Improved Evidence Extraction for Document Inconsistency Detection with LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) are becoming useful in many domains due to their impressive abilities that arise from large training datasets and large model sizes. However, research on LLM-based approaches to document inconsistency detection is relatively limited. There are two key aspects of document inconsistency detection: (i) classification of whether there exists any inconsistency, and (ii) providing evidence of the inconsistent sentences. We focus on the latter, and introduce new comprehensive evidence-extraction metrics and a redact-and-retry framework with constrained filtering that substantially improves LLM-based document inconsistency detection over direct prompting. We back our claims with promising experimental results.

1 Introduction

Large language models (LLMs) are becoming useful in many domains (Li et al., 2023a; Sakai and Lam, 2025; Tan et al., 2025b,a) due to their impressive abilities that arise from large training datasets (Li et al., 2023b) and large model sizes (Naveed et al., 2023). Despite their wide applications, research on LLM-based approaches to document inconsistency detection is very limited (Li et al., 2024). Unlike natural language understanding (Harabagiu et al., 2006), where detecting inconsistencies (or contradictions) is often defined as determining the relation between a hypothesis and a piece of premise, we study the detection of inconsistencies that occur within the confines of a single input document.

Motivation. Research from the field of psychology (Graesser and McMahen, 1993; Otero and Kintsch, 1992) indicates that humans struggle to identify and detect inconsistencies (or contradictions) in unfamiliar and informative texts, especially when contradictions are widely separated in long documents. This motivates a need for an automated system to tackle this challenge.

Problem setup. The objective of document inconsistency detection is defined as follows: Given an input document $\boldsymbol{x}$, the goal is to correctly classify the presence of inconsistency $y \in \{\text{Yes}, \text{No}\}$ and identify the set of inconsistent sentences $\mathcal{E} = \{s_1, \ldots, s_k\}$ if there exists any inconsistency. Therefore, the LLM inference approach is required to output two things: (i) Classification $\hat{y} \in \{\text{Yes}, \text{No}\}$ of whether there is any inconsistency. (ii) If $\hat{y} = \text{Yes}$, then the evidence set of sentences $\widehat{\mathcal{E}} = \{\hat{s}_1, \ldots, \hat{s}_{k'}\}$ is also required. Note that k' may not equal k. While improving the performance of both points (i) and (ii) is important, in this paper, we focus solely on (ii).

2 Related Work

Sentence-level detection. Most of the previous work in inconsistency detection focused on the sentence level. Specifically, prior work studied contradictions under the natural language inference (NLI) framework of evaluating contradictory pairs of sentences (Dagan et al., 2005; Bowman et al., 2015).

Document-level detection. Some NLI research has more recently been extended to document-level reasoning (Yin et al., 2021; Schuster et al., 2022; Mathur et al., 2022). However, these works do not consider inconsistency detection via LLMs holistically at the document level. The most relevant work is that of Li et al. (2024), which provided a dataset for document inconsistency detection and tested LLM direct prompting (DP) on it.

Applications. Document inconsistency detection can be applied to various domains: (i) detecting rumorous posts on Twitter (Li et al., 2018), (ii) detecting contradicting reviews on Amazon (Li et al., 2018), and (iii) detecting and fixing contradictions

in financial (Deußer et al., 2023) and biomedical reports (Rosemblat et al., 2019).

Main contributions. Building on the contributions of Li et al. (2024), we propose new comprehensive metrics for evaluating evidence extraction (Section 3). Furthemore, we provide a new novel approach, which utilizes *redact-and-retry* (Algorithm 1) and a constrained filter call, that outperforms DP in the comprehensive metrics that we proposed. We show this empirically via experiments.

3 Metrics for Evidence Extraction

We define $\mathcal{D} = \{1, \ldots, n\}$ to be the indices of the datapoints, $\mathcal{D}_+ = \{i \,|\, i \in \mathcal{D} \text{ and } y_i = \text{Yes}\}$, and $\mathcal{D}_{++} = \{i \,|\, i \in \mathcal{D} \text{ and } y_i = \hat{y}_i = \text{Yes}\}$. We now define our metrics.

Evidence hit rate (EHR) and evidence hit rate when correct (EHRC). For index $i \in \mathcal{D}$, given document $\boldsymbol{x}_i$, true classification y_i, and true evidence set $\mathcal{E}_i$, we have the evidence hit $\text{EH}_i = \mathbb{1}\{\mathcal{E}_i \subseteq \widehat{\mathcal{E}}_i\}$. We define EHR and EHRC as follows:

$$\text{EHR} = \frac{1}{|\mathcal{D}_+|} \sum_{i \in \mathcal{D}_+} \text{EH}_i \tag{1}$$

$$\text{EHRC} = \frac{1}{|\mathcal{D}_{++}|} \sum_{i \in \mathcal{D}_{++}} \text{EH}_i. \tag{2}$$

They are connected by the identity EHR = TPR × EHRC, where TPR is the true positive rate of classification – we derive this in Appendix B. These metrics were previously introduced by Li et al. (2024).

Evidence precision rate (EPR) and evidence precision rate when correct (EPRC). We define evidence precision $\text{EP}_i = |\mathcal{E}_i \cap \widehat{\mathcal{E}}_i| / |\widehat{\mathcal{E}}_i|$. We define EPR and EPRC as follows:

$$\text{EPR} = \frac{1}{|\mathcal{D}_+|} \sum_{i \in \mathcal{D}_+} \text{EP}_i \tag{3}$$

$$\text{EPRC} = \frac{1}{|\mathcal{D}_{++}|} \sum_{i \in \mathcal{D}_{++}} \text{EP}_i. \tag{4}$$

They are connected by the identity EPR = TPR × EPRC; we derive this in Appendix B.

