Where Is the Lie? Supervised Divergence Span Detection for User-Injected Misinformation in LLM Inputs

Anonymous ACL submission

Abstract

Large language models (LLMs) silently comply with false premises in user queries, propagating misinformation they could otherwise correct. Detecting *whether* a prompt is corrupted is insufficient for interpretable defense—one needs to know *which tokens* carry the falsehood. We introduce **SpanDetector**, a LoRA-fine-tuned span localization model that performs token-level BIO sequence labeling to identify the corrupted span in user inputs. A key technical contribution is our *alignment-derived annotation*: given paired clean/corrupted prompts, we automatically generate BIO labels via longest-common-prefix/suffix alignment—zero human annotation cost. At inference time, the trained model localizes spans from a *single* corrupted prompt without any clean reference, resolving the deployment bottleneck of prior surprisal-based approaches. Evaluated on PIH-Bench v2 (803 pairs, 4 domains, 43 corruption types), SpanDetector achieves **AUC = 0.9778** and **Span-F1 = 0.818** on Llama-3.1-8B, exceeding the best unsupervised baseline (divergence-span surprisal, AUC = 0.887) by **+9.1 pp**. The gain is consistent across four model families (Llama, Qwen2.5-7B/14B, DeepSeek-R1-Distill-7B), with all models surpassing the baseline by $\geq$7.7 pp AUC.

1 Introduction

Large language models are increasingly deployed as general-purpose assistants that must handle open-ended user queries—including queries that embed factually incorrect premises. When a user asks *"Why did Apple release the first iPhone in 1995?"*, a capable assistant should detect the false premise (*1995*), correct it, and then answer. Instead, state-of-the-art models routinely comply as if the false date were true—a failure mode that compounds at scale. document that user-injected misinformation degrades LLM accuracy by 12–66% across Llama-3, Mixtral, Qwen-2, and GPT-4o-mini, and that explicit verification instructions recover only $\approx$5 percentage points. The underlying cause is a well-documented tension between *parametric knowledge* (facts encoded in model weights during pretraining, Petroni et al. 2019) and *contextual knowledge* (facts supplied at inference time): when the two conflict, modern instruction-tuned LLMs tend to defer to the user's input even when it is wrong (Longpre et al., 2021; Xie et al., 2024; Shi et al., 2023; Zhou et al., 2023).

Why localization matters. Prior work on this problem focuses on *detection*: determining *whether* a given prompt contains a false premise (Yu et al., 2023; Lin et al., 2022). Detection is necessary but insufficient for deployable defense. A binary "corrupted / not corrupted" signal cannot produce a user-facing explanation (*which claim is wrong?*), cannot support targeted correction pipelines that edit only the suspicious tokens, and cannot enable fine-grained audit of where misinformation enters the reasoning chain. Token-level *localization*—identifying the exact span of corrupted tokens—addresses all three: it enables inline warnings to users, pinpoints the precise input region for retrieval-based fact-checking (Lewis et al., 2020), and is directly actionable by downstream correction modules (Meng et al., 2022; Neeman et al., 2023).

The deployment bottleneck. A concurrent analysis (Anonymous, 2026) establishes that the conflict between a false premise and parametric knowledge manifests as a measurable surprisal spike at the *corrupted token span*. Divergence-span surprisal achieves AUC = 0.887 at separating corrupted from clean prompts—a compelling signal. Its critical limitation, however, is that computing surprisal *within* the span requires first *finding* the span via alignment against a paired clean reference. In production, the user's message arrives alone: this paired-alignment requirement makes the ap-

proach fundamentally undeployable. Furthermore, because the detection must be triggered by the input rather than the output, it cannot be substituted by generation-based detectors that read the model's response (Manakul et al., 2023; Kadavath et al., 2022).

Our approach: SpanDetector. We address this gap by separating the *training regime* (where paired data is available to supervise span boundaries) from the *inference regime* (where only a single corrupted prompt is available). During training, we use paired (clean, corrupted) prompts to automatically derive BIO (Begin/Inside/Outside) labels via token-level longest-common-prefix/suffix alignment—a fully deterministic process requiring zero human annotation. At inference, the fine-tuned model processes a *single* corrupted prompt and outputs span coordinates with a single encoder forward pass. No clean reference, no extra LLM call, no generation overhead.

