

MUCH: A Multilingual Claim Hallucination Benchmark

Jérémie Dentan¹, Alexi Canesse¹, Davide Buscaldi^{1,2},
Aymen Shabou³, Sonia Vanier¹

¹LIX (École Polytechnique, IP Paris, CNRC), ²LIPN (Sorbonne Paris Nord), ³Crédit Agricole SA
{jeremie.dentan, sonia.vanier}@polytechnique.edu

Abstract

Claim-level Uncertainty Quantification (UQ) is a promising approach to mitigate the lack of reliability in Large Language Models (LLMs). We introduce MUCH, the first claim-level UQ benchmark designed for fair and reproducible evaluation of future methods under realistic conditions. It includes 4,873 samples across four European languages (English, French, Spanish, and German) and four instruction-tuned open-weight LLMs. Unlike prior claim-level benchmarks, we release 24 generation logits per token, facilitating the development of future white-box methods without re-generating data. Moreover, in contrast to previous benchmarks that rely on manual or LLM-based segmentation, we propose a new deterministic algorithm capable of segmenting claims using as little as 0.2% of the LLM generation time. This makes our segmentation approach suitable for real-time monitoring of LLM outputs, ensuring that MUCH evaluates UQ methods under realistic deployment constraints. Finally, our evaluations show that current methods still have substantial room for improvement in both performance and efficiency.

Keywords: Large Language Models, Uncertainty Quantification, Evaluation Benchmark

 orailix/much

 orailix/MUCH

 much-segmenter

1. Introduction

Despite significant improvements in their performance, the reliability of Large Language Models (LLMs) remains an open challenge, as these models are prone to producing plausible yet non-factual content, usually referred to as hallucinations (Huang et al., 2025; Zhang et al., 2025; Sahoo et al., 2024; Ji et al., 2023). To mitigate the consequences of factuality hallucinations, Uncertainty Quantification (UQ) techniques have been developed to estimate an LLM’s confidence in its responses (Shorinwa et al., 2026; Xia et al., 2025; Geng et al., 2024; Huang et al., 2024). Most UQ benchmarks focus on a single aggregate uncertainty score for the entire generation. However, this response-level score does not provide a fine-grained view of uncertainty, and a long LLM generations risk being rejected even if only a small part is incorrect. To address these limitations, claim-level UQ methods were recently proposed (Fadeeva et al., 2024). These methods provide a local uncertainty score for each distinct idea in the output.

Limitations of existing UQ benchmarks Despite promising impact across various application scenarios, claim-level UQ remains an emerging field. We identify two key limitations in existing benchmarks, summarized in Table 1. First, no benchmark releases multiple logits per LLM-generated token, making these datasets unsuitable for developing and evaluating new white-box UQ

Benchmark	Scale	Segmenter	Size	Logits
Mu-SHROOM	Span	Human	2.4k	✗
LM-Polygraph	Claim	LLM	818	✗
PsiloQA	Span	LLM	70k	✗
HalluEntity	Entity	Human+LLM	157	✗
RAGTruthQA	Span	Human	18k	✗
FAVA	Span	Human	902	✗
MUCH	Claim	Algorithmic	4.8k	✓

Table 1: Existing benchmarks: Mu-SHROOM (Vázquez et al., 2025), LM-Polygraph (Vashurin et al., 2025), PsiloQA (Rykov et al., 2025), HalluEntity (Yeh et al., 2025), RAGTruthQA (Niu et al., 2024), FAVA (Min et al., 2023), and MUCH (ours). We show the hallucination scale, segmentation method “Segmenter”), dataset size, and whether logits are available.

methods, which typically require access to multiple logits. Second, segmentation is performed under unrealistic conditions, which does not allow for a reliable estimation of UQ performance for real-time monitoring of LLM outputs in production. Existing benchmarks either split LLM generations into claims that are then annotated (“Claim” in Tab. 1), or directly annotate spans within generations (“Span”), or annotate entities from human-defined claims (“Entity”). Human-based approaches are clearly impractical in production, while LLM-based ones are computationally expensive, requiring as much computation as the output generation itself. They are also stochastic and non-reproducible, limiting

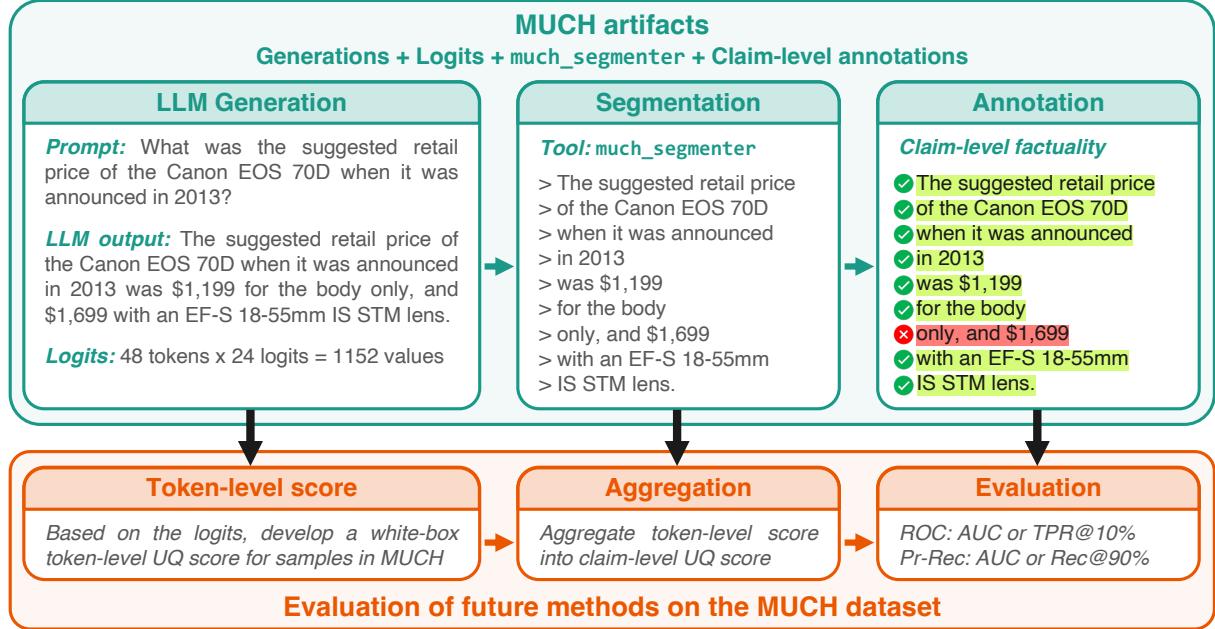


Figure 1: We open-source four artifacts as part of the MUCH benchmark: (1) 4,873 LLM generations spanning four languages (English, French, Spanish, and German) and four models (Llama 3.1 8B, Llama 3.2 3B, Minstral 8B, and Gemma 3 4B); (2) 24 logits per generation token; (3) much_segmenter, a fast and reproducible claim segmenter; and (4) claim-level factuality annotations for every sample, totaling 20,751 binary annotations. This framework facilitates the evaluation of future methods, which only requires defining a new token-level score, aggregating it, and comparing it to the claim-level annotations.

