SummExecEdit: A Factual Consistency Benchmark in Summarization with Executable Edits

Anonymous ACL submission

Abstract

Detecting factual inconsistencies in summarization is critical, yet existing benchmarks lack the necessary challenge and interpretability for robust evaluation. In this paper, we introduce SummExecEdit, a novel benchmark leveraging executable edits to assess models on their ability to both detect factual errors and provide accurate explanations. The top-performing model, Claude3-Opus, achieves a joint detection and explanation score of only 0.49 in our benchmark, with individual scores of 0.67 for detection and 0.73 for explanation. Furthermore, we identify four primary types of explanation errors, with 45.4% of errors focusing on completely unrelated parts of the summary.

1 Introduction

Large language models (LLMs) have shown remarkable performance across various tasks by generating coherent responses while facing a major challenge with factual hallucination (Huang et al., 2024). Various evaluation methods (Laban et al., 2022; Fabbri et al., 2021, 2022; Tang et al., 2023; Luo et al., 2023; Yang et al., 2024) and inconsistency detection benchmarks have been proposed, and they used approaches such as editing ground truth texts to create intentional inconsistencies (Laban et al., 2023) or generating benchmarks with LLMs where they label the factual accuracy of target texts post-generation. (Tang et al., 2024b)

However, previous approaches usually have broad and sweeping edits, i.e., trivial edits and multiple edits in one sample. Kim et al. (2024) in their work chose to entirely rely on humans to generate edits, because initial experiments with LLM-based edits yielded mostly trivial edits. Such edits make factual inconsistency errors easier to be detected by LLMs, for example, GPT4 achieves 82.4% detection accuracy on a previous inconsistency benchmark - SummEdits (Laban et al., 2023). This shows that we need mechanisms to generate more complex edits. Moreover, high quality explanations about inconsistency in these detection benchmarks are missing, and they are necessary to evaluate the model's ability to explain their detection result and reason over facts.

To address these drawbacks, we propose to leverage the concept of executable edits (Laban et al., 2024) to generate a challenging and interpretable benchmark called SummExecEdit - extending SummEdits (Laban et al., 2023) [1]. While normal editing rewrites the entire summary with edits already applied, executable editing focuses on isolating and replacing a specific substring in the text showing what has changed, allowing precise changes that can introduce factual inconsistencies. Such edits helps LLM to better concentrate and come up with a complex yet granular and controlled edit with a meaningful explanation for same in a structured way, helping to eliminate broad and sweeping multiple edits, as shown in Figure 1.

We compare executable edits to existing editing and show that executable edits are superior, providing in range of 18-25% of higher number of challenging samples with better explanations. We evaluate a wide range of open-source and open-api access LLMs on SummExecEdit and find that most of these models lack on the combined task of detection and explanation of error, with the best model - Claude3-Opus achieving a joint score of only 0.49, with individual scores of 0.67 for detection and 0.73 for explanation. We also perform error analyses for incorrect explanations generated by candidate LLMs and categorize them into four reasons, with 45.4% of errors focusing on completely unrelated parts of the summary.

2 Related Works

Editing and Annotation in Previous Benchmarks. Several previous works have assessed

[1] Our data is released at www.anonymous.com

various metrics in relation to human judgments associated with hallucination and factual inconsistencies. Works prior to advances in language models focused on human and expert level (Fabbri et al., 2021; Falke et al., 2019) while the very recent works focused on generating annotations with the help of language models (Chhikara et al., 2024; Tang et al., 2024b; Laban et al., 2023). Although these models generate coherent texts with a good level of inter-annotator agreements with respect to consistency (Laban et al., 2023), such benchmarks may struggle to keep the changes in the original data to minimal (Nguyen et al., 2024). Executable edits with help of language models was first explored by Laban et al. (2024), where it focused on replacing a substring for document editing. We build upon the works of SummEdits (Laban et al., 2023) to use executable edits to generate new factual consistency benchmark.