Evidence recall rate (ERR) and evidence overlap rate when correct (ERRC). We define evidence recall $\text{ER}_i = |\mathcal{E}_i \cap \widehat{\mathcal{E}}_i| / |\mathcal{E}_i|$. We define ERR and ERRC as follows:

$$\text{ERR} = \frac{1}{|\mathcal{D}_+|} \sum_{i \in \mathcal{D}_+} \text{ER}_i \tag{5}$$

$$\text{ERRC} = \frac{1}{|\mathcal{D}_{++}|} \sum_{i \in \mathcal{D}_{++}} \text{ER}_i. \tag{6}$$

They are connected by the identity ERR = TPR × ERRC; we derive this in Appendix B.

Average evidence coverage ratio (AECR). We define the number of sentences in document $\boldsymbol{x}_i$ to be ℓ_i, and $\text{ECR}_i = |\widehat{\mathcal{E}}_i| / \ell_i$. We define AECR as

$$\text{AECR} = \frac{1}{|\mathcal{D}_+|} \sum_{i \in \mathcal{D}_+} \text{ECR}_i. \tag{7}$$

Remark. We desire high EHR, EHRC, EPR, EPRC, ERR, ERRC, and a low AECR. Aside from EHR and EHRC, the rest are new metrics that we propose in this paper.

Research question. *Can we improve the EHR, EHRC, EPR, EPRC, ERR, and ERRC of direct prompting, while keeping the AECR low?* We show empirically that the answer is yes.

4 Methodology

Redact. We start by defining the function $\text{Redact}(\boldsymbol{x}, \widehat{\mathcal{E}})$ which takes in a document $\boldsymbol{x}$ and outputs the same document, but with all the sentences present in $\widehat{\mathcal{E}}$ removed.

Redact-and-retry. Our algorithm is shown in Algorithm 1 – a visual representation is presented in Figure 3 in the appendix. The LLM prompt in the retry step is the same as that of direct prompting – the prompt is given in Appendix F. The intuition behind redacting and retrying is that redacting preserves the sentence order in the document and also reduces the search space for the LLM, which we expect would make it easier for the LLM to detect the remaining inconsistencies (if there are any left). Since $\widehat{\mathcal{E}}_i^{(1)} \subseteq \bigcup_j \widehat{\mathcal{E}}_i^{(j)}$, EHR, EHRC, ERR, and ERRC can only improve from DP.

Filter. We explore the idea of applying a filter (i.e., another LLM call) to the evidence set output $\bigcup_j \widehat{\mathcal{E}}_i^{(j)}$ of redact-and-retry – this is shown visually in Figure 4 in the appendix. The role of the filter call is for the LLM to re-analyze the sentences in $\bigcup_j \widehat{\mathcal{E}}_i^{(j)}$ and only output the sentences that it thinks are truly inconsistent. The filter call is expected to

reduce the size of the evidence set, but at the cost of an additional LLM call. We study 2 types of filter calls: *(i) Unconstrained.* The filter call is allowed to return any number of sentences, including zero. If it returns an empty evidence list, then we will change the initial classification from Yes to No. *(ii) Constrained.* The filter call is constrained to return at least one sentence. Hence, we do not change the LLM's initial classification (from redact-and-retry). The prompts for both filter calls are given in Appendix F.

Remark. The idea of using redact-and-retry with a filter call is loosely inspired by Zhang et al. (2024), where they used a sequence of LLM calls (i.e., worker agents) followed by a final LLM call (i.e., manager agent) to synthesize the outputs of prior LLM calls.

Algorithm 1 Redact-and-Retry Algorithm

Input: document $\boldsymbol{x}_i$.
Initialize $\boldsymbol{x}_i^{(1)} = \boldsymbol{x}_i$ and $j = 1$.
Get $\hat{y}_i^{(1)}, \widehat{\mathcal{E}}_i^{(1)} = \text{LLM}(\boldsymbol{x}_i^{(1)})$.
while $\hat{y}_i^{(j)} =$ Yes **do**
    $j = j + 1$.
    $\boldsymbol{x}_i^{(j)} = \text{Redact}(\boldsymbol{x}_i^{(j-1)}, \widehat{\mathcal{E}}_i^{(j-1)})$.
    $\hat{y}_i^{(j)}, \widehat{\mathcal{E}}_i^{(j)} = \text{LLM}(\boldsymbol{x}_i^{(j)})$.
end while
Set $\hat{y}_i = \hat{y}_i^{(1)}$ and $\widehat{\mathcal{E}}_i = \bigcup_j \widehat{\mathcal{E}}_i^{(j)}$.
return $\hat{y}_i$ and $\widehat{\mathcal{E}}_i$.