the generalizability of evaluation results. These limitations often force new methods to regenerate evaluation data and re-implement baselines for comparison (Fadeeva et al., 2024; Farquhar et al., 2024)

Benchmarking future methods with MUCH To address these limitations, and to support the development of new UQ approaches, we introduce MUCH, a new multilingual claim hallucination dataset and evaluation protocol, illustrated in Figure 1. The dataset contains 4,873 pairs of question and LLM-response. Importantly, we provide 24 logits per LLM-generated token, enabling future research to directly evaluate new white-box, logit-based approaches without re-generating data. We also release much_segmenter, a new claim segmentation algorithm that addresses the limitations of LLM-based and human-based segmentation. Relying on keyword and punctuation cues, it is fully deterministic and extremely fast: segmenting the entire dataset requires only 0.2% of the computation time needed for LLM generation. It is independent of any UQ method, ensuring fair comparisons with future approaches and eliminating the need for re-annotation. Finally, we provide an annotation in $\{-1; +1\}$ to assess the factuality of each of the 20,751 claims in the dataset.

The construction of MUCH is presented in Figure 2. The questions are taken from the Mu-

SHROOM test set (Vázquez et al., 2025) and consist of factual questions answerable from a single Wikipedia page provided with each question. This ensures that the factuality of responses can be evaluated objectively. We retained four European languages (English, French, Spanish and German), with approximately 200 questions per language. For each question, we generated 8 LLM responses using two different temperatures and four instruction-tuned open-weight models (Llama 3.1 8B Instruct and 3.2 3B Instruct (Grattafiori et al., 2024), Minstral 8B Instruct (Minstral AI Team, 2025), and Gemma 3 4B Instruct (Team et al., 2025)). Then we automatically annotated claim-level factuality using the corresponding Wikipedia page and two different LLMs (GPT-4o and GPT-4.1). To ensure annotation quality, we retained only the 4.8k generations for which GPT-4o and GPT-4.1 labels perfectly matched. Finally, a small portion of the dataset was double-annotated by humans, showing that the gap between automatic and human labels is comparable to inter-human variability.

Summary of contributions

- We release MUCH, a benchmark for claim-level uncertainty quantification in English, French, Spanish and German;
- We release generation logits to support the development of new white-box UQ methods;

- We release `much_segmenter`, a fast and deterministic claim segmentation method;
- We document the construction of MUCH and compare automatic annotations to human labels to ensure quality and reproducibility;
- We benchmark the current best claim-level UQ methods and discuss directions for improving both performance and efficiency.

2. Background and Related Works

Fact uncertainty quantification benchmarks
 Literature distinguishes between *faithfulness* and *factuality* hallucinations (Huang et al., 2025), or alternatively between *input-conflicting*, *context-conflicting* and *fact-conflicting* hallucinations (Zhang et al., 2025). Our benchmark focuses exclusively on factuality hallucinations in LLM responses. Numerous response-level benchmarks have been proposed for this task, including BioASQ (Krithara et al., 2023), TruthfulQA (Lin et al., 2022), NQ (Kwiatkowski et al., 2019), SQuAD (Rajpurkar et al., 2018), SVAMP (Patel et al., 2021) or TriviaQA (Joshi et al., 2017). However, this paper only focuses on claim-level UQ, which provides a finer-grained view of uncertainty in LLM responses than response-level approaches. The existing claim-level UQ benchmarks we are aware of are listed in Table 1. As discussed in the Introduction, the lack of efficient and deterministic segmentation methods, along with the absence of logits, makes these benchmarks inadequate for evaluating future claim-level UQ methods in settings that generalize to real-world scenarios.

White-Box, Sample-Specific UQ UQ literature distinguishes between *white-box* approaches, such as Fadieeva et al. (2024); Sriramanan et al. (2024); Chen et al. (2024); Kadavath et al. (2022); Duan et al. (2024), which require access to logits, internal activations, or the model itself, and *black-box* approaches, such as Kuhn et al. (2023); Nikitin et al. (2024); Lin et al. (2023), which rely only on the generated text. UQ methods can also be classified as sample-specific or population-level (Sriramanan et al., 2024). The former assign a UQ score to a single LLM response, while the latter assign a score to multiple LLM generations for the same prompt.

Our benchmark focuses on white-box, sample-specific UQ. First, we posit that the reliability of an LLM’s output is primarily the responsibility of the provider who generates them and, consequently, has white-box access to the model. Second, substantial efforts are made to reduce inference time and computational costs. Population-level approaches require multiple generations, which contradicts these goals and limits their adoption.

Claim segmentation All claim-level UQ methods we are aware of rely on prompt-based segmentation by an LLM or human-based segmentation (see Table 1). However, such LLM-based segmentation is impractical in realistic scenarios for three main reasons, which substantially limits the utility of the benchmarks proposed in these works. First, LLM-based segmentation is intrinsically non-deterministic, and some papers do not report the exact LLM version used, preventing results from generalizing to future applications. Second, LLM segmentation is costly, requiring the segmenter-LLM to re-generate all claims, which is at least as costly as generating the target LLM outputs for which UQ is computed. Finally, LLM-based segmentation requires mapping target LLM output tokens to the claims produced by the segmenter-LLM, which is a non-trivial problem. For instance, Fadieeva et al. (2024) report that about 5% of claims could not be mapped to original tokens, which is unacceptable in many applications. On the opposite, our segmenter is deterministic, extremely fast (only 0.2% of the computation cost of LLM generation), and directly aligns with the target generation tokens, making it suitable for realistic applications.

Existing sample-specific, claim-level methods
 Few approaches have been developed specifically for claim-level UQ. In Section 5, we evaluate on MUCH several baselines often cited as top-performing in existing benchmarks: CCP (Fadieeva et al., 2024), SAR (Duan et al., 2024), Token Likelihood (Guerreiro et al., 2023), Token Entropy (Malinin and Gales, 2021), and Maximum Likelihood (Aichberger et al., 2024). Although some of these methods were not designed for claim-level UQ, they produce token-level scores that can be aggregated at the claim level. We did not include FOCUS (Zhang et al., 2023), despite its strong performance on several benchmarks (Yeh et al., 2025; Rykov et al., 2025), because it relies on attention scores, which are unavailable in modern implementations such as Flash-Attention (Dao et al., 2022; Dao, 2023), making it impractical for production.

3. Methodology

The construction of MUCH is illustrated in Figure 2 and detailed in the following sections.

