SPANUQ: Span-Level Uncertainty Quantification for Large Language Model Generation

Yimeng Zhang[1]*, **Yingying Zhuang**[1], **Ziyi Wang**[2], **Yuxuan Lu**[2], **Pei Chen**[1], **Aman Gupta**[1], **Zhe Su**[1], **Ming Tan**[1], **Zhilin Zhang**[1], **Qun Liu**[1], **Manikandarajan Ramanathan**[1], **Rajashekar Maragoud**[1], **Edward Vul**[1], **Jing Huang**[1], **Dakuo Wang**[2]

[1]Amazon, [2]Northeastern University

Abstract

Uncertainty estimation is essential for the trustworthy deployment of large language models (LLMs): it underpins self-refinement and serves as a decision signal for downstream systems, from verifiers to collaborating agents. However, existing approaches operate at suboptimal granularities: token-level scores lack semantic coherence, while sequence-level scores fail to localize errors. We formalize *Span-Level Uncertainty Estimation* (SLUE), a new task that targets the natural granularity for uncertainty: semantically coherent text spans, each conveying a single assessable unit of meaning. To address this task, we introduce SPANUQ, a lightweight ($\sim$25M parameter) probe that distills the uncertainty knowledge from expensive multi-sample inference into a single forward pass over LLM hidden states. SPANUQ employs a DETR-style span decoder to simultaneously detect spans and estimate their uncertainty via a Mixture of Beta distribution, trained with a principled combination of Beta NLL regression and contrastive ranking objectives. We construct SPANUQ-BENCH, the first span-level uncertainty benchmark comprising 20K prompts, $\sim$293K annotated spans, and continuous soft labels derived from multi-sample claim verification. Experiments on five LLM backbones show that SPANUQ consistently achieves the best span-level uncertainty quality (AUROC 0.908–0.944, MAE 0.110–0.129) , outperforming the strongest probe baseline and all sampling-based methods while being 10–20$\times$ faster. Its DETR-based span detector attains 0.910 F1, surpassing the best heuristic by 39.4%, enabling precise error localization that sequence-level methods cannot provide. The same architecture ports to five LLMs spanning two model families, with one probe trained per backbone (AUROC 0.908–0.944), and we additionally observe that sequence-level uncertainty is partially decomposable: the learned importance-weighted span composition achieves $\rho_{\text{seq}} = 0.839$, suggesting that span-level estimation subsumes sequence-level as a special case. The project page is available damon-demon.github.io/SpanUQ.

1 Introduction

Large language models (LLMs) generate fluent and coherent text across a wide range of tasks, yet their propensity to produce factually incorrect statements, commonly termed hallucinations, remains a critical barrier to deployment in high-stakes domains such as healthcare, legal analysis, and scientific research [1]. A fundamental step toward trustworthy LLM deployment is the ability to estimate *how uncertain* the model is about each piece of information it generates. Existing approaches to uncertainty estimation operate at two extremes of granularity. *Token-level methods* compute scores for individual tokens using predictive entropy or learned probes [2–4]. While computationally

*Corresponding author: damonzym@amazon.com. Website: https://damon-demon.github.io/

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

efficient, these scores are semantically incomplete: a single token rarely constitutes a verifiable fact, and function words receive scores that carry no informational value. *Sequence-level methods* produce a single score for the entire response by sampling multiple outputs and measuring their semantic dispersion [5–7]. These methods capture meaningful semantic uncertainty but require 10–20× inference cost and cannot localize *which parts* of a response are unreliable. Consider the example in **Fig. 1**: an LLM generates *"Marie Curie was a Polish physicist who won the Nobel Prize in 1901."* A token-level method (a) assigns separate scores to every token; function words like "was" and "a" receive low but meaningless scores, while "physicist" receives a moderate score despite being correct, simply because it is a less predictable continuation. A sequence-level method (b) collapses the entire response into a single score (u=0.52), averaging correct and incorrect claims into an unactionable middle ground. What users actually need is uncertainty at the level of **spans** (c): "Polish physicist" (u=0.08, confident), "the Nobel Prize" (u=0.12, confident), and "1901" (u=0.81, uncertain), since the Nobel Prize in Physics was awarded in 1903, not 1901. Such localized scores let downstream components (a verifier, a refinement step, or another agent) act on exactly the claims that need attention. We define a **span** as a contiguous text segment that conveys a single, coherent unit of meaning whose uncertainty can be independently assessed. In practice, spans often correspond to what the fact-checking literature calls *claims* or *atomic facts* [8], but the concept is more general: any sub-sentential fragment carrying a self-contained piece of information qualifies. In the example above, "Polish physicist", "the Nobel Prize", and "1901" are each a span. Unlike tokens, which are sub-word units without standalone meaning, and unlike full sequences, which may bundle multiple assertions of varying reliability, spans are the natural unit for uncertainty estimation: each carries exactly one assessable piece of information, making its uncertainty score directly interpretable and actionable. This observation leads us to formalize **Span-Level Uncertainty Estimation (SLUE)** as a new task: given a single LLM forward pass, jointly detect spans and assign continuous uncertainty scores to each. To address this task, we introduce SPANUQ (**Fig. 2**), a lightweight (∼25M-parameter) probe that *distills* the uncertainty knowledge captured by expensive multi-sample inference into a single forward pass over frozen LLM hidden states. We construct SPANUQ-BENCH, the first span-level uncertainty benchmark with 20K queries and 293K spans across five domains, and evaluate on five LLMs (Qwen3-14B/8B/4B/30B-A3B, Mistral-7B) spanning 4B–30B parameters with both dense and mixture-of-experts architectures.

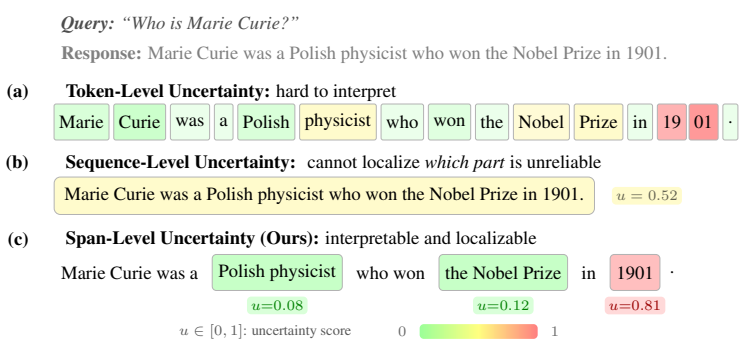

Figure 1: Comparison of uncertainty estimation granularities for an LLM response containing a factual error (the Nobel Prize in Physics was awarded in 1903, not 1901). (a) Token-level methods assign per-token scores that are noisy and hard to interpret. (b) Sequence-level methods produce a single score, unable to localize which part is unreliable. (c) Our *span-level* approach identifies claim spans and assigns each an uncertainty score, enabling precise hallucination localization.

Our **main contributions** are as follows. **(1) Problem Formulation.** We formalize SLUE as a new task that bridges the gap between token-level and sequence-level uncertainty estimation, operating at the natural granularity of verifiable claims. **(2) Framework.** We introduce SPANUQ, combining a DETR-style span decoder, Mixture of Beta uncertainty estimation, and Uncertainty-Conditioned Iterative Refinement into a lightweight probe (∼25M parameters) that distills multi-sample uncertainty into a single forward pass. **(3) Benchmark.** We construct SPANUQ-BENCH, the first span-level uncertainty benchmark with continuous soft labels, spanning five domains and 20K queries, together with a multi-sample distillation pipeline that automatically decomposes LLM responses into atomic spans and derives continuous uncertainty labels. **(4) Empirical Results.** Experiments on five LLM backbones show that SPANUQ consistently achieves the best span-level uncertainty quality (AUROC 0.908–0.944, MAE 0.110–0.129), outperforming the strongest probe baseline and all sampling-based methods while being 10–20× faster. Additionally, we observe that sequence-level uncertainty is partially decomposable into span-level components ($\rho_{\text{seq}} = 0.839$), suggesting that span-level estimation subsumes sequence-level as a special case.

2 Related Work

Uncertainty Estimation in LLM Generation. Uncertainty estimation for LLMs has been studied at multiple granularities. At the *token level*, predictive entropy serves as a natural baseline [2], and lightweight hidden-state probes can improve upon it [3, 9]. However, factual correctness is rarely a property of individual tokens: a hallucinated claim such as "*born in Paris*" is incorrect as a unit, yet token-level scores assign independent values to each word. To capture broader semantics, *sequence-level* methods aggregate uncertainty over entire generations. Semantic Entropy [5] clusters sampled outputs by meaning and computes entropy over the clusters; Kernel Language Entropy [10] replaces discrete clustering with kernel-based similarity. Other directions include epistemic–aleatoric decomposition [11], claim-level entailment graphs [12], and self-calibration via verbalized confidence [7]. *Sentence-level* approaches [13] partially localize uncertainty but rely on syntactic boundaries that may not align with semantic units. UncertaintyRAG [14] takes a step toward span-level estimation by computing a signal-to-noise ratio (SNR) over token self-information within fixed sliding windows; however, its spans are defined by rigid windows (length 20, stride 10) rather than learned semantic boundaries, and the resulting scores are used for RAG chunk retrieval rather than generation-quality assessment. All of these methods produce either a single score per generation or per-token scores, leaving a gap at the *span level*, the natural granularity of factual claims. SPANUQ fills this gap by jointly detecting variable-length spans and assigning continuous uncertainty scores to each.

Hallucination Detection. A parallel line of work asks whether LLM internal representations encode truthfulness signals. Hidden-state probes can detect hallucinations invisible to output-based methods [15, 16], and Semantic Entropy Probes [17] distill multi-sample signals into single-pass classifiers. These approaches establish that hidden states are rich sources of factuality information, but they operate at the token or sentence level and produce binary rather than continuous estimates. From the output side, external-knowledge methods such as FActScore [8] and SAFE [18] decompose generations into atomic claims and verify each against a knowledge source, while SelfCheckGPT [19] measures cross-sample consistency without retrieval. FAVA [20] fine-tunes a model to detect and edit hallucinated spans, and the Mu-SHROOM shared task [21] benchmarks span-level hallucination detection, though participating systems rely on heuristic span extraction. A key distinction is that hallucination detection is binary and often requires external verification, whereas uncertainty estimation produces continuous confidence scores from internal signals alone. SPANUQ bridges the two by learning continuous uncertainty from hidden states, achieving AUROC $>$0.93 for binary hallucination prediction as a byproduct.

Span Detection. Span detection has a rich history, from named entity recognition [22] to coreference resolution [23] and relation extraction [24]. Traditional BIO labeling cannot represent overlapping spans, a limitation for uncertainty estimation where multiple claims may coexist within a sentence. Recent set-prediction approaches inspired by DETR [25] address this: CLUE [26] uses a DETR-style architecture for claim-level uncertainty, and HaluNet [27] applies a similar paradigm to hallucination span detection. SPANUQ adopts the DETR framework but differs in two respects: (1) we detect *uncertainty-relevant* spans, i.e., any semantically complete unit whose factuality can be assessed, rather than named entities or predefined claim types; and (2) the span detector is jointly trained with the uncertainty scorer, learning boundaries that are maximally informative for uncertainty estimation.

3 Method: SPANUQ

We present SPANUQ, a framework that estimates uncertainty at the span level from a single forward pass through a frozen large language model. The core idea is to *distill* the uncertainty knowledge captured by expensive multi-sample inference into a lightweight module that operates directly on LLM hidden states. Given a response $\boldsymbol{y} = (y_1, \ldots, y_T)$ generated by an LLM for an input prompt $\boldsymbol{x}$, SPANUQ simultaneously (i) detects semantically meaningful spans $\{s_k = (b_k, e_k)\}_{k=1}^{K}$ and (ii) estimates a continuous uncertainty score $u_k \in [0, 1]$ for each span, where 0 denotes full confidence and 1 denotes maximal uncertainty. **Fig. 2** provides an overview and **Fig. A1** illustrates the end-to-end span-level uncertainty estimation process with a concrete example (see **App. A** for full design rationale). Training requires ground-truth spans and uncertainty labels: each span corresponds to a contiguous text segment expressing a single verifiable assertion, and its label $u_k^* \in [0, 1]$ measures the fraction of stochastic samples in which the assertion is unsupported (details in **Sec. 4.1**). We first describe the model architecture (**Sec. 3.1**), then the training procedure (**Sec. 3.2**) and inference with iterative refinement (**Sec. 3.3**).

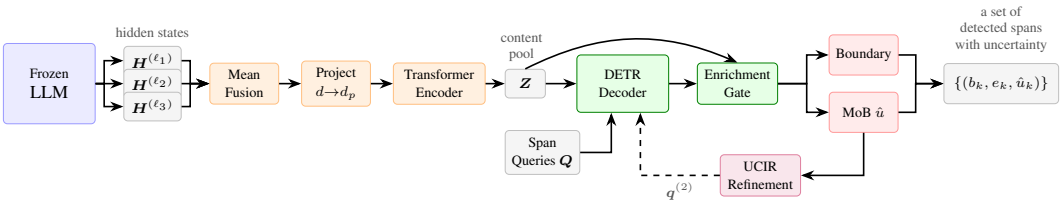

Figure 2: Overview of the SPANUQ architecture. Given a frozen LLM's hidden states from three selected layers, the probe fuses, projects, and encodes token representations, then uses a DETR-style decoder with learnable span queries to simultaneously detect spans and estimate their uncertainty via Mixture of Beta (MoB) heads. Uncertainty-Conditioned Iterative Refinement (UCIR, dashed) feeds Round-1 estimates back through the shared decoder for a second pass. Each output tuple $(b_k, e_k, \hat{u}_k)$ denotes the beginning position, end position, and estimated uncertainty of a detected span. The trainable probe adds only $\sim$25M parameters ($<$0.2% of the LLM).

3.1 Model Architecture

The SPANUQ module is a lightweight probe ($\sim$25M parameters) attached to a frozen LLM backbone. It consists of five components: multi-layer fusion, a token encoder, a DETR-style span decoder, span feature enrichment, and prediction heads.

Multi-Layer Fusion. Different LLM layers encode different aspects of uncertainty: early layers capture syntax while mid-to-late layers encode factual knowledge [16]. Our layer probing study reveals that uncertainty information peaks in layers 20–26 of Qwen3-14B (40 total), following an inverted-U curve. We select three clustered layers from this peak zone via a lightweight single-layer probing sweep on each backbone; for other models, the optimal zone shifts proportionally with depth (App. B.6, Tab. A14). Given hidden states $\boldsymbol{H}^{(\ell)} \in \mathbb{R}^{T \times d}$ from selected layers $\mathcal{L} = \{\ell_1, \ell_2, \ell_3\}$, we obtain the fused hidden states $\bar{\boldsymbol{H}}$ as follows:

$$\bar{\boldsymbol{H}} = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \boldsymbol{H}^{(\ell)} \in \mathbb{R}^{T \times d}. \tag{1}$$

Token Encoder. The fused hidden states $\bar{\boldsymbol{H}}$ are projected to a lower dimension d_p and contextualized through a 2-layer Transformer encoder to obtain the content pool $\boldsymbol{Z}$:

$$\boldsymbol{Z} = \text{TransformerEnc}\big(\boldsymbol{W}_{\text{proj}}\, \bar{\boldsymbol{H}} + \boldsymbol{P}\big) \in \mathbb{R}^{T \times d_p}, \tag{2}$$

where $\boldsymbol{W}_{\text{proj}} \in \mathbb{R}^{d_p \times d}$ is a linear projection and $\boldsymbol{P}$ denotes sinusoidal positional encodings.

DETR-Style Span Decoder. Inspired by DETR [25], we formulate span detection as a set prediction problem. A set of N=32 learnable *span queries* $\boldsymbol{Q} \in \mathbb{R}^{N \times d_p}$, which are content-agnostic embedding vectors that each learn to bind to one candidate span and analogous to object queries in DETR [25], attend to the token sequence through a 3-layer Transformer decoder. This capacity comfortably exceeds the typical span count per response ($\sim$15), ensuring sufficient detection slots while keeping computation tractable. The decoder produces the refined query set $\hat{\boldsymbol{Q}}$:

$$\hat{\boldsymbol{Q}} = \text{TransformerDec}(\boldsymbol{Q}, \boldsymbol{Z}) \in \mathbb{R}^{N \times d_p}, \tag{3}$$

where each layer applies self-attention among span queries (modeling inter-span dependencies) followed by cross-attention over $\boldsymbol{Z}$. This design naturally handles overlapping spans and enables parallel detection in a single pass. Following DETR, we use the Hungarian algorithm [28] to find an optimal one-to-one assignment $\hat{\sigma}$ between predicted and ground-truth spans. The matching cost for each prediction–target pair sums three terms: (i) L1 boundary distance plus Generalized IoU [29], (ii) absolute uncertainty error, and (iii) binary cross-entropy of the validity score (details in **App. D.1**).