SpanDetector builds on a frozen large language model backbone augmented with parameter-efficient LoRA adapters (Hu et al., 2022) and a linear classification head—the same token classification paradigm established by Devlin et al. (2019) for NER, here applied to a novel target: *corruption span detection* in user inputs. The architecture keeps over 99.9% of the base model's weights frozen, making SpanDetector compatible with any open-weight LLM family without modification.

Contributions.

- **Task formulation.** We are the first to frame user-injected misinformation detection as token-level BIO span localization, upgrading from instance-level detection to token-level attribution.
- **Zero-cost annotation.** Alignment-derived BIO labels require only paired prompts, which any misinformation benchmark provides for free. No human labeling is needed at any stage.
- **Deployment-ready detection.** The trained model infers from a single input, removing the paired-alignment bottleneck, adding no generation latency, and remaining agnostic to the model's output style (direct answer or chain-of-thought, Wei et al. 2022).
- **Empirical validation.** SpanDetector achieves AUC ≥ 0.963 across four diverse model families (7B–14B), with the best configuration (Llama-3.1-8B, $r = 16$) reaching F1 = 0.818 and AUC = 0.9778—a 9.1 pp improvement over the strongest unsupervised baseline.

End-to-end demonstrations on real LLM conversations (Table 6) show that SpanDetector's span flags enable the same base LLM to self-correct answers it would otherwise get wrong—turning a passive misinformation follower into an active premise verifier.

2 Related Work

False-premise detection and misinformation. A growing body of work studies how LLMs respond to queries that embed incorrect factual premises. quantify the vulnerability across multiple instruction-tuned models and propose early-step supervised fine-tuning as a remedy. Yu et al. (2023) study false-premise questions in open-domain QA, finding that models often fail to challenge incorrect presuppositions. Shafiei et al. (2025) provide multi-hop false-premise data, extending the challenge to chained misinformation. Lin et al. (2022) measure models' systematic tendency to generate human-endorsed falsehoods. The broader hallucination problem is surveyed in Ji et al. (2023). Our work is orthogonal to all of these: rather than proposing a new defense or evaluation protocol, we address the more fundamental question of *where* in the input the misinformation resides—a prerequisite for interpretable, targeted correction.

Knowledge conflicts in LLMs. When contextual and parametric knowledge disagree, LLMs exhibit inconsistent behavior. Longpre et al. (2021) first systematically study entity-based knowledge conflicts in question answering, showing that models often override correct parametric knowledge with incorrect contextual information. Xie et al. (2024) characterize this as a dilemma between "adaptive chameleon" (following context) and "stubborn sloth" (following parametric knowledge) behaviors—precisely the failure mode SpanDetector is designed to diagnose. Shi et al. (2023) show that irrelevant context in math problems consistently distracts LLMs from correct solutions, motivating the need for explicit span-level conflict detection. Zhou et al. (2023) propose context-faithful prompting to mitigate sycophantic behavior, and Neeman et al. (2023) use adversarial training to disentangle parametric from contextual knowledge in

QA. Mallen et al. (2023) identify conditions under which parametric memory is unreliable, complementing our token-level diagnostic perspective.

Parametric knowledge and internal representations. Petroni et al. (2019) establish that LLMs encode factual knowledge parametrically, enabling probing via cloze queries. Meng et al. (2022) localize factual associations to specific mid-layer MLP activations, showing that knowledge is spatially organized. Tenney et al. (2019) and Belinkov (2022) probe transformer layers for linguistic properties at different granularities. The concurrent analysis (Anonymous, 2026) extends probing to the factual-conflict setting, showing that divergence-span surprisal achieves AUC = 0.887—a near-perfect signal that is, however, conditioned on having a clean reference, motivating our supervised training approach.

Sequence labeling and span extraction. SpanDetector's core mechanism—BIO labeling of token sequences— builds on a rich tradition in NLP. Lafferty et al. (2001) introduce conditional random fields for sequence labeling, and Lample et al. (2016) demonstrate the effectiveness of neural architectures (BiLSTM-CRF) for NER, establishing the recall-before-precision learning dynamic we observe in training (§5). Devlin et al. (2019) show that pre-trained transformers fine-tuned with a token classification head achieve state-of-the-art NER performance—the same paradigm we apply to corruption span detection. Conneau and Lample (2019) extend neural sequence labeling to multilingual settings. Our work differs from classical NER in its target (semantic corruption spans rather than named entities) and supervision source (automatic alignment rather than manual annotation).

