. – `Antonym Swap`
Employees of the European Commission are now forced instructed to delete remove TikTok from their work devices, and delete get rid of it from their personal devices too if they have work-related apps applications installed.	Employees of the European Commission are now forced not required to delete TikTok from their work devices, and delete but should still remove it from their personal devices too if they have work-related apps installed. – `Hallucinated Fact`
A conversation between a sales agent and a potential client possible customer. The sales agent provides information on different home insurance plans options and pricing, as well as available discounts for clients with good credit scores and other factors.	A conversation between a sales agent and a potential client. The sales agent provides information on different home insurance plans and, but not on pricing, as well as or available discounts for clients with good credit scores and other factors. – `Negation Insertion`

Table 7: Example edit summaries – deletions, insertions – for four domains of SUMMEDITS (top-to-bottom: Shakespeare Plays, SciTLDR, News, Sales Call). Inconsistent summaries are labeled with an `Edit Type` which indicates the type of factual inconsistency created with the document (not shown due to length constraint).

5 SUMMEDITS Protocol

5.1 Design Principles

Based on the analysis of previous benchmarks, we set several design principles that can help create higher quality factual consistency benchmark:

P1. **Binary Classification Task:** In the benchmark, a summary should either be labeled as inconsistent if any factual inconsistency is identified with the document or consistent otherwise, to improve label interpretability.

P2. **Focus on Factual Consistency:** Summaries in the benchmark should be flawless on aspects unrelated to consistency, such as fluency, coherence, and formatting, to avoid confounding effects on the quality of the benchmark.

P3. **Reproducibility:** Benchmark labels should not depend on annotator identity, and high annotator agreement should confirm the validity of the benchmark, as well as estimate human performance on the benchmark.

P4. **Benchmark Diversity:** Inconsistency errors in the benchmark should represent a wide range of errors in realistic textual domains, to increase understanding of model strengths and weaknesses, and better establish gaps in performance between models and human annotators at factual reasoning, if there are any.

5.2 Creation Procedure

We now describe the creation procedure we design for the SUMMEDITS benchmark with an objective to satisfy the design principles stated above, the procedure is visually introduced in Figure 3.

At a high level, the procedure consists of three steps: (1) seed summary verification, (2) generation of summary edits, and (3) annotation of edited summaries.

Seed Summary Verification. Benchmark creators select a small collection of documents in a domain of choice, and a *seed summary* is collected for each document, which can either be human-written or model generated. An annotator answers two questions about each `(document, seed summary)` tuple: (a) "Are there any flaws with the summary? (fluency, format, etc.)", (b) "Is the summary factually consistent with the document?". If the annotator identifies a flaw in the summary (e.g., an incomplete or disfluent sentence), or any inconsistency, the tuple is filtered out (**P2**), otherwise, it proceeds to Step 2.

Generation of Edited Summaries. Once a seed summary has been verified, the second step consists in generating multiple *minor edits* to the summary, which might or might not affect the consistency of the summary. This procedure can be carried out manually, or automatically with an LLM. Proposed edits should be atomic and localized, not entirely rewriting a novel summary. Example edits of summaries are shown in Table 7.

Annotation of Edited Summaries. The annotator who completed the seed verification task (Step 1) is tasked with reviewing each edited summary and assigning it with one of three labels: (a) *consistent* if an edit does not lead to an inconsistency in the summary, (b) *inconsistent* if the edit modifies the seed summary in a way that introduces a factual inconsistency, (c) *borderline* for any other case such as the edit making the summary unclear, or the edit requiring subjective interpretation.

Crucially, we note that a single annotator should complete both Steps 1 and 3, as once they have invested the time in reading the `(document, summary seed)` tuple, the time required to judge the consistency of edits is greatly reduced. We also recommend including a large number of edits (e.g., 30 edits) to maximize edit diversity (**P4**), and encouraging annotators to assign the borderline label if they are unsure about any aspect of an edit, in order to maximize reproducibility (**P3**).

A benchmark can be formed by retaining edited summaries that are labeled as consistent and inconsistent and filtering out borderline cases.

We note that this procedure only requires a small number of documents and seed summaries, as each seed summary is derived into many edited summaries. This flexibility facilitates the creation of factual consistency benchmarks in application domains that lack such resources, such as legal (Kornilova and Eidelman, 2019) or podcast summarization (Clifton et al., 2020).

6 SUMMEDITS Benchmark

6.1 Benchmark Creation

We implemented the SUMMEDITS protocol on ten realistic summarization domains to explore the reliability of the protocol. For five domains, seed summaries are automatically generated due to the lack or low quality of existing reference summaries. In such cases, we used ChatGPT and domain-specific prompts to generate seed summaries. We note that for all domains, the quality of seed summaries is ultimately manually confirmed in step 1 of the protocol, which consists of ensuring seed summaries are factually consistent and flawless in terms of fluency, formatting, etc.