Detecting Factual Inconsistencies. Since automatic evaluation metrics such as ROUGE (Lin, 2004) or BLEU (Papineni et al., 2002) cannot reliably detect factual inconsistencies, several non-LLM based methods have been developed such as question-based (Fabbri et al., 2022) and entailment-based (Laban et al., 2022; Zha et al., 2023; Tang et al., 2024a). In various recent works, LLMs have shown superior performance when used as a judge to evaluate factual consistencies (Luo et al., 2023). We explore this further in various settings and also re-evaluate the modern-day LLMs on our benchmark generated using executable edits.

ORIGINAL SUMMARY – BillSum Dataset

Libyan Assets Taxpayer Reimbursement Act of 2011 authorizes the President to vest blocked Libyan assets in an amount necessary to ensure: (1) reimbursement for the cost of U.S. military and humanitarian operations undertaken in Libya, and (2) compensation for U.S. victims of Libyan-sponsored terrorism.

NON-EXECUTABLE EDIT

{"*edited_sum*": "The **2011** Libyan Assets Taxpayer Reimbursement Act **permits** the President to **seize** Libyan assets to **fund** U.S. military and humanitarian **activities in** Libya, **alongside** compensating **Libyan victims of U.S.-sponsored terrorism.**",
"*explanation*": "The direction of compensation in the summary is incorrect."}

EXECUTABLE EDIT

{"*original_text*": "**U.S.** victims of **Libyan**-sponsored terrorism",
"*replace_text*": "**Libyan** victims of **U.S.**-sponsored terrorism",
"*explanation*": "The direction of compensation in the summary is incorrect."}

Figure 1: Example comparison between non-executable and executable edit.

3 Executable Edits

Given a *(source, target)* pair with a goal to edit the target such that it becomes factually inconsistent with the source, we can either ask the LLM to rewrite the target token-by-token, or ask it to isolate and highlight only a specific substring in the target text to be replaced. The latter can also be considered a program that can be executed and used. Executable editing is simple but it can help us generate more controlled and complex edits. Moreover, this minimizes synthetic data in the benchmarks since a majority of the original data remains same. Executable edits can also have a wide usage for data augmentation.

We first experiment and verify on a smaller sample to show that executable edits work better compared to non-executable edits. We selected around 100 original (document, summary) pairs from Laban et al. (2023) and used GPT4-Turbo, Claude3 Opus, and GPT3.5-Turbo to generate edited summaries and explanation of inconsistencies using both executable and non-executable prompts (Appendix F) in a structured json format. This generates around 600 edits, which are shuffled, anonymized, and annotated by two of the authors manually based on the four questions: a) is the edit inconsistent? b) is the edit complex/good quality? c) is the edit controlled/granular? d) is the explanation quality good? Each annotator annotated nearly 400 edits with 200 in common to verify the inter-annotator agreement in Table 3.

The result for our manual annotation is given in Table 1. We use a filtering mechanism - each subsequent column filters out the edits deemed inappropriate by either of the annotators in the previous column. The models show a similar trend - executable edits lead to a higher score towards the end implying a higher number of good edits and explanations. For example, Claude3-Opus provides nearly 18% more controlled and high quality edits with executable prompt.

4 SummExecEdit Benchmark

We leverage Laban et al. (2023) to build our benchmark across 10 domains such as News, Podcast, Bill, Sales calls, etc. Based on Table 1, we use both GPT4-Turbo and Claude3-Opus models to generate our benchmark using the executable prompt. Both the models are asked to generate six edits for each (document, summary) pair.

After generating these edits, we remove the trivial edits with help of GPT4-Turbo. We ask GPT4-

Condition	N	%Controlled (↑)	%Inconsistent (↑)	%Complex (↑)	%Explanation (↑)
GPT4-Turbo (Exec)	144	86.81	81.25	45.83	44.44
GPT4-Turbo (Non-Exec)	134	90.30	81.34	23.13	20.90
GPT3.5-Turbo (Exec)	134	76.87	73.13	17.16	16.42
GPT3.5-Turbo (Non-Exec)	133	86.47	72.18	18.05	12.03
Claude3-Opus (Exec)	138	92.03	84.78	**49.28**	**48.55**
Claude3-Opus (Non-Exec)	136	**97.06**	**88.24**	31.62	30.15

Table 1: Manually annotated scores for comparison between executable and non-executable edits. Each column shows percentage of N but does not consider the edits filtered out in its previous columns.