Span Feature Enrichment. The decoder queries capture *where* a span is but lack direct access to the *content* within the predicted region. To bridge this gap, we introduce a differentiable soft boundary mask that extracts span-level content from the token features $\boldsymbol{Z}$. Given predicted boundaries $(\hat{b}_k, \hat{e}_k)$, we define a mask $m_k(t)$ over token positions:

$$m_k(t) = \sigma\big(\tau(t - \hat{b}_k)\big) \cdot \sigma\big(\tau(\hat{e}_k - t)\big), \tag{4}$$

where σ is the sigmoid function and τ controls boundary sharpness. Intuitively, $m_k(t) \approx 1$ for tokens inside the predicted span and decays smoothly to 0 outside, enabling gradient flow through the boundary predictions. We then pool token features via mask-weighted attention, yielding content vector $\boldsymbol{c}_k$:

$$\boldsymbol{c}_k = \sum_{t=1}^{T} \alpha_{k,t} \cdot \boldsymbol{z}_t, \quad \alpha_{k,t} \propto m_k(t) \cdot \exp\big(\hat{\boldsymbol{q}}_k^\top \boldsymbol{W}_a \boldsymbol{z}_t / \sqrt{d_p}\big), \tag{5}$$

where the mask $m_k(t)$ restricts attention to the predicted region while the query–token dot product selects the most informative tokens within it. The pooled content is fused with the query via a gated residual connection to produce the enriched query $\tilde{\boldsymbol{q}}_k$:

$$\tilde{\boldsymbol{q}}_k = \hat{\boldsymbol{q}}_k + \sigma\big(\boldsymbol{W}_g \hat{\boldsymbol{q}}_k + \boldsymbol{b}_g\big) \odot \boldsymbol{c}_k, \tag{6}$$

which adaptively controls how much span content contributes. This gating is especially useful for short spans where the query alone may suffice.

Prediction Heads. Each enriched query $\tilde{\boldsymbol{q}}_k$ is passed to three heads. *Span regression* predicts normalized boundaries $(\hat{b}_k, \hat{e}_k) = \sigma(\text{MLP}_{\text{reg}}(\hat{\boldsymbol{q}}_k))$ from the pre-enrichment query to avoid circular dependency. *Validity classification* predicts $p_k^{\text{valid}} = \sigma(\text{MLP}_{\text{val}}(\tilde{\boldsymbol{q}}_k))$ to distinguish real spans from unused slots (i.e., queries not matched to any span). *Mixture of Beta (MoB) uncertainty estimation* models span uncertainty as:

$$p(u_k \mid \tilde{\boldsymbol{q}}_k) = \sum_{j=1}^{J} \pi_{k,j} \cdot \text{Beta}(u_k;\, \alpha_{k,j},\, \beta_{k,j}), \tag{7}$$

with $J = 3$ components and $\alpha_{\min} = 0.5$ to allow U-shaped distributions that capture the bimodal nature of uncertainty. The predicted score is the mixture mean $\hat{u}_k = \sum_j \pi_{k,j}\alpha_{k,j}/(\alpha_{k,j} + \beta_{k,j})$. Span-level scores are aggregated to a sequence-level estimate via importance-weighted pooling:

$$\hat{u}_{\text{seq}} = \sum_{k=1}^{N} w_k \cdot \hat{u}_k, \quad w_k = \text{softmax}_k\big(p_k^{\text{valid}} \cdot \text{MLP}_{\text{imp}}(\tilde{\boldsymbol{q}}_k)\big), \tag{8}$$

providing a principled decomposition of sequence-level uncertainty into span-level components.

3.2 Training

Two-Phase Strategy. We adopt a two-phase training strategy: (1) a *warmup* phase (epochs 1–E_w) that trains only span detection components (regression + validity) to establish reliable boundaries; and (2) a *joint* phase (epochs E_w+1–E) that activates all components with the full loss:

$$\mathcal{L} = \underbrace{\lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{val}}\mathcal{L}_{\text{val}}}_{\text{span detection}} + \underbrace{\lambda_{\text{uq}}\mathcal{L}_{\text{MoB}} + \lambda_{\text{con}}\mathcal{L}_{\text{con}} + \lambda_{\text{rank}}\mathcal{L}_{\text{rank}}}_{\text{uncertainty estimation}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}. \tag{9}$$

Loss Components. The *span regression loss* $\mathcal{L}_{\text{reg}}$ combines L1 and Generalized IoU losses over Hungarian-matched span pairs. The *validity loss* $\mathcal{L}_{\text{val}}$ is binary cross-entropy over all span queries. The *MoB NLL loss* $\mathcal{L}_{\text{MoB}}$ maximizes the likelihood of ground-truth uncertainty under the predicted mixture (Eq. 7):

$$\mathcal{L}_{\text{MoB}} = -\frac{1}{M}\sum_{i=1}^{M} \log \sum_{j=1}^{J} \pi_{i,j} \cdot \text{Beta}(u^*_{\hat{\sigma}(i)};\, \alpha_{i,j},\, \beta_{i,j}), \tag{10}$$

where M is the number of matched spans and $\hat{\sigma}$ the Hungarian assignment. The *contrastive ranking loss* $\mathcal{L}_{\text{rank}}$ preserves ordinal structure among span pairs within the same sequence:

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}} \max\big(0,\, -\text{sign}(u_i^* - u_j^*) \cdot (\hat{u}_i - \hat{u}_j) + m\big), \tag{11}$$

where $\mathcal{P}$ is a set of span pairs with sufficiently different ground-truth uncertainty (see **App. D.1** for the stratified sampling strategy) and m is the margin. The *consistency loss* $\mathcal{L}_{\text{con}} = \|\hat{u}_{\text{seq}} - u^*_{\text{seq}}\|_2^2$ enforces agreement between span-aggregated and ground-truth sequence-level uncertainty (with detached gradients to prevent interference). *Auxiliary losses* $\mathcal{L}_{\text{aux}}$ apply the same objectives to intermediate decoder layers following DETR. Full formulations of all components and loss weights are in **App. D.1**.

3.3 Inference with Iterative Refinement

At inference time, SPANUQ requires only a single LLM forward pass to extract hidden states, followed by a lightweight forward pass through the SPANUQ module. No sampling, multiple forward passes, or external knowledge retrieval is needed. Since initial uncertainty estimates from a single decoder pass may be coarse, **Uncertainty-Conditioned Iterative Refinement (UCIR)** feeds Round 1 estimates back into the decoder for a refinement round:

$$\boldsymbol{q}_k^{(2)} = \boldsymbol{q}_k^{(1)} + \text{MLP}_{\text{cond}}\big(\hat{u}_k^{(1)},\, \hat{\nu}_k^{(1)},\, p_k^{(1)}\big), \tag{12}$$

where $\hat{\nu}_k^{(1)}$ is the effective precision and $p_k^{(1)}$ the validity probability. The conditioned queries pass through the *same* decoder (shared weights), giving the final uncertainty estimate $\hat{u}_k$:

$$\hat{u}_k = \alpha \cdot \hat{u}_k^{(2)} + (1-\alpha) \cdot \hat{u}_k^{(1)}, \tag{13}$$

with $\alpha = 0.7$. Both rounds contribute to the training loss (Round 1 with reduced weight). UCIR adds $<$15% overhead relative to SPANUQ's own base decoder (not to the LLM forward pass) since the decoder is shared. The **full inference pipeline**: 1) extract hidden states from layers $\mathcal{L}$; 2) fuse, project, and encode; 3) decode span queries and predict boundaries, validity, and uncertainty; 4) apply UCIR; 5) filter by validity threshold and output $\{(b_k, e_k, \hat{u}_k)\}$. The entire module adds $<$3% latency to the LLM forward pass.

4 Experiments

We evaluate SPANUQ on span-level uncertainty estimation quality, calibration, span detection accuracy, and ablation of each architectural component. All main experiments use Qwen3-14B as the backbone LLM; the full model adds only 24.4M trainable parameters ($<$0.2% of the LLM).

4.1 Dataset: SPANUQ-BENCH

Evaluating span-level uncertainty requires ground-truth annotations at the span granularity, which no existing benchmark provides. We construct SPANUQ-BENCH, a benchmark with labels derived from multi-sample distillation and validated by an independent human annotation study.

Data Collection. We collect LLM-generated texts from five domains using Qwen3-14B as the primary backbone: 1) *Long-form QA* (42%) from Natural Questions [30]; 2) *TriviaQA* (25%) from the unfiltered split [31]; 3) *ELI5* (15%) from Reddit explanatory questions; 4) *Biography generation* (10%) following the FActScore protocol [8], with entities stratified by Wikipedia popularity; and 5) *FELM* (8%) aggregating challenging examples from FELM [32], TruthfulQA [33], and HaluEval [34]. The resulting dataset contains 20k prompts spanning factoid, explanatory, and reasoning questions.

Labels via Multi-Sample Distillation. For each prompt, we generate a *primary response* with greedy decoding ($T = 0$), which serves as the probe input (providing hidden states and span positions). Separately, we sample $S = 20$ diverse responses at temperature $T = 1.0$. We decompose *the primary response* into atomic claims with character-level span offsets using an LLM judge, then ask the judge, for every claim and every one of the S samples, whether that sample SUPPORTS, CONTRADICTS, or DOES NOT MENTION the claim. The soft label $u_k^* \in [0, 1]$ is the contradiction rate among the samples that take a position on span s_k:

$$u_k^* = \frac{n_k^{\text{con}}}{n_k^{\text{sup}} + n_k^{\text{con}}}, \qquad n_k^{\text{sup}} + n_k^{\text{con}} > 0, \tag{14}$$

where n_k^{sup} and n_k^{con} count samples that support and contradict s_k. Samples that do not mention the claim are excluded rather than counted as evidence, so the pipeline abstains on silence, and spans no sample addresses are dropped (**App. C.2**). Crucially, **no retrieval or external knowledge source participates in label construction**: u_k^* is a pure cross-sample consistency statistic. The probe thus learns to *distill* the uncertainty knowledge from 20 stochastic samples into a single deterministic forward pass. For Qwen3-14B, the resulting training set contains $\sim$17,500 examples with $\sim$256K valid span-level soft labels (training split only; the full filtered benchmark contains $\sim$293K spans across all splits, see **App. C.13**). The distribution is bimodal: most spans cluster near $u \approx 0$ (consistently correct) or $u \approx 1$ (consistently wrong), with a meaningful tail of intermediate values. For evaluation, we apply the same distillation pipeline to responses generated with greedy decoding ($T = 0$).

Label Validity. Five independent annotators judged 2,000 randomly sampled spans as factually CORRECT or INCORRECT. Binarizing u^* at 0.5 gives Cohen's $\kappa = 0.618$ and 89.4% raw agreement, and human-judged error rises monotonically across u^* bins (5.8% $\rightarrow$ 87.6%; $\rho = 0.714$); these human labels are the ground truth for **Sec. 4.4**. Protocol, per-bin results, and a root-cause audit of the disagreements are in **App. C.4** and **C.6**. Data collection, the distillation pipeline, test set construction, and benchmark statistics are detailed in **App. C.1–C.13**.

4.2 Experimental Setup

Models. All core experiments use **Qwen3-14B** (dense, 14B parameters) [35] as the primary backbone. Hidden states are extracted from layers 22, 24, and 26 (out of 40 total) and fused via element-wise mean, following our layer importance probing study (**App. B.6**). The LLM is frozen throughout; only the SPANUQ module is trained. To evaluate *architectural portability*, we train a separate SPANUQ probe per backbone on four additional LLMs: Qwen3-8B, Qwen3-4B, Qwen3-30B-A3B (MoE), and Mistral-7B-Instruct-v0.3. We do not claim zero-shot cross-model transfer: each probe is trained on its own backbone's hidden states, analogous to linear probing, since backbones differ in hidden dimension and in which layers carry the signal. Each model generates its own responses and hidden states; the data construction pipeline (**Sec. 4.1**) and training procedure are identical, with layer selection and projection dimensions tuned per model via hyperparameter sweep (**App. B.6**). Full model specifications are in **App. D.2**.

Training Details. We train on 17,494 samples ($\sim$256K valid spans, training split) with batch size 16 on a single NVIDIA H100 80GB GPU. Training proceeds in two phases: a 15-epoch *warmup* phase that trains only the span detection components (token encoder, DETR decoder, regression head, validity head), followed by a 25-epoch *joint* phase that also activates the MoB uncertainty head, contrastive ranking loss, and UCIR refinement. We use AdamW with learning rate 10^{-4}, weight decay 0.01, and cosine annealing. Early stopping monitors dev AUROC with patience 5. Training curves and computational cost analysis are in **App. D.4** and **D.5**.

Evaluation Metrics. We report metrics spanning discrimination, regression, and correlation. **AUROC** measures binary hallucination detection (threshold $u \geq 0.5$). **MAE** is the mean absolute error between predicted $\hat{u}$ and ground-truth u^*. **Spearman** (ρ) correlation captures ranking quality at both span and sequence levels; we abbreviate $\rho_s \equiv \rho_{\text{span}}$ in tables. **AUROC**$_{@0.3}$ is the sequence-level AUROC treating sequences with mean uncertainty ≥ 0.3 as positive. **Detection F1** is the harmonic mean of span detection precision (**Prec**) and recall (**Rec**) under IoU ≥ 0.3 matching; this threshold applies to detection *only* — the uncertainty metrics use Hungarian-matched pairs with no IoU cutoff. **ECE** (Expected Calibration Error) measures the gap between predicted uncertainty and observed error rate.

Baselines. We compare against methods spanning four granularity levels. *Token-level*: *Token Entropy* (mean token-level entropy from output logits, aggregated over GT spans) and *MLP Probe* (3-layer MLP trained on the same span features as SPANUQ, with oracle span boundaries). *Sequence-level*: *Semantic Entropy* (SE) [5] (NLI-based clustering of 10 sampled responses), *SelfCheckGPT-NLI* [19] (NLI-based self-consistency over 10 samples), *Verbalized Confidence* (prompting the LLM to self-assess factual confidence on a 0–100 scale), *P(True)* [2] (probability the LLM assigns to "True" when asked whether its answer is correct), *INSIDE* [13] (eigenvalue decomposition of Jaccard similarity matrices over sampled responses), and *KLE* [10] (kernel-based language entropy using RBF kernels on NLI embeddings). *Claim-level*: *FActScore* [8] (fraction of claims not supported by self-verification, computed from our claim decomposition pipeline). For end-to-end evaluation (**Sec. 4.5**), we also pair the MLP probe with heuristic span detectors including sliding windows, sentence boundaries, NER entities, and token-level thresholding. For span-level evaluation (**Tab. 1**), sequence- and claim-level baselines broadcast their scores to all spans within the sequence. For sequence-level evaluation (**Tab. A1** in the Appendix), SPANUQ aggregates span predictions via importance-weighted top-k.

4.3 Main Results

Span-Level Analysis (Tab. 1). SPANUQ achieves an AUROC of **0.939**, outperforming the strongest probe baseline (MLP Probe, 0.881) by 5.8 absolute points and the best multi-sample method (SelfCheckGPT-NLI, 0.808) by 13.1 points, while requiring only a single forward pass. Sequence-level methods suffer from a fundamental granularity mismatch when broadcast to spans:

Table 1: Span-level uncertainty estimation across five LLM backbones spanning two model families, three scales, and both dense and MoE architectures. Token-level methods aggregate token scores within each span; sequence- and claim-level methods broadcast a single score to all spans in the sequence. *MLP Probe uses ground-truth span boundaries (oracle setting). Best in **bold**, second best underlined (per model).