Domain	**N**	**%Balance**	**IAA**
News	819	39.2%	0.91
Podcast	500	32.6%	0.91
Billsum	853	42.3%	0.90
Samsum	664	36.4%	0.90
Shakespeare	814	46.4%	0.96
SciTLDR	466	31.1%	0.93
QMSum	431	42.5%	0.92
ECTSum	668	38.0%	0.96
Sales Email	613	29.2%	0.87
Sales Call	520	33.3%	0.93
Overall	6,348	37.10%	0.92

Table 8: Statistics of the ten domains included in the SUMMEDITS benchmark, including the number of samples (**N**), the percentage of consistent summaries (**%Balance**), and the inter-annotator agreement (**IAA**).

For all domains, we use GPT3.5-turbo as the LLM to produce edited summaries[3]. The model chosen to produce summary edits has an important impact on the benchmark. We experimented with integrating multiple LLMs in the edit generation process, but preliminary results indicated that many LLMs were not successful at generating minorly edited summaries and often attempted to write entirely novel summaries, which led us to use ChatGPT as the single model to generate edited summaries. This choice is discussed further in Section 7.

We hired two professional annotators, who were compensated at a rate of \$20/hour to perform steps 1 and 3 of the protocol. Three authors of the paper also participated in the annotation for quality control purposes. Appendix C has further detail on annotation protocol, and an overview of the annotation interface, which ensured that each annotator completed Task 1 and 3 sequentially for any sample in the benchmark. We next introduce the ten domains included in the SUMMEDITS benchmark.

News To avoid selecting documents and summaries that are in the training corpora of evaluated models, we follow prior work (Goyal et al., 2022) and select `(document, summary)` tuples from recent news articles. We obtained news articles from the Google News top events feed in February 2023, selecting at most one sample per news source to increase coverage diversity (Laban

[3]The prompts we use are listed in our open-source release.

Model	Podcast	BillSum	SAMSum	News	Sales C	Sales E	Shkspr	SciTLDR	QMSum	ECTSum	Overall (↓)
DAE	54.9	55.1	59.5	61.7	50.8	55.0	54.5	55.2	52.0	58.6	55.7
SummaC	58.5	55.7	54.7	62.1	59.0	57.7	59.3	59.7	56.6	64.4	58.8
QAFactEval	64.0	54.4	66.3	74.6	68.5	64.2	61.9	67.5	62.4	72.9	65.7
Dav001	53.3	50.2	51.0	54.4	55.3	52.5	50.0	51.0	50.3	50.9	51.9
Cohere-cmd-XL	51.1	52.7	52.0	52.6	60.3	59.5	50.0	60.5	53.9	60.5	55.1
Vicuna-13b	52.8	52.6	50.8	63.0	58.1	51.8	55.5	59.7	54.0	62.5	56.1
Claude v1.3	59.9	52.1	64.1	63.3	61.7	56.6	58.0	57.6	56.9	67.8	59.8
Dav002	56.4	53.9	57.1	61.9	65.1	59.1	56.6	64.6	60.6	66.2	60.1
Bard	50.0	58.3	61.3	72.8	73.8	69.0	58.4	66.1	53.9	73.1	63.7
PaLM2-bison	66.0	62.0	69.0	68.4	74.5	68.1	61.6	78.1	70.2	72.3	69.0
Dav003	65.7	59.9	67.5	71.2	78.8	69.4	69.6	74.4	72.2	77.9	70.7
GPT3.5-turbo	68.4	63.6	69.1	74.5	79.7	65.5	68.1	75.6	69.2	78.9	71.3
GPT4	83.3	71.1	82.9	83.3	87.6	80.1	84.6	82.4	80.4	88.0	82.4
GPT4 Oracle	90.2	85.5	86.3	88.3	91.1	83.5	96.6	86.3	89.9	91.7	88.9
Human Perf.	90.8	87.5	89.4	90.0	91.8	87.4	96.9	89.3	90.7	95.4	90.9

Table 9: Balanced accuracy of models on the SUMMEDITS benchmark. The top three models are non-LLM specialized models, the middle section are LLMs. We also report a GPT4 oracle performance and an estimate of human performance.

et al., 2023). Seed summaries are extracted from the article's metadata.

Podcast (Clifton et al., 2020) We collected 40 podcast transcripts from the unreleased test set of Spotify's podcast summarization dataset. Due to low reference summary quality, we generated seed summaries automatically.