Turbo to classify an edit as a date change, number change, antonym change, or others. Any edit classified as date, number, or antonym change is removed from the benchmark.

The final benchmark results in 2,121 factually inconsistent summaries. To balance out the entire benchmark, we use all 2,120 factually consistent edits from SummEdits benchmark resulting in a total of 4,241 samples in the final benchmark. Each of the 10 domains provide around 200-300 inconsistent summaries. The distribution of each domains is given in Table 6.

5 Results

To evaluate LLMs on SummExecEdit benchmark, we use two types of prompts. **D&E** - Detect and Explain error. Models need to detect if there is any factual inconsistency in summary, if yes, explain the inconsistency (Appendix F.3). **E|D** - Explain error given Detection. Given that the summary is inconsistent, models need to explain the inconsistency in the summary (Appendix F.4).

Table 2 provides the Detection Accuracy (DA) of all the models using prompt D&E. The best performing model on SummExecEdit , GPT4o, provides an accuracy of only around 73%. The overall results show that many LLMs struggle in detecting the factual error. As a reference, we also evaluate two non-LLM based approaches which use far lesser compute - AlignScore (Zha et al., 2023) receiving 57.4% and MiniCheck (Tang et al., 2024a) receiving 60.0% accuracy, which are better than several open-source LLMs.

5.1 Evaluating Explanations

Two of the authors manually annotated around 1200 explanations with 300 in common. The results for manual annotation are given in appendix in Table 5. Based on these annotations we evaluated different LLM-as-Judge (Zheng et al., 2023) by asking them to assign a label as Entirely Correct (1), Partially Correct (0.5), or Not Correct (0). We use following four types of prompts for explanation evaluation using GPT4o, GPT3.5-Turbo, Claude3-Opus, and Claude3-Haiku.

- Reference less and no seed (EvalV1) - We provide document, edited summary, and LLM's explanation to be evaluated.
- Reference less and with seed (EvalV2) - We provide seed summary (ground truth), edited summary, and LLM's explanation to be evaluated.
- Reference based with seed (EvalV3) - We provide seed summary (ground truth), edited summary, reference explanation, and LLM's explanation to be evaluated.
- Reference only (EvalV4) - We provide reference explanation and LLM's explanation to be evaluated.

The correlations for four prompts with different LLMs with respect to our manually annotated explanations are provided in appendix in Table 4. Selected explanations for manual annotations were shuffled and randomly selected. The model and either of the two prompts that generated those explanations were anonymized. The IAA between both the annotators are - Correlation of 0.885 and Cohen Kappa of 0.81. We think that the reference explanations generated at the time of edit itself are the best and of high quality, scoring 0.95 for 40 samples. The prompt EvalV4 works the best which suggests that evaluating explanations or reasoning of models, works better when we have access to what the edit/reason is or a reference explanation.

We use prompt EvalV4 with GPT4o model to give the **Explanation Score (ES)** of all models on 2121 inconsistent summaries generated using both our prompts - D&E and E|D. The results are provided in Table 2.

5.2 Joint Scores on SummExecEdit

We define the joint scores comprising of both detection and explanation scores on factually incon-

	D&E				E|D
Model	DA(↑)	DS(↑)	ES(↑)	JS(↑)	ES(↑)
claude3-opus	0.71	0.67	0.733	**0.491**	0.684
gemini-1.5-pro	0.728	0.728	0.665	0.484	0.628
gpt-4o	**0.733**	0.725	0.629	0.456	0.684
gpt4-turbo	0.729	0.581	0.714	0.415	**0.711**
gemini-1.5-flash	0.708	0.556	0.633	0.352	0.62
gpt3.5-turbo	0.585	0.676	0.467	0.315	0.488
command-r-plus	0.655	0.369	0.703	0.259	0.595
gemini-pro	0.625	0.432	0.499	0.216	0.456
claude3-haiku	0.622	0.54	0.387	0.209	0.473
claude3-sonnet	0.636	0.33	0.626	0.206	0.633
command-r	0.609	0.414	0.489	0.202	0.49
palm-bison	0.608	0.248	0.473	0.117	0.407
mixtral-8x7b	0.609	0.73	0.503	0.367	0.568
mistral-large	0.703	0.512	0.714	0.366	0.624
llama3-70b	0.678	0.423	0.713	0.302	0.585
llama3-8b	0.555	0.73	0.239	0.175	0.458
mistral-7b	0.541	0.107	0.735	0.079	0.488