5 Experiments

Due to the non-deterministic nature of some LLMs, the sentences that the LLM outputs might not always be exactly the same as the sentence in the input documents. Therefore, we allow approximate matching instead exact matching in certain aspects of our implementation: (i) For the computation of EH_i, when checking whether $\mathcal{E}_i \subseteq \widehat{\mathcal{E}}_i$, we check whether each sentence in $\mathcal{E}_i$ has a match in $\widehat{\mathcal{E}}_i$ that has a cosine similarity of at least 0.8 using the TF-IDF vectorizer (Sparck Jones, 1972). If it is a yes for all such sentences, then we determine that $\mathcal{E}_i \subseteq \widehat{\mathcal{E}}_i$. (ii) For the computation of EP_i, when checking whether $\mathcal{E}_i \cap \widehat{\mathcal{E}}_i$, we check whether each sentence in $\mathcal{E}_i$ has a match in $\widehat{\mathcal{E}}_i$ that has a cosine similarity of at least 0.8 using the TF-IDF vectorizer (Sparck Jones, 1972). If it is a yes, then that sentence is in $\mathcal{E}_i \cap \widehat{\mathcal{E}}_i$. (iii) For the computation of $\text{Redact}(\boldsymbol{x}, \widehat{\mathcal{E}})$, when checking whether a sentence s in $\boldsymbol{x}$ is in $\widehat{\mathcal{E}}$, we check whether there exists a sentence in $\widehat{\mathcal{E}}$ that has a cosine similarity of at least 0.8 with s using the TF-IDF vectorizer (Sparck Jones, 1972). We do not consider ERR and ERRC since they are redundant for our dataset – explained shortly in Section 5.1.

	G4o	L3.2	L3
acc (DP,RnR,RnR+CF)	0.680	0.681	0.601
acc (RnR+UF)	0.677	0.675	0.601
TPR (DP,RnR,RnR+CF)	0.763	0.616	0.225
TPR (RnR+UF)	0.712	0.583	0.225
AECR (DP)	0.051	0.047	0.016
AECR (RnR)	0.133	0.118	0.022
AECR (RnR+UF)	0.059	0.050	0.016
AECR (RnR+CF)	0.054	0.046	0.015
avg #retries (RnR)	2.92	2.50	1.41

Table 1: Performance comparison of LLMs across various metrics. G4o, L3.2, and L3 are shorthands for GPT-4o, LLaMA3.2-90B, and LLaMA3-70B respectively. 'acc' is the shorthand for accuracy and 'avg' is the shorthand for average.

5.1 Dataset and Models

We use the ContraDoc dataset from Li et al. (2024), which contains 449 positive (inconsistent) documents and 442 negative (consistent) documents – every positive document has exactly one inconsistent sentence. The average number of sentences per document for the positive documents, negative documents, and all documents are 38.7, 36.5, and 37.6 respectively. We test our approach using 3 different LLMs with temperature set to 0 for consistency of results: GPT-4o (strong), LLaMA3.2-90B (medium), and LLaMA3-70B (weak). Since $|\mathcal{E}_i| = 1$ for all $i \in \mathcal{D}$, we have $\text{EH}_i = \text{ER}_i$ for all i. Therefore, we will not consider ERR and ERRC in our experiments. More details on ContraDoc are in Appendix G.

5.2 Main Results

We use the shorthands DP for direct prompting, RnR for redact-and-retry, RnR+UF for redact-and-retry with an unconstrained filter, and RnR+CF for redact-and-retry with a constrained filter. The results are stated in Table 1, Figure 1, and Figure 2. We remark that DP, RnR, and RnR+CF all share the same classification metrics (i.e., accuracy and TPR) since their classifications are all the same. We proceed to give some key observations.

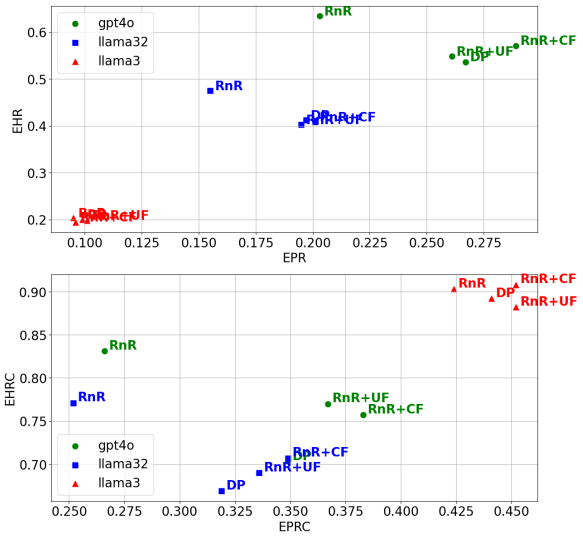

Figure 1: Plots of EHR vs. EPR and EHRC vs. EPRC.

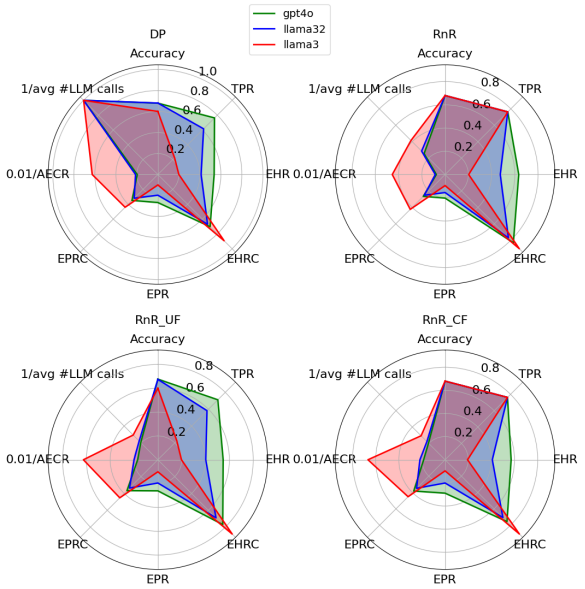

Figure 2: Radar charts for different approaches and LLMs.