BillSum (Kornilova and Eidelman, 2019) We collected 40 US bills and their accompanying summaries as seeds from the training portion of BillSum, a challenging dataset for summarization in the legal domain.

SamSum (Gliwa et al., 2019) We collected 40 dialogues and their accompanying summaries from the training portion of SamSum, a common dialogue summarization dataset for messenger-like conversations.

Shakespeare (Karpathy, 2015) We collected 40 scenes from Shakespeare plays from the Tiny Shakespeare corpus, each roughly 700 words long. We generated seed summaries automatically.

SciTLDR (Cachola et al., 2020) We collected 40 research paper abstracts and their corresponding TLDRs from the training portion of SciTLDR, a dataset for scientific paper summarization.

QMSum (Zhong et al., 2021) We collected 40 document and seed summaries from QMSum, a dataset for query-based meeting summarization.

ECTSum (Mukherjee et al., 2022) We collected 40 documents from the ECTSum dataset, a summarization dataset for the financial earnings call transcripts. Due to low reference summary quality, we generated seed summaries automatically.

Sales Call & Email We generated fictional sales call transcripts and sales emails – 40 for each – and corresponding seed summaries using ChatGPT. This subset of the benchmark evaluates the protocol's validity with textual data entirely machine-generated in targeted domains that lack pre-existing summarization datasets.

6.2 SUMMEDITS Statistics

Table 8 provides statistics of the finalized SUMMEDITS benchmark. Each domain yielded between 400-900 edited summaries, depending on the fraction of seed summaries that pass the first step validation (58% overall pass rate) and the percentage of edited summaries that are annotated as borderline and filtered out (around 6%). In the five domains where seed summaries were generated by ChatGPT, 17.8% of the seed summaries were labeled as factually inconsistent, indicating that modern LLMs like ChatGPT still struggle to remain consistent when summarizing documents.

At least 20% of each domain's samples were annotated by multiple annotators, allowing us to measure the agreement level when completing the annotation. When considering all three labels (consistent, inconsistent, borderline), Cohen's Kappa in each domain varies between 0.72-0.90, averaging 0.82. When removing samples annotated as borderline by any annotator, the average Cohen's Kappa rises to 0.92, **empirically validating the importance of labeling and filtering out borderline samples to create a reproducible benchmark.**

In the final benchmark, 37% of summaries are consistent, and the rest are inconsistent, approaching our objective of a balanced benchmark to facilitate robust evaluation and minimize metric fluctuations (Luque et al., 2019).

The total annotation cost of SUMMEDITS is around USD 3,000, representing around 150 hours of annotator work. The average cost of adding a domain to SUMMEDITS is therefore around USD 300, within reach for NLP practitioners looking to evaluate the model's ability to detect factual errors in their domain of choice. Authors of the FRANK benchmark (Pagnoni et al., 2021) – samples of which are in AggreFact – estimate that each sample in their benchmark required 30 minutes of annotator time. At similar annotator pay, annotating a benchmark for a new domain similar to the ones in SummEdits would cost an estimated USD 6,000: twenty times more. This cost analysis reveals the dual advantage of our protocol: by focusing the annotation task on atomic edits, costs can be drastically reduced while maintaining high reproducibility.

6.3 SUMMEDITS Results

Table 9 reports the average performance of specialized models, LLMs with a zero-shot prompt, an oracle version for the LLM in which it has access to additional information and an estimate of human performance computed on the subset of the benchmark which was plurally annotated.

Overall, model performance on the benchmark is low, with a single model – GPT4 – getting within 10% of human performance. Larger or more recent LLMs perform better on the benchmark, illustrated by the performance of models in the OpenAI family, with each model generation leading to an improvement in performance and confirming that the SUMMEDITS benchmark assesses model ability at factual reasoning.

PaLM2-Bison, Dav003, ChatGPT, and GPT4 are the only four LLMs that outperform the best non-LLM approach QAFactEval, **providing evidence that most LLMs are not yet capable to reason out-of-the-box about the consistency of facts**.

All three specialized models achieve their highest performance in the news domain, unlike LLM models. The specialized models are likely calibrated to the news domain, which they are most frequently tested on (Goyal and Durrett, 2020; Laban et al., 2022a; Tang et al., 2022; Fabbri et al., 2022; **?**). This finding confirms the importance of creating multi-domain benchmarks to measure model ability in diverse and realistic scenarios.

Some domains such as Shakespeare's plays or the legal BillSum are more challenging to the majority of models, with the latter seeing no model score higher than 71.1%. Yet, factual reasoning in the legal domain is an important application area of NLP (Chalkidis et al., 2020; Shen et al., 2022).