Table 2: Scores for D&E (Prompt Detect and Explain error) and E|D (Prompt Explain error given Detection) for inconsistent summaries. DA - Detection Accuracy, DS - Detection Score, ES - Explanation Score, JS - Joint Score. ES evaluated by EvalV4 prompt using GPT4o. DA based on entire benchmark. DS, ES, and JS based only on inconsistent summaries. Last 5 models are open-source.

sistent summaries in SummExecEdit . **Detection Score (DS)** is calculated only on 2,121 factually inconsistent summaries, and we give a score of 1 if model correctly detects the summary being factually incorrect and 0 otherwise. **Joint score (JS)** is calculated by multiplying both DS and ES element-wise, and the results are presented in Table 2.

The best model Claude3-Opus achieves a JS of 0.49 which suggests that the task of detecting factual inconsistency and explaining the same is still a challenging task for most modern-day LLMs, making them incapable to reason out-of-the-box. It is worth noting the big JS gap for open-api and open-source models. At the same time, it is also good to see Mixtral-8x7b achieving the best DS. This also brings up an interesting finding - some models are good at detecting the factual errors but struggle to explain the error, and also vice-versa. Models belonging to the same family also show differing behavior, for example GPT-4o and GPT4-Turbo or Mixtral-8x7b and Mistral-Large show different trends.

5.3 Error Analysis for Explanations

We analysed 350 of our manually annotated explanations that were incorrect or partially correct and observed that most of the errors in these explanations mainly fall under the following categories. If an explanation contains multiple errors, we report the first found error in the explanation.

Misattribution of Error - This is the most common type of error, accounting for 45.4% of incorrect explanations. The explanation would focus on a completely unrelated part of the summary or the document and assign the blame on it.

Additional Unrelevant Explanation - The LLM provides the correct explanation but also continues to generate some unrelated explanation. Such explanations make 28.9% of incorrect explanations.

Concentrating on Completeness - The explanation focuses on completeness showing missing details in summary rather than focusing on factual correctness. This accounts for 15.4% of incorrect explanations.

Vague Explanation - These are either complex to understand or incomplete explanations missing out on details. They may correctly identify the error but not effectively explain it. They account for 10.3% of incorrect explanation.

While we do not find any relation of errors with specific models or specific prompts, we find a relation where different models happen to make similar errors in explanations belonging to the same document and factually inconsistent summary pairs.

6 Conclusion

In this work, we first explored the executable editing with LLMs for generating benchmarks. Through our experiments, we show its superiority in generating a challenging benchmark and use it to generate a new factual consistency detection and explanation benchmark - SummExecEdit . We evaluate various LLMs over this benchmark for both detection and explanation of factual inconsistencies. The best performing model for detection - GPT4o achieves an accuracy of 73% while Claude3-Opus performs best on combined task of detection and explanation achieving a joint score of 0.49. These scores suggest that SummExecEdit is a challenging benchmark. We explore various prompt formats that lead to better auto-evaluation by LLMs and perform error-analysis for incorrect explanations generated by LLMs. We hope researchers and practitioners use executable edits to generate and augment data, and SummExecEdit can be used by LLM Developers for evaluating LLMs' abilities to detect factual errors and reason about facts.