3.1. Collecting questions

First, we collect questions by filtering the 200 questions in English, French, Spanish and German from Mu-SHROOM dataset (Vázquez et al., 2025) (see [A] in Fig. 2). We chose Mu-SHROOM because it is multilingual and well-documented, containing only factual and non-ambiguous questions. Moreover, it

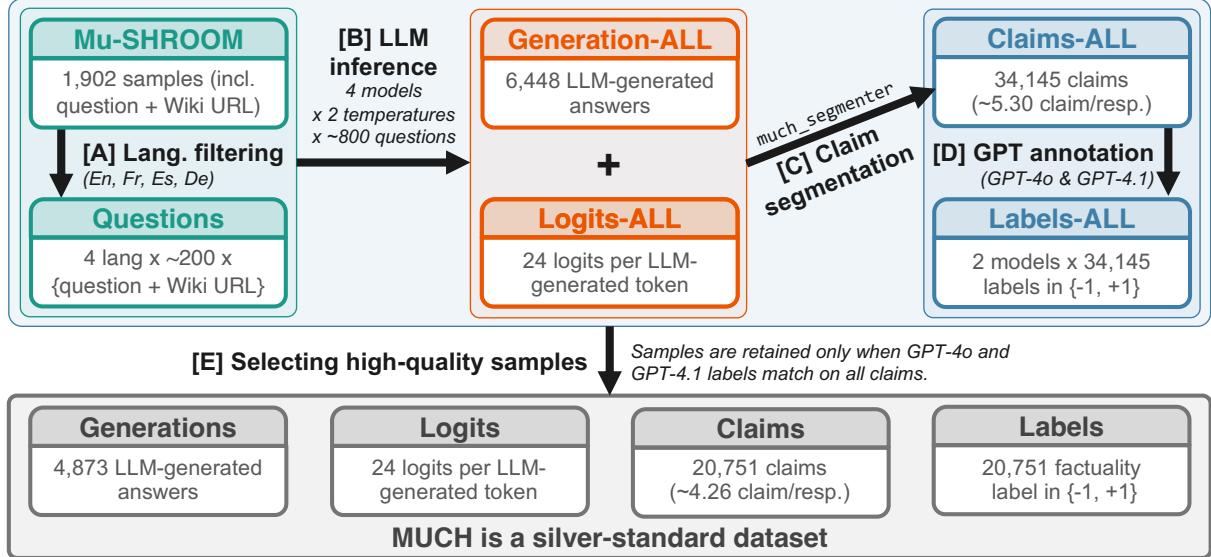


Figure 2: Construction pipeline of MUCH benchmark. We filter English, French, Spanish, and German questions from the Mu-SHROOM test set (Vázquez et al., 2025) (see [A]). We then generate eight LLM answers per question, and retain 24 logits per generated token (see [B]). Next, we use `much_segmenter` to parse LLM generations (see [C]). We automatically assign two binary labels to each claim, one using GPT-4o and one using GPT-4.1 (see [D]). Finally, we retain only high-quality annotations by filtering out samples where GPT-4o and GPT-4.1 labels mismatch on at least one claim (see [E]).

is designed for hallucination detection, containing difficult questions for which small LLMs are likely to hallucinate, which is necessary to obtain interesting annotations in MUCH. Finally, Mu-SHROOM provides the URL of a Wikipedia page containing the answer to each question, which is necessary for high-quality automatic annotation (see Section 3.4).

3.2. LLM generation

For LLM generation (see [B] in Fig. 2), we use four open-weight models for generation: “gemma-3-4b-it” (Team et al., 2025); “Minstral-8B-Instruct-2410” (Mistral AI Team, 2025); “Llama-3.1-8B-Instruct” (Grattafiori et al., 2024); and “Llama-3.2-8B-Instruct” (Grattafiori et al., 2024). We used the instruct versions to ensure the models accurately follow the prompt. Answers were generated using two temperatures (1.0 and 0.7), resulting in eight generations per question. Generations have an average length of 24.5 tokens and 97.3 characters. Finally, we retain 24 logits per generation token, which is sufficient for most white-box methods. For example, the state-of-the-art method CCP (Fadeeva et al., 2024) uses only 10 logits per token. Out of the 6,448 LLM generations, four contained a token that was not among the top-24 most likely tokens. For simplicity, we excluded these four samples in step [E]. The generation hyperparameters, including system prompt, user prompt, temperature, seed, library versions and other relevant settings, are provided in Appendix A and B.

3.3. Claim Segmentation

Claim segmentation is a core component of our pipeline (see [C] in Fig. 2). As explained in the introduction, the segmenter needs to be fast to minimise the overhead of uncertainty quantification. It must also be reproducible and independent of any UQ method to ensure fair comparison and avoid re-annotating evaluation data for future evaluations. Finally, it must always map claims to chunks of tokens generated by the target LLM, since most white-box UQ methods rely on token-level computations. We introduce `much_segmenter`, a new claim segmentation algorithm that meets these four requirements. It is fully rule-based and does not require external models or internet access, making it suitable for offline or computation-limited use cases. It is designed for English, French, Spanish, and German. We retain only these four European languages because their stopword and punctuation systems are similar. We expect our segmenter to be easily adaptable to languages with similar punctuation and stopwords, although we have not tested it beyond the four languages mentioned.

On the importance of segmentation for UQ

The most straightforward approach to annotate the factuality of LLM generations in a fine-grained manner would be to annotate each token individually. However, individual tokens lack the contextual structure needed to represent semantic uncertainty, which arises over spans of text rather than single

units. For instance, in the example of Figure 1, the non-factual part mainly involves four tokens: “\$”, “1”, “,” and “699”. Yet, it would be arbitrary to decide whether tokens “\$” and “,” should be marked as non-factual, since they could appear in a correct response, or whether all four tokens should be marked as non-factual. This difficulty makes detecting precise span-level boundaries particularly challenging in span-level UQ benchmarks (Rykov et al., 2025). Consequently, to focus on semantic hallucinations and avoid imposing token-level factuality conventions that could unnecessarily constrain future methods, we focus on claim-level factuality.

Our segmentation algorithm. We release a PyPI implementation of our segmentation algorithm for real-world use. Algorithm 1 presents the corresponding pseudo-code, which consists of two main steps. First, we split the LLM generation `inpt` into words using an external word tokenizer, and we use these words to identify the character indices of claim starts (ll. 2-22). Second, we map these character indices to the tokens of the LLM generation (ll. 24-41). As a result, `much_segmenter` outputs a list of lists: each contains the token indices for a claim, for example: `[[0,1], [2,3,4], [5]]`.

In the first step, we use nltk’s TreebankWordTokenizer to split `inpt` into words and punctuation (l. 3), and we compare them to a list of stopwords and punctuation marks (ll. 13-14). This list contains nltk’s stopword list in each language, plus string punctuation list as well as additional custom punctuation marks. We avoid LLM tokenizers because they differ across models, which would make this step unreliable. When we detect a stopword or punctuation mark, we add the index of its first character to the list of claim starts, since stopwords and punctuation often introduce new ideas that form separate claims (l. 17). However, in the case of several stopwords or punctuation marks in a row, we only start one single claim (see `stop_prev` in ll. 15-18). Finally, nltk word tokenizer merges points “.” to the end of the preceding word, contrary to other punctuation marks. For that reason, we introduce `stop_next` to start a new claim right after a word ending with a point (ll. 15 and 19). An example of claim segmentation is provided below, with stopwords and punctuation marks in bold. We see that the three ideas in the sentence are correctly separated into three separate claims.

No, Xining **is** the largest city **in** Qinghai.