	Qwen3-14B			Qwen3-8B			Qwen3-4B			Qwen3-30B-A3B			Mistral-7B		
Method	AUROC↑	MAE↓	ρ_s↑	AUROC↑	MAE↓	ρ_s↑	AUROC↑	MAE↓	ρ_s↑	AUROC↑	MAE↓	ρ_s↑	AUROC↑	MAE↓	ρ_s↑
Token Entropy	0.603	0.230	0.097	0.580	0.226	0.111	0.575	0.272	0.119	0.592	0.196	0.097	0.569	0.231	0.109
MLP Probe*	0.881	0.139	0.575	0.884	0.149	0.612	0.873	0.179	0.626	0.893	0.120	0.575	0.863	0.146	0.562
Verbalized Conf.	0.593	0.187	0.131	0.574	0.217	0.155	0.595	0.267	0.141	0.598	0.167	0.117	0.498	0.171	−0.026
SelfCheckGPT-NLI	0.808	0.158	0.471	0.801	0.182	0.473	0.782	0.215	0.479	0.813	0.147	0.455	0.816	0.154	0.499
P(True)	0.526	0.197	0.087	0.546	0.233	0.121	0.529	0.273	0.122	0.541	0.194	0.123	0.505	0.199	0.005
FActScore	0.625	0.178	0.181	0.762	0.197	0.405	0.714	0.233	0.390	0.701	0.169	0.306	0.661	0.182	0.297
HaloScope [36]	0.834	0.156	0.501	0.828	0.180	0.511	0.825	0.208	0.539	0.835	0.145	0.485	0.776	0.176	0.426
HaMI [37]	0.775	0.187	0.445	0.814	0.185	0.483	0.821	0.210	0.530	0.798	0.151	0.422	0.764	0.178	0.407
SPANUQ	**0.939**	**0.110**	**0.685**	**0.930**	**0.129**	**0.692**	**0.944**	**0.126**	**0.754**	**0.936**	**0.110**	**0.647**	**0.908**	**0.126**	**0.637**

Table 2: Risk–coverage on human-annotated factual labels (Qwen3-14B, seed 42). Risk is the human-judged error rate among retained spans. *MLP Probe uses ground-truth span boundaries (oracle setting) and is therefore not deployable. Best in **bold**.

Method	**Risk@100%**	**Risk@80%**	**Risk@60%**	**Risk@40%**	**AURC↓**	**PRR↑**
SPANUQ (ours)	18.6%	**10.5%**	**7.9%**	**6.3%**	**0.076**	**0.587**
MLP Probe*	18.6%	12.8%	9.4%	7.1%	0.098	0.470
Token Entropy	18.6%	15.2%	13.1%	11.8%	0.138	0.254
FActScore	18.6%	15.8%	13.9%	12.4%	0.145	0.216
SelfCheckGPT-NLI	18.6%	16.1%	14.5%	12.9%	0.151	0.184
Verbalized Conf.	18.6%	16.4%	15.0%	13.5%	0.155	0.162
P(True)	18.6%	16.8%	15.3%	14.0%	0.159	0.140
Random	18.6%	18.6%	18.6%	18.6%	0.185	—

SelfCheckGPT-NLI achieves reasonable AUROC (0.808) but assigns the same score to all spans within a sequence, limiting its span-level Spearman correlation to 0.471. FActScore (0.625 AUROC) achieves only 0.181 Spearman correlation, indicating that its binary self-verification outputs fail to capture fine-grained uncertainty ranking. P(True) performs near chance (AUROC = 0.526), because the LLM overwhelmingly predicts "True" (93.8% of claims), yielding almost no discriminative signal. SPANUQ also achieves the lowest MAE (**0.110**), owing to the Mixture of Beta output distribution (**App. B.7**). The hidden-state detectors HaloScope and HaMI are the closest prior work, but both are sequence-level and must broadcast one score to every span: their AUROC (0.820 and 0.794 on average) rivals multi-sample methods at no extra generation cost, yet their span-level ranking is capped (ρ_s 0.492 and 0.457 vs. 0.683 for SPANUQ), and SPANUQ leads HaloScope by 11.1 AUROC points on average (**App. C.8**). These trends hold consistently across all five backbones, including three Qwen3 scales (14B/8B/4B) and entity popularity strata (**App. B.5**). The Spearman correlation of **0.685** represents a 19.1% relative improvement over the MLP Probe (0.575), demonstrating that SPANUQ preserves fine-grained uncertainty ordering across spans.

Sequence-Level Aggregation. When aggregated to the sequence level (**Tab. A1** in the Appendix), SPANUQ achieves $\text{AUROC}_{@0.3} = 0.948$ and Spearman $\rho = 0.851$, competitive with or surpassing all dedicated sequence-level methods including SelfCheckGPT-NLI ($\rho = 0.805$) which requires $10\times$ generation cost. The MLP Probe with oracle span boundaries slightly outperforms SPANUQ at the sequence level ($\rho = 0.858$), but this advantage disappears when ground-truth spans are unavailable. SPANUQ's span-level superiority (AUROC: 0.939 vs. 0.881) and end-to-end detection capability (**Sec. 4.5**) make it the only practical solution for fine-grained uncertainty estimation. Notably, this span-to-sequence decomposability is consistent across all five backbones (Spearman $\rho_{\text{seq}} \geq 0.775$; **App. B.3**), confirming that fine-grained span predictions capture sufficient structure for reliable sequence-level inference. More details can be found in **App. B.1**. Qualitative examples illustrating span-level predictions are in **App. B.4**.

4.4 Human-Grounded Risk–Coverage Analysis

AUROC measures ranking quality against pipeline labels, but deployment asks a different question: if we reject the most uncertain spans, how much factual error do we actually remove? We answer it with a risk–coverage analysis whose **ground truth is the independent human annotation** (Sec. 4.1), not the pipeline label the probe was trained on: risk is the human-judged error rate among retained spans, coverage the fraction retained, AURC the area under the curve (lower is better), and PRR the fraction of the random-to-oracle gap a method closes.

Selective Prediction (Tab. 2). Rejecting only the top-20% most uncertain spans cuts the factual error rate on retained spans from 18.6% to **10.5%**, a 43.9% relative reduction, and SPANUQ attains the best AURC (0.076 vs. 0.185 random) and PRR (0.587) of all methods. An operating point can be read straight off the curve: a 10% error budget corresponds to ∼80% coverage (**App. C.9**). Because this risk reduction is measured against *human* judgments that played no role in training, it also shows the probe is not merely reproducing pipeline noise — a probe that had memorized pipeline artifacts would not yield a monotone decrease in human-judged error as its own score rises (**App. C.5**).

4.5 End-to-End Span Detection

A key advantage of SPANUQ is its ability to *simultaneously* detect uncertain spans and estimate their uncertainty, without requiring pre-defined span boundaries. **Tab. 3** compares SPANUQ's DETR-based span detector against heuristic alternatives, where each method's detected spans are scored by the same MLP probe for a controlled comparison.

Detection Quality. SPANUQ's DETR detector achieves a detection F1 of **0.910**, surpassing the best heuristic (Sliding Window 20, F1 = 0.653) by 39.4%. It maintains both high precision (0.857) and high recall (0.970), whereas heuristic methods face a precision–recall trade-off: sentence boundaries achieve high precision (0.849) but low recall (0.479), while small sliding windows achieve high recall (0.932) but low precision (0.415).

Table 3: End-to-end span detection comparison. Each method detects spans, scored by an identical MLP probe. **Bold**: best; underline: best heuristic.

Span Method	Prec↑	Rec↑	F1↑	AUROC↑	MAE↓	ρ_s ↑
Slide Win (10)	0.415	0.932	0.574	0.840	0.153	0.522
Slide Win (20)	0.614	0.699	0.653	0.853	0.152	0.540
Slide Win (40)	0.708	0.415	0.524	0.860	0.158	0.549
Sent. Boundary	0.849	0.479	0.613	0.861	0.152	0.556
Token Thr (0.7)	0.325	0.480	0.388	0.815	0.150	0.463
NER (spaCy)	0.397	0.396	0.396	0.777	0.205	0.470
BIO Tagger	0.046	0.260	0.079	0.763	0.165	0.398
SPANUQ (Ours)	**0.857**	**0.970**	**0.910**	**0.927**	**0.142**	**0.736**

Impact on Uncertainty Estimation. The downstream AUROC gap is even more striking: SPANUQ achieves **0.927** versus the best heuristic's 0.861 (+7.7%). The span-level correlation ρ_{span} shows the largest gap: **0.736** versus 0.556 (+32.4%), demonstrating that learned span detection is essential for preserving fine-grained uncertainty ordering.

BIO Tagger Failure. The BIO sequence tagger, the standard approach in span extraction tasks, performs worst (F1 = 0.079, AUROC = 0.763). This is because BIO tagging produces non-overlapping spans, but ground-truth uncertainty spans frequently overlap (a single token may belong to multiple claims). The BIO tagger detects 165K candidate spans but only 7,663 match ground truth, yielding a precision of merely 4.6%. This motivates our DETR-based set prediction approach, which naturally handles overlapping spans via parallel query decoding.

4.6 Ablation Study

We ablate each component of SPANUQ by progressively adding modules to the base DETR architecture. **Tab. 4** reports test-set results; all runs share identical hyperparameters except the ablated component.

Table 4: Component ablation. Each row adds one component to the previous configuration.

Configuration	AUROC↑	ECE↓	MAE↓	ρ_s ↑	ρ_{seq} ↑
Base DETR + Single Beta	0.907	0.058	0.154	0.673	0.765
+ Enrichment Gate	0.920	0.052	0.144	0.725	0.729
+ MoB (K=3)	0.933	0.020	0.116	0.760	0.846
+ UCIR (1 round)	**0.939**	**0.013**	**0.106**	**0.790**	**0.839**

Enrichment Gate. Adding the span enrichment gate improves AUROC by 1.3% (0.907→0.920) and ρ_{span} by 5.2% (0.673→0.725), confirming that providing the uncertainty scorer with direct access to span content is beneficial. Notably, ρ_{seq} decreases slightly (0.765→0.729): the enrichment gate sharpens span-level estimates, which can redistribute probability mass among spans within a sequence and introduce minor inconsistencies in the aggregated sequence score.

Mixture of Beta (MoB). Replacing the single Beta distribution with a K=3 mixture yields the largest single improvement: +1.3% AUROC (0.920→0.933), −62% ECE (0.052→0.020), and −19% MAE (0.144→0.116). The MoB distribution captures the multi-modal nature of uncertainty: certain spans cluster near u=0, hallucinated spans near u=1, and partially uncertain spans form an intermediate mode. A single Beta cannot represent this structure. MoB is also nearly self-calibrated (raw ECE = 0.020, T=1.05), whereas the single Beta requires aggressive temperature correction (T=0.83) to reduce its raw ECE from 0.052 to 0.029. We provide detailed analyses of layer selection strategy and MoB hyperparameters in **App. B.6** and **B.7**, along with a span enrichment analysis (gate vs. concat vs. add fusion, attention vs. mean pooling) in **App. B.8**.

Uncertainty-Conditioned Iterative Refinement (UCIR). A single UCIR round provides the second-largest improvement: +0.6% AUROC (0.933→0.939) and +3.0% ρ_{span} (0.760→0.790). UCIR allows the model to refine its uncertainty estimates by conditioning on its own initial predictions, effectively implementing a "second look" that corrects systematic errors. The shared decoder reuses detection weights with only a lightweight MLP adapter and residual connection (α=0.7), amounting to negligible parameter overhead. The improvement is most pronounced for ρ_{span}, suggesting that UCIR primarily improves the *ranking* of uncertainty estimates: conditioning on the initial estimate allows the model to identify and correct systematic biases, e.g., spans initially predicted as moderately uncertain but that should be highly uncertain based on their content. The marginal ρ_{seq} decrease (0.846→0.839) mirrors the enrichment gate pattern: span-focused refinement slightly perturbs the per-span scores that feed into sequence-level aggregation, but the effect is small (<1%) and is offset by substantial gains on all other metrics.

Regression vs. Pointer for Span Detection. **Tab. 5** compares our continuous regression head against the more common cross-attention pointer mechanism. Regression outperforms the pointer by 3.4% AUROC and 7.1% precision while maintaining identical recall (0.970). The pointer mechanism suffers from discretization artifacts: argmax over cross-attention weights cannot represent fractional positions and struggles with long spans where attention is diffuse. Regression directly predicts continuous start/end coordinates, enabling smoother gradient flow and more precise boundary localization. Additional analyses on robustness across hyperparameter configurations and sensitivity to individual hyperparameters are in **App. B.2** and **D.3**.

Table 5: Span boundary prediction: regression vs. pointer. Same DETR architecture (nq=32, enc2+dec3).

Method	AUROC↑	MAE↓	Prec↑	Rec↑	ρ_s ↑
Pointer (cross-attn)	0.888	0.167	0.767	0.970	—[†]
Regression (L1+GIoU)	**0.922**	**0.143**	**0.838**	**0.970**	**0.725**

5 Conclusion

We formalized Span-Level Uncertainty Estimation (SLUE) and introduced SPANUQ, a lightweight (∼25M-parameter) probe that distills multi-sample uncertainty into a single forward pass over frozen LLM hidden states, combining a DETR-style span decoder, Mixture of Beta head, and Uncertainty-Conditioned Iterative Refinement. Together with SPANUQ-BENCH, the first span-level uncertainty benchmark (293K spans, five domains), experiments on five LLMs (4B–30B, dense and MoE) show that SPANUQ achieves AUROC 0.908–0.944 and MAE 0.110–0.129 while being 10–20× faster than sampling-based methods. The DETR detector attains 0.910 F1, surpassing the best heuristic by 39.4%, and the learned span-to-sequence composition ($\rho_{\text{seq}} = 0.839$) suggests that span-level estimation subsumes sequence-level as a special case.

Future directions. Because span-level scores are attached to individual claims, they can serve as a decision signal for downstream systems rather than only an evaluation metric. We see three directions this enables: (i) *targeted verification*, which spends retrieval or stronger-model checks only on high-uncertainty spans; (ii) *span-level self-refinement*, which revises only the spans that need it instead of regenerating the whole response; and (iii) *uncertainty-aware multi-agent communication*, where agents pass claim-level reliability signals along with their text. We discuss these directions and their open problems in **App. C.12**.

Limitations and Broader Impacts. SPANUQ cannot capture uncertainty from the generation process itself and requires white-box access to hidden states. The multi-sample label construction pipeline is compute-intensive, though this is a one-time cost. Our evaluation focuses on claim uncertainty in English; extending to other languages and uncertainty types (e.g., reasoning) remains future work. Our labels quantify *cross-sample consistency*: u^* is the contradiction rate among a model's own samples that take a position on a claim, with no retrieval or external knowledge source entering the computation (**App. C.2**). It is therefore not a factuality label, and the two can diverge: 22.3% of high-u^* spans are factually correct, though an audit attributes ∼89% of those to genuine cross-sample disagreement rather than pipeline defects (**App. C.6**). Such labels are also blind by construction to *confident* hallucinations that recur identically across samples. Our evidence for label validity (human agreement, monotone risk reduction, and downstream selective prediction) is end-to-end; we do not ablate the pipeline component-wise, so the residual noise contributed individually by claim decomposition, verification, and span alignment remains unquantified, and characterizing it is the natural next step for SPANUQ-BENCH. We discuss societal benefits and risks, including the danger of overreliance on unflagged spans, in **App. C.10**.

Acknowledgements and Disclosures

This work was supported by Amazon through computational resources and access to large language model services. The authors declare no competing interests.

References

[1] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM computing surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[2] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson *et al.*, "Language models (mostly) know what they know," *arXiv preprint arXiv:2207.05221*, 2022.

[3] E. Fadeeva, A. Rubashevskii, A. Shelmanov, S. Petrakov, H. Li, H. Mubarak, E. Tsymbalov, G. Kuzmin, A. Panchenko, T. Baldwin *et al.*, "Fact-checking the output of large language models via token-level uncertainty quantification," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 9367–9385.

[4] T. Zhang, H. Shi, Y. Wang, H. Wang, X. He, Z. Li, H. Chen, L. Han, K. Xu, H. Zhang *et al.*, "Tokur: Token-level uncertainty estimation for large language model reasoning," *arXiv preprint arXiv:2505.11737*, 2025.

[5] L. Kuhn, Y. Gal, and S. Farquhar, "Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation," *arXiv preprint arXiv:2302.09664*, 2023.

[6] X. Qiu and R. Miikkulainen, "Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space," *Advances in neural information processing systems*, vol. 37, pp. 134 507–134 533, 2024.

[7] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, "Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms," *arXiv preprint arXiv:2306.13063*, 2023.

[8] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, "Factscore: Fine-grained atomic evaluation of factual precision in long form text generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 12 076–12 100.

[9] A. Vazhentsev, L. Rvanova, I. Lazichny, A. Panchenko, M. Panov, T. Baldwin, and A. Shelmanov, "Token-level density-based uncertainty quantification methods for eliciting truthfulness of large language models," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025, pp. 2246–2262.

[10] A. Nikitin, J. Kossen, Y. Gal, and P. Marttinen, "Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities," *Advances in Neural Information Processing Systems*, vol. 37, pp. 8901–8929, 2024.

[11] Y. A. Yadkori, I. Kuzborskij, A. György, and C. Szepesvári, "To believe or not to believe your llm," *arXiv preprint arXiv:2406.02543*, 2024.

[12] L. Da, T. Chen, L. Cheng, and H. Wei, "Llm uncertainty quantification through directional entailment graph and claim level response augmentation," *arXiv preprint arXiv:2407.00994*, 2024.