To assess the feasibility of the benchmark, we experiment with an oracle setting of the benchmark, in which the model is provided the seed summary in addition to the input document and the seed summary. The seed summary serves as an information scaffold, enabling the model to directly assess the modifications between the seed and edited summaries when assessing factual consistency. The oracle setting leads to a large boost in performance for the GPT4 model across domains, with the model performing within 2% of human performance. The GPT4 oracle experiment confirms that high model performance on the benchmark is attainable and that the challenge of SUMMEDITS lies in aligning the facts of the edited summary with the document, without knowing that it has been edited.

6.4 Edit Type Analysis

To gain more specific insights into the types of edits present in SUMMEDITS, we annotated each inconsistent sample in the benchmark with tags of edit types that lead to factual inconsistency.

The four types are: (1) `Entity Modification` in which an entity or phrase in the summary has been changed in a way that alters the meaning, (2) `Antonym Swap` is when a word or phrase is replaced by a word of opposite meaning (e.g., increasing vs. decreasing), (3) `hallucinated fact insertion` is a novel fact is introduced in the summary which is not supported by the document, and (4) `negation insertion` is the use of any negator word (e.g., not, neither) which modifies summary meaning. Figure 7 provides an example of each edit type in SUMMEDITS.

To annotate the entire benchmark, one author of the paper first manually annotated 200 samples

Model	Inconsistent Edit Type			
	EntMod	Anto	Hallu	Neg
DAE	52.0	53.0	52.9	53.9
SummaC	56.8	56.8	55.3	57.3
QAFactEval	61.4	65.0	64.3	70.4
Dav001	50.0	50.9	50.8	53.7
Cohere-cmd-XL	53.7	55.8	55.5	63.8
Vicuna-13b	55.2	57.1	56.2	61.0
Claude v1.3	58.8	60.3	61.5	66.7
Dav002	58.3	61.4	62.4	72.0
Bard	63.2	65.3	65.6	71.3
PaLM2-Bison	67.0	70.0	71.7	80.3
Dav003	69.2	71.1	76.3	83.3
GPT3.5-turbo	70.7	70.6	74.2	79.7
GPT4	82.2	81.3	87.0	92.7
Average	61.4	62.9	64.1	69.7

Table 10: Balanced accuracy of models on the SUMMEDITS benchmark, broken down by type of factual error: Entity Modification (`EntMod`), Antonyms (`Anto`), Hallucination (`Hallu`) and Negation (`Neg`) insertion.

Model	#Distinct Edit Types			
	1	2	3	4
DAE	50.2	53.5	55.4	64.9
SummaC	58.2	56.3	57.6	67.3
QAFactEval	59.4	63.7	72.3	76.5
Dav001	50.0	50.5	53.9	63.1
Vicuna-13b	52.8	57.0	60.2	58.5
Cohere-cmd-XL	50.0	55.9	63.7	70.0
Claude v1.3	57.5	60.6	65.4	64.3
Dav002	56.3	61.2	69.4	81.7
Bard	61.0	64.9	72.4	73.4
PaLM2-Bison	66.1	69.5	79.6	69.4
ChatGPT	68.5	71.4	82.0	86.6
Dav003	65.3	72.0	85.8	88.8
GPT4	81.0	83.0	92.0	94.3
Average	59.2	62.5	69.2	74.1

Table 11: Relationship between the number of edits types in the summary and balanced accuracy of models on SUMMEDITS. Models generally perform better as the number of introduced edits in a summary increases.

of the dataset, which was used to evaluate several GPT4-based Zero-Shot and Few-Shot approaches. The best approach was then used to annotate each edited summary with edit types.

The best-performing prompt provides the definition of each edit type and a canonical example of each, and it achieved a performance of 0.85 F-1 and 0.92 recall, which was deemed sufficient for analysis purposes.[4]

Overall in SUMMEDITS, 78% of inconsistent summaries contain an entity modification, 48% an antonym swap, 22% a hallucinated fact insertion, and 18% a negator insertion. We note that the distribution of edit types is highly influenced by the LLM used to produce the edits, which is ChatGPT in our case. Table 10 presents model performance across each of the edit types.

All models detect inconsistencies due to negator insertions the best, a sign that such errors are more discernable to models. Fact hallucinations are relatively harder to detect for non-LLM models but gradually become more evident to more performant LLMs. Finally, the entity modification and antonym error types generally see the lowest rate of detection by models across the board, perhaps due to such edits modifying an existing consistent fact in a more nuanced way.

[4]We provide the prompt with the code release.