The second step maps the character indices to the tokens generated by the LLM (ll. 24-41). Because claims are defined by their character indices, we can always map them to LLM tokens, unlike

```

def much_segmenter(inpt, llm_tokenizer):
    # Parsing claims with a word tokenizer
    word_spans ← word_tokenizer(inpt)
    claim_starts ← []
    stop_prev, stop_next ← False, False
    eos_index ← index(inpt, eos)
    # Analysing each word
    foreach (start, stop) in word_spans:
        if stop ≥ eos_index:
            add eos_index to claim_starts
            break
        w ← inpt[start:stop]
        is_stop ← (stop_next or w in stpwd
                   or w in punct)
        if is_stop and not(stop_prev)
            and start ≠ 0:
            add start to claim_starts
        stop_prev ← is_stop
        stop_next ← (w ends with ".")
    # End of the last claim
    if claim_starts[-1] ≠ length(inpt):
        add length(inpt) to claim_starts
    #
    # Mapping claims to LLM tokens
    result, current_claim ← [], []
    tokenized ← llm_tokenizer(inpt)
    for idx in 1, ..., length(tokenized):
        idx_end ← index(last char
                         of token idx)
        if (claim_starts is not empty
            and current_claim is not empty
            and idx_end > claim_starts[0]):
            add current_claim to result
            claim_starts ← claim_starts[1:]
            current_claim ← [idx]
        else:
            add idx to current_claim
    # Final flush
    if current_claim is not empty:
        add current_claim to result
    return result

```

Algorithm 1: Pseudocode of `much_segmenter`

prompt-based segmentation methods such as the one used in (Fadeeva et al., 2024), which fails in about 5% of cases. Note that the end-of-sequence token always form its own claim, but we exclude this special claim in the next steps of the pipeline.

3.4. Automated Annotation

We automatically annotate the factuality of each claim (see [D] in Fig. 2). Although human annotation remains the gold standard for assessing factuality, it is time-consuming and costly, particularly for large datasets. When we manually annotated a subset of the data (see Section 3.5), we found that humans process roughly 350 claims per hour, which would translate to about 100 hours to annotate the full dataset. Moreover, relying on human annotation would make extending the dataset

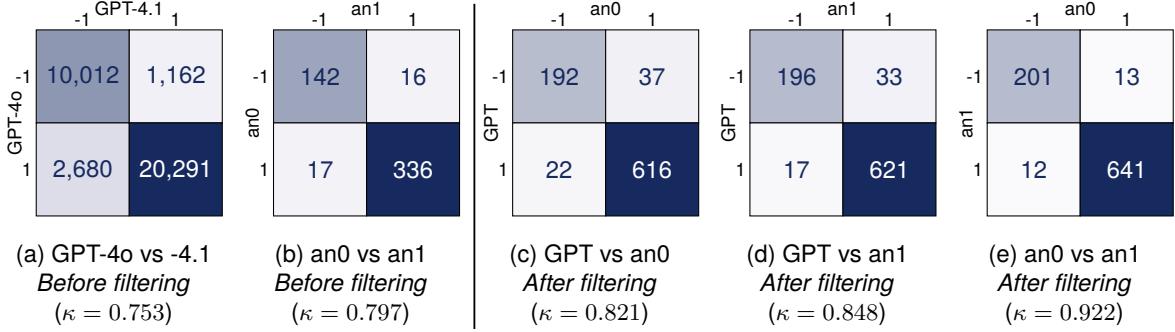


Figure 3: Confusion matrices comparing claim annotations from GPT-4o, GPT-4.1, and human annotators (an0, an1), before and after sample filtering (see [E] in Figure 2 and Section 3.5). Cohen’s kappa (κ) quantifies inter-annotator agreement. Figure 3a: GPT-4o vs GPT-4.1 on the 34.1k claims of the 6.4k samples before filtering. Figure 3b: two human annotators on the 511 claims of 100 random samples before filtering. Figure 3c-3e: GPT vs an0 vs an1 on the 865 claims of 200 random samples after filtering.

or adding new languages prohibitively expensive. Therefore, we adopt automated annotation. We discuss the quality of these annotations and compare them with human annotations in Section 3.5.

Two models are used independently to provide two sets of annotations: GPT-4o (“gpt-4o-2024-11-20”) and GPT-4.1 (“gpt-4.1-2025-04-14”). Both models are prompted to classify each claim as either “correct” or “incorrect”. As contextual knowledge, we provide the Wikipedia page associated with the question (see Section 3.1). We select three relevant chunks of approximately 120 tokens based on cosine similarity with the question, using OpenAI’s “text-embedding-3-large” model to compute embeddings. The entire annotation process involved about 17M GPT completion tokens and 13k requests, for a total cost of roughly USD 50.

A binary annotation Each claim is labeled as either factual or non-factual. In the initial stages of the project, we included a neutral label for cases where the information was ambiguous or irrelevant to the question. However, in practice, the LLM annotation rarely contained this label, and the neutral label was the root cause of most disagreements between annotators. We therefore decided to drop the neutral label and re-annotate the dataset with only positive and negative labels allowed. The filtering stage [E] (see Section 3.5) replaces the neutral label by removing samples containing ambiguous claims on which GPT-4o and GPT-4.1 disagree.

Annotation guidelines The complete annotation guidelines, as provided in the annotation prompt and to human annotators, are available in Appendix C. We instruct annotators to consider denial of answer as correct, since they do not introduce hallucinations. Moreover, the label “incorrect” should be specific to non-factual claims and not to the surrounding phrasing, as in:

Question: Who was the first Carolingian king?

Answer: The first Carolingian king was Clovis.

✓ ✗

3.5. Filtering and Quality Verifications

Automated annotations are generally less reliable than human annotations. Nevertheless, recent advances in LLM technology allow them to perform many tasks with sufficiently high quality, especially when provided with a trusted knowledge base known to contain the answer (here: the Wikipedia page, see Section 3.4). To ensure the quality of the data provided in MUCH, we filter out the least reliable annotations (see [E] in Figure 2) and manually annotate a random subset of the remaining samples for quality verification (see Figure 3).

Filtering Out of the 6,448 generations containing 34,145 claims obtained so far, we only retain 4,873 samples (75.6% of the total) containing 20,751 claims (60.8% of the total) in the final version of MUCH. We perform filtering by retaining only samples where GPT-4o and GPT-4.1 annotations agree on all claims. See Appendix E for details and post-filtering statistics on the MUCH benchmark. We observe that Cohen’s kappa (Cohen, 1960) between GPT-4o and GPT-4.1 is $\kappa = 0.753$, which is close to inter-human agreement $\kappa = 0.797$ (see Figures 3a and 3b). Such values of $\kappa \in [0.6, 0.8]$ are usually considered as a “substantial” agreement (Landis and Koch, 1977). However, these values indicate that many claims are ambiguous, even for humans. We remove samples containing such claims from our benchmark to avoid biasing the evaluation of future methods with ambiguous claims or unreliable annotations. On average, these ambiguous samples are longer, with 5.30 claims per sample compared to 4.26 claims per sample in MUCH.