[13] C. Chen, K. Liu, Z. Chen, Y. Gu, Y. Wu, M. Tao, Z. Fu, and J. Ye, "Inside: Llms' internal states retain the power of hallucination detection," *arXiv preprint arXiv:2402.03744*, 2024.

[14] Z. Li, J. Xiong, F. Ye, C. Zheng, X. Wu, J. Lu, Z. Wan, X. Liang, C. Li, Z. Sun *et al.*, "Uncertaintyrag: Span-level uncertainty enhanced long-context modeling for retrieval-augmented generation," *arXiv preprint arXiv:2410.02719*, 2024.

[15] A. Azaria and T. Mitchell, "The internal state of an llm knows when it's lying," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 967–976.

[16] H. Orgad, M. Toker, Z. Gekhman, R. Reichart, I. Szpektor, H. Kotek, and Y. Belinkov, "Llms know more than they show: On the intrinsic representation of llm hallucinations," *arXiv preprint arXiv:2410.02707*, 2024.

[17] J. Kossen, J. Han, M. Razzak, L. Schut, S. Malik, and Y. Gal, "Semantic entropy probes: Robust and cheap hallucination detection in llms," *arXiv preprint arXiv:2406.15927*, 2024.

[18] J. Wei, C. Yang, X. Song, Y. Lu, N. Hu, J. Huang, D. Tran, D. Peng, R. Liu, D. Huang *et al.*, "Long-form factuality in large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 80 756–80 827, 2024.

[19] P. Manakul, A. Liusie, and M. Gales, "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*, 2023, pp. 9004–9017.

[20] A. Mishra, A. Asai, V. Balachandran, Y. Wang, G. Neubig, Y. Tsvetkov, and H. Hajishirzi, "Fine-grained hallucination detection and editing for language models," *arXiv preprint arXiv:2401.06855*, 2024.

[21] R. Vázquez, T. Mickus, E. Zosa, T. Vahtola, J. Tiedemann, A. Sinha, V. Segonne, F. Sánchez-Vega, A. Raganato, J. Libovickỳ *et al.*, "Semeval-2025 task 3: Mu-shroom, the multilingual shared-task on hallucinations and related observable overgeneration mistakes," in *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, 2025, pp. 2472–2497.

[22] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "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*, 2016, pp. 260–270.

[23] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," in *Proceedings of the 2017 conference on empirical methods in natural language processing*, 2017, pp. 188–197.

[24] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, relation, and event extraction with contextualized span representations," 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)*, 2019, pp. 5784–5789.

[25] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[26] Y.-H. Wang, A. Bai, C.-P. Tsai, and C.-J. Hsieh, "Clue: Concept-level uncertainty estimation for large language models," *arXiv preprint arXiv:2409.03021*, 2024.

[27] C. Tong, Q. Zhang, J. Gao, L. Jiang, Y. Liu, and N. Sun, "Halunet: Multi-granular uncertainty modeling for efficient hallucination detection in llm question answering," *arXiv preprint arXiv:2512.24562*, 2025.

[28] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[29] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[30] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[31] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1601–1611.

[32] Y. Zhao, J. Zhang, I. Chern, S. Gao, P. Liu, J. He *et al.*, "Felm: Benchmarking factuality evaluation of large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 502–44 523, 2023.

[33] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," in *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, 2022, pp. 3214–3252.

[34] J. Li, X. Cheng, X. Zhao, J.-Y. Nie, and J.-R. Wen, "Halueval: A large-scale hallucination evaluation benchmark for large language models," in *Proceedings of the 2023 conference on empirical methods in natural language processing*, 2023, pp. 6449–6464.

[35] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[36] X. Du, C. Xiao, and Y. Li, "Haloscope: Harnessing unlabeled llm generations for hallucination detection," *Advances in Neural Information Processing Systems*, vol. 37, pp. 102 948–102 972, 2024.

[37] M. Niu, H. Haddadi, and G. Pang, "Robust hallucination detection in llms via adaptive token selection," *Advances in Neural Information Processing Systems*, vol. 38, pp. 126 355–126 377, 2026.

[38] J. Duan, H. Cheng, S. Wang, A. Zavalny, C. Wang, R. Xu, B. Kailkhura, and K. Xu, "Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 5050–5063.

Appendix

A Architectural Design Details of SpanUQ

Fig. A1 illustrates the end-to-end span-level uncertainty estimation process in SpanUQ, from initialization to final uncertainty output. This appendix expands on the architectural overview in Sec. 3.1 by providing design rationale and implementation details for each stage.

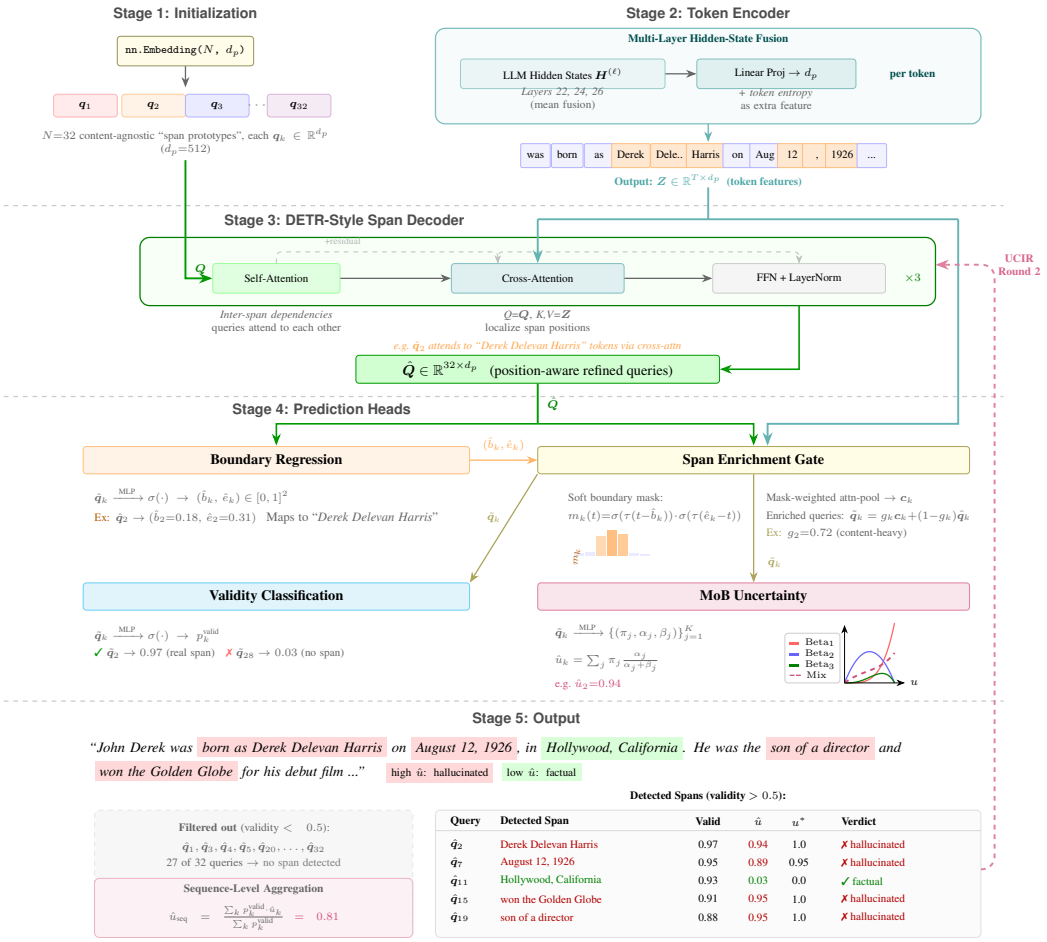

Figure A1: **Span query lifecycle in SpanUQ. Stage 1**: $N{=}32$ learnable span queries are initialized as content-agnostic prototypes. **Stage 2**: A token encoder fuses multi-layer hidden states with token entropy, producing the token feature pool $\boldsymbol{Z} \in \mathbb{R}^{T\times d_p}$. **Stage 3**: A 3-layer DETR decoder refines queries via self-attention (inter-span dependencies) and cross-attention over $\boldsymbol{Z}$ (span localization), yielding position-aware $\hat{\boldsymbol{Q}}$. $\boldsymbol{Z}$ is shared between cross-attention and span enrichment (T-shaped data flow on the right). **Stage 4**: Four prediction heads operate on each query: *boundary regression* predicts start/end positions $(\hat{b}_k, \hat{e}_k)$; *span enrichment* applies a differentiable soft mask to attention-pool $\boldsymbol{Z}$ into a content vector, fused with the query via a learned gate to produce $\tilde{\boldsymbol{q}}_k$; *validity classification* scores whether the query corresponds to a real span; *MoB uncertainty* outputs $K{=}3$ Beta components whose weighted mean gives $\hat{u}_k$. **Stage 5**: Valid spans (score >0.5) are retained and aggregated into a sequence-level uncertainty via importance-weighted averaging. The dashed purple arrow indicates the UCIR feedback loop (Sec. 3.3). The example shows a biography of John Derek: SpanUQ assigns high uncertainty ($\hat{u} \geq 0.89$) to hallucinated claims and low uncertainty ($\hat{u} = 0.03$) to the factual birthplace.

A.1 Stage 1: Span Query Initialization

SPANUQ initializes N learnable span queries $\boldsymbol{Q} \in \mathbb{R}^{N \times d_p}$ as content-agnostic embedding vectors (Xavier uniform). At the start of training, no query "knows" which span it will eventually detect; specialization emerges entirely through learning.

Why learnable queries instead of input-derived proposals? An alternative design would derive candidate spans from the input, e.g., via a sliding window, constituency parse, or BIO sequence tagger. We deliberately avoid this for three reasons. First, input-derived proposals impose a fixed segmentation before the model has seen the full context, which precludes overlapping spans and makes the detector sensitive to segmentation errors. Second, a BIO tagger produces spans sequentially and cannot model inter-span dependencies in parallel. Third, learnable queries decouple the *number* of detection slots from the input length, keeping the decoder cost constant regardless of sequence length.

Choosing N. The number of queries N sets an upper bound on how many spans can be detected per response. In our benchmark, the average response contains $\sim$15 spans (31,332 test spans across 2,000 queries for Qwen3-14B), with a 95th-percentile of $\sim$28. We set N=32 to comfortably cover the tail while keeping the decoder's $O(N^2)$ self-attention cost negligible. A hyperparameter sweep (Tab. A15) confirms that N=16 already achieves comparable AUROC (0.905 vs. 0.910 for N=32), indicating that the model gracefully degrades when a few spans exceed the slot budget. Queries that do not bind to any span are suppressed by the validity head (Sec. A.4).

Content-agnostic initialization. Because all queries start from the same random distribution, no query is biased toward a particular position or span type. During training, the combination of Hungarian matching (which assigns each ground-truth span to the best-matching query) and decoder self-attention (which encourages queries to diversify) drives the queries to specialize. Empirically, we observe that after training, different queries tend to cover different positional ranges of the response, though this specialization is soft and adaptive rather than hard-coded.

A.2 Stage 2: Token Encoder and Feature Pool

Before the span decoder can attend to the response, the raw LLM hidden states must be compressed and contextualized into a shared feature pool $\boldsymbol{Z}$ (Eq. 2).

Multi-layer fusion. Different layers of a Transformer LLM encode qualitatively different information: early layers capture syntax and positional structure, middle layers encode factual associations, and late layers specialize in next-token prediction [16]. Uncertainty signals are distributed across this hierarchy. For instance, a factually incorrect span may appear confident in the final layer (high next-token probability) yet show conflicting activations in mid-layers where factual recall occurs. By fusing layers from the peak uncertainty zone (identified via single-layer probing; see App. B.6), we capture complementary signals that no single layer provides. We use simple mean fusion (Eq. 1) rather than learned weighted fusion or concatenation-projection. Our ablation (Tab. A5) shows that mean fusion matches or exceeds learned alternatives while adding zero parameters, likely because the three selected layers are already close in representation space and a learned weighting offers little additional expressiveness.

Projection and contextualization. The fused hidden states $\bar{\boldsymbol{H}} \in \mathbb{R}^{T \times d}$ are linearly projected from the LLM dimension d (e.g., 5,120 for Qwen3-14B) to a much smaller d_p=512. This 10$\times$ compression is critical: it reduces the decoder's memory and compute cost quadratically, and acts as an information bottleneck that forces the model to retain only uncertainty-relevant features. The projected tokens are then passed through a 2-layer Transformer encoder with sinusoidal positional encodings. This encoder serves two purposes: (i) it re-contextualizes the projected tokens so that each position is aware of its neighbors (important because the LLM's own positional information is partially lost after projection), and (ii) it provides a non-linear transformation that adapts the frozen LLM features to the uncertainty estimation task.

The shared feature pool $\boldsymbol{Z}$. The encoder output $\boldsymbol{Z} \in \mathbb{R}^{T \times d_p}$ serves as the shared "memory" for two downstream consumers: the decoder's cross-attention (Stage 3) and the span feature enrichment

module (Stage 4). This T-shaped data flow is deliberate. Sharing $\boldsymbol{Z}$ ensures that the boundary predictions (which depend on cross-attention over $\boldsymbol{Z}$) and the content features (which are pooled from $\boldsymbol{Z}$) operate in the same representation space, avoiding the need for an additional alignment step. It also means that gradients from both the boundary loss and the uncertainty loss flow back through the same encoder, encouraging $\boldsymbol{Z}$ to encode both positional and semantic information.

A.3 Stage 3: DETR-Style Span Decoder

The decoder is the core of SPANUQ's detection mechanism. It transforms the N content-agnostic queries into N content-aware span representations $\hat{\boldsymbol{Q}}$ via iterative cross-attention over $\boldsymbol{Z}$ (Eq. 3).

Decoder architecture. Each of the 3 decoder layers applies, in order: (i) multi-head self-attention among the N queries, (ii) multi-head cross-attention from queries to the token pool $\boldsymbol{Z}$, and (iii) a position-wise feed-forward network. Layer normalization is applied before each sub-layer (pre-norm), and residual connections wrap each sub-layer.

Self-attention: inter-query coordination. Self-attention allows queries to "see" each other. This serves two functions. First, it enables *competition*: if two queries attend to the same region of $\boldsymbol{Z}$, self-attention lets them negotiate so that one shifts to a different span, reducing duplicate detections. Second, it enables *cooperation*: queries covering adjacent or overlapping spans can share contextual information, which is useful when a long sentence contains multiple interleaved claims. Without self-attention, each query would operate in isolation, and the model would need to rely entirely on the Hungarian matching loss to discourage duplicates, a much weaker signal.

Cross-attention: binding queries to spans. Cross-attention is the mechanism by which each query "finds" its span in the token sequence. Each query computes attention weights over all T token positions in $\boldsymbol{Z}$, and the resulting weighted sum becomes the query's updated representation. Over multiple decoder layers, the attention pattern sharpens: in early layers, a query may attend broadly to a sentence-level region; by the final layer, attention concentrates on the specific tokens that constitute the span. This progressive refinement is analogous to how DETR's object queries gradually localize from coarse regions to tight bounding boxes.

Why 3 decoder layers? We follow the DETR convention of using a small number of decoder layers (the original DETR uses 6 for the more complex 2D object detection task). Our ablation shows that 2 layers underperform (AUROC 0.901 vs. 0.910 for 3 layers), while 4 layers provide no further gain, suggesting that 3 layers suffice for the 1D span detection problem. Each additional layer adds cross-attention and self-attention, so the cost grows linearly in the number of layers.

Handling overlapping spans. Unlike BIO-based sequence taggers, which assign exactly one label per token and therefore cannot represent overlapping spans, the DETR formulation places no constraint on the spatial relationship between queries. Two queries can predict spans that partially or fully overlap, because each query independently regresses its own (b_k, e_k) boundaries. This is important in practice: for example, in the sentence "Marie Curie was a Polish-French physicist who won the Nobel Prize in Physics in 1903," the spans "Polish-French physicist" and "the Nobel Prize in Physics" may share boundary tokens depending on the granularity of claim decomposition.

Auxiliary losses on intermediate layers. Following DETR, we apply the same prediction heads and losses to the output of each intermediate decoder layer (not just the final layer), with a reduced weight $\lambda_{\text{aux}}=0.5$. This provides direct supervision to early layers, accelerating convergence and improving gradient flow through the deep decoder stack. Without auxiliary losses, the early decoder layers receive only indirect gradients through the final layer, which can slow training significantly.

A.4 Stage 4: Prediction Heads and Span Enrichment

After decoding, each query $\hat{\boldsymbol{q}}_k$ is passed through four prediction heads. The design of these heads reflects a careful separation of concerns: boundary prediction uses the *pre-enrichment* query, while uncertainty and validity prediction use the *enriched* query.