Evidence hit and precision. Figure 1 shows how EHR varies with EPR (top), and how EHRC varies with EPRC (bottom). Recall that we desire both a higher EHR/EHRC and a higher EPR/EPRC. We make some observations:

- For every LLM, we notice that RnR has a better EHR (and EHRC) than DP, at the cost of a lower EPR (and EPRC). This is expected since RnR outputs a larger evidence set than DP, making it easier to hit the correct evidence but harder to be precise.
- RnR+CF generally outperforms DP in both EHR (and EHRC) and EPR (and EPRC) for GPT-4o and LLaMA3.2-90B, showing that the constrained filter is effective. The performance of RnR+CF is better than RnR+UF in most cases.
- Interestingly, while stronger LLMs lead to a better EHR and EPR, this is not the case of EHRC and EPRC since LLaMA3-70B has the best EHRC and EPRC for all approaches. This might imply that when the LLM is weaker, it is more cautious in selecting evidence sentences, leading to a higher precision when it is correct. Differences between LLMs are further illustrated in Figure 2.

Observe that for RnR+UF, the accuracy and TPR dropped slightly for GPT-4o and LLaMA3.2 but remained the same for LLaMA3, implying that the UF is not effective in improving classification – we provide a detailed error analysis in Appendix C.

Evidence coverage ratio. As expected, RnR leads to a larger AECR. Notice that RnR+CF has AECR that is very close to DP, showing the effectiveness of the constrained filter in reducing the evidence set size. RnR+UF also leads to a significant reduction in AECR when compared to RnR, but is still slightly larger than DP.

Number of retries and sentences. Table 1 states the average number of retries across all datapoints in the dataset (both positive and negative), and shows that the average number of retries for all LLMs does not exceed 3, implying that it is well controlled. Note that the number of retries for RnR+UF and RnR+CF are just 1 more than RnR. We conduct a deeper analysis on the number of retries and sentences in Appendix E.

Overall, by considering all the metrics, we observe that RnR+CF is the best all-rounder performer. The only cost of attaining this better performance through RnR+CF is additional LLM calls – however, we argue that the cost is low since we only require around 3 additional LLM calls (when compared to DP). Full experimental results are provided in Appendix D.

6 Conclusion

We introduced comprehensive evidence-extraction metrics for document inconsistency detection, and showed empirically that our RnR+CF approach improves the EHR, EHRC, EPR, and EPRC, while keeping the AECR close to that of DP.

7 Limitations

We only considered one dataset, but argue that it may be sufficient since it contains many datapoints (i.e., documents). We note that our dataset only has evidence set size $|\mathcal{E}_i| = 1$ for every datapoint, resulting in $\text{EH}_i = \text{ER}_i$ for all i. Hence, we are unable to highlight experimentally the differences between EHR (EHRC) and ERR (ERRC). While we would like to test our approach on datasets where $|\mathcal{E}_i| > 1$, we were unable to find such a dataset – document inconsistency datasets are generally not readily available, as the field is relatively new compared to other LLM applications.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. *Machine learning challenges workshop*, pages 177–190.

Tobias Deußer, Maren Pielka, Lisa Pucknat, Basil Jacob, Tim Dilmaghani, Mahdis Nourimand, Bernd Kliem, Rüdiger Loitz, Christian Bauckhage, and Rafet Sifa. 2023. Contradiction detection in financial reports. *Proceedings of the Northern Lights Deep Learning Workshop*, 4.

Arthur C Graesser and Cathy L McMahen. 1993. Anomalous information triggers questions when adults solve quantitative problems and comprehend stories. *Journal of Educational Psychology*, 85:136.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 6:755–762.

Chuqin Li, Xi Niu, Ahmad Al-Doulat, and Noseong Park. 2018. A computational approach to finding contradictions in user opinionated text. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 351–356.

Jierui Li, Vipul Raheja, and Dhruv Kumar. 2024. ContraDoc: Understanding self-contradictions in documents with large language models. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 6509–6523.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023a. Large language models in finance: A survey. *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023b. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Puneet Mathur, Gautam Kunapuli, Riyaz Bhat, Manish Shrivastava, Dinesh Manocha, and Maneesh Singh. 2022. DocInfer: Document-level natural language inference using optimal evidence selection. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 809–824.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

José Otero and Walter Kintsch. 1992. Failures to detect contradictions in a text: What readers believe versus what they read. *Psychological Science*, 3:229–236.

Graciela Rosemblat, Marcelo Fiszman, Dongwook Shin, and Halil Kilicoglu. 2019. Towards a characterization of apparent contradictions in the biomedical literature using context analysis. *Journal of biomedical informatics*, 98:103275.

Hajar Sakai and Sarah S Lam. 2025. Large language models for healthcare text classification: A systematic review. *arXiv preprint arXiv:2503.01159*.

Tal Schuster, Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, and Donald Metzler. 2022. Stretching sentence-pair NLI models to reason over long documents and clusters. *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 394–412.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28:11–21.

Nelvin Tan, James Asikin Cheung, Yu-Ching Shih, Dong Yang, and Amol Salunkhe. 2025a. Does using counterfactual help llms explain textual importance in classification? *arXiv preprint arXiv:2510.04031*.