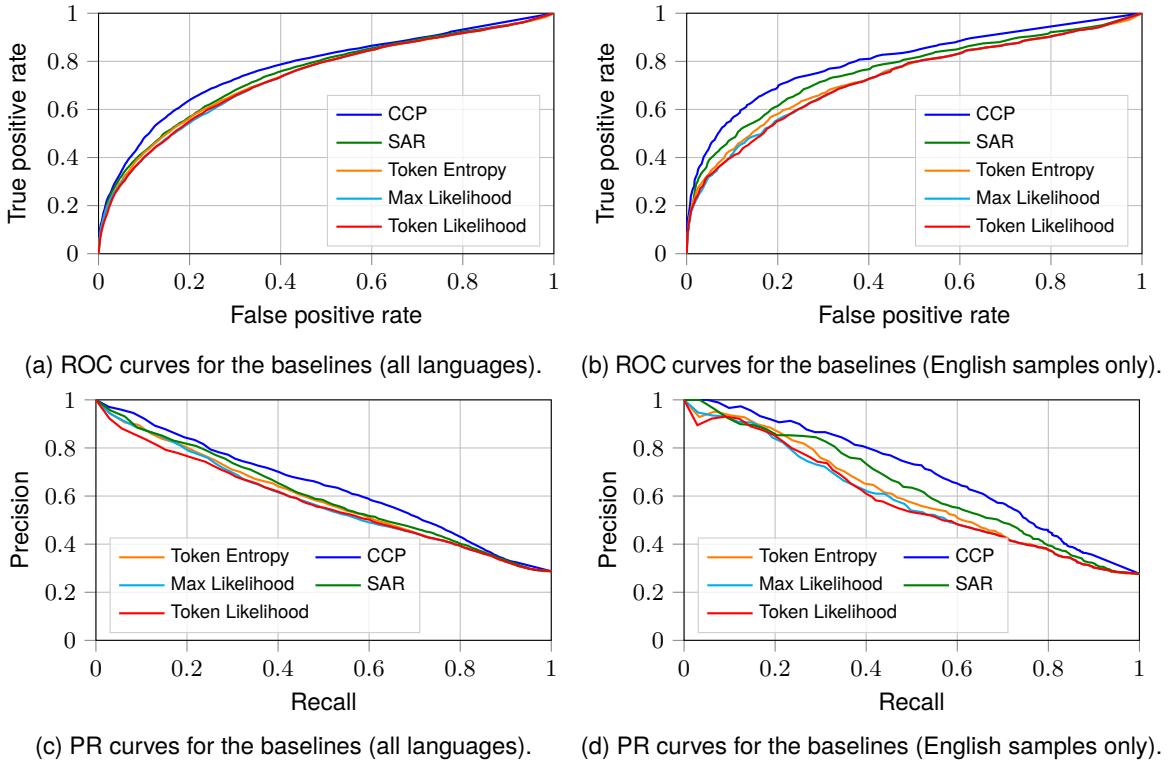


Figure 4: Evaluation of baseline methods on MUCH. The state-of-the-art CCP method (Fadeeva et al., 2024) outperforms other approaches, but there remains considerable room for improvement.

Quality Verifications After filtering, we randomly selected 50 samples per language, for a total of 867 claims. Two human annotators fluent in all four languages then independently annotated these 200 samples. They received the same instructions as those used for automated annotation (see Section 3.4) and were allowed to consult any Wikipedia page. Our results appear in Figures 3c-3e. We refer to automated labels as "GPT" because GPT-4o and GPT-4.1 annotations match for these samples. We release these human annotations alongside the MUCH benchmark as a gold-standard subset.

Inter-human agreement increases substantially after filtering, confirming that ambiguous samples were effectively removed ($\kappa = 0.922$ vs 0.797 , see Fig. 3e). The fact that humans still disagree on some claims highlights the task's inherent subjectivity. GPT-vs-human confusion matrices are nearly diagonal, indicating strong alignment between human and automated annotators. Both annotators show similar agreement with GPT labels ($\kappa = 0.821$ and $\kappa = 0.848$), values usually considered as "almost perfect" agreement (Landis and Koch, 1977). These results suggest that automated annotations are nearly as consistent with human judgments as humans are with each other. Remaining disagreements often occur when the Wikipedia page lacks the information needed to verify a claim. Overall, the automated annotations provide a reliable silver-standard benchmark for evaluating UQ methods.

	Llama -3.1-8B	Llama -3.2-3B	Gemma -3-4b	Minstral -8B	All
EN	44.0%	38.0%	74.0%	61.7%	55.6%
FR	54.9%	48.2%	81.9%	65.0%	63.8%
ES	40.6%	51.8%	65.2%	51.2%	53.0%
DE	55.9%	55.5%	87.1%	75.6%	69.1%
All	48.8%	48.4%	76.9%	63.7%	60.4%

Table 2: Proportion of MUCH samples containing at least one wrong claim, per model and per language.

4. Hallucination statistics

The proportion of sample in MUCH containing at least one non-factual claim is 60.4% overall, but it varies considerably across models and languages, as shown in Table 2. Although Gemma-3-4b is not the smallest model in our evaluation, it shows the highest hallucination rates across all languages. The two Llama models exhibit similar overall performance, though results differ notably across languages. Minstral exhibits similar or higher values than the Llama models across all languages.

Among samples containing at least one non-factual claim, the proportion of non-factual claims per answer is 49.8% on average, though it varies considerably across models. It is 42.4%, 44.2%, and 46.6% for Llama 3.1, Llama 3.2, and Minstral, respectively, and rises to 68.0% for Gemma-3.

Method	Absolute runtime (and relative to generation)			Performance	
	Segmentation	UQ	Total	ROC-AUC	PR-AUC
CCP (adapted)	6s (0.2%)	3,410s (124%)	3,416s (124%)	0.772	0.639
SAR (adapted)	6s (0.2%)	613s (22.2%)	619s (22.4%)	0.746	0.603
Max Likelihood	6s (0.2%)	8s (0.3%)	14s (0.5%)	0.732	0.582
Token Likelihood	6s (0.2%)	8s (0.3%)	14s (0.5%)	0.732	0.574
Token Entropy	6s (0.2%)	9s (0.3%)	15s (0.5%)	0.737	0.591

Table 3: Execution time and performance of each baseline on the MUGH benchmark. Runtime is broken down into segmentation (with `much_segmenter`) and UQ, and reported both in absolute terms and as a proportion of LLM generation (2,758s). We report ROC-AUC and Precision–Recall (PR) AUC.

5. Evaluating existing baselines

Aggregating token-level scores into claims
 We evaluated four baselines on MUGH. Each baseline produces a token-level UQ score, which we aggregate at the claim level. In all cases, we exclude stopwords from the aggregation, following common practice (Fadeeva et al., 2024). We tested four aggregation strategies: the (arithmetic) mean, the maximum value, the geometric mean, and the product of token-level UQ scores. Consistent with the findings of Fadeeva et al. (2024), we empirically observed that using the product yields the best performance. The results presented here rely on this aggregation strategy. Results with other aggregation strategies are available in Appendix F.