6.5 Number of Edits Effect

It is common for the LLM to introduce multiple edits in each of its candidate summaries, as can be seen in the examples in Table 7, in which each edited summary contains multiple inserted and deleted words. We group inconsistent summaries by the number of distinct edit types they contain (1 to 4) and compute model performance on each group, with results summarized in Table 11.

As the number of edit types in a summary increases, most models see sizable performance improvements, with average performance increasing from 59.2 to 74.1 between summaries with 1 or 4 edit types represented.

This analysis confirms the perspective the task in the SUMMEDITS benchmark corresponds to a *detection* task: as the number of introduced errors increases, model performance increases as there is generally more evidence of inconsistencies for the models to detect. This also points in the direction of a more challenging explanation analysis, in which one could annotate whether a model can detect all inconsistencies in a summary.

In turn, future work looking to create more challenging versions of benchmarks using the SummEdits protocol can focus on editing summaries with a single edit type, as such inconsistent summaries are more challenging to detect.

7 Limitations and Discussion

Why not fix existing benchmarks? In Section 5, analysis reveals limitations with existing benchmarks that in theory can be fixed to yield improved versions of known benchmarks. The analysis we performed however only helps us invalidate a subset of samples in an opportunistic way, by looking at samples where benchmark labels and GPT4 disagree. However, this methodology cannot help us efficiently correct or confirm all samples, and improving existing benchmarks would require re-annotating a large portion of the benchmarks, and we do not have a guarantee that new annotations would improve on previous ones. By designing a new protocol for sample annotation that relies on clear, atomic edits, we simplify the annotation process, improving reproducibility.

Effect of LLM in benchmark creation. Step 2 of the protocol described in Section 5 relies on an LLM to generate many edits of the seed summary, which are subsequently manually annotated and included in the benchmark. The choice of LLM likely has an effect on the benchmark which could favor a subset of LLMs most similar to the one used for benchmark creation. Initial attempts to use a pool of LLMs to produce edits were unsuccessful as we found that only ChatGPT and GPT4 were currently capable of following editing instructions that do not fully rewrite summaries. Future iterations on similar benchmarks should consider including diverse pools of LLMs in benchmark creation processes to avoid model-specific bias.

Evalutating Summarizers. Previous benchmarks were in part collected to evaluate which summarization models are least likely to generate factual inconsistencies (Falke et al., 2019). Since the summaries in SUMMEDITS are synthetic modifications of summaries, the benchmark cannot directly provide insights on summarizers and their ability to remain consistent. Future work can explore using methods such as Near-Negative Distinction (NND) (Laban et al., 2022b) to adapt SUMMEDITS into a set of tests to evaluate summarizer performance, and model ability to avoid generating inconsistent samples in the first place.

Build Your Own Benchmark. By implementing the protocol in ten diverse domains for an average cost of around USD300 per domain, we've demonstrated that the protocol can be adapted to widely different textual domains – from US legal bills to Shakespeare plays – and produce domain-specific benchmarks. Although we hope that the domains we've selected span a range of practical use cases, we hope that others will adopt and adapt the protocol to new domains, languages, and NLP tasks.

8 Conclusion

In this work, we explore the capabilities of LLMs to act as factual reasoners through the lens of factual evaluation in text summarization. We show that on a surface level, LLMs perform on par with specialized non-LLM evaluators, but the performance substantially degrades in more advanced evaluation settings. As part of this analysis, we also uncover and discuss shortcomings of existing benchmarks for factual evaluation. Using those insights we develop a new protocol for creating inconsistency detection benchmarks, which we implement in a 10-domain benchmark called SUMMEDITS. The SUMMEDITS benchmark is highly reproducible and more cost-effective per sample than previous benchmarks. Our experiments show that the benchmark is challenging for most current LLMs, with the best-performing model, GPT-4, still 8% below estimated human performance. We believe that SUMMEDITS can serve as a valuable tool for evaluating LLMs' abilities to reason about facts, detect factual errors and promote more reliable NLG systems. We encourage LLM developers to report their performance on the benchmark, and practitioners to adapt the protocol to generate in-domain benchmarks for model evaluation.

References

Negar Arabzadeh, Ali Ahmadvand, Julia Kiseleva, Yang Liu, Ahmed Hassan Awadallah, Ming Zhong, and Milad Shokouhi. 2022. Preme: Preference-based meeting exploration through an interactive questionnaire. *arXiv preprint arXiv:2205.02370*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind

Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S Weld. 2020. Tldr: Extreme summarization of scientific documents. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4766–4777.