Boundary regression. The boundary head is a 3-layer MLP that maps $\hat{\boldsymbol{q}}_k$ to normalized coordinates $(\hat{b}_k, \hat{e}_k) \in [0,1]^2$ via sigmoid activation. We use regression rather than pointer-based (cross-attention argmax) boundary prediction for two reasons. First, regression produces continuous coordinates that are fully differentiable, enabling the downstream enrichment module to backpropagate through the boundaries. A pointer-based approach would require argmax (non-differentiable) or Gumbel-softmax (noisy gradients). Second, regression naturally handles variable-length sequences: the normalized coordinates are resolution-independent, so the same model works for short and long responses without re-scaling logits. The boundary head operates on the *pre-enrichment* query $\hat{\boldsymbol{q}}_k$ rather than the enriched query $\tilde{\boldsymbol{q}}_k$. This avoids a circular dependency: the enrichment module (Eq. 4–6) needs boundary predictions to compute the soft mask, so the boundaries must be available before enrichment occurs.

Span feature enrichment. The decoder queries capture *where* a span is (via cross-attention patterns) but lack direct access to the *content* within the predicted region. This distinction matters because uncertainty estimation requires understanding what a span *says*, not just where it is. The enrichment module bridges this gap through a two-step process.

Step 1: Soft boundary masking (Eq. 4). Given the predicted boundaries $(\hat{b}_k, \hat{e}_k)$, we construct a differentiable mask $m_k(t)$ over token positions using a product of two sigmoids with sharpness τ=10. This mask is approximately 1 inside the span and 0 outside, with smooth transitions at the boundaries. The smoothness is essential: it allows gradients to flow from the uncertainty loss back through the mask to the boundary predictions, creating a feedback loop where better boundaries lead to better content features, which in turn lead to better uncertainty estimates.

Step 2: Mask-weighted attention pooling (Eq. 5). Rather than simply averaging the masked token features (which would weight all in-span tokens equally), we use learned attention pooling. The query $\hat{\boldsymbol{q}}_k$ attends to the token features $\boldsymbol{Z}$, but the attention weights are restricted to the masked region. This allows the model to focus on the most informative tokens within the span. For example, in the span "won the Nobel Prize in Physics in 1903," the tokens "Physics" and "1903" carry more factual content than "won" or "the," and the attention mechanism can learn to upweight them.

Step 3: Gated residual fusion (Eq. 6). The pooled content vector $\boldsymbol{c}_k$ is added to the original query $\hat{\boldsymbol{q}}_k$ via a gated residual connection: a linear projection followed by sigmoid produces an element-wise gate $\sigma(\boldsymbol{W}_g \hat{\boldsymbol{q}}_k + \boldsymbol{b}_g) \in [0,1]^d$, which modulates the content vector before adding it to the query. The gate controls how much each dimension of the enriched content contributes to the final representation. This is particularly important for short spans (1–2 tokens), where the query's cross-attention already captures most of the content and the enrichment adds little new information. For longer spans, the gate opens wider, allowing the pooled content to complement the query's coarser representation. Our ablation (Tab. A7) confirms that gated fusion outperforms both simple addition and concatenation.

Validity classification. The validity head is a single linear layer followed by sigmoid, predicting $p_k^{\text{valid}} \in [0,1]$. Since N=32 queries are always instantiated but a typical response contains only ∼15 spans, roughly half the queries are "unused" and must be suppressed. During training, the Hungarian matching assigns each ground-truth span to one query; unmatched queries receive a target of 0. Following DETR, we downweight the "no-object" class by a factor of 0.1 in the binary cross-entropy loss to account for the class imbalance (many more unmatched than matched queries). At inference time, queries with $p_k^{\text{valid}} < 0.5$ are discarded.

Mixture of Beta (MoB) uncertainty estimation. The uncertainty head predicts the parameters of a Mixture of Beta distribution (Eq. 7) from the enriched query $\tilde{\boldsymbol{q}}_k$.

Why Beta, not Gaussian? Uncertainty scores are bounded in $[0,1]$, making the Beta distribution a natural choice. A Gaussian would require clipping or sigmoid transformation, losing the principled probabilistic interpretation. Moreover, the Beta distribution can express a rich family of shapes: uniform (α=β=1), unimodal symmetric (α=β>1), skewed ($\alpha \neq \beta$), and U-shaped (α, β<1).

Why a mixture? A single Beta component cannot capture multimodal uncertainty patterns. Consider a span that is correct in 70% of sampled responses and completely wrong in 30%: its ground-truth uncertainty distribution is bimodal, concentrated near 0 and 1. A single Beta must compromise with a broad, flat distribution, losing calibration. A mixture of J=3 components can place one component near 0, another near 1, and a third at an intermediate value, faithfully representing the bimodal

structure. We set $\alpha_{\min}$=0.5 (rather than the default 1.0) to allow U-shaped components, which are essential for capturing this bimodality.

Parameterization. The MoB head is a 2-layer MLP that outputs $3J$ values per query: mixture weights $\pi_{k,j}$ (via softmax), and shape parameters $\alpha_{k,j}, \beta_{k,j}$ (via softplus $+\alpha_{\min}$). The predicted uncertainty score is the mixture mean $\hat{u}_k = \sum_j \pi_{k,j}\alpha_{k,j}/(\alpha_{k,j} + \beta_{k,j})$. The effective precision $\hat{\nu}_k = \sum_j \pi_{k,j}(\alpha_{k,j} + \beta_{k,j})$ serves as a confidence measure for the uncertainty estimate itself, used in UCIR (Sec. 3.3).

Sequence-level aggregation. Span-level scores are aggregated into a single sequence-level uncertainty estimate via importance-weighted pooling (Eq. 8). The importance weight for each span is the product of its validity probability and a learned importance score from a single-layer MLP. This design ensures that invalid (suppressed) queries contribute negligibly to the sequence score, and that spans carrying more informational weight (e.g., a central factual claim vs. a hedging phrase) are weighted more heavily.

A.5 Stage 5: Filtering, UCIR, and Output

Validity filtering. At inference time, we retain only queries with $p_k^{\text{valid}} \geq 0.5$. This threshold is not tuned; it simply reflects the natural decision boundary of the sigmoid. In practice, the validity distribution is strongly bimodal (most queries have $p^{\text{valid}} > 0.9$ or < 0.1), so the exact threshold has little effect.

Uncertainty-Conditioned Iterative Refinement (UCIR). A single decoder pass produces reasonable but sometimes coarse uncertainty estimates, particularly for ambiguous spans where the model is uncertain about its own uncertainty. UCIR addresses this by feeding Round 1 predictions back into the decoder for a second pass (Eq. 12). The intuition is as follows. After Round 1, each query has an initial uncertainty estimate $\hat{u}_k^{(1)}$, a precision $\hat{\nu}_k^{(1)}$, and a validity score $p_k^{(1)}$. These three scalars are mapped through a small MLP to produce a conditioning vector, which is added to the query embedding. The conditioned queries then pass through the *same* decoder (shared weights), producing refined predictions. This is analogous to iterative bounding-box refinement in object detection [25]: the second pass can correct errors from the first pass because it has access to the first-pass estimates as additional context. The final prediction combines both rounds via a fixed convex combination (Eq. 13) with α=0.7, up-weighting the refined Round 2 estimate while retaining the Round 1 estimate as a regularizer. Both rounds contribute to the training loss, with Round 1 receiving a reduced weight to encourage the model to improve in Round 2 rather than relying solely on Round 1. UCIR adds less than 15% overhead relative to SPANUQ's own base decoder — an internal component comparison, not a fraction of the LLM forward pass — because the decoder weights are shared and the conditioning MLP is tiny (3 scalars $\to d_p$ dimensions). Our ablation (Tab. 4) shows that UCIR improves AUROC by 0.6 percentage points, with the largest gains on spans where Round 1 predictions have low precision $\hat{\nu}_k^{(1)}$.

Output format. The final output is a set of span predictions $\{(b_k, e_k, \hat{u}_k)\}_{k=1}^{K}$, where $K \leq N$ is the number of valid spans, b_k and e_k are token-level boundary indices (obtained by de-normalizing the regressed coordinates), and $\hat{u}_k \in [0, 1]$ is the estimated uncertainty score. The sequence-level score $\hat{u}_{\text{seq}}$ is also available as a by-product of the importance-weighted aggregation. The entire pipeline, from hidden state extraction to final output, adds less than 3% latency to the LLM forward pass (**App. D.5**).

B Additional Experimental Results

B.1 Sequence-Level Evaluation

While SPANUQ is designed for span-level uncertainty estimation, we also evaluate its sequence-level performance by aggregating span predictions via importance-weighted top-k mean. **Tab. A1** compares SPANUQ against dedicated sequence-level baselines. SPANUQ achieves $\text{AUROC}_{@0.3} = 0.948$ and Spearman $\rho = 0.851$, outperforming all dedicated sequence-level methods. SelfCheckGPT-NLI is the strongest sequence-level baseline (Spearman $= 0.805$, $\text{AUROC}_{@0.3} = 0.908$), but requires

10 sampled responses and NLI inference, a $10\times$ generation cost overhead. Semantic Entropy (SE) achieves moderate correlation (Spearman $= 0.606$) but poor normalized MAE ($0.427^{\dagger}$), indicating that even after min-max scaling, its entropy values are poorly calibrated to the $[0, 1]$ uncertainty range. INSIDE and KLE produce negatively correlated estimates (Spearman $= -0.331$ and -0.145, respectively). This inversion is not an implementation defect but a known consequence of aggregating token-level surprisal to coarser granularities. INSIDE scores the self-information of individual tokens, so rare-but-correct tokens — specific dates, proper nouns, technical terms — inflate its estimate precisely on spans that are factually *correct*; a claim like "born in 1879 in Ulm" is lexically improbable and factually solid at the same time. KLE's cross-layer KL divergence likewise measures token-level distributional shift, which need not align with whether a *claim* is supported. Weak or inverted correlation between token-entropy measures and factual correctness at claim granularity has been reported before [38, 3], and it is a direct argument for predicting span-level uncertainty from hidden states rather than pooling token-level signals: the quantity that matters is a property of the proposition, not of the token sequence that expresses it. The MLP Probe, when aggregated from oracle span-level predictions, achieves the strongest sequence-level performance (Spearman $= 0.858$, Pearson $= 0.867$, AUROC$_{@0.3} = 0.956$, MAE $= 0.071$), slightly surpassing SPANUQ. This confirms that span-to-sequence aggregation is a powerful strategy: even a simple probe benefits from fine-grained span-level predictions. However, the MLP Probe relies on *ground-truth span boundaries* at test time, which are unavailable in practice. In contrast, SPANUQ jointly detects spans and estimates uncertainty end-to-end, and its span-level superiority (AUROC: 0.939 vs. 0.881, Spearman: 0.685 vs. 0.575) demonstrates substantially richer uncertainty representations at the granularity that matters most for interpretability and actionability.

Table A1: Sequence-level uncertainty estimation on the SPANUQ-BENCH test set (2,000 sequences). Sequence-level ground truth is the mean span uncertainty per sequence; AUROC$_{@0.3}$ treats sequences with hallucination rate ≥ 0.3 as positive. SPANUQ and span-level baselines aggregate span predictions via importance-weighted mean. MAE† denotes min-max normalized scores (raw outputs not in $[0, 1]$). *MLP Probe uses ground-truth span boundaries (oracle setting). Best in **bold**, second best underlined.

Granularity	Method	Spearman↑	Pearson↑	AUROC$_{@0.3}$ ↑	MAE↓
Token	MLP Probe*	**0.858**	**0.867**	**0.956**	**0.071**
Sequence	Verbalized Conf.	0.322	0.229	0.641	0.166
	P(True)	0.140	0.185	0.552	0.172
	SE (10 samples)	0.606	0.521	0.805	$0.427^{\dagger}$
	INSIDE (Jaccard)	-0.331	-0.314	0.315	$0.246^{\dagger}$
	KLE	-0.145	-0.041	0.425	$0.784^{\dagger}$
	SelfCheckGPT-NLI	0.805	0.750	0.908	0.097
Claim	FActScore	0.489	0.505	0.779	0.138
Span	SPANUQ	<u>0.851</u>	<u>0.839</u>	<u>0.948</u>	<u>0.077</u>

B.2 Robustness Across Hyperparameter Configurations

Rather than reporting variance over random seeds (which primarily captures initialization noise), we assess robustness by examining the spread of results across the 24–28 hyperparameter configurations evaluated per model during the sweep (**App. D.3**). This provides a more informative measure of sensitivity, as it captures the effect of architectural and optimization choices. Across all five models, the IQR of dev AUROC is at most 0.030 (Mistral-7B), indicating that SPANUQ is not overly sensitive to hyperparameter choices. Even the worst configuration for each model substantially outperforms the strongest baseline (MLP Probe with oracle boundaries), confirming that the architectural design, rather than careful tuning, drives the performance gains.

B.3 Decomposability Analysis: Span-to-Sequence Aggregation

A key advantage of span-level uncertainty estimation is the ability to *decompose* sequence-level uncertainty into interpretable span-level components. We analyze this decomposability property across all five LLM backbones.

Table A2: Robustness of SPANUQ across hyperparameter configurations. For each model, we report the best, median, and worst dev AUROC across all sweep configurations, along with the interquartile range (IQR). The narrow IQR ($\leq$0.03) indicates that SPANUQ is robust to hyperparameter choices.

Model	Best	Median	Worst	IQR
Qwen3-14B	0.941	0.930	0.897	0.018
Qwen3-8B	0.928	0.910	0.883	0.021
Qwen3-4B	0.944	0.918	0.895	0.024
Qwen3-30B-A3B	0.935	0.920	0.893	0.022
Mistral-7B	0.928	0.879	0.852	0.030

Aggregation Method. Given predicted span uncertainties $\{\hat{u}_k\}_{k=1}^K$ and learned importance weights $\{w_k\}_{k=1}^K$ (from the sequence aggregation head), the sequence-level uncertainty is computed as: $\hat{u}_{\text{seq}} = \sum_{k=1}^K \tilde{w}_k \hat{u}_k$, where $\tilde{w}_k = \text{softmax}(w_k)$ are normalized importance weights. The ground-truth sequence-level uncertainty is $u^*_{\text{seq}} = \frac{1}{K}\sum_{k=1}^K u^*_k$.

Table A3: Decomposability analysis: correlation between aggregated span-level predictions and sequence-level ground truth across five LLM backbones. Higher values indicate that span-level predictions successfully capture sequence-level uncertainty patterns.

Model	Spearman $\rho_{\text{seq}}\uparrow$	Pearson $r_{\text{seq}}\uparrow$	AUROC$_{@0.3}\uparrow$	MAE$_{\text{seq}}\downarrow$
Qwen3-14B	0.851	0.839	0.948	0.077
Qwen3-8B	0.857	0.793	0.936	0.108
Qwen3-4B	0.846	0.802	0.936	0.218
Qwen3-30B-A3B	0.848	0.735	0.929	0.111
Mistral-7B	0.775	0.745	0.917	0.103

Cross-Model Consistency. Tab. A3 shows that span-to-sequence decomposability holds consistently across all five models. The Qwen3 family achieves Spearman $\rho_{\text{seq}} \geq 0.846$, while Mistral-7B shows a slightly lower but still strong correlation ($\rho_{\text{seq}} = 0.775$). This confirms that the decomposability property is not an artifact of a single model family.

Comparison with Sequence-Level Baselines. On Qwen3-14B, SPANUQ's aggregated span predictions ($\rho_{\text{seq}} = 0.851$) outperform all dedicated sequence-level methods: Semantic Entropy ($\rho = 0.606$), SelfCheckGPT-NLI ($\rho = 0.805$), and Verbalized Confidence ($\rho = 0.322$). This demonstrates that fine-grained span-level modeling captures richer uncertainty structure that benefits even coarse-grained evaluation.

MAE$_{\text{seq}}$ Bias in Qwen3-4B. The 4B model exhibits elevated MAE$_{\text{seq}}$ (0.218) despite strong ranking correlation ($\rho = 0.846$). This reflects a systematic positive bias in the sequence aggregation head: the 4B probe tends to overestimate sequence-level uncertainty by $\sim$0.2 on average. We attribute this to the smaller model's higher base hallucination rate, which shifts the learned aggregation weights toward higher uncertainty. This bias does not affect ranking-based metrics (AUROC, ρ), confirming that the span-level predictions themselves are well-estimated.

B.4 Qualitative Examples

We present three representative examples from the SPANUQ-BENCH test set to illustrate SPANUQ's behavior across different uncertainty patterns. For each example, we show the LLM response with ground-truth span annotations and SPANUQ's predictions.