Limitations

While executable edits help in creating more challenging benchmark, it is important to note a few limitations of these edits and this work in general. First, generating benchmarks with executable edits require availability of ground truth *(source, target)* pairs which might not always be the case. Second, non-executable edits are equally capable of generating controlled edits but these are simple and low quality. If the end-goal is to only generate controlled edits and complexity does not matter, non-executable edits could be used as well. Last, we have only experimented using these edits for creating a single benchmark related to summarization. Further experiments in other domains, as well as for data augmentations, are required.

References

Garima Chhikara, Anurag Sharma, V. Gurucharan, Kripabandhu Ghosh, and Abhijnan Chakraborty. 2024. Lamsum: Creating extractive summaries of user generated content using llms. *Preprint*, arXiv:2406.15809.

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 R. Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. Qafacteval: Improved qa-based factual consistency evaluation for summarization. *Preprint*, arXiv:2112.08542.

Tobias Falke, Leonardo F. R. 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, Florence, Italy. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2024. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.* Just Accepted.

Yekyung Kim, Yapei Chang, Marzena Karpinska, Aparna Garimella, Varun Manjunatha, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. Fables: Evaluating faithfulness and content selection in book-length summarization. *Preprint*, arXiv:2404.01261.

Philippe Laban, Wojciech Kryscinski, Divyansh Agarwal, Alexander Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. 2023. SummEdits: Measuring LLM ability at factual reasoning through the lens of summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9662–9676, Singapore. Association for Computational Linguistics.

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

Philippe Laban, Jesse Vig, Marti Hearst, Caiming Xiong, and Chien-Sheng Wu. 2024. Beyond the chat: Executable and verifiable text-editing with llms. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–23.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for text summarization. *Preprint*, arXiv:2303.15621.

Van Bach Nguyen, Paul Youssef, Christin Seifert, and Jörg Schlötterer. 2024. LLMs for generating and evaluating counterfactuals: A comprehensive study. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14809–14824, Miami, Florida, USA. Association for Computational Linguistics.

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

Liyan Tang, Tanya Goyal, Alexander R. Fabbri, Philippe Laban, Jiacheng Xu, Semih Yavuz, Wojciech Kryściński, Justin F. Rousseau, and Greg Durrett. 2023. Understanding factual errors in summarization: Errors, summarizers, datasets, error detectors. *Preprint*, arXiv:2205.12854.

Liyan Tang, Philippe Laban, and Greg Durrett. 2024a. Minicheck: Efficient fact-checking of llms on grounding documents. *Preprint*, arXiv:2404.10774.

Liyan Tang, Igor Shalyminov, Amy Wing mei Wong, Jon Burnsky, Jake W. Vincent, Yu'an Yang, Siffi Singh, Song Feng, Hwanjun Song, Hang Su, Lijia Sun, Yi Zhang, Saab Mansour, and Kathleen McKeown. 2024b. Tofueval: Evaluating hallucinations of llms on topic-focused dialogue summarization. *Preprint*, arXiv:2402.13249.

Jiuding Yang, Hui Liu, Weidong Guo, Zhuwei Rao, Yu Xu, and Di Niu. 2024. Sifid: Reassess summary factual inconsistency detection with llm. *Preprint*, arXiv:2403.07557.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. AlignScore: Evaluating factual consistency with a unified alignment function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Model Details

OpenAI Models. We used three OpenAI models in our experiments - GPT4o, GPT4-Turbo, and GPT3.5-Turbo. All the models were accessed from the official OpenAI API.

Google Models. We used four Google models in our experiments - Gemini-1.5-Pro, Gemini-1.5-Flash, Gemini-Pro, and Palm-Bison. We used Google Cloud Platform to access Google models, which relies on the VertexAI API.

Anthropic, Cohere, Llama3, Mistral models. We used the Claude models in Anthropic family, Command-R models in Cohere family, models in Llama3 family, and models in Mistral family on AWS bedrock.

B License Details and Terms of Use

We leverage Laban et al. (2023) for raw data which was released under Apache-2.0 license, granting copyright license.

C Annotation Tool

Figure 2 provides an example of our annotation tool. The text highlighted in red, is replaced by the text highlighted in green and on the right side we perform the required annotation. This helps us annotate more reliably and easily.