Baselines We evaluated five baselines on MUGH that are frequently cited as top-performing methods. We excluded FOCUS (Zhang et al., 2023) because it relies on attention scores, which are typically unavailable in production (see discussion in Section 2). First, we adapted **CCP** (Fadeeva et al., 2024). The main adaptation arises from the fact that the meaning of claims in MUGH is often not self-contained, whereas the original method operates on self-contained sentences. To preserve the performance of the Natural Language Inference (NLI) model used in CCP, we provide the model with the $\sigma = 8$ preceding and succeeding tokens of the token for which the CCP score is computed. Second, we adapted **SAR** (Duan et al., 2024). Like CCP, it relies on an entailment score computed on sentences. For the same reason, we feed the NLI model with the $\sigma = 8$ preceding and succeeding tokens around the target token (see discussion on hyperparameters in Appendix D). Third, we adapted **Maximum Likelihood** (Aichberger et al., 2024), where the token-level UQ score corresponds to the probability of the most likely token at each generation step. Fourth, **Token Likelihood** (Guerreiro et al., 2023) is defined as the likelihood of the sampled token. Finally, **Token Entropy** (Malinin and Gales, 2021) is the entropy of the probability distribution over the top-24 tokens provided in MUGH.

Evaluation metrics For performance, we report the Area Under the ROC Curve (ROC-AUC) and the Area Under the Precision–Recall Curve (PR-AUC), which are standard metrics for claim-level UQ (Vashurin et al., 2025). We also report the computation time of each method, a critical factor often omitted from benchmarks. We report computation time both in absolute terms and as a proportion of the time required for LLM generation, because UQ overhead relative to generation should remain minimal to enable large-scale, real-time uncertainty quantification. Conversely, methods such as CCP exclude segmentation from their runtime estimation, even though it adds at least a 100% overhead due to the LLMs involved (see Introduction). The introduction of `much_segmenter` in this paper substantially reduces this overhead (see Table 3).

A poor performance–efficiency trade-off Evaluation results are shown in Figure 4 and Table 3. Existing baselines achieve reasonable performance, with CCP reaching a ROC-AUC of 0.772. This value is close to the ROC-AUC of 0.74 reported for CCP on English/Mistral 7B in the private test of Vashurin et al. (2025), indicating that our adaptation preserved the baseline’s original behavior. The other baselines achieve slightly lower ROC-AUC scores. However, CCP is substantially more computationally expensive, with an overhead of about 124% of LLM generation time, for a modest gain in UQ performance.

Discussion Our evaluation shows that existing claim-level UQ methods still have substantial room for improvement. First, realistic applications require a low false positive rate and high precision. However, the best method evaluated here achieves only a TPR@FPR=10% of 48% and a Rec@Prec=80% of 23.5%, which remains too low for reliable UQ. We encourage future methods to focus on improving performance in these regions of the ROC and PR curves. Second, current methods perform better in English than in other languages (see Figure 4b) because they rely on NLI models that are more performant in English. Future methods should aim to

close the gap in UQ performance across languages. Third, future methods should report computation time relative to LLM generation and aim for minimal overhead, on the order of a few percent, to enable large-scale adoption for real-time monitoring of LLM outputs. To promote transparency regarding computation time and comparison with LLM generation runtime, we release a script to estimate the generation runtime on a given machine.

Conclusion

We release MUCH, a multilingual claim-level uncertainty quantification benchmark comprising 4.8k samples representing 20.7k claims. The benchmark includes factuality labels and 24 logits per token, supporting the development of new white-box methods and their fair and reproducible evaluation.

We also advocate for the use of a reproducible and computation-efficient segmentation algorithm, independent of any uncertainty quantification method. To this end, we introduce `much_segmenter`, a claim segmentation package for English, French, Spanish, and German, available on PyPI under a permissive license.

Finally, we benchmark existing strong baselines for claim-level uncertainty quantification. Our results show substantial room for improvement in both accuracy and efficiency. For performance, future methods should target low-FPR/high-precision regimes to support realistic deployment. For efficiency, future method should report runtime relative to LLM generation, aiming for minimal overhead to enable real-time monitoring of LLM outputs.

Limitations

Limited number of languages Our benchmark includes only four European languages: English, French, Spanish, and German. We focus on these languages because their punctuation and stopword systems are very similar, enabling the use of a single claim segmentation algorithm without requiring language detection. This choice simplifies and accelerates the segmentation. While these four languages cover many practical scenarios, expanding the benchmark to include additional languages would improve its generality. In particular, incorporating languages that do not use the Latin alphabet, such as Chinese, Arabic, or Russian, would be valuable, though it would increase the complexity of the segmenter.

Automated annotation vs human annotations During manual annotations performed for quality verification, we observed that, for some samples, assessing the factuality of an LLM generation requires consulting pages other than the one linked

to the question in Mu-SHROOM (Vázquez et al., 2025). This typically occurs when the LLM produces an unexpected yet potentially factual answer whose verification is nontrivial. Incorporating a human-generated *gold answer* for each question, as in (Rykov et al., 2025), could help improve the quality of automated annotations. Finally, allowing the annotation model to access multiple Wikipedia pages through a multi-agent pipeline with web search could further enhance annotation quality, albeit at a substantial computational cost.

Unavailability of attention weights We do not provide the attention weights of samples in MUCH, making our benchmark unsuitable for evaluating attention-based UQ methods such as FOCUS (Zhang et al., 2023). To mimic realistic LLM generation conditions, the MUCH samples were generated with the FlashAttention (Dao et al., 2022; Dao, 2023) algorithm, which does not allow retrieving attention weights. Because such implementations are now widely adopted, attention-based UQ methods are inapplicable in most scenarios.

Code and Data

Alongside this paper, we provide: the MUCH dataset (including LLM generations, logits, annotations, and generation configurations); the `much_segmenter` source code and PyPI package; token-level scores of all baselines evaluated; and a repository containing the source code, seeds, library versions, and hyperparameters necessary to reproduce the generation of MUCH and the evaluation of the baselines, along with a script to estimate LLM generation time on any machine.

 [orailix/much](#)  [orailix/MUCH](#)
 [much-segmenter](#)

Acknowledgement

As detailed in Section 3, the construction of MUCH benchmark involves some fields extracted from Mu-SHROOM dataset (Vázquez et al., 2025), a dataset released under CC-BY-4.0 license.

We thank Mohammed El Sharkawy, Lucas Thil, Mohamed Dhouib, Clément Elliker, Mathis Le Bail, Benoit Goupil and Martin Bonsergent-Brachet for discussions on early versions of this paper.

This work received financial support from the research chair *Trustworthy and Responsible AI* at École Polytechnique.

This work was granted access to the HPC resources of IDRIS under the allocation AD011014843R1 made by GENCI.