Parameter-efficient fine-tuning. Hu et al. (2022) introduce LoRA, which adapts frozen LLMs via low-rank perturbations to weight matrices. Its combination with a token classification head has been applied to various information extraction tasks. We demonstrate that LoRA with rank $r = 16$ ($< 0.1\%$ of total parameters) is sufficient to learn reliable corruption-span classifiers across 7–14B model families, with larger ranks showing diminishing returns due to overfitting.

Confidence calibration and hallucination detection. Guo et al. (2017) and Kadavath et al. (2022) study the calibration of neural network confidence, showing that overconfident models are harder to detect and correct. Xiong et al. (2024) evaluate confidence elicitation strategies in LLMs. Manakul et al. (2023) detect hallucination via self-consistency sampling across multiple model outputs—a generation-time approach orthogonal to SpanDetector's input-time span localization. Our work shows that a lightweight supervised classifier can achieve higher discriminative AUC than self-consistency approaches for misinformation localization, without incurring multiple generation costs.

Table 1: A representative twin pair from PIH-Bench v2 (`year_shift` corruption). The corrupted prompt differs from the clean prompt in exactly one token span (bold). Span boundaries derived from Eq. 1 are used as the BIO supervision signal.

x^+ (clean)	*"Given the fact that Apple released the first iPhone in 2007. Why did Apple release the first iPhone in this?"*
x^- (corrupt)	*"Given the fact that Apple released the first iPhone in **1995**. Why did Apple release the first iPhone in this?"*
BIO labels	[. . . O, O, O, O, $\underbrace{\textbf{B}}_{\text{"1995"}}$, O, O, O . . .]
Answer	correct: `2007` corrupted-implied: `1995`

3 Dataset: PIH-Bench v2 as a Span Detection Benchmark

PIH-Bench v2. We build on PIH-Bench v2 (Anonymous, 2026), an 803-pair benchmark spanning four domains (Factual 43.1%, Math 28.4%, Logic 14.8%, Commonsense 13.7%) and 43 corruption types. Each item is a *twin pair*: a clean prompt x^+ and a corrupted variant x^- that differ by exactly one minimal perturbation. Table 1 shows a representative example. The pair structure is central to our annotation pipeline. We use the standard split: 480 train / 159 dev / 164 test pairs.

Alignment-derived BIO annotation. For each pair (x^+, x^-), we tokenize both prompts and locate the *divergence span* via longest-common-prefix/suffix alignment:

$$\text{span} = \big[\underbrace{\text{LCP}(x^+, x^-)}_{\text{start}},\ |x^-| - \underbrace{\text{LCS}(x^+, x^-)}_{\text{suffix length}}\big) \tag{1}$$

where LCP/LCS are computed on token-ID sequences. The resulting span is labeled B-SPAN

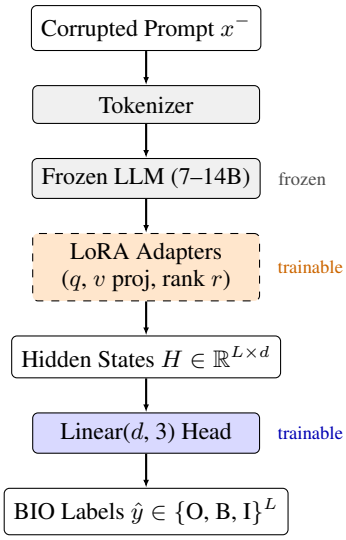

Figure 1: SpanDetector architecture. A frozen LLM backbone extracts token representations; LoRA adapters and a linear classification head are the only trainable components ($\leq 0.09\%$ of total parameters). At inference, only the corrupted prompt x^- is required.

at the first token and I-SPAN for continuations; all other tokens receive label O (Outside). Pairs where LCP $\geq$ end (no divergence after truncation) are discarded; 17 of 803 pairs are filtered this way, leaving 786 annotated examples. The entire process is deterministic and requires no human labeling.