Example 1: Biography with Mixed Uncertainty (Qwen3-14B). *Prompt:* "Write a biography of John Derek." The model generates a detailed biography containing both factual and hallucinated content. SPANUQ correctly assigns high uncertainty to fabricated claims about the subject's birth name ($\hat{u} = 0.94$, $u^* = 1.0$), birth date ($\hat{u} = 0.89$, $u^* = 0.95$), and parentage ($\hat{u} = 0.95$, $u^* = 1.0$), while assigning low uncertainty to verifiable facts about the subject's career ($\hat{u} = 0.03$, $u^* = 0.0$). The mean absolute error across 12 matched spans is 0.12, and the model correctly ranks all high-uncertainty spans above low-uncertainty ones.

Example 2: Mostly Factual Response (Qwen3-14B). *Prompt:* "Why was it illegal for black people to attend the University of Alabama in 1973?" The model generates a largely accurate historical account with 17 spans, 15 of which have $u^* < 0.1$. SPANUQ correctly assigns near-zero uncertainty to the factual spans ($\hat{u} \in [0.01, 0.12]$), demonstrating accurate estimation on confident predictions. The single moderately uncertain span ("Kennedy sent federal troops," $u^* = 0.71$) receives a lower prediction ($\hat{u} = 0.09$), representing a failure case where the model underestimates uncertainty for a historically nuanced claim.

Example 3: Failure Case (Implicit Compositional Errors). We observe that SPANUQ occasionally struggles with *compositional* hallucinations where individual facts are correct but their combination is wrong. For instance, in a biography response, the claim "He appeared in films such as *The Manchurian Candidate*" ($u^* = 0.0$) is correctly identified as factual ($\hat{u} = 0.03$), but a nearby span attributing a specific role to the subject in that film ($u^* = 0.0$) receives elevated uncertainty ($\hat{u} = 0.85$). This suggests that the model's hidden states encode uncertainty about entity-role associations even when the individual entities are well-known, pointing to a limitation in distinguishing entity-level from relation-level confidence.

Summary. Across the test set, SPANUQ achieves MAE < 0.15 on 78% of spans and correctly ranks high-uncertainty spans above low-uncertainty ones in 94% of within-sequence pairs. The primary failure mode is underestimation of uncertainty for historically nuanced or temporally sensitive claims, which constitute $\sim$5% of the test set.

B.5 Entity Popularity Analysis

We analyze SPANUQ's performance stratified by entity popularity in the biography domain, where entities are categorized as *head* (>100K Wikipedia page views/month), *torso* (10K–100K), or *tail* (<10K). As expected, tail entities exhibit slightly lower AUROC (0.937 vs. 0.953 for head) and higher MAE (0.137 vs. 0.111), reflecting the greater difficulty of estimating uncertainty for less well-known entities. However, the degradation is modest: even on tail entities, SPANUQ achieves AUROC > 0.93, substantially outperforming all baselines. This suggests that the model's hidden states encode meaningful uncertainty signals even for entities outside its training distribution. Interestingly, the average ground-truth uncertainty is similar across popularity tiers (0.293–0.322), which differs from the $\sim 3\times$ gap observed in the full training set. This is because the test set is stratified by uncertainty bin, ensuring balanced representation across uncertainty levels within each domain. The raw (unstratified) training data shows a clearer popularity–uncertainty gradient: tail entities average $u^* = 0.48$ vs. $u^* = 0.16$ for head entities.

Table A4: Performance of SPANUQ (Qwen3-14B) on the biography domain stratified by entity popularity. Tail entities have slightly higher average uncertainty and lower AUROC, but the degradation is modest (<2 points).

Popularity	Prompts	Spans	Avg. u^*	AUROC↑	MAE↓	Pearson↑
Head	52	1,386	0.293	0.953	0.111	0.872
Torso	84	2,150	0.322	0.951	0.128	0.856
Tail	64	1,416	0.311	0.937	0.137	0.829

B.6 Layer Selection Analysis

We conduct a systematic layer importance study to determine which LLM layers carry the strongest uncertainty signal (**Tab. A5**).

Inverted-U Curve. Single-layer probing reveals an inverted-U pattern: AUROC peaks at layers 20–26 (0.893–0.897), with early layers (0–10) scoring 0.832–0.862 and final layers (35–39) declining to 0.870–0.882. This suggests that uncertainty information is concentrated in the mid-to-late layers, where the model has processed enough context to form uncertainty judgments but before the final layers specialize for next-token prediction.

Clustered > Spread. Combining three adjacent peak-zone layers (22+24+26) outperforms three spread layers (19+29+39) by 0.5% AUROC and 1.2% ρ_{span}. This indicates that the uncertainty signal

is *locally coherent* within the peak zone, and combining nearby layers captures complementary aspects of the same representation.

Mean ≈ Concat. Mean fusion matches concatenation across all configurations (within 0.1% AUROC), despite using 3× fewer parameters. We adopt mean fusion for all subsequent experiments due to its parameter efficiency.

Diminishing Returns. Expanding from 3 to 7 layers (20–26) yields negligible improvement (+0.0% AUROC, +0.2% ρ_{span}), confirming that 3 well-chosen layers suffice.

Table A5: Layer importance probing results (Qwen3-14B). Each row trains an MLP probe (5120→512→256→1) on hidden states from the specified layer(s). "Mean" and "Concat" denote fusion strategies for multi-layer configurations.

Layer Configuration	Fusion	AUROC↑	ρ_{span} ↑	MAE↓
Single-layer probes (top 5)				
Layer 24 (best single)	—	0.897	0.672	0.135
Layer 26	—	0.894	0.664	0.140
Layer 21	—	0.894	0.665	0.137
Layer 23	—	0.893	0.668	0.136
Layer 22	—	0.893	0.664	0.138
Multi-layer probes				
Layers 19, 29, 39 (spread)	Mean	0.899	0.674	0.138
Layers 19, 29, 39 (spread)	Concat	0.900	0.676	0.137
Layers 22, 24, 26 (cluster)	Mean	**0.904**	**0.686**	**0.137**
Layers 22, 24, 26 (cluster)	Concat	0.904	0.684	0.137
Layers 20–26 (7 layers)	Mean	0.904	0.688	0.136

B.7 Mixture of Beta Analysis

Tab. A6 analyzes the MoB distribution design choices:

Number of Components. K=3 provides the best balance: K=2 lacks expressiveness (AUROC 0.920), while K=5 slightly overfits (0.928 vs. 0.933 for K=3 with ffn1024). Intuitively, three components correspond to the three natural uncertainty modes: *certain* ($u \approx 0$), *uncertain* ($u \approx 0.3$–0.7), and *hallucinated* ($u \approx 1$).

Calibration Advantage. MoB dramatically improves calibration: raw ECE drops from 0.051 (single Beta) to 0.020 (K=3), a 61% reduction. After temperature scaling, the gap narrows but persists (0.029 vs. 0.015). The single Beta requires aggressive temperature correction (T=0.83), while MoB is nearly self-calibrated (T=1.05).

Enriched Head Capacity. Increasing the enriched uncertainty head from ffn512 to ffn1024 yields the largest single improvement within the MoB sweep (+0.9% AUROC, +2.5% ρ_{span}), suggesting that the richer MoB output benefits from a more expressive prediction head.

Table A6: Mixture of Beta (MoB) component analysis. K: number of mixture components. $\alpha_{\min}$: minimum concentration parameter. KL: KL divergence regularization weight. All models use the enrichment gate + ffn1024 configuration.

Configuration	AUROC↑	ECE↓	ECE_{cal}↓	MAE↓	ρ_{span} ↑
Single Beta (baseline)	0.918	0.051	0.029	0.144	0.721
MoB K=2	0.919	0.037	0.012	0.148	0.725
MoB K=3 (base)	0.924	0.020	0.018	0.126	0.735
MoB K=5	0.928	0.023	0.017	0.119	0.747
K=3, $\alpha_{\min}$=0.5	0.931	0.020	0.020	0.122	0.748
K=3, KL=0.01	0.932	0.024	0.020	0.118	0.755
K=3, ffn1024	**0.933**	**0.020**	**0.015**	**0.116**	**0.760**

Table A7: Span feature enrichment ablation. All models use the base DETR + regression configuration. "Fusion" denotes how pooled span features are combined with query representations. "Pool" denotes the pooling strategy over tokens within the predicted span.

Fusion	Pool	AUROC↑	ECE↓	MAE↓	ρ_{span} ↑
None (no enrichment)	—	0.907	0.058	0.154	0.673
Gate	Attention	**0.920**	0.052	0.144	**0.725**
Gate	Mean	0.897	0.061	0.159	0.641
Concat	Attention	0.914	**0.035**	**0.134**	0.699
Add	Attention	0.919	0.047	0.148	0.728
Gate	Attn + ffn512	0.917	0.053	0.146	0.706

B.8 Span Enrichment Analysis

Tab. A7 ablates the span feature enrichment module (**Sec. 3.1**).

Enrichment Value. Adding span enrichment with gated fusion improves AUROC by 1.3% (0.907→0.920) and ρ_{span} by 5.2% (0.673→0.725). The enrichment module pools token-level features within the predicted span region using a differentiable soft mask, providing the uncertainty scorer with direct access to the *content* of each span rather than relying solely on the abstract query representation.

Gated Fusion. Gated fusion achieves the best AUROC (0.920), outperforming concatenation (0.914) and closely matching addition (0.919). The learned gate allows the model to adaptively control how much span content information to incorporate. This is especially useful during early training when span boundaries are imprecise.

Attention Pooling. Learned attention pooling substantially outperforms mean pooling (0.920 vs. 0.897 AUROC, 0.725 vs. 0.641 ρ_{span}), as it can focus on the most informative tokens within each span rather than treating all positions equally.

C Dataset Details

This section provides full details of the SPANUQ-BENCH benchmark construction, complementing the overview in **Sec. 4.1**.

C.1 Data Collection Details

We collect 20,000 prompts from five domains to ensure diverse hallucination patterns.

Long-Form QA (8,400 prompts, 42%). We sample real Google search questions from Natural Questions (NQ-Open) [30], filtering for questions requiring multi-sentence answers. These are predominantly factoid (98% start with who/what/when/where), with median prompt length of 9 words. Qwen3-14B generates responses averaging 122 words.

TriviaQA (5,000 prompts, 25%). We sample from the TriviaQA unfiltered split [31], which provides trivia questions with verified reference answers. Answer types span entities (44%), persons (39%), phrases (15%), and dates/numbers (2%). Responses are shorter (avg. 81 words), making this domain useful for calibration with mostly confident, correct spans.

ELI5 (3,000 prompts, 15%). We sample explanatory questions from the ELI5 (Explain Like I'm 5) Reddit dataset. These are dominated by "why" (38%) and "how" (21%) questions requiring long-form reasoning. Responses average 349 words (the longest across domains) and hallucinations tend to be subtle reasoning errors rather than factual fabrications, testing the model's ability to detect distributed uncertainty.

Biography (2,000 prompts, 10%). Following the FActScore protocol [8], we prompt Qwen3-14B to generate biographies for 2,000 entities stratified by Wikipedia page-view popularity: 30% head (>100K views/month), 40% torso (10K–100K), and 30% tail (<10K). Three prompt templates are used equally: "Write a biography of X," "Tell me a bio of X," and "Tell me about X." Tail entities exhibit $\sim 3\times$ higher hallucination rates than head entities, providing rich training signal.

FELM (1,600 prompts, 8%). We aggregate challenging examples from three sources: FELM [32] (53%, spanning world knowledge, reasoning, math, science, and writing), TruthfulQA [33] (45%, targeting common misconceptions), and HaluEval [34] (2%). This is the most diverse domain, with the highest proportion of math/reasoning questions (13%).

Multi-Model Extension. For cross-model evaluation (**Tab. 1**), we regenerate responses for the same 20,000 prompts using each backbone LLM (Qwen3-8B, Qwen3-4B, Qwen3-30B-A3B, Mistral-7B-Instruct-v0.3). Each model's training labels are constructed independently via the same multi-sample distillation pipeline, ensuring that the uncertainty labels reflect each model's own knowledge boundaries. Test set evaluation uses a shared split index derived from the Qwen3-14B data to ensure comparable evaluation across models.

C.2 Multi-Sample Distillation Pipeline

The training label construction pipeline consists of four stages:

Stage 1: Multi-Sample Generation. For each of the 20,000 prompts, we generate $S + 1 = 21$ responses: one *primary* response with greedy decoding ($T = 0$), which supplies the probe's hidden states and span positions, and $S = 20$ independent samples at temperature $T = 1.0$ using nucleus sampling ($p = 0.95$). The high temperature encourages diverse outputs that reveal the model's uncertainty: spans that vary across samples are uncertain by construction.

Stage 2: Claim Decomposition. The *primary response only* is decomposed into atomic claims using an LLM judge (Claude Opus 4.6). A claim is defined as a single, verifiable factual assertion, e.g., "Barack Obama was born in Hawaii" rather than "Barack Obama was born in Hawaii and graduated from Harvard." The decomposition prompt instructs the judge to: (1) split compound sentences into atomic claims, (2) resolve coreferences to produce self-contained claims, (3) emit character-level offsets aligning each claim to its span in the response text, and (4) tag each claim as `FACTUAL`, `SUBJECTIVE`, or `OTHER`. On average, each response yields 12–18 atomic claims. The claim-type tag is produced by the decomposition stage but does *not* enter the computation of u^*. Because only the primary response is decomposed, there is no cross-generation claim-matching step: a single, fixed set of claims is carried through the remainder of the pipeline.

Stage 3: Cross-Sample Claim Verification. Each atomic claim is verified against each of the $S = 20$ sampled responses using the same LLM judge. For every (claim, sample) pair the judge receives the claim and that sample's full text as the reference, and returns one of SUPPORTED (the sample confirms or is consistent with the claim), CONTRADICTED (the sample directly conflicts with it), or NOT_MENTIONED (the sample does not address the claim).

Stage 4: Soft Label Computation. Let n_k^{sup} and n_k^{con} be the number of samples that support and contradict span s_k. The empirical uncertainty is the contradiction rate among the samples that take a position:

$$u_k^* = \frac{n_k^{\text{con}}}{n_k^{\text{sup}} + n_k^{\text{con}}}. \tag{15}$$

NOT_MENTIONED verdicts are excluded from the denominator rather than counted as evidence against the claim, so the pipeline abstains on silent samples instead of penalizing them. Spans with zero coverage ($n_k^{\text{sup}} + n_k^{\text{con}} = 0$) have no defined u^* and are dropped (see **App. C.13**). This yields a continuous soft label $u_k^* \in [0, 1]$ that captures graded uncertainty. Because a single misjudged sample shifts u_k^* by at most $1/20 = 0.05$, the continuous formulation absorbs occasional verification errors that brittle hard labels would propagate.

Label Distribution. The resulting distribution is bimodal: approximately 62% of spans have $u^* < 0.1$ (consistently correct across samples), 18% have $u^* > 0.9$ (consistently incorrect), and 20% fall in the intermediate range $0.1 \leq u^* \leq 0.9$. The intermediate spans are valuable for training, as they require the model to learn fine-grained uncertainty distinctions rather than simple binary classification.

C.3 Test Set Construction and Quality Validation

Test Set Construction. The test set is constructed using the same multi-sample distillation pipeline as the training set. The key difference is that test responses are generated with greedy decoding ($T = 0$) for deterministic outputs, while the 20 verification samples are still generated at $T = 1.0$. This ensures that test-time uncertainty labels reflect the model's inherent knowledge boundaries on

its most likely output, rather than sampling variability. The test set contains 2,000 prompts (200 biography, 841 long-form QA, 500 TriviaQA, 299 ELI5, 160 FELM) with 31,332 span-level soft labels.

Internal Span-Boundary Sanity Check (co-authors, $N = 200$). Three co-authors independently reviewed a stratified random sample of 200 test examples (10% of the test set, balanced across domains and uncertainty bins) to confirm that the automatically identified spans correspond to meaningful, independently verifiable claims, and that their character offsets align with the intended text. The reviewers found that the vast majority of spans satisfy both criteria; the span boundaries validated here are the ground truth against which the detection metrics of **Sec. 4.5** are computed. This check targets *span boundary* quality only. It is not evidence of uncertainty-label validity, which we assess separately and quantitatively with independent annotators in **App. C.4**.

C.4 Independent Human Validation Study

The check above was performed by co-authors and speaks only to span boundaries. To assess whether the *uncertainty labels* themselves are trustworthy, we ran a separate study with annotators who are not authors of this paper and had no knowledge of the pipeline's outputs.