D Manual Annotation and Inter-Annotator Agreement

In Section 3, we manually annotate the generated inconsistent summaries to compare between executable and non-executable edits based on following 4 questions:

Is the edit inconsistent? - We annotate if the edited summary is factually inconsistent or the replaced text still kept the edited summary consistent.

Is the edit complex/good quality? - We annotate if the factual inconsistency is complex or interesting, meaning the edits should not be trivial such as easy antonym swap, negating sentence, or numerical and date changes. Trivial edits make the benchmark easier.

Is the edit controlled/granular? - We annotate if the edit is controlled and granular or still very broad and sweeping, a drawback of previous benchmarks.

Is the explanation quality good? - We annotate if the reference explanation specifying the reason of inconsistency generated by LLM at the time of edit is correct.

Table 3 provides the Cohen Kappa as the inter-annotator agreement.

Question	N	Cohen Kappa
Inconsistent	202	0.76
Interesting	177	0.61
Controlled	156	0.6
Explanation quality	43	0.49

Table 3: Inter-annotator agreement between both the authors. We keep filtering out the examples in every row to remove the bias of non-matching annotations since the subsequent annotation of questions for every example depend on the previous questions. The Cohen for Explanation looks low but the disagreement is on a very small samples. Since Cohen Kappa is looking at classes with equal weights and due to a big imbalance, it has a larger effect.

Table 4 provides the Correlation and Balanced Accuracy for manual annotation of explanations based on EvalV1, EvalV2, EvalV3, and EvalV4 for four models under consideration.

Table 5 provides explanation scores for our manual annotations of explanation evaluation. We can see a comparable performance gap between the open-source and open-api access models.

E SummExecEdit Statistics

Table 6 provides statistics for different domains present in the DEFES benchmark.