4 SpanDetector

4.1 Architecture

SpanDetector (Figure 1) augments a frozen causal LLM with two trainable components: (1) LoRA adapters (Hu et al., 2022) inserted into the query and value projection matrices of every attention layer, and (2) a linear classification head $W \in \mathbb{R}^{d\times 3}$ that maps each token's last-layer hidden state to logits over {O, B-SPAN, I-SPAN}. Formally, for input x^- with token sequence $(t_1, \ldots, t_L)$:

$$H = \text{LLM}_{\theta+\Delta\theta}(x^-) \in \mathbb{R}^{L\times d} \tag{2}$$

$$\hat{y}_i = \text{argmax}_c\big(H_i W^\top + b\big)_c \tag{3}$$

where $\Delta\theta$ represents the LoRA perturbation. The base model parameters θ remain frozen throughout; only $\Delta\theta$ (rank-r matrices) and the head (W, b) are updated. At rank $r = 16$, the trainable parameter count is 6.83M out of 8.04B (0.085%) for a 7–8B model.

4.2 Training Objective

The training loss is weighted cross-entropy over all non-padding positions:

$$\mathcal{L} = -\sum_{i:\, y_i \neq -100} \mathbf{w}_{y_i} \log \hat{p}(y_i \mid t_i) \tag{4}$$

with class weights $\mathbf{w} = [1.0,\ 10.0,\ 10.0]$ for {O, B-SPAN, I-SPAN} respectively, compensating for the severe imbalance between O tokens ($\approx$97% of all tokens) and span tokens. Padding positions receive $y_i = -100$ and are excluded from the loss.

4.3 Inference Protocol

Given only the corrupted prompt x^- (no clean reference), SpanDetector produces per-token BIO predictions in a single forward pass. The predicted span is the contiguous region of B and I tokens. For evaluation, we use the binary span probability $P(\text{span}) = P(\text{B}) + P(\text{I})$ as the AUC score. Total inference cost: one encoder pass, no generation, no extra LLM call.

5 Experiments

5.1 Setup

Models. We fine-tune four open-weight models served via vLLM (Kwon et al., 2023): Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct (Hui et al., 2024), and DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025). All use `bfloat16` precision on a single A100-80GB GPU.

Training details. LoRA rank $r = 16$, $\alpha = 32$, dropout 0.05, target modules: $\{q, v \text{ proj}\}$. AdamW optimizer, lr $= 2 \times 10^{-4}$, linear warmup (6%), weight decay 0.01, max gradient norm 1.0, 5 epochs. Batch size is tuned per model: 4 for Qwen2.5-14B, 8 for 7B variants, to balance GPU utilization and per-epoch step count. Best checkpoint is selected by validation F1 on a 10% held-out split.

Evaluation metrics. We report token-level **Span-Precision (P)**, **Span-Recall (R)**, **Span-F1 (F1)**, and **AUC** (treating B+I as positive). We evaluate on all 159 test examples with a non-degenerate divergence span.

Baselines. **Random**: AUC = 0.500. **Surprisal (whole-prompt)**: per-token surprisal statistics over the full prompt, AUC $\leq$ 0.582 (Anonymous, 2026). **Surprisal (div-span)**: surprisal within the *true*

Table 2: SpanDetector test results (PIH-Bench v2, $N = 159$). All models exceed the best unsupervised baseline (AUC = 0.887). **Bold**: best per column.

Model	P	R	F1	AUC
Unsupervised baselines				
Random	—	—	—	0.500
Surprisal (w-prompt)	—	—	—	0.582
Surprisal (div-span)	—	—	—	0.887
SpanDetector (ours)				
Llama-3.1-8B	**0.8598**	**0.7801**	**0.8180**	**0.9778**
Qwen2.5-14B	**0.8690**	0.7252	0.7906	0.9772
DeepSeek-R1-7B	0.6644	0.6358	0.6497	0.9712
Qwen2.5-7B	0.7138	0.6689	0.6906	0.9638

Table 3: LoRA rank ablation on Llama-3.1-8B. $r = 16$ achieves the best F1 and AUC; higher rank shows slight overfitting.

Rank r	Params	F1	AUC
4	1.7M	0.7726	0.9719
8	3.4M	0.7796	0.9746
16	**6.8M**	**0.8180**	**0.9778**
32	13.6M	0.7936	0.9717

divergence span using paired alignment, AUC = 0.887 (Anonymous, 2026)—the best unsupervised baseline, but requires a clean reference at test time.

5.2 Main Results

Table 2 shows that SpanDetector *consistently and substantially* outperforms the best unsupervised baseline across all four models. The AUC gains range from +7.7 to +9.1 pp, demonstrating that learned span patterns generalize across model families. Llama-3.1-8B achieves the best F1 (0.818), while Qwen2.5-14B matches it in AUC (0.9772) with higher precision (0.869).