Shuyang Cao and Lu Wang. 2021. Cliff: Contrastive learning for improving faithfulness and factuality in abstractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6633–6649.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *ArXiv*, abs/2010.02559.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Ann Clifton, Aasish Pappu, Sravana Reddy, Yongze Yu, Jussi Karlgren, Ben Carterette, and Rosie Jones. 2020. The spotify podcast dataset. *arXiv preprint arXiv:2004.04270*.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Alexander Richard Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. Qafacteval: Improved qa-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601.

Tobias Falke, Leonardo FR Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220.

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

Mingqi Gao and Xiaojun Wan. 2022. Dialsummeval: Revisiting summarization evaluation for dialogues. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5693–5709.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, page 70.

Tanya Goyal and Greg Durrett. 2020. Evaluating factuality in generation with dependency-level entailment.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*.

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 446–469.

Raghav Jain, Anubhav Jangra, Sriparna Saha, and Adam Jatowt. 2022. A survey on medical document summarization. *ArXiv*, abs/2212.01669.

Andrej Karpathy. 2015. char-rnn. `https://github.com/karpathy/char-rnn`.

Anastassia Kornilova and Vladimir Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56.

Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346.

Philippe Laban, Tobias Schnabel, Paul N Bennett, and Marti A Hearst. 2022a. Summac: Re-visiting nli-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Philippe Laban, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022b. Near-negative distinction: Giving a second life to human evaluation datasets. In *Conference on Empirical Methods in Natural Language Processing*.

Philippe Laban, Chien-Sheng Wu, Lidiya Murakhovs' Ka, Xiang 'Anthony' Chen, and Caiming Xiong. 2023. Designing and evaluating interfaces that highlight news coverage diversity using discord questions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context?

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

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

Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de Las Heras. 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

Rajdeep Mukherjee, Abhinav Bohra, Akash Banerjee, Soumya Sharma, Manjunath Hegde, Afreen Shaikh, Shivani Shrivastava, Koustuv Dasgupta, Niloy Ganguly, Saptarshi Ghosh, et al. 2022. Ectsum: A new benchmark dataset for bullet point summarization of long earnings call transcripts. *arXiv preprint arXiv:2210.12467*.

Sharan Narang and Aakanksha Chowdhery. 2022. Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, Alex Wang, and Patrick Gallinari. 2021. Questeval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6594–6604.

Zejiang Shen, Kyle Lo, Lauren Yu, Nathan Dahlberg, Margo Schlanger, and Doug Downey. 2022. Multilexsum: Real-world summaries of civil rights lawsuits at multiple granularities. *arXiv preprint arXiv:2206.10883*.

Liyan Tang, Tanya Goyal, Alexander R Fabbri, Philippe Laban, Jiacheng Xu, Semih Yahvuz, Wojciech Kryściński, Justin F Rousseau, and Greg Durrett. 2022. Understanding factual errors in summarization: Errors, summarizers, datasets, error detectors. *arXiv preprint arXiv:2205.12854*.

Liyan Tang, Zhaoyi Sun, Betina Idnay, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin Rousseau, et al. 2023. Evaluating large language models on medical evidence summarization. *medRxiv*, pages 2023–04.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, ly usable llms. Accessed: 2023-05-16.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

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

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*, pages 5008–5020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot in-context learning. *arXiv preprint arXiv:2205.03401*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921.

A Model Access Detail

In Section 3, we experiment with a wide range of models. For each model, we specify its model card, and how it was accessed.

Non-LLM models. The three specialized models – SummaC[5], DAE[6], and QAFactEval[7] – were implemented through their online public repositories, and run locally on a multi-GPU machine (with 2 V-100 GPUs).

Open-source Models. We experimented with five open-source LLM models: LLama-13b (Touvron et al., 2023), Alpaca-13b (Taori et al., 2023), Dolly-V2-12b (`databricks/dolly-v2-12b`), Vicuna-13b (Chiang et al., 2023), and MosaicML's MPT-7b-chat (Team, 2023). All models were accessed through the public, online demonstration of LMSys.org[8]. Model responses were collected between April 15th, 2023, and May 15th, 2023.

Google Models. We experiment with two Google models, the Bard (Thoppilan et al., 2022) which we accessed through a web-based interface[9] which does not specify an exact model card, but model responses were collected between April 15th, 2023 and May 15th, 2023. Second, the PaLM-v2-bison model (Narang and Chowdhery, 2022) (model card `text-bison@001`), which was accessed through the Google Cloud VertexAI API.

Anthropic Model. We collected outputs of the Claude V1.3 model (model card: `claude-v1.3`), the latest and largest Anthropic model at the time of publication, using the official API hosted by Anthropic[10].

Cohere Model. We collected outputs of Cohere's `command-xlarge` model, the latest and largest Cohere model at the time of publication, using the official API hosted by Cohere[11].