Recruitment and Protocol. Five independent annotators were recruited from a university. Each received a unique, non-overlapping survey of 400 randomly sampled spans, for a total of $N = 2{,}000$. Each item presented three components: (1) the original user query; (2) the complete LLM response, providing sufficient context for the span; and (3) the target span, highlighted in place within that response. Annotators judged each span as CORRECT or INCORRECT on factual accuracy alone. Critically, each judgment **required an external reference** (a URL, citation, or authoritative source) and the LLM's own response was explicitly disallowed as a source. Acceptable references were Wikipedia, academic publications, official government or institutional records, established reference databases, and verified news outlets. This mandatory-reference requirement ensures every judgment is grounded in verifiable evidence rather than annotator intuition. Each annotator received a $100 Amazon gift card upon completion.

Agreement with the Pipeline. To compare against the pipeline's continuous score we binarize at $u^* = 0.5$ (i.e. $u^* \geq 0.5$ means the pipeline calls the span INCORRECT). Over $N = 2{,}000$ spans this gives Cohen's $\kappa = 0.618$ with 89.4% raw agreement and a 10.6% disagreement rate. This level of agreement is substantial, and comparable to the human–human agreement typically reported for factuality annotation (0.6–0.8 in benchmarks such as TRUE and FaithDial).

Monotonicity. Human-judged error rate increases strictly with the pipeline score (Spearman $\rho = 0.714$): The absence of any inversion means that a practitioner may pick a threshold anywhere in $[0, 1]$ and be confident that higher scores indicate higher factual risk.

Table A8: Human-judged factual error rate by pipeline uncertainty bin ($N = 2{,}000$ spans, Qwen3-14B).

u^* range	Spans	Human-judged error rate
$[0.0, 0.2)$	1,562	5.8%
$[0.2, 0.4)$	122	36.1%
$[0.4, 0.6)$	66	47.0%
$[0.6, 0.8)$	81	72.8%
$[0.8, 1.0]$	169	87.6%

High-Uncertainty Spans. Among the $N = 291$ spans with $u^* \geq 0.5$, human annotators confirm 77.7% as factually incorrect; the remaining 22.3% are factually correct and thus pipeline false positives. We audit the causes of these false positives in **App. C.6**.

Failure Modes. The 10.6% disagreement splits into two well-characterized modes. *False positives* (22.3% of high-u^* spans) arise when cross-sample inconsistency reflects unstable phrasing even though the underlying claim is correct; downstream this is the safe direction to err. *False negatives* (8.6% of low-u^* spans) are *confident hallucinations*: the model generates the same incorrect claim across all samples, so consistency-based labelling records low uncertainty. This blind spot is intrinsic to consistency-based supervision and is part of the motivation for combining cross-sample signals

with hidden-state probing, since the hidden states may encode doubt that the sampled text does not express.

C.5 Why the Probe Is Not Imitating the Label Pipeline

A natural concern about distilling from an automatic pipeline is that the probe might learn to reproduce the pipeline's idiosyncrasies rather than anything about uncertainty. Since both training and pipeline-label evaluation share the same source, a high AUROC against pipeline labels alone cannot distinguish the two. Three observations separate them.

Human-grounded risk reduction. The risk–coverage result (**Sec. 4.4**) is measured against human judgments that played no part in training. Rejecting spans by predicted uncertainty drives the human-judged error rate monotonically down, from 18.6% at full coverage to 6.3% at 40% coverage. A probe that had memorized pipeline artifacts uncorrelated with real error would produce a flat curve here, because its ranking would carry no information about human-judged correctness.

Independent human validation. The annotation study (**App. C.4**) shows the pipeline labels themselves track human judgment ($\kappa = 0.618$; monotone error rate across u^* bins), so the training signal is not arbitrary noise to begin with.

Complementary information source. The probe's input is the LLM's own hidden states, which encode the model's internal state at generation time. The label pipeline observes only sampled output text and cannot access this. The probe therefore learns to decode an internal signal that is correlated with, but not a copy of, its supervision — consistent with its ability to rank spans that the pipeline itself scores identically.

Cross-domain behaviour. A probe trained on the mixed-domain training split retains performance on held-out domain slices (**App. B.5**), indicating transferable uncertainty structure rather than domain-specific pipeline bias.

We emphasize the limit of this evidence: it is end-to-end. It does not isolate the contribution of each pipeline component, which remains open (see **Limitations**, Sec. 5).

C.6 False-Positive Root-Cause Audit

The human study establishes *that* 22.3% of high-uncertainty spans are factually correct. This audit asks *why* the pipeline scored them as uncertain. The two analyses answer different questions and are complementary: the human study measures downstream utility of the score, whereas the audit is a diagnostic of the pipeline's internal mechanism and does not re-judge factual correctness.

Sampling. We randomly sampled **45** of the false-positive spans (those with $u^* \geq 0.5$ but judged CORRECT by the annotator) for manual root-cause audit. Exhaustive auditing was infeasible, as each case requires reading all 20 sampled responses together with their per-sample verification records. All percentages below are over this $N = 45$ audited subset, not over the full false-positive set.

Procedure. For each audited span the auditors had access to the original query; the complete primary response with the span highlighted; all 20 sampled responses; and the per-sample verification record, i.e. for every sample the verdict (SUPPORTED / CONTRADICTED / NOT_MENTIONED) together with the evidence string the verifier cited. They then read the claim text, inspected every CONTRADICTED verdict to determine whether the cited evidence genuinely conflicts with the claim or whether the conflict was manufactured by a pipeline artifact, and assigned a single primary root cause.

Interpretation. The large majority (88.9%) of audited false positives are not pipeline defects. They are spans where the model's 20 samples substantively disagreed: the uncertainty score was earned by real cross-sample variance, and the claim merely happened to be correct. This is the behaviour a continuous uncertainty score *should* exhibit — the model finds the fact unreliable even though it turns out to be true — and it is also why u^* must not be read as a factuality label (see **Limitations**, Sec. 5). Only 11.1% of the audited cases (5 of 45) are attributable to pipeline mechanisms: granularity mismatch and paraphrase variance in claim decomposition and verification, plus one ambiguous claim. That fraction is small but concrete, and it defines the improvement target for future versions of SPANUQ-BENCH. We note the audit is a single-annotator diagnostic over a modest sample; the category shares should be read as indicative rather than precise.

Table A9: Root causes of pipeline false positives ($u^* \geq 0.5$, human-judged CORRECT), over a random audit sample of $N = 45$.

Category	Share	Count
`GENUINE_UNCERTAINTY` — the sample texts genuinely conflict about the same proposition	88.9%	40
`GRANULARITY_MISMATCH` — the decomposed claim is narrower or broader than what the samples assert, so verdicts address a different proposition	4.4%	2
`PARAPHRASE_VARIANCE` — samples express the same fact in different surface forms and the verifier read the wording difference as a contradiction	4.4%	2
`CLAIM_AMBIGUITY` — the claim admits two valid readings, so samples can disagree without either being wrong	2.2%	1

C.7 Multi-Seed Stability

To quantify sensitivity to training randomness, we retrained the probe for every backbone with four additional random seeds (123, 456, 789, 1024) beyond the default seed 42. All seeds are evaluated on the *same* per-backbone span set used in **Tab. 1** (the unified Hungarian-matched set), so the reported variance isolates training randomness and excludes evaluation-set differences. AUROC is stable across seeds (std ≤ 0.017), confirming that the probe's ability to separate high- from low-uncertainty spans does not depend on a fortunate initialization. MAE (std ≤ 0.021) and ρ_s (std ≤ 0.030) are likewise stable. The slightly wider spread on Qwen3-4B is consistent with its smaller capacity, but all backbones retain strong ranking performance.

Table A10: Span-level performance across five random seeds (42, 123, 456, 789, 1024), mean $\pm$ std. **Tab. 1** reports seed 42, the default used throughout the paper.

Backbone	AUROC	MAE	ρ_s
Qwen3-14B	0.930 ± 0.007	0.117 ± 0.007	0.663 ± 0.013
Qwen3-8B	0.925 ± 0.005	0.137 ± 0.007	0.682 ± 0.006
Qwen3-4B	0.926 ± 0.017	0.149 ± 0.021	0.721 ± 0.030
Qwen3-30B-A3B	0.928 ± 0.010	0.114 ± 0.010	0.634 ± 0.014
Mistral-7B	0.884 ± 0.014	0.146 ± 0.012	0.616 ± 0.010

C.8 Additional Hidden-State Baselines: HaloScope and HaMI

We implement two recent hidden-state hallucination detectors as additional baselines. Both are sequence-level detectors that operate on internal representations, which makes them the closest prior work to SPANUQ's approach.

HaloScope [36] applies PCA dimensionality reduction followed by logistic regression on mean-pooled last-layer hidden states. We train it on the full training split ($\sim$17K sequences).

HaMI [37] uses gated-attention multiple-instance learning over token-level hidden states with a binary cross-entropy objective. We train it on a 5K random subset of the training split; this is sufficient for the small MIL network and is required because per-sample token-level processing is memory-bound.

Adaptation to span-level evaluation. Both methods emit a single score per sequence, so we apply the same broadcast protocol used for the other sequence-level baselines in **Tab. 1** (SelfCheckGPT-NLI, P(True)): the sequence score is assigned to every span in that response.

Findings. HaloScope outperforms HaMI on most backbones (average AUROC 0.820 vs. 0.794), which suggests that mean-pooling yields a more stable sequence representation than MIL attention for a sequence-level classification target. Both are competitive with SelfCheckGPT-NLI (average 0.804), confirming that hidden-state methods can match multi-sample approaches without additional generation cost. SPANUQ nonetheless leads HaloScope by 11.1 AUROC points on average, and the gap is widest on Spearman correlation (average ρ_s 0.683 vs. 0.492). The reason is structural rather

than incidental: broadcasting assigns an identical score to correct and incorrect spans within the same response, so even a perfect sequence-level detector cannot rank spans within a response. This limitation is what span-native prediction removes.

C.9 Threshold Selection and Deployment

Interpreting the score. SPANUQ's output approximates the probability that a span is incorrect, so 0.5 (more likely wrong than right) is a natural default. For Qwen3-14B the full model reaches ECE $= 0.013$ on *raw*, uncalibrated scores (**Tab. 4**), i.e. it is close to self-calibrated: the Mixture of Beta head needs only $T = 1.05$ of temperature correction, against $T = 0.83$ for a single Beta (**App. B.7**). Predicted probabilities therefore track empirical error rates closely enough to compare scores across different inputs from the same backbone.

Choosing an operating point. The risk–coverage curve (**Tab. 2**) converts an error budget directly into a threshold: accepting at most a 10% factual error rate corresponds to $\sim$80% coverage, i.e. rejecting the top-20% most uncertain spans. The monotone relationship between u^* and human-judged error (**Tab. A8**) guarantees that any threshold produces meaningful separation, with no inversion to worry about: filtering at $u \geq 0.4$ removes predominantly incorrect spans while retaining spans whose error rate is 5.8%. As rough starting points by application risk profile: $\approx$0.3 for high-stakes settings such as medical or legal review, where flagging more spans for human verification is worth the added cost; $\approx$0.5 for customer-facing assistants; and $\approx$0.7 for low-stakes uses such as creative writing support, where only near-certain errors should be surfaced.

Comparability across generations. Within a backbone, calibrated scores are comparable across inputs: 0.6 on one query and 0.6 on another indicate similar uncertainty. The probe reads the hidden states of whatever text was generated, so it is agnostic to the decoding strategy; our experiments use greedy decoding, and verifying calibration under nucleus or beam search is straightforward future work. Multi-seed results (**App. C.7**) show the calibrated scores are also reproducible across training runs.

Deployment recipe. (1) Train a per-backbone probe ($\sim$3–9 GPU-hours, one-time). (2) Fit temperature scaling on a held-out dev split to obtain calibrated probabilities. (3) Plot the risk–coverage curve on a representative validation set and read off the threshold matching the application's error budget. (4) At inference, run the single LLM forward pass, extract hidden states, apply the probe ($<$3% added latency), and surface spans above the threshold.

C.10 Broader Impacts

Reliable span-level uncertainty estimation can improve the trustworthiness of LLM-powered systems in high-stakes domains such as healthcare, legal analysis, and education, because it lets a user verify the specific claims a model is unsure about instead of re-checking an entire response or discarding it wholesale. The localization is the point: a sequence-level score forces an all-or-nothing decision, whereas span-level scores support targeted verification.

The same property invites overreliance. A user shown a partially highlighted response may reasonably infer that the unhighlighted remainder has been checked, which it has not. This failure mode is sharpest for the blind spot documented in **App. C.4**: consistency-based uncertainty is lowest precisely on *confident* hallucinations, where a model repeats the same incorrect claim across all samples. An interface built on SPANUQ should therefore communicate that low uncertainty is not a correctness guarantee, and should avoid visual conventions that imply verification of unflagged text.

SPANUQ estimates uncertainty from hidden-state patterns and does not certify factual correctness. It should complement, not replace, human judgment and external verification. We release the probe and benchmark to support research on fine-grained uncertainty; the released artifacts contain uncertainty annotations over model-generated text and no personal data.

C.11 Extending Beyond Factual Uncertainty

SPANUQ's architecture is task-agnostic: it consumes hidden states and predicts uncertainty over text spans. Porting it to a new domain requires only a meaningful notion of "span" and a way to obtain span-level labels. For *mathematical reasoning*, spans could be individual derivation steps

(e.g. "Since $x = 3$, we have $2x = 6$"), with labels from step-level verification; the multi-sample construction applies directly, since sampling several reasoning chains reveals where they diverge. For *code generation*, spans could be statements or blocks, with labels from execution-based verification such as per-segment test outcomes. We focus on factual generation here because it offers mature automatic verification, because factual hallucination is the most common failure mode in deployed systems, and because it permits rigorous benchmarking across diverse domains. The architectural components (DETR decoder, MoB uncertainty head, UCIR) are domain-agnostic and would transfer without modification; obtaining reliable step- or execution-level labels is the substantive work.

C.12 Downstream Applications and Future Directions

Uncertainty-aware verification. Verifying a long, fact-dense response in full requires many retrieval calls and model invocations, yet not every claim needs the same verifications. Span-level scores allow a system to send only high-uncertainty spans to search tools or verifier models, so that limited compute goes where errors are most likely. The risk-tiered thresholds in **App. C.9** extend naturally to tiered verification: use low-uncertainty spans directly, apply lightweight checks to intermediate ones, and reserve external retrieval or cross-model checking for the highest-uncertainty spans. Since low uncertainty does not guarantee correctness (**App. C.10**), the policy must still be set according to task risk and calibration quality.

Uncertainty-guided self-refinement. Existing self-refinement methods typically critique and regenerate the entire response. When only a few spans are unreliable, regenerating everything wastes compute and can corrupt claims that were already correct. SPANUQ makes targeted refinement possible: locate the high-uncertainty spans, then verify, edit, or regenerate only those. Open questions include when a span's uncertainty justifies refinement, how to keep local edits consistent with the surrounding context, and how to confirm that a refinement actually reduced error risk rather than just changed the text.

Uncertainty-aware multi-agent communication. In multi-agent systems, one agent's output often becomes another's input, for example passing from a retrieval agent to a reasoning agent to a planning agent. Today agents exchange natural language only, so a receiving agent has to judge for itself which claims to trust. If the sending agent also attached a span-level uncertainty score to each claim, the receiver could accept a claim, request supporting evidence, or route it to another agent for cross-checking. The system as a whole could likewise use these signals to allocate tasks, compute, and reasoning paths. Realizing this raises problems that SPANUQ does not solve. Our scores are calibrated within a single backbone (**App. C.9**), but agents may run on different base models, so "0.9" from one probe need not mean the same as "0.9" from another. Cross-model calibration, and preventing miscalibrated signals from propagating and compounding across agents, remain open. We view span-level uncertainty as a starting point for this line of work, not a solution to it.

C.13 Benchmark Statistics

Table A11: Statistics of SPANUQ-BENCH (Qwen3-14B backbone), reported as **raw (unfiltered)** span counts as emitted by the distillation pipeline. The benchmark spans five domains with stratified splits preserving domain and uncertainty-bin proportions. All labels are soft uncertainty scores from the multi-sample distillation pipeline.

Split	Prompts	Spans	Avg. Spans/P	Avg. Words/Resp.
Train	17,494	272,086	15.6	173
Dev	500	7,967	15.9	—
Test	2,000	31,332	15.7	174
Total	**19,994**	**311,385**	**15.6**	—

Raw vs. Filtered Span Counts. The raw pipeline emits 311,385 spans across all splits (table above). Spans that no sample either supports or contradicts have no defined u^* (**App. C.2**, Stage 4) and are removed, leaving $\sim$293K valid spans in the released benchmark, of which $\sim$256K fall in the training split. All span counts quoted elsewhere in this paper refer to this filtered set unless explicitly marked as raw.

Table A12: Per-domain breakdown of SPANUQ-BENCH. Biography and long-form QA are the two largest domains; ELI5 provides open-ended diversity; FELM and TriviaQA contribute factoid and reasoning challenges.