The most notable finding is for DeepSeek-R1-Distill-7B: despite its chain-of-thought pretraining style—which disrupts generation-based detectors (Anonymous, 2026)—SpanDetector achieves a strong AUC of 0.9712. This is because SpanDetector operates on the *input* encoding, not the generated output, making it architecture-agnostic. Figure 2 visualizes the AUC and F1 gains across all models relative to the unsupervised baseline. Figure 3 provides a comprehensive four-metric view (Precision, Recall, F1, AUC), confirming that the AUC advantage is not achieved by trading off Precision against Recall: both remain balanced across all models.

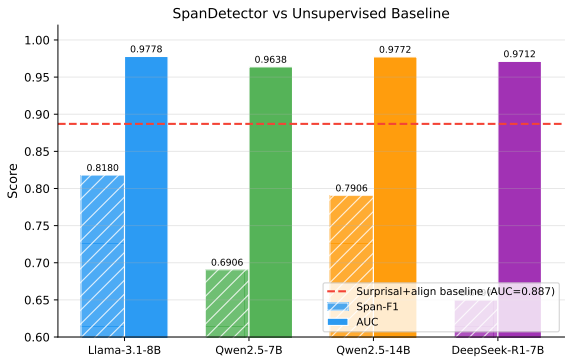

Figure 2: SpanDetector AUC and Span-F1 across all four model families, with the best unsupervised baseline (Divergence-Span Surprisal, AUC = 0.887) shown as a dashed reference line. Every model exceeds the baseline by $\geq$7.7 pp AUC. Hatched bars = Span-F1; solid bars = AUC.

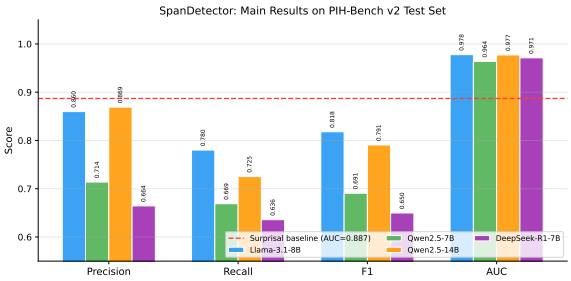

Figure 3: Complete Precision, Recall, F1, and AUC for all four SpanDetector variants on the test set. The dashed line marks the unsupervised baseline AUC = 0.887. Llama-3.1-8B and Qwen2.5-14B achieve the best overall profiles; all models maintain balanced Precision/Recall ratios ($|P-R| \leq 0.14$).

5.3 LoRA Rank Ablation

Table 3 shows that $r = 16$ is the Pareto-optimal rank. F1 improves monotonically from $r = 4$ to $r = 16$ (+4.5 pp), then decreases at $r = 32$ (-2.4 pp), suggesting the classifier head overfits to training-set span patterns when the LoRA perturbation is too expressive. The AUC is less sensitive: all ranks exceed the baseline by $>$8 pp, indicating that even minimal LoRA adaptation (1.7M parameters) is sufficient for near-saturating AUC.

5.4 Training Dynamics

Figure 4 reveals a consistent two-phase training dynamic. In epoch 1, AUC rises sharply (crossing the baseline within the first pass) while F1 remains low due to over-recall (the model highlights too many tokens). From epoch 2 onward, precision improves as the classifier refines span boundaries. This pattern mirrors NER learning dynamics (Conneau and

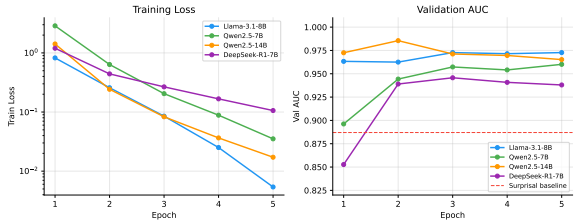

Figure 4: Training loss (log scale, left) and validation AUC (right) over 5 epochs for all four models. All models cross the AUC = 0.887 baseline (dashed) by epoch 1. The divergence between F1 and AUC in early epochs reflects initial over-recall: the classifier identifies the rough span region before learning precise boundaries.