OpenAI Models. We collected outputs for eight OpenAI models. Six models are from the GPT-3 family: Ada001 (`text-ada-001`), Bab001 (`text-babbage-001`), Cur001 (`text-curie-001`), Dav001 (`text-davinci-001`), Dav002 (`text-davinci-002`), and Dav003 (`text-davinci-003`). We also include GT3.5-turbo (`gpt-3.5-turbo`) and GPT-4 (`gpt-4`). All models were accessed through OpenAI's official API[12].

B Explanation Annotation Guidelines

We hired two professional annotators to complete the annotation of model-generated explanations on the FactCC and AggreFact domains. The annotators were compensated at $20/hour. They received onboarding documentation that introduced them to the task, and provided the following definition for each type of explanation:

- **No Explanation**: If the model did not provide any explanation. (For example just saying: "The summary is inconsistent"),
- **Entirely Correct**: if the explanation correctly identifies and explains one or more factual inconsistencies in the summary,
- **Partially Correct**: if the explanation provided contains several elements and at least one of them correctly identifies and explains a factual inconsistency in the summary,
- **Unrelated**: if the explanation given does not directly relate to a factual inconsistency between the summary and the document,
- **Incorrect**: if the explanation given does not correctly identify a factual inconsistency in the summary, for example, making a logical error.

[5] `https://github.com/tingofurro/summac`

[6] `https://github.com/tagoyal/factuality-datasets`

[7] `https://github.com/salesforce/QAFactEval`

[8] `https://chat.lmsys.org/`

[9] `https://bard.google.com/`

[10] `https://github.com/anthropics/anthropic-sdk-python`

[11] `https://docs.cohere.com/docs/the-cohere-platform`

[12] `https://github.com/openai/opeai-python`

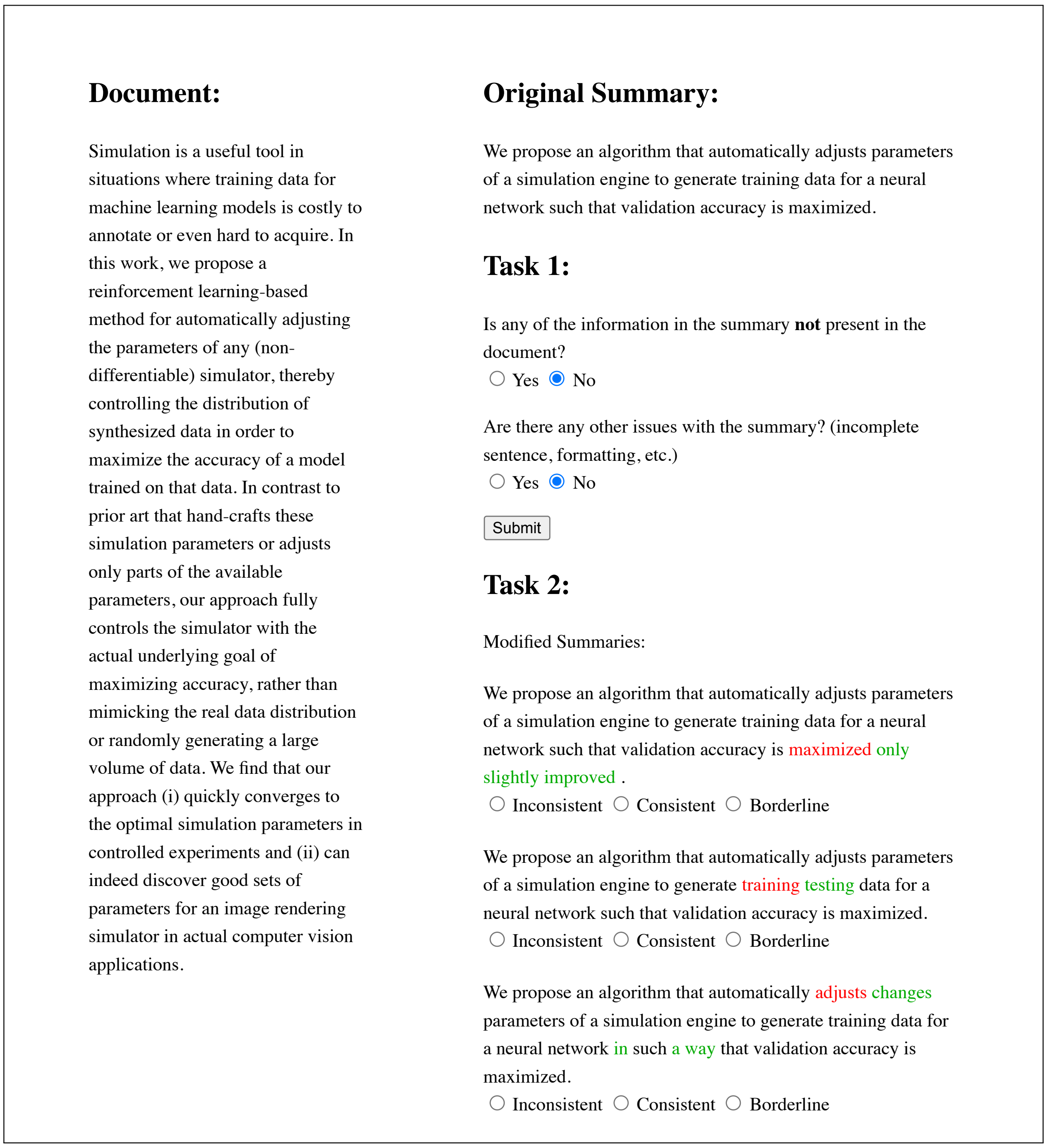
An example for each type of explanation was provided during onboarding, similar to the ones given in Table 3. In order to obtain impartial results that do not benefit or disadvantage any model, for cases where multiple explanations were annotated for the same `(document, summary)` sample, the explanations' order was shuffled, and annotators were not aware of the model origin of any explanation.