Bibliographical References

- Lukas Aichberger, Kajetan Schweighofer, and Sepp Hochreiter. 2024. [Rethinking Uncertainty Estimation in Natural Language Generation](#).
- Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. [INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection](#). In *ICLR*.
- Jacob Cohen. 1960. [A Coefficient of Agreement for Nominal Scales](#). *Educational and Psychological Measurement*, 20(1):37–46.
- Tri Dao. 2023. [FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning](#).
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. [FLASHATTENTION: fast and memory-efficient exact attention with IO-awareness](#). In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22.
- Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. [Shifting Attention to Relevance: Towards the Predictive Uncertainty Quantification of Free-Form Large Language Models](#). In *ACL*, volume 1, pages 5050–5063.
- Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, et al. 2024. [Fact-Checking the Output of Large Language Models via Token-Level Uncertainty Quantification](#). In *Findings of the ACL*, pages 9367–9385.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. [Detecting hallucinations in large language models using semantic entropy](#). *Nature*, 630(8017):625–630.
- Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koepll, Preslav Nakov, and Iryna Gurevych. 2024. [A Survey of Confidence Estimation and Calibration in Large Language Models](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. 2024. [The Llama 3 Herd of Models](#).
- Nuno M. Guerreiro, Elena Voita, and André Martins. 2023. [Looking for a Needle in a Haystack: A Comprehensive Study of Hallucinations in Neural Machine Translation](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1059–1075.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION](#). In *ICLR*.
- Hsiu-Yuan Huang, Yutong Yang, Zhaoxi Zhang, Sanwoo Lee, and Yunfang Wu. 2024. [A Survey of Uncertainty Estimation in LLMs: Theory Meets Practice](#).
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, et al. 2025. [A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions](#). *ACM Transactions on Information Systems*, 43(2):1–55.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, et al. 2023. [Survey of Hallucination in Natural Language Generation](#). *ACM Computing Surveys*, 55(12):1–38.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [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)*, pages 1601–1611.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, et al. 2022. [Language Models \(Mostly\) Know What They Know](#).
- Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Palioras. 2023. [BioASQ-QA: A manually curated corpus for Biomedical Question Answering](#). *Scientific Data*, 10(1):170.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. [Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation](#). In *ICLR*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, et al. 2019. [Natural Questions: A Benchmark for Question](#)

- Answering Research.** *Transactions of the Association for Computational Linguistics*, 7:453–466.
- J. Richard Landis and Gary G. Koch. 1977. **The Measurement of Observer Agreement for Categorical Data.** *Biometrics*, 33(1):159.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. **TruthfulQA: Measuring How Models Mimic Human Falsehoods.** In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.
- Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models. *TMLR*.
- Andrey Malinin and Mark Gales. 2021. **Uncertainty Estimation in Autoregressive Structured Prediction.** In *ICLR*.
- Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, et al. 2023. **FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation.**
- Mistral AI Team. 2025. Minstral 8B Instruct 2410.
- Alexander Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. 2024. Kernel Language Entropy: Fine-grained Uncertainty Quantification for LLMs from Semantic Similarities. In *NeurIPS*, volume 37, pages 8901–8929.
- Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. 2024. **RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models.** In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10862–10878.
- Arkil Patel, Satwik Bhattacharya, and Navin Goyal. 2021. **Are NLP Models really able to Solve Simple Math Word Problems?** In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. **Know What You Don’t Know: Unanswerable Questions for SQuAD.** In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.
- Elisei Rykov, Kseniia Petrushina, Maksim Savkin, Valerii Olisov, Artem Vazhentsev, Kseniia Titova, Alexander Panchenko, Vasily Konovalov, et al. 2025. **When Models Lie, We Learn: Multilingual Span-Level Hallucination Detection with Psil-oQA.**
- Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. **A Comprehensive Survey of Hallucination in Large Language, Image, Video and Audio Foundation Models.** In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724.
- Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. 2026. **A Survey on Uncertainty Quantification of Large Language Models: Taxonomy, Open Research Challenges, and Future Directions.** *ACM Computing Surveys*, 58(3):1–38.
- Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Katkanda, and Soheil Feizi. 2024. **LLM-Check: Investigating Detection of Hallucinations in Large Language Models.** In *NeurIPS*, volume 37, pages 34188–34216.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, et al. 2025. **Gemma 3 Technical Report.**
- Roman Vashurin, Ekaterina Fadeeva, Artem Vazhentsev, Lyudmila Rvanova, Akim Tsvigun, Daniil Vasilev, Rui Xing, Abdelrahman Boda Sadallah, et al. 2025. **Benchmarking Uncertainty Quantification Methods for Large Language Models with LM-Polygraph.** *TACL*, 13:220–248.
- Zhiqiu Xia, Jinxuan Xu, Yuqian Zhang, and Hang Liu. 2025. **A Survey of Uncertainty Estimation Methods on Large Language Models.** In *Findings of the ACL*, pages 21381–21396.
- Min-Hsuan Yeh, Max Kamachee, Seongheon Park, and Yixuan Li. 2025. **HalluEntity: Benchmarking and Understanding Entity-Level Hallucination Detection.** *Transactions on Machine Learning Research*.
- Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, et al. 2023. **Enhancing Uncertainty-Based Hallucination Detection with Stronger Focus.** In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 915–932.
- Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao,

et al. 2025. Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models.

Language Resource References

Vázquez, Raúl and Mickus, Timothee and Zosa, Elaine and Vahtola, Teemu and Tiedemann, Jörg and Sinha, Aman and Segonne, Vincent et al. 2025. *SemEval-2025 Task 3: Mu-SHROOM, the Multilingual Shared Task on Hallucinations and Related Observable Overagegeneration Mistakes*. Association for Computational Linguistics. PID <https://aclanthology.org/2025.semeval-1.322/>.

A. Computing infrastructure

Experiments were conducted on two devices: an HPC cluster and a local machine. The HPC cluster comprises nodes equipped with 2x AMD EPYC 7543 CPUs (32 cores, 64 threads each), 468 GB of RAM, and 8x NVIDIA A100 GPUs (80 GB each), running Red Hat Enterprise Linux 9.4. The computations on the cluster were deployed on jobs allocated 1/8 of a node (1 GPU, 8 cores, and 58.5 GB of RAM). The local machine is equipped with an Apple M4 Pro and 48 GB of RAM, running MacOS 15.6.1. Relevant software versions: Python 3.12.0, accelerate 1.17.0, datasets 3.6.0, numpy 2.3.0, torch 2.7.1, transformers 4.52.4.

The main computation steps were divided as described in Table 4. The HPC cluster was used for GPU-intensive computations (LLM generations, as well as CCP and SAR baselines due to the use of an NLI model).

Step	Machine
Question generation	Local
LLM generation	HPC cluster
Segmentation	Local
Wikipedia page caching	Local
GPT annotation	Local
Baselines: TL, MxL, TE	Local
Baselines: CCP and SAR	HPC cluster
Baseline evaluation	Local

Table 4: Separation of computation steps between HPC cluster and local machine.

B. Generation details

The hyperparameters used for LLM generation of MUCH samples are provided in Table 6. The system and user prompts used for generation are

shown in Figure 6. Generating the 6,448 samples took 4,540 seconds. Since only 4,873 samples are retained in MUCH and these samples contain an average of 4.26 claims, compared to 5.30 for all samples, we estimate the LLM generation time for MUCH as $4,540 \times (4,873/6,448) \times (4.26/5.30) = 2,758$ seconds.