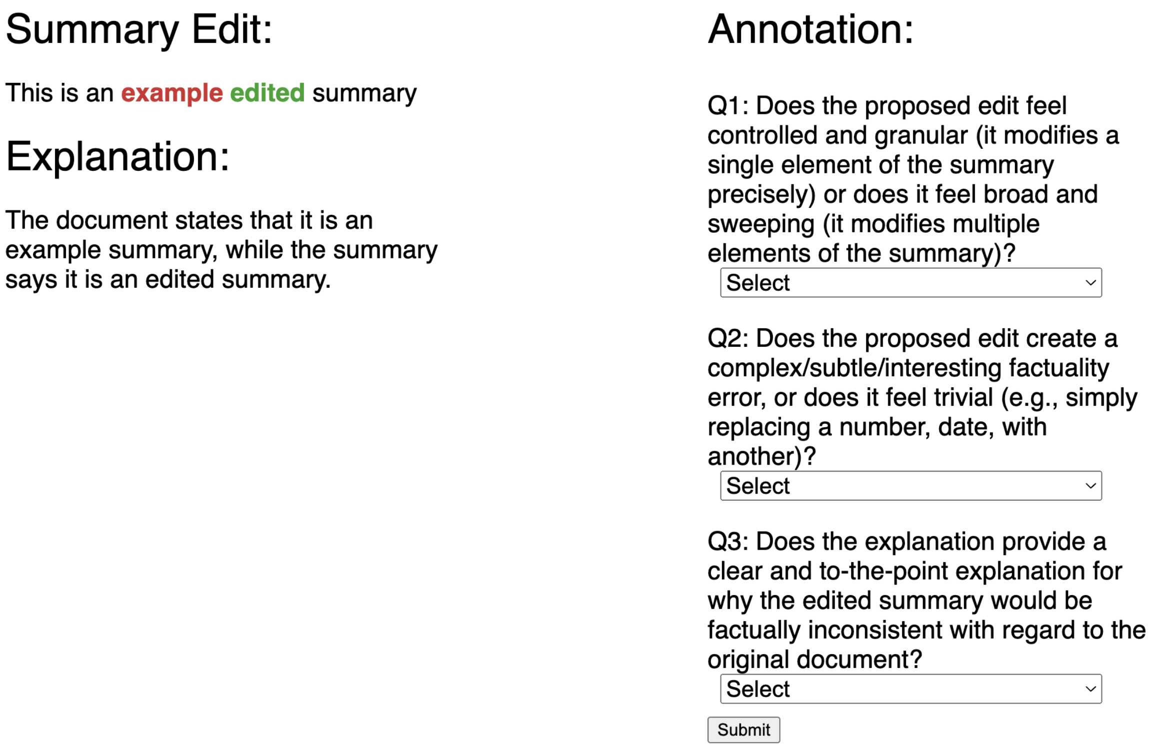

Figure 2: Two-Column annotation interface with highlights used for annotation

	Correlation				BAcc			
Model	**V1**	**V2**	**V3**	**V4**	**V1**	**V2**	**V3**	**V4**
claude3-haiku	0.076	0.478	0.732	0.725	0.385	0.49	0.655	0.618
claude3-opus	0.484	0.782	0.801	0.774	0.46	0.76	0.776	0.738
gpt3.5-turbo	0.103	0.118	0.325	0.563	0.375	0.401	0.495	0.625
gpt4o	0.603	0.816	0.804	**0.833**	0.533	0.792	0.794	**0.811**

Table 4: Correlation and Balanced Accuracy of prompts and models for explanation evaluation with respect to manual evaluation.

	Prompt V1		Prompt V2	
Model	N	ES(↑)	N	ES(↑)
claude3-opus	29	**0.897**	39	**0.884**
gpt-4o	28	0.821	39	**0.884**
gpt4-turbo	26	0.807	20	0.8
mixtral-8x7b	31	0.516	40	0.662
gpt3.5-turbo	30	0.567	40	0.7
llama3-70b	20	0.775	40	0.725
command-r-plus	17	0.706	40	0.762
gemini-pro	17	0.676	40	0.587
claude3-haiku	25	0.54	40	0.537
claude3-sonnet	19	0.763	40	0.725
command-r	15	0.6	40	0.625
llama3-8b	36	0.319	40	0.625
palm-bison	10	0.6	40	0.6
mistral-7b	6	0.833	40	0.612

Table 5: Scores for prompt V1 and V2 for inconsistent summaries. N - Number of samples, ES - Explanation Score. ES for Manual Annotation based on 887 explanations annotated by one author. The score for 40 manually annotated reference explanation is 0.95.

Domain	**N**	**%Inconsistent**
SciTLDR	307	58.31
News	567	47.8
Podcast	344	58.72
BillSum	608	44.9
SamSum	450	51.11
Shakespeare	511	41.3
QMSum	334	50.3
ECTSum	416	47.12
Sales Email	368	57.1
Sales Call	336	53.87
Total	4241	50

Table 6: Statistics of the domains in SummExecEdit

F Prompts

F.1 Executable Prompt

```
You are given a document and its corresponding summary. You must
   generate 6 edits to the summary which should strictly follow the
   guidelines mentioned below.

While making edits, think of what in a summary could be changed,
   added, or removed that is difficult to detect but makes the
   summary inconsistent with the document. Be mindful, that you
   cannot change the figures, numbers, dates, digits mentioned in the
    summary.

The edit should introduce an error which makes the summary factually
   incorrect, or inconsistent with the document. Few examples of good
    error types are given after the guidelines. Refer to those edits
   and generate similar error types. For each edit only one of the
   error type should be used.

[Guidelines]
- You can generate only one error per edited you generate.
- Your output should be a valid JSON string: that starts with : {"
   edits": [ ... ]} and contains edits in the format described below.
- Remember to not introduce following types of errors:
    - You cannot use antonyms or opposite words to introduce an error
      .
    - You cannot negate the sentence to introduce an error.
    - You cannot replace a date by some other random date or random
       time of the year to introduce an error.
    - You cannot replace some amount or money by other random amount
       or money.
    - You cannot replace a number or a percentage by some other
       random number or percentage to introduce an error.
    - You cannot replace a name by some other random name to
       introduce an error.
- Your explanation should not mention the original summary or the
   edit, it should only describe the difference between the new
   summary and document with respect to the document: "The document
   says X but the document says Y."
- Make sure your "original_text" is a substring of the summary.
- You should diversify the types of errors you generate as much as
   possible. Do not generate all edits of the same type.
- In the explanations you generated, you must refer to the edited
   summary simply as the summary. You must not mention edits or
   editing, or the "edited summary" in any way, simply call it the
   summary.

[Edit format]
Each edit should be a valid JSON object, with three keys as show: "
   original_text", "replace_text", and "explanation". The "
   original_text" key should be the original text in the summary that
```

```
    you are editing which is a substring of the summary. The "
   replace_text" key should be the new text that you are replacing
   the original text with. The "explanation" key should be a 1-2
   sentence explanation of why the edit is an error.

[Document]
[DOCUMENT]

[Summary]
[SUMMARY]
```