Annotation was performed in batches, and the first two batches of annotation of each annotator were reviewed by the authors of the paper. Incorrect annotations were discussed, allowing annotators to better understand edge cases of the task, and modify their annotation in the first batches. The annotators were added to a Slack channel with one of the authors and regularly discussed edge cases to maintain a common understanding of the task. For example, both annotators raised the question of how to deal with cut-off explanations, in which the last sentence is incomplete (due to the max-length of generation). Annotators were both instructed to disregard any incomplete sentence and only consider full sentences in their assessment.

C SUMMEDITS Annotation Guidelines

We hired two professional annotators to complete the annotation of Steps 1 and 3 of the SUMMEDITS protocol (see Section 5). The annotators were compensated at $20/hour. They received onboarding documentation that introduced them to the task and used the interface shown in Figure 4.

Annotators were first assigned 10 warm-up seed summaries, each with roughly 30 edited summaries, which had been pre-annotated by the authors of the paper. The authors reviewed the completed warm-up exercises, and a strong agreement level on the warm-up task with both annotators was observed. We discussed disagreement cases with the annotators and added both annotators to a Slack channel with one of the authors of the paper to allow them to discuss any edge case or domain-specific question. For example, since the QMSumm domain is the more specific query-focused summarization, the annotators were given updated instructions on Slack on how to deal with the "query" component when evaluating summaries. Namely, during Step 1 of the protocol, participants were asked to additionally judge whether the summary accurately responded to the query, and otherwise mark summaries as inadequate.


Document:

Simulation is a useful tool in situations where training data for machine learning models is costly to annotate or even hard to acquire. In this work, we propose a reinforcement learning-based method for automatically adjusting the parameters of any (non-differentiable) simulator, thereby controlling the distribution of synthesized data in order to maximize the accuracy of a model trained on that data. In contrast to prior art that hand-crafts these simulation parameters or adjusts only parts of the available parameters, our approach fully controls the simulator with the actual underlying goal of maximizing accuracy, rather than mimicking the real data distribution or randomly generating a large volume of data. We find that our approach (i) quickly converges to the optimal simulation parameters in controlled experiments and (ii) can indeed discover good sets of parameters for an image rendering simulator in actual computer vision applications.

Original Summary:

We propose an algorithm that automatically adjusts parameters of a simulation engine to generate training data for a neural network such that validation accuracy is maximized.

Task 1:

Is any of the information in the summary **not** present in the document?
Yes [✓ No]

Are there any other issues with the summary? (incomplete sentence, formatting, etc.)
Yes [✓ No]

Task 2:

Modified Summaries:

We propose an algorithm that automatically adjusts parameters of a simulation engine to generate training data for a neural network such that validation accuracy is maximized only slightly improved .
Inconsistent Consistent Borderline

We propose an algorithm that automatically adjusts parameters of a simulation engine to generate training testing data for a neural network such that validation accuracy is maximized.
Inconsistent Consistent Borderline

We propose an algorithm that automatically adjusts changes parameters of a simulation engine to generate training data for a neural network in such a way that validation accuracy is maximized.
Inconsistent Consistent Borderline

Figure 4: Two-column annotation interface used to annotate samples in the SUMMEDITS benchmark. Participants could read the document on the left-hand column. Once they completed Task 1 in the right-hand column, the second annotation task became visible.