C. Annotation instructions

The instructions provided to human annotators and the GPT annotation pipeline are detailed in Figure 5. These instructions were concatenated with the reference knowledge extracted from the Wikipedia page associated with each question. The reference knowledge consists of (i) the three chunks of approximately 120 tokens from the page that exhibit the highest cosine similarity with the question, and (ii) the *infobox* of the page. The *infobox* is the fixed-format table typically displayed in the top right-hand corner of articles to present a consistent summary of relevant information.

D. Baseline Details

NLI model for SAR and CCP SAR and CCP rely on an NLI model. We used the same model for both baselines: `microsoft/deberta-large-mnli` (He et al., 2021). This choice was motivated by its use in the original implementation of CCP (Fadeeva et al., 2024). The main limitation of this model is that it is not optimized for French, Spanish, and German, which is particularly problematic for Spanish, where CCP and SAR achieved poor results (see Appendix F).

Hyperparameters of CCP, SAR and Token Entropy We evaluated SAR with three hyperparameters: $\sigma \in \{3, 5, 8\}$. We recall that σ refers to the number of preceding and succeeding tokens passed as context to the NLI model. In the main part of the paper, “SAR” refers to the results with $\sigma = 8$, as this configuration achieves the best performance while maintaining an acceptable execution time (see Table 7). On the other hand, CCP relies on two hyperparameters. The first one is σ , which plays the same role as in SAR. The second metric is the number of possibilities evaluated per token, denoted by δ . For instance, since we only provide 24 logits per generated token, the maximum value of δ compatible with MUCH is 24. In the original paper, Fadeeva et al. (2024) used $\delta = 10$. We evaluated the CCP baseline with six hyperparameters: $\delta \in \{10, 24\}$ and $\sigma \in \{3, 5, 8\}$. In the main part of the paper, “CCP” refers to the results with $\delta = 10$ and $\sigma = 8$, since this configuration provides near-optimal performance while keeping computation time lower than the other CCP settings

(see Table 7). Moreover, the choice of $\delta = 10$ is consistent with the original paper (Fadeeva et al., 2024), and the choice of $\sigma = 8$ is consistent with that made for the SAR baseline. Finally, Token Entropy depends on a single hyperparameter: the number of possibilities considered when computing the entropy of the distribution for each generated token. We denote it by δ and evaluate three values, $\delta \in \{5, 10, 24\}$. In the main part of the paper, we selected $\delta = 24$ because it obtains the best results with a negligible increase of the computational cost. The performance of each configuration is discussed in Section F.

E. Samples filtered out for quality reasons

Out of the 6,448 generated samples, we retain only 4,873 in MUCH. Samples were filtered out for three reasons:

- 1,568 samples were removed because the labels annotated by GPT-4o and GPT-4.1 mismatch on at least one claim;
- 4 samples were removed because one of their tokens was sampled outside the top-24 most likely tokens;
- 3 samples were removed because they did not include an EOS token. These cases correspond to generation loops where the model endlessly repeats the same text snippet.

The proportion of samples per language and per model after the filtering step is presented in Table 5.

	Llama -3.1-8B	Llama -3.2-3B	Gemma -3-4b	Minstral -8B	Sum
EN	6.1%	6.2%	7.3%	7.2%	26.8%
FR	5.2%	5.2%	6.7%	6.1%	23.3%
ES	5.5%	5.7%	7.0%	5.9%	24.0%
DE	6.0%	6.5%	6.9%	6.6%	26.0%
Sum	22.8%	23.6%	27.8%	25.8%	100%

Table 5: Proportion samples per language and per model in MUCH benchmark.

F. Additional Results

Table 7 extends Table 3 with the performance for all hyperparameter configurations of SAR, CCP, and Token Entropy. The numbers appended to the baseline names refer to the hyperparameters. For CCP, the first number corresponds to δ and the second to σ . For SAR, the number corresponds

to σ , and for Token Entropy, it corresponds to δ . Figure 7 (resp. 9) display the ROC curve (resp. Precision–Recall curve) for all baselines, depending on the aggregator used for evaluation. The aggregator refers to the algorithm used to combine token-level UQ scores into a claim-level UQ value. Following (Fadeeva et al., 2024), we evaluated four aggregators: the (arithmetic) mean of token values in the claim, the maximum, the geometric mean, and the product. Consistent with the observations of (Fadeeva et al., 2024), our best results are obtained when using the product as aggregator. Figures 8 and 10 display the ROC and Precision–Recall curves for all baselines grouped by language. We observe that SAR and CCP achieve poor performance for languages other than English, and especially in Spanish. This can be explained by the fact that the NLI model used for these baselines was only optimized for English (see Appendix D). Evaluating computation time and performance across languages for alternative NLI models would be a promising direction for improving these baselines.

Parameter	Type	Value(s)
Model Name	Variable	meta-llama/Llama-3.2-3B-Instruct meta-llama/Llama-3.1-8B-Instruct mistralai/Minstral-8B-Instruct-2410 google/gemma-3-4b-bit
Temperature	Variable	{1.0, 0.7}
Eager	Fixed	False
Greedy	Fixed	False
Seed	Fixed	1234
Top p	Fixed	0.9
Top k	Fixed	20
Max new tokens	Fixed	500

Table 6: Hyperparameters of the LLM generations used for MUGH benchmark. We used two variable hyperparameter with 4 and 2 possibilities, leading to a total of 8 different configurations.

Method	Absolute runtime (and relative to generation)			Performance	
	Segmentation	UQ	Total	ROC-AUC	PR-AUC
CCP-24-8	6s (0.2%)	5,429s (197%)	5,435s (197%)	0.775	0.643
CCP-10-5	6s (0.2%)	3,230s (117%)	3,236s (117%)	0.772	0.633
CCP-24-5	6s (0.2%)	4,268s (155%)	4,274s (155%)	0.772	0.636
CCP-10-8	6s (0.2%)	3,410s (124%)	3,416s (124%)	0.772	0.639
CCP-24-3	6s (0.2%)	3,508s (127%)	3,514s (127%)	0.766	0.619
CCP-10-3	6s (0.2%)	4,047s (147%)	4,053s (147%)	0.766	0.616
SAR-8	6s (0.2%)	613s (22.2%)	619s (22.4%)	0.746	0.603
SAR-5	6s (0.2%)	510s (18.5%)	516s (18.7%)	0.745	0.603
SAR-3	6s (0.2%)	419s (15.2%)	425s (15.5%)	0.742	0.595
Token Entropy-24	6s (0.2%)	9s (0.3%)	15s (0.5%)	0.737	0.591
Token Entropy-10	6s (0.2%)	9s (0.3%)	15s (0.5%)	0.736	0.590
Token Entropy-5	6s (0.2%)	9s (0.3%)	15s (0.5%)	0.733	0.578
Max Likelihood	6s (0.2%)	8s (0.3%)	14s (0.5%)	0.732	0.582
Token Likelihood	6s (0.2%)	8s (0.3%)	14s (0.5%)	0.732	0.574

Table 7: Execution time and performance of each baseline on the MUGH benchmark. Runtime is broken down into segmentation (with