Nelvin Tan, Zian Seng, Liang Zhang, Yu-Ching Shih, Dong Yang, and Amol Salunkhe. 2025b. Improved LLM agents for financial document question answering. *arXiv preprint arXiv:2506.08726*.

Wenpeng Yin, Dragomir Radev, and Caiming Xiong. 2021. DocNLI: A large-scale dataset for document-level natural language inference. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4913–4922.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

A Visualization of Algorithms

The visualizations of RnR and RnR+filter (both unconstrained and constrained filters) are provided in Figure 3 and Figure 4, respectively.

B Derivation of Identities

Identity 1. EHR = TPR × EHRC.
Proof: We have

$$\begin{aligned}\text{EHR} &\overset{(a)}{=} \frac{1}{|\mathcal{D}_+|}\sum_{i\in\mathcal{D}_+}\text{EH}_i \\ &\overset{(b)}{=} \frac{1}{|\mathcal{D}_+|}\sum_{i\in\mathcal{D}_{++}}\text{EH}_i \\ &\overset{(c)}{=} \frac{1}{\text{TP}+\text{FN}}\sum_{i\in\mathcal{D}_{++}}\text{EH}_i \\ &= \frac{\text{TP}}{\text{TP}+\text{FN}}\cdot\frac{1}{\text{TP}}\sum_{i\in\mathcal{D}_{++}}\text{EH}_i \\ &\overset{(d)}{=} \text{TPR}\times\text{EHRC},\end{aligned}$$

where:

- (a) uses the definition of EHR.
- (b) uses the fact that $\text{EH}_i = 0$ for all $i \in \mathcal{D}_+ \setminus \mathcal{D}_{++}$ since $\widehat{\mathcal{E}}_i = \emptyset$ when $\hat{y}_i = \text{No}$.
- (c) uses the fact that $\text{TP} + \text{FN} = |\mathcal{D}_+|$.
- (d) uses the definition of TPR and EHRC, along with the fact that $\text{TP} = |\mathcal{D}_{++}|$.

Identity 2. EPR = TPR × EPRC. *Proof:* This can be derived by using similar steps to the proof of identity 1, and noting that $\text{EP}_i = 0$ for all $i \in \mathcal{D}_+ \setminus \mathcal{D}_{++}$ since $\widehat{\mathcal{E}}_i = \emptyset$ when $\hat{y}_i = \text{No}$.

Identity 3. ERR = TPR × ERRC. *Proof:* This can be derived by using similar steps to the proof of identity 1, and noting that $\text{ER}_i = 0$ for all $i \in \mathcal{D}_+ \setminus \mathcal{D}_{++}$ since $\widehat{\mathcal{E}}_i = \emptyset$ when $\hat{y}_i = \text{No}$.

C Error Analysis of the Unconstrained Filter

To provide a detailed error analysis of the unconstrained filter (UF), we compute the following metrics:

- Rate wrong (-) to correct (+) given flipped classification:

$$R_{\text{UF}}(- \to + \,|\, \text{flip}) = \frac{\#\{- \to +\}}{\#\text{flips}}.$$

  A similar definition applies to $R_{\text{CF}}(- \to + \,|\, \text{flip})$, where instead of UF, we look at the constrained filter.

- Rate correct (+) to wrong (-) given flipped classification:

$$R_{\text{UF}}(+ \to - \,|\, \text{flip}) = \frac{\#\{+ \to -\}}{\#\text{flips}}.$$

  A similar definition applies to $R_{\text{CF}}(+ \to - \,|\, \text{flip})$, where instead of UF, we look at the constrained filter.

- Rate of UF keeping true evidence given the earlier RnR sub-algorithm found true evidence:

$$R_{\text{UF}}(\mathcal{E} \text{ kept} \,|\, \mathcal{E} \text{ found}) = \frac{\#\mathcal{E} \text{ kept}}{\#\mathcal{E} \text{ found}}.$$

  A similar definition applies to $R_{\text{CF}}(\mathcal{E} \text{ kept} \,|\, \mathcal{E} \text{ found})$, where instead of UF, we look at the constrained filter.

- Rate of UF discarding true evidence given the earlier RnR sub-algorithm found true evidence:

$$R_{\text{UF}}(\mathcal{E} \text{ discarded} \,|\, \mathcal{E} \text{ found}) = \frac{\#\mathcal{E} \text{ discarded}}{\#\mathcal{E} \text{ found}}.$$

  A similar definition applies to $R_{\text{CF}}(\mathcal{E} \text{ discarded} \,|\, \mathcal{E} \text{ found})$, where instead of UF, we look at the constrained filter.

The results are displayed in Table 2. We make the following observations:

- As expected, for GPT-4o and LLaMA3.2, we have $R_{\text{UF}}(+ \to - \,|\, \text{flip}) > R_{\text{UF}}(- \to + \,|\, \text{flip})$, resulting in a lower accuracy and TPR after applying UF.
- Comparing $R_{\text{UF}}(\mathcal{E} \text{ kept} \,|\, \mathcal{E} \text{ found})$ with $R_{\text{UF}}(\mathcal{E} \text{ discarded} \,|\, \mathcal{E} \text{ found})$ for each LLM, we observe that the rate of discarding the true evidence by UF is around the same for GPT-4o and LLaMA3.2, whereas it is significantly lower for LLaMA3.
- For every LLM, we have

$$R_{\text{UF}}(\mathcal{E} \text{ kept}|\mathcal{E} \text{ found}) < R_{\text{CF}}(\mathcal{E} \text{ kept}|\mathcal{E} \text{ found})$$
$$R_{\text{UF}}(\mathcal{E} \text{ disc.}|\mathcal{E} \text{ found}) > R_{\text{CF}}(\mathcal{E} \text{ disc.}|\mathcal{E} \text{ found}).$$