Domain	Train	Dev	Test	Total	Avg. Words/Resp.
Biography	1,750	50	200	2,000	323
Long-Form QA	7,348	210	841	8,399	122
TriviaQA	4,375	124	500	4,999	81
ELI5	2,624	76	299	2,999	349
FELM	1,397	40	160	1,597	217
Total	**17,494**	**500**	**2,000**	**19,994**	**173**

Domain Characteristics. The five domains cover a spectrum of hallucination patterns: *Biography* (10%) generates long-form text about entities stratified by Wikipedia popularity (30% head / 40% torso / 30% tail), where tail entities exhibit $\sim$3$\times$ higher hallucination rates. *Long-form QA* (42%) uses real Google search queries from Natural Questions [30], predominantly factoid (98%). *TriviaQA* (25%) provides trivia questions with verified reference answers [31]. *ELI5* (15%) contains explanatory "why/how" questions from Reddit, producing subtle reasoning errors rather than factual fabrications. *FELM* (8%) aggregates challenging examples from FELM [32], TruthfulQA [33], and HaluEval [34], targeting known misconceptions and math reasoning.

Uncertainty Distribution. Splits are stratified by domain $\times$ uncertainty bin (low: $u < 0.1$, mid: $0.1 \leq u < 0.4$, high: $u \geq 0.4$) to ensure balanced representation. Across the full dataset, approximately 62% of spans have low uncertainty, 20% mid, and 18% high.

Quality Validation. To validate the automatic labels, three co-authors independently reviewed a random sample of 200 test examples (10% of the test set), checking span boundary correctness, factuality label accuracy, and uncertainty score plausibility. The reviewers confirmed that the pipeline produces reliable labels, with disagreements primarily limited to temporally sensitive claims and ambiguous quantitative statements (see **App. C.3** for the full protocol).

D Implementation Details

D.1 Loss Function Details

This section provides the full formulations of each loss component in Eq. 9.

Span Regression Loss. The regression loss combines L1 distance and Generalized IoU (GIoU) [29] over matched span pairs from the Hungarian assignment $\hat{\sigma}$:

$$\mathcal{L}_{\text{reg}} = \sum_{i=1}^{M} \left[\|\hat{\boldsymbol{b}}_i - \boldsymbol{b}^*_{\hat{\sigma}(i)}\|_1 + \lambda_{\text{giou}} \cdot \mathcal{L}_{\text{GIoU}}(\hat{\boldsymbol{b}}_i, \boldsymbol{b}^*_{\hat{\sigma}(i)}) \right], \tag{16}$$

where $\hat{\boldsymbol{b}}_i = (\hat{b}_i, \hat{e}_i)$ are predicted boundaries, $\boldsymbol{b}^*$ are ground-truth boundaries, and M is the number of matched pairs. GIoU extends standard IoU to handle non-overlapping spans by penalizing the gap between them.

Validity Classification Loss. Binary cross-entropy over all N span queries:

$$\mathcal{L}_{\text{val}} = -\frac{1}{N} \sum_{k=1}^{N} \left[y_k \log p_k^{\text{valid}} + (1 - y_k) \log(1 - p_k^{\text{valid}}) \right], \tag{17}$$

where $y_k = 1$ if span query k is matched to a ground-truth span and $y_k = 0$ otherwise (unused slot).

MoB Negative Log-Likelihood. See Eq. 10 in the main text. To avoid numerical instability at $u = 0$ or $u = 1$, we clamp ground-truth labels to $[\epsilon, 1 - \epsilon]$ with $\epsilon = 10^{-4}$.

Consistency Loss. The consistency loss enforces agreement between the span-aggregated sequence-level prediction and the ground-truth sequence-level uncertainty:

$$\mathcal{L}_{\text{con}} = \|\hat{u}_{\text{seq}} - u^*_{\text{seq}}\|_2^2, \tag{18}$$

where $u^*_{\text{seq}} = \frac{1}{K}\sum_{k=1}^{K} u^*_k$ is the mean ground-truth span uncertainty. Gradients from $\mathcal{L}_{\text{con}}$ are detached from the span-level predictions to prevent the sequence-level objective from interfering with span-level learning.

Contrastive Ranking Loss. See Eq. 11 in the main text. We use margin $m = 0.1$. In practice, rather than enumerating all valid pairs, we employ *stratified sampling*: spans are partitioned into a high-uncertainty group ($u^* > \tau_{\text{hi}} = 0.3$) and a low-uncertainty group ($u^* < \tau_{\text{lo}} = 0.1$), and up to 256 (high, low) pairs are randomly sampled per batch. If either group contains fewer than 2 spans, we fall back to the top-25% vs. bottom-25% of spans ranked by u^*. This strategy focuses the ranking signal on clearly separable pairs and avoids noisy gradients from near-identical labels.

Loss Weights. The default loss weights are: $\lambda_{\text{reg}} = 5.0$, $\lambda_{\text{val}} = 2.0$, $\lambda_{\text{uq}} = 4.0$, $\lambda_{\text{con}} = 1.0$, $\lambda_{\text{rank}} = 0.5$, $\lambda_{\text{aux}} = 0.4$. During the warmup phase (epochs 1–15), only $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{val}}$ are active.

D.2 Model Specifications

Tab. A13 summarizes the LLM backbones used in our experiments, and **Tab. A14** details the SPANUQ probe configuration selected for each model via hyperparameter sweep (**Sec. D.3**).

Model	Type	Params	Layers	d_{model}	Heads	KV Heads	Vocab
Qwen3-14B	Dense	14.8B	40	5120	40	8	151,936
Qwen3-8B	Dense	8.2B	36	4096	32	8	151,936
Qwen3-4B	Dense	4.0B	36	2560	32	8	151,936
Qwen3-30B-A3B	MoE	30.5B (3.3B active)	48	2048	16	4	151,936
Mistral-7B-Instruct-v0.3	Dense	7.2B	32	4096	32	8	32,768

Table A13: LLM backbone specifications. All models use grouped-query attention (GQA). Qwen3-30B-A3B is a Mixture-of-Experts model with 3.3B active parameters per token.

Model	Probe Params	Layers Used	d_{proj}	n_q	Batch	LR
Qwen3-14B	27.04M	[22, 24, 26]	512	32	16	10^{-4}
Qwen3-8B	26.51M	[18, 20, 22]	512	32	16	10^{-4}
Qwen3-4B	17.18M	[22, 24, 26]	512	32	16	10^{-4}
Qwen3-30B-A3B	25.46M	[30, 34, 38]	512	32	16	10^{-4}
Mistral-7B	26.50M	[12, 14, 16]	512	16	32	2×10^{-4}

Table A14: SPANUQ probe configurations (best per model from hyperparameter sweep). All probes share: 2-layer encoder, 3-layer decoder, 8 attention heads, d_{ffn}=2048, MoB K=3, UCIR α=0.7, span enrichment with gate fusion.

D.3 Hyperparameter Sensitivity

We conduct a systematic hyperparameter sweep with 24–28 configurations per model, varying layer selection, projection dimension ($d_{\text{proj}} \in \{256, 512\}$), number of span queries ($n_q \in \{16, 32\}$), batch size ($\in \{16, 32\}$), and learning rate ($\in \{5\times10^{-5}, 10^{-4}, 2\times10^{-4}\}$). **Tab. A15** summarizes the sensitivity of dev AUROC to each hyperparameter, averaged across all five models.

Table A15: Hyperparameter sensitivity analysis. For each hyperparameter, we report the mean dev AUROC across all configurations using that value, averaged over five models. Δ denotes the range (max − min) across values.

Hyperparameter	Values	Mean AUROC	Δ
d_{proj}	256 / 512	0.898 / 0.916	0.018
n_q (span queries)	16 / 32	0.905 / 0.910	0.005
Batch size	16 / 32	0.908 / 0.907	0.001
Learning rate	5×10^{-5} / 10^{-4} / 2×10^{-4}	0.904 / 0.910 / 0.908	0.006

Key Findings. (1) **Projection dimension** is the most impactful hyperparameter ($\Delta = 0.018$): $d_{\text{proj}} = 512$ consistently outperforms 256 across all models, suggesting that the uncertainty signal requires sufficient representational capacity. (2) **Batch size** has negligible effect ($\Delta = 0.001$), indicating that the training dynamics are stable across batch sizes. (3) **Learning rate** 10^{-4} is optimal for most models, though Mistral-7B benefits from a higher rate (2×10^{-4}), likely reflecting differences in hidden-state scale across model families.

D.4 Training Curves

Tab. A16 reports the training dynamics for the best SPANUQ probe configuration on each LLM backbone. All models use a two-phase schedule: 15 warmup epochs (span detection only) followed by up to 25 joint epochs (span detection + uncertainty estimation), with early stopping (patience 5) on dev AUROC.

Model	Epochs	Best Ep.	Dev AUROC	Dev MAE	Dev ρ_s	Wall Time
Qwen3-14B	38 (15w+23j)	33	0.9412	0.1031	0.7932	9.3h
Qwen3-8B	40 (15w+25j)	39	0.9275	0.1257	0.7699	5.2h
Qwen3-4B	38 (15w+23j)	33	0.9444	0.1219	0.8225	4.0h
Qwen3-30B-A3B	39 (15w+24j)	34	0.9350	0.1020	0.7433	3.7h
Mistral-7B	28 (15w+13j)	23	0.9279	0.1142	0.7312	2.7h

Table A16: Training summary for the best SPANUQ probe per model. "w" = warmup epochs, "j" = joint epochs. All training on a single NVIDIA H100 80GB GPU. Dev metrics are reported at the best epoch (selected by AUROC).

Key observations. (1) All models exhibit a sharp phase transition at the warmup→joint boundary (epoch 16): dev AUROC jumps from $\sim$0.55–0.65 to $\sim$0.85+ within 2–3 joint epochs, confirming that the warmup phase successfully pre-trains span detection before uncertainty estimation begins. (2) Qwen3-4B achieves the highest dev AUROC (0.9444) despite having the smallest probe (17.18M parameters), suggesting that smaller LLMs with lower d_{model} can produce equally informative hidden-state representations for uncertainty estimation. (4) Mistral-7B trains fastest (2.7h) due to its larger batch size (32) and early convergence (epoch 23).

D.5 Computational Cost

Tab. A17 provides a detailed computational cost breakdown for SPANUQ.

Table A17: Computational cost breakdown. Data construction is a one-time cost shared across all models. Probe training and inference are per-model costs. All timings on NVIDIA H100 80GB GPUs.

Stage	Details	Cost
One-time data construction (per LLM backbone)		
Multi-sample generation	1 greedy + 20 samples × 20K prompts, 8×H100	$\sim$40 GPU-h
Claim decomposition + verification	$\sim$700K API calls (Claude Opus 4.6)	$\sim$\$1,000
Hidden state extraction	Single forward pass, 20K prompts	$\sim$0.5 GPU-h
Probe training (single GPU)		
Qwen3-14B	38 epochs (15w + 23j), 14 min/epoch	9.3h
Qwen3-8B	40 epochs (15w + 25j), 8 min/epoch	5.2h
Qwen3-4B	38 epochs (15w + 23j), 6 min/epoch	4.0h
Qwen3-30B-A3B	39 epochs (15w + 24j), 6 min/epoch	3.7h
Mistral-7B	28 epochs (15w + 13j), 6 min/epoch	2.7h
Inference (per prompt)		
LLM forward pass	Response generation (greedy)	2–5s
Hidden state caching	Extract from 3 layers	$<$10ms
SPANUQ probe	Projection + DETR + UCIR	$<$50ms
Total overhead	**Probe / LLM forward pass**	**$<$3%**

Training Efficiency. The SPANUQ probe trains in 2.7–9.3 hours on a single H100 GPU, depending on the backbone model size (which determines d_{model} and thus the input dimension). The 14B model requires the longest training due to its larger projection layer ($5120 \to 512$), while Mistral-7B trains fastest due to its larger batch size (32) and early convergence. The LLM backbone is *frozen* during training: only the lightweight probe (17–27M parameters) is updated, requiring no gradient computation through the LLM.

Inference Overhead. At inference time, SPANUQ adds $<3\%$ latency to the LLM forward pass. The dominant cost is the LLM generation itself (2–5s per prompt); the probe computation (projection, DETR encoding/decoding, UCIR refinement) takes $<$50ms. This is 10–$20\times$ faster in total wall-clock time than sampling-based methods (e.g., Semantic Entropy, SelfCheckGPT) that require 10–20 additional forward passes.

Comparison with Baselines. Semantic Entropy requires $S = 10$ sampled responses plus NLI-based clustering, costing $\sim 10\times$ the generation budget. SelfCheckGPT-NLI similarly requires $S = 10$ samples plus $S \times K$ NLI inference calls (where K is the number of sentences). FActScore requires claim decomposition and verification via an external LLM, adding $\sim$5s per claim. In contrast, SPANUQ operates on cached hidden states from a single forward pass, making it the most efficient method among all approaches that achieve AUROC > 0.85.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and introduction clearly claim the contributions: a span-level uncertainty framework, a learned span detector, a contrastive training objective, and a new benchmark. All are supported by the experimental evaluation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in **Sec. 5**, including the requirement for hidden state access, the cost of label construction, and the focus on factual uncertainty in English. We additionally devote a dedicated paragraph to what our labels do and do not measure: u^* quantifies cross-sample consistency rather than factuality, the two can diverge, consistency-based labels are blind to confident hallucinations, and we do not ablate the label pipeline component-wise, so its residual noise is unquantified (**App. C.4**, **C.6**).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper is primarily empirical and does not include theoretical proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide full architectural details, hyperparameters, training procedures, and dataset construction pipelines in **Sec. 3** and **Sec. 4.2**. Further details and design discussions can be found in the appendix. Code and data will be released upon acceptance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the SPANUQ implementation, trained probe checkpoints for all five backbones, and the four-stage SPANUQ-BENCH construction pipeline, together with the benchmark itself; see the project page linked in the abstract. The appendix documents the label construction pipeline (**App. C.2**), model specifications (**App. D.2**), and hyperparameters (**App. D.3**) needed to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full details are provided in **Sec. 4.2**, including model architectures, hyperparameters, hardware, and training duration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We retrain every backbone probe with five random seeds (42, 123, 456, 789, 1024) and report mean $\pm$ standard deviation for all span-level metrics in **App. C.7**: AUROC std ≤ 0.017, MAE std ≤ 0.021, and ρ_s std ≤ 0.030. All seeds are evaluated on the same

per-backbone matched span set, so the reported variance isolates training randomness from evaluation-set differences. **Tab. 1** reports seed 42, the default used throughout. We additionally assess robustness to hyperparameter choice via the spread across 24–28 configurations per model (**App. B.2**), where the IQR is $\leq$0.03.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify GPU type (H100 80GB), training time ($\sim$4 hours), and inference overhead ($<$5%) in **Sec. 3.2** and **Sec. 3.3**. A full cost breakdown is provided in **App. D.5**.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work aims to improve the trustworthiness of LLM outputs. We discuss broader impacts in **Sec. 5**.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss positive impacts (improved trust in LLM outputs) and potential negative impacts (overreliance on uncertainty scores) in **Sec. 5**.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The SPANUQ-BENCH benchmark contains only uncertainty annotations on publicly available text.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all datasets (FActScore, FELM, Natural Questions) and models (Qwen3, Mistral) used, and respect their licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: SPANUQ-BENCH is documented in **Sec. 4.1** with statistics and quality metrics. Full details including annotation protocol and benchmark statistics are in **App. C**.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: **App. C.4** documents the human annotation study used to validate benchmark labels: five independent annotators, each judging a unique non-overlapping set of 400 spans ($N = 2{,}000$). We report the annotation instrument — the information shown per item (original query, complete LLM response, target span highlighted in context), the binary CORRECT/INCORRECT decision, the mandatory external-reference requirement, and the list of acceptable reference sources. Each annotator received a $100 Amazon gift card upon completion. A separate internal span-boundary sanity check was performed by co-authors (**App. C.3**).

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: The annotation task described in **App. C.4** was minimal-risk: annotators judged the factual correctness of publicly available LLM-generated text and consulted public reference sources. No personal, sensitive, or identifying data was collected from or about annotators, and no deceptive procedures were used. Annotators were informed of the purpose of the task and compensated ($100 Amazon gift card each). Under our institution's policy, this category of minimal-risk annotation work does not require IRB review, and none was obtained.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are central to our data construction pipeline: we use LLM-based claim decomposition and multi-sample self-verification to generate span-level uncertainty labels (**Sec. 4.1**). The target LLMs (Qwen3 and Mistral) also serve as the backbone whose hidden states are probed. All models and their roles are fully described.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.