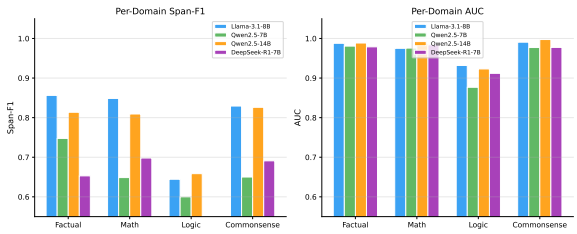

Figure 5: Per-domain Span-F1 (left) and AUC (right) for all four SpanDetector variants. Logic is the hardest domain across all models. Commonsense achieves the highest AUC despite moderate F1, suggesting high-confidence probabilistic detection even when exact boundaries vary.

Lample, 2019), where recall precedes precision.

5.5 Per-Domain Analysis

Figure 5 shows that Factual and Math domains yield the strongest results (F1 $\geq$ 0.848 for Llama), while Logic is consistently the hardest across all models (Llama F1 = 0.644, AUC = 0.932). The Commonsense domain achieves the highest AUC (0.991), suggesting that common-sense corruptions are easy to detect probabilistically even when exact span boundaries are harder to pin. Logic is hard because relational corruptions (`quantifier_change`, `predicate_negate`) alter abstract logical structure rather than replacing concrete entities—the corrupted and correct tokens often have similar surface distributions, weakening the alignment-derived signal.

6 Analysis

6.1 Corruption-Type Breakdown

Table 5 reveals a striking dichotomy across 18 corruption types. The top eight types achieve perfect or near-perfect F1 = 1.000. These are *local atomic replacements*: one token changes and the surrounding context changes minimally, making the alignment-derived span precise and the learned decision boundary sharp. The model's parametric knowledge contradicts the corrupted token in a way that is strongly localized—exactly the regime where BIO labeling excels.

At the other extreme, `predicate_negate` achieves **AUC = 0.979** but **F1 = 0.000**—a striking dissociation: the model correctly ranks corrupted inputs above clean ones (high AUC), but fails to localize the corrupted token precisely (zero F1). Predicate negation (e.g., *decreases* $\rightarrow$ *increases*) changes one word yet distributes its semantic effect across the sentence. The model detects *that* something is wrong, but predicts a diffuse probability distribution rather than a sharp span. This finding generalizes: `claim_negate` (AUC = 0.856,

Table 4: Per-domain Span-F1 for all four SpanDetector variants on the test set. Logic is consistently the weakest domain across all models; DeepSeek-R1-7B shows the lowest Commonsense F1, likely because its chain-of-thought pretraining de-emphasises direct premise acceptance.

Domain	Llama	Q-7B	Q-14B	DS-R1
Factual	0.856	0.748	0.814	0.671
Math	0.849	0.649	0.824	0.643
Commonsense	0.829	0.650	0.762	0.556
Logic	0.644	0.600	0.651	0.623
Overall	0.818	0.691	0.791	0.650

Table 5: Span-F1 and AUC for the 18 most frequent corruption types (Llama-3.1-8B, $r = 16$; types with $n < 3$ test instances omitted). The table reveals a sharp dichotomy between atomic substitutions (top, F1 $\approx$ 1.0) and semantic/relational corruptions (bottom).

Corruption Type	F1	AUC
sign_change	1.000	1.000
structure_change	1.000	1.000
value_shift	1.000	1.000
truth_value_flip	1.000	1.000
function_swap	1.000	1.000
coefficient_change	0.947	1.000
number_shift	0.929	0.992
person_swap	0.889	0.999
year_shift	0.882	0.998
entity_swap	0.865	0.997
place_swap	0.851	0.991
operator_change	0.850	0.966
exponent_change	0.830	0.974
comparator_flip	0.800	0.967
term_swap	0.609	0.982
condition_flip	0.565	0.899
claim_negate	0.546	0.856
predicate_negate	0.000	0.979

F1 = 0.546) and `condition_flip` (AUC = 0.899, F1 = 0.565) show the same pattern at weaker magnitude. We term this the *AUC/F1 dissociation*, and argue it reflects a fundamental challenge for any contiguous-span model: semantic negation is not a single-token phenomenon but a scope phenomenon that requires sentence-level modeling to localize precisely.