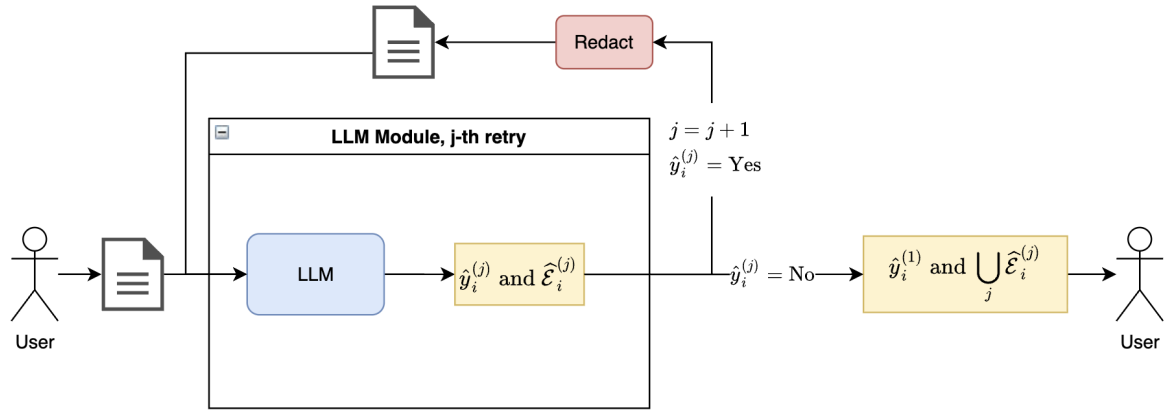

Figure 3: Algorithm 1 visualized.

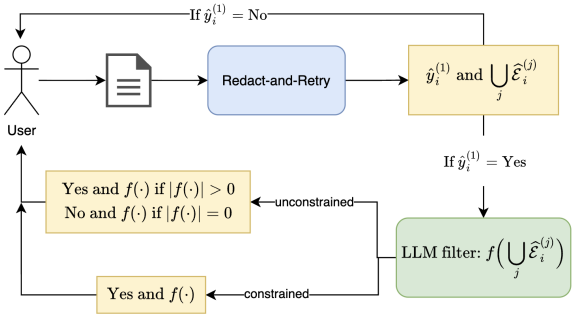

Figure 4: Algorithm 1 with LLM filter call visualized.

	G4o	L3.2	L3
$R_{\text{UF}}(- \to +\|\text{flip})$	0.472	0.380	0
$R_{\text{UF}}(+ \to -\|\text{flip})$	0.528	0.619	0
$R_{\text{CF}}(- \to +\|\text{flip})$	0	0	0
$R_{\text{CF}}(+ \to -\|\text{flip})$	0	0	0
$R_{\text{UF}}(\mathcal{E}$ kept$\|\mathcal{E}$ found)	0.861	0.847	0.976
$R_{\text{UF}}(\mathcal{E}$ disc.$\|\mathcal{E}$ found)	0.139	0.153	0.024
$R_{\text{CF}}(\mathcal{E}$ kept$\|\mathcal{E}$ found)	0.908	0.884	0.988
$R_{\text{CF}}(\mathcal{E}$ disc.$\|\mathcal{E}$ found)	0.092	0.116	0.012

Table 2: Performance comparison of LLMs across various metrics. G4o, L3.2, and L3 are shorthands for GPT-4o, LLaMA3.2-90B, and LLaMA3-70B respectively. 'disc.' is the shorthand for discarded.

D Full Experimental Results

We present the following metrics for our experiment:

- Accuracy $\frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}} \mathbb{1}\{y_i = \hat{y}_i\}$, where $\mathcal{D}$ is the set of all data points.
- Precision $\text{TP}/(\text{TP}+\text{FP})$, where TP is the number of true positives and FP is the number of false positives.
- Recall/true positive rate $\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$ where FN is the number of false negatives.
- F1 score $(2\times\text{Precision}\times\text{Recall})/(\text{Precision}+\text{Recall})$.
- FPR false positive rate $\text{FP}/(\text{TN} + \text{FP})$.
- TNR true negative rate $\text{TN}/(\text{TN}+\text{FP})$, where TN is the number of true negatives.
- FNR false negative rate $\text{FN}/(\text{TP} + \text{FN})$.
- EHR, EHRC, EPR, EPRC, and AECR are defined in Section 3.

The full experimental results for DP, RnR, and RnR+CF are presented in Table 3, and the full experimental results for RnR+UF are presented in Table 4 – we have separated them up in two tables for clarity since DP, RnR, and RnR+CF all share the same classification metrics (i.e., accuracy, precision, recall, F1 score, TPR, FPR, TNR, and FNR) whereas RnR+UF does not.

E Additional Analysis on Number of Retries and Sentences

Number of retries. Figure 5 shows visually how the number of retries for RnR are distributed – we observe that the stronger the LLM, the more retries it will make. The same conclusion follows through for RnR+UF and RnR+CF since their number of retries are always just 1 more than RnR. This implies that stronger LLMs are more aggressive in detecting inconsistencies.