F.2 Non-Executable/Normal Prompt

```
You are given a document and its corresponding summary. You must
   generate 6 edits to the summary which should strictly follow the
   guidelines mentioned below.

While making edits, think of what in a summary could be changed,
   added, or removed that is difficult to detect but makes the
   summary inconsistent with the document. Be mindful, that you
   cannot change the figures, numbers, dates, digits mentioned in the
    summary.

The edit should introduce an error that makes the summary factually
   incorrect, or inconsistent with the document.

[Guidelines]
- You can generate only one error per edited you generate.
- You must not rewrite an entirely new summary. Your edited summary
   should modify, insert or delete AT MOST 5 words of the original
   summary.
- Remember to not introduce following types of errors:
    - You cannot use antonyms or opposite words to introduce an error
       .
    - You cannot negate the sentence to introduce an error.
    - You cannot replace a date by some other random date or random
       time of the year to introduce an error.
    - You cannot replace some amount or money by other random amount
       or money.
    - You cannot replace a number or a percentage by some other
       random number or percentage to introduce an error.
    - You cannot replace a name by some other random name to
       introduce an error.
- You should diversify the types of errors you generate as much as
   possible. Do not generate all edits of the same type.
- You do not need to mention the type of edit that you made.
- For each of the three edits, rewrite the entire summary with the
   edit you make.
- In the explanations you generated, you must refer to the edited
   summary simply as the summary. You must not mention edits or
   editing, or the "edited summary" in any way, simply call it the
   summary.
```

```
[Edit format]
You should generate six edits to the original summary, and for each
   generate an one-sentence explanation of what the error in the
   edited summary is.
Each edit should follow following json format:
{"edits": [
        {"edited_summary": "[Edited version of the summary]", "
           explanation": "A natural language explanation of the error
           ."},
    ...
]}

[Document]
[DOCUMENT]

[Summary]
[SUMMARY]
```

F.3 Detect and Explain error (D&E)

```
You are given a document and its corresponding summary which may or
   may not be factually correct and consistent with the document.
   Your task is to generate a valid json string (use escape
   characters if double quotes and new lines are used within value)
   that has following two fields:
1. "consistent" - This field gives whether the summary is factually
   correct and consistent with respect to the document. It should be
   "yes" if the summary is factually consistent with respect to the
   document and "no" otherwise.
2. "explanation" - If you set "consistent" to "no", then this field
   should give the explanation as why you think it is incorrect or
   inconsistent. If you set "consistent" to "yes", then the
   explanation should be an empty string. Example format:

{"consistent": "no", "explanation": "[...]"}
or
{"consistent": "yes", "explanation": ""}

Now complete the task for the following document, summary pair:

[Document]
[DOCUMENT]

[Summary]
[SUMMARY]
```

F.4 Explain error given Detection (E|D)

```
You are given a document and its corresponding summary. We know that
   the summary contains a factual error which renders it inconsistent
    with the facts in the document. Your task is to provide a 1-2
```

```
   sentence explanation that identifies what fact or facts in the
   summary is/are inconsistent with the document. You should output
   your explanation in a valid json format (use escape characters for
    double quotes and new lines if used within value), in the
   following format:

{"explanation": "[...]"}

Now complete the task for the following document, summary pair:

[Document]
[DOCUMENT]

[Summary]
[SUMMARY]
```