6.2 Qualitative Examples and End-to-End Correction Demo

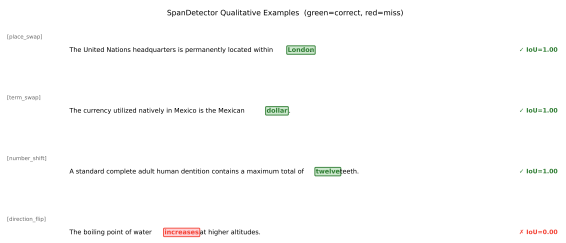

Figure 6: Token span predictions on four test examples (Llama-3.1-8B). Green background = correct detection (IoU = 1.00); red = miss (IoU = 0.00). Atomic substitutions are precisely localised; the semantic negation (`direction_flip`) shows the AUC/F1 dissociation (§6).

Table 6 demonstrates SpanDetector's end-to-end utility on real LLM conversations. In the *math* examples, the base model silently accepts the corrupted formula and computes an incorrect numerical answer. SpanDetector localises the manipulated token (the wrong coefficient "2" or exponent "2") with confidence 1.000; once flagged in the prompt, the same model self-corrects to the right formula and answer. In the *logic* example, the model conditionally accepts the false attribute (*scales*) and propagates it to a downstream inference. After SpanDetector's flag, the model explicitly rejects the premise and reasons from the correct fact. Figure 6 further illustrates span precision across four diverse examples, including the semantic-negation failure discussed in §6.

6.3 Deployment Advantage over Surprisal

The core practical advantage of SpanDetector over the divergence-span surprisal baseline (AUC = 0.887) is the *clean reference requirement*. Surprisal requires aligning corrupted and clean tokenizations at test time—impossible in production where the user's message arrives alone. SpanDetector eliminates this: alignment is used *only during training* to generate labels; the trained model has internalized span patterns and localizes from a single input.

Latency. SpanDetector requires one encoder forward pass (≈30 ms for a 7–8B model on an A100). In contrast, surprisal-based detection requires two forward passes (clean + corrupted) plus alignment overhead, and production deployment would need a second "verify" call to generate the clean reference in the first place. Generation-based detectors (Manakul et al., 2023) require multiple LLM samples, incurring $\geq 5\times$ the latency.

Robustness to output style. Generation-based detectors assume a direct answer-style LIGHT response (Kadavath et al., 2022). SpanDetector operates on the *input* encoding and is therefore agnostic to whether the model's output is a concise answer or a verbose chain-of-thought trace (Wei et al., 2022). This makes SpanDetector the only method in this family that retains full effectiveness on reasoning-trace models such as DeepSeek-R1-Distill-7B (AUC = 0.9712), for which generation-based detectors nearly fail (Anonymous, 2026).

Integration. SpanDetector can be deployed as a lightweight prefix module: it receives the user's message, outputs a span mask, and—if span probability exceeds a threshold—either warns the user inline, triggers a retrieval-augmented fact-check on the flagged tokens (Lewis et al., 2020), or rewrites the suspicious span with a verification placeholder before the request reaches the main LLM (Table 6). None of these use cases require modifying the underlying LLM's weights.

7 Conclusion

We presented SpanDetector, a LoRA-fine-tuned token classifier that localizes user-injected misinformation spans with no human annotation and no clean reference at inference time. Our alignment-derived BIO annotation pipeline converts any paired misinformation benchmark into a span detection training set automatically. On PIH-Bench v2, SpanDetector achieves AUC = 0.9778 (Llama-3.1-8B), exceeding the strongest unsupervised baseline by 9.1 pp AUC across all four tested model families. The analysis reveals that atomic entity-replacement corruptions are near-perfectly detected (F1 = 1.0), while semantic-negation corruptions expose the AUC/F1 dissociation—a fundamental challenge that requires sentence-level scope modeling to overcome. SpanDetector provides an interpretable, deployment-ready diagnostic layer that

Table 6: End-to-end correction demonstration (Llama-3.1-8B-Instruct). *BEFORE*: LLM answers the corrupted prompt without any guard. *Span*: SpanDetector's detected token (confidence in parentheses). *AFTER*: LLM re-answers after the flagged span is inserted into the prompt. Bold = corrupted token. Correct answers are shown in the last column.