Number of sentences. Table 4 shows that the average number of sentences for all LLMs, when

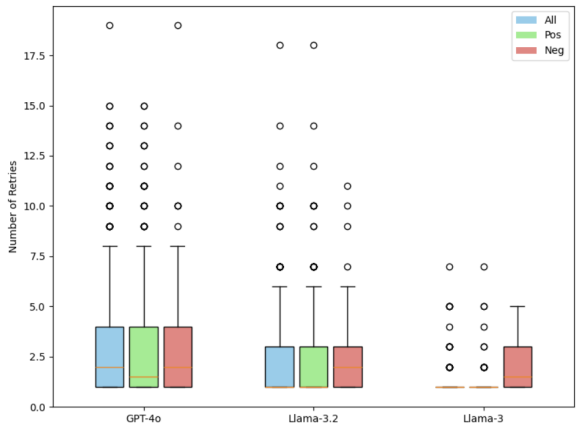

Figure 5: Boxplot for number of retries for RnR. 'Pos' refers to datapoints i such that truth y_i = Yes.'Neg' refers to datapoints i such that truth y_i = No.

using RnR is at most 6.85. Figure 6 shows visually how the number of sentences are distributed – we observe that the stronger LLM (GPT-4o and LLaMA3.2-90B) have around the same distribution of sentences, whereas the weaker LLM (LLaMA3-70B) produces a smaller number of sentences.

F Prompts and Code

The prompts used are stated here. Note that for redact-and-retry, the same prompt for direct prompting is used during every retry step.

Prompt for direct prompting and redact-and-retry:

```
The task is to determine whether the document contains any
↪ self-contradictions. If yes, provide evidence by quoting mutually
↪ contradictory sentences in a list of strings in Python. If no, then
↪ give an empty list. Your response must follow this JSON format (OR
↪ options are provided), and provide absolutely nothing else.
↪ Strictly follow the double quotation marks and only use single
↪ quotations within each sentence.

### JSON format
{
    "judgement": "yes" OR "no",
    "evidence": ["sentence1", "sentence2", ..., "sentenceN"] OR []
}

### document
{document}
```

Prompts for filtering (unconstrained):

```
You will be given a list of sentences that are flagged to be
↪ potentially inconsistent. Your task is to identify all inconsistent
↪ sentences and output them in the following JSON format, and provide
↪ absolutely nothing else. Strictly follow the double quotation marks
↪ and only use single quotations within each sentence.

### JSON format
{
    "evidence": ["sentence1", "sentence2", ..., "sentenceN"] OR []
}

### potentially inconsistent list of sentences
{list of inconsistent sentences}
```

Prompts for filtering (constrained):

```
You will be given a list of sentences that are flagged to be
↪ potentially inconsistent. Your task is to identify all inconsistent
↪ sentences and output them in the following JSON format, and provide
↪ absolutely nothing else. You must output at least 1 sentence.
↪ Strictly follow the double quotation marks and only use single
↪ quotations within each sentence.

### JSON format
{
    "evidence": ["sentence1", "sentence2", ..., "sentenceN"]
}

### potentially inconsistent list of sentences
{list of inconsistent sentences}
```

G ContraDoc Details

The ContraDoc dataset from Li et al. (2024) is a human-annotated dataset consisting of self-contradictory documents across varying document domains and lengths and self-contradiction types. More specifically, each positive datapoint in their dataset contains exactly one sentence with one of the following 8 types of self-contradiction:

1. **Negation.** There exists one sentence which is a negation of another sentence. Example: 'Zully donated her kidney.' vs. 'Zully never donated her kidney.'

2. **Numeric.** There exists a numerical mismatch between sentences. Example: 'All the donors are between 20 to 45 years old.' vs. 'Lisa, who donates her kidney, she is 70 years old.'

3. **Content.** There exists one sentence changing one or multiple attributes of an event or entity that was previously stated in another sentence. Example: 'Zully Broussard donated her kidney to a stranger.' vs. 'Zully Broussard donated her kidney to her close friend.'

4. **Perspective/View/Opinion.** Inconsistency in perspective/view/opinion between sentences. Example: 'The doctor spoke highly of the project and called it a breakthrough' vs. 'The doctor disliked the project, saying it had no impact at all.'

5. **Emotion/Mood/Feeling.** Inconsistency in emotion/mood/feeling between sentences. Example: 'The rescue team searched for the boy worriedly.' vs. 'The rescue team searched for the boy happily.'

6. **Relation.** Presence of two mutually exclusive relations between entities. Example: 'Jane and Tom are a married couple.' vs. 'Jane is Tom's sister.'