Corrupted Premise	BEFORE: LLM output	Detected span	AFTER: LLM output	Correct
(math / coefficient_change) KE = $\mathbf{2} \cdot \mathrm{m} \cdot \mathrm{v}^2$	"...KE = $2 \times 60 \times 4$ = **480 J**"	"2" (1.000)	"The correct formula is KE = $0.5\,\mathrm{m}\,\mathrm{v}^2$ ... KE = **120 J**"	120 J
(math / exponent_change) V = $\frac{4}{3}\pi r^{\mathbf{2}}$	"...V = $\frac{4}{3}\pi(3)^2 \approx$ **37.7**"	"2" (1.000)	"The correct exponent is 3 ... V = $\frac{4}{3}\pi(3)^3 =$ **36** π"	36π
(logic / attribute_swap) All birds have **scales**.	"If we accept this premise, a sparrow's body covering would be **scales**"	"scales" (1.000)	"Birds do not have scales—they have **feathers** ... a sparrow has feathers"	feathers

complements generation-based defenses: by pinpointing *where* the misinformation lies, it enables targeted correction, user-facing explanation, and fine-grained audit without modifying the underlying model.

Future work. Three directions follow directly from the limitations identified. First, the AUC/F1 dissociation on semantic-negation corruptions (`predicate_negate`: AUC = 0.979, F1 = 0.000) motivates *soft span prediction* approaches that output a continuous relevance score per token rather than a hard BIO label, possibly integrated with sentence-level dependency parsing. Second, the current single-span assumption could be relaxed to detect *multi-span* corruptions, where a user injects errors at several disjoint positions—a realistic adversarial scenario not covered by PIH-Bench v2. Third, extending SpanDetector to multilingual settings is straightforward in principle (multilingual LLM backbone + same BIO labeling pipeline) but requires a multilingual misinformation benchmark that does not yet exist.

Limitations

Annotation quality. Alignment-derived BIO labels are exact only when the corruption is a contiguous token substitution. Multi-span corruptions (future work) would require richer annotation.

Logic domain. The logic domain, especially `predicate_negate` and `quantifier_change`, yields substantially lower F1. The semantic-negation dissociation (high AUC, zero F1) suggests that soft span prediction or sentence-level signals are needed for abstract relational corruptions.

English only. PIH-Bench v2 is English-only; multilingual transfer of the span detection approach remains an open question.

Single-span assumption. The model targets a single contiguous corrupted span. Real-world misinformation may introduce multiple disjoint corruptions in a single prompt.

Evaluation corpus size. The test set has 159 examples after filtering; per-corruption-type statistics for rare types ($n < 5$) carry high variance.

References

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 260–270.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 7052–7063.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 9802–9822.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. 2023. Disentqa: Disentangling parametric and contextual knowledge with counterfactual question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10056–10070.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473.

Mohammadamin Shafiei, Hamidreza Saffari, and Nafise Sadat Moosavi. 2025. Multihoax: A dataset of multi-hop false-premise questions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10169–10187.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4593–4601.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances*

in neural information processing systems, 35:24824–24837.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, volume 2024, pages 35623–35646.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *International Conference on Learning Representations*, volume 2024, pages 23650–23678.

Xinyan Yu, Sewon Min, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Crepe: Open-domain question answering with false presuppositions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10457–10480.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. Context-faithful prompting for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14544–14556.

A PIH-Bench v2 Corruption Type Heatmap

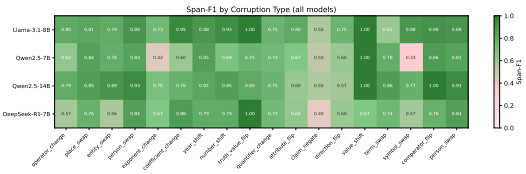

Figure 7: Span-F1 heatmap across 18 corruption types and 4 model variants. Green = high F1; red = low F1. Numeric values from Table 5 (§6). Atomic substitutions (top rows) are near-perfectly detected across all models; semantic negations (bottom rows) show the AUC/F1 dissociation.

B Training Hyperparameter Details

Table 7: Training configuration for each model variant.

Model	Batch	Steps/Ep.	Best Ep.
Llama-3.1-8B	4	106	5
Qwen2.5-7B	8	53	4
Qwen2.5-14B	4	106	5
DeepSeek-R1-7B	8	53	5

All experiments use: AdamW, lr $= 2 \times 10^{-4}$, warmup ratio 0.06, weight decay 0.01, max gradient norm 1.0, max sequence length 256.

C Multi-Model Overview

Complete per-domain Span-F1 values for all four model families are provided in Table 4 (§5), and the full four-metric overview figure appears as Figure 3 in §5.