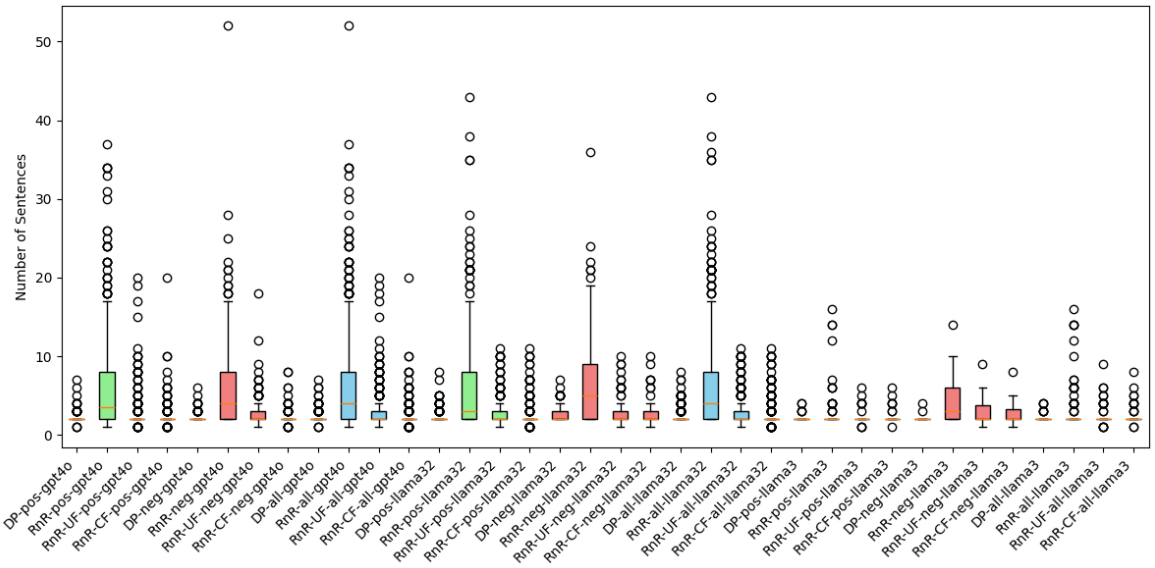

Figure 6: Boxplot for number of sentences. 'pos' refers to datapoints i such that $y_i =$ Yes, 'neg' refers to datapoints i such that $y_i =$ No, and 'all' refers to all datapoints.

7. **Factual.** There exist sentence(s) in disagreement with external world knowledge/facts. Example: 'The road T51 was located in New York.' vs. 'The road T51 was located in California.'

8. **Causal.** There exist sentences where the effect does not match the cause. Example: 'I slam the door.' vs. 'After I do that, the door opens.'

	G4o	L3.2	L3
accuracy	0.680	0.681	0.601
precision	0.642	0.704	0.869
F1 score	0.697	0.657	0.357
TPR/recall	0.763	0.616	0.225
FPR	0.399	0.254	0.033
TNR	0.601	0.746	0.967
FNR	0.237	0.384	0.775
EHR (DP)	0.536	0.412	0.200
EHR (RnR)	0.634	0.475	0.203
EHR (RnR+CF)	0.571	0.408	0.194
EHRC (DP)	0.703	0.669	0.892
EHRC (RnR)	0.831	0.771	0.903
EHRC (RnR+CF)	0.757	0.707	0.908
EPR (DP)	0.267	0.197	0.099
EPR (RnR)	0.203	0.155	0.095
EPR (RnR+CF)	0.289	0.201	0.096
EPRC (DP)	0.349	0.319	0.441
EPRC (RnR)	0.266	0.252	0.424
EPRC (RnR+CF)	0.383	0.349	0.452
AECR (DP)	0.051	0.047	0.016
AECR (RnR)	0.133	0.118	0.022
AECR (RnR+CF)	0.054	0.046	0.015
avg #sen, pos (DP)	2.11	2.33	2.08
avg #sen, neg (DP)	2.22	2.61	2.21
avg #sen, all (DP)	2.15	2.41	2.09
avg #sen, pos (RnR)	6.40	6.42	2.84
avg #sen, neg (RnR)	6.41	6.85	4.71
avg #sen, all (RnR)	6.40	6.55	3.08
avg #sen, pos (RnR+CF)	2.33	2.57	2.16
avg #sen, neg (RnR+CF)	2.26	2.67	2.92
avg #sen, all (RnR+CF)	2.31	2.60	2.25
avg #retries, pos (RnR)	2.95	2.47	1.30
avg #retries, neg (RnR)	2.88	2.55	2.14
avg #retries, all (RnR)	2.92	2.50	1.41

Table 3: Performance comparison of models across various metrics for the LLM approaches DP, RnR, and RnR+CF. G4o, L3.2, and L3 are shorthands for GPT-4o, LLaMA3.2-90B, and LLaMA3-70B respectively. 'pos' refers to datapoints i such that $y_i =$ Yes, 'neg' refers to datapoints i such that $y_i =$ No, and 'all' refers to all datapoints.

	G4o	L3.2	L3
accuracy	0.677	0.675	0.601
precision	0.651	0.709	0.869
F1 score	0.680	0.640	0.357
TPR/recall	0.712	0.583	0.225
FPR	0.356	0.235	0.0329
TNR	0.644	0.765	0.967
FNR	0.288	0.417	0.775
EHR	0.548	0.402	0.198
EHRC	0.770	0.690	0.882
EPR	0.261	0.195	0.101
EPRC	0.367	0.336	0.452
AECR	0.059	0.050	0.016
avg #sen, pos	2.85	2.71	2.14
avg #sen, neg	2.90	2.84	3.14
avg #sen, all	2.87	2.75	2.27
avg #retries, pos	3.95	3.47	2.30
avg #retries, neg	3.88	3.55	3.14
avg #retries, all	3.92	3.50	2.41

Table 4: Performance comparison of models across various metrics for the LLM approach RnR+UF. G4o, L3.2, and L3 are shorthands for GPT-4o, LLaMA3.2-90B, and LLaMA3-70B respectively. 'pos' refers to datapoints i such that $y_i =$ Yes, 'neg' refers to datapoints i such that $y_i =$ No, and 'all' refers to all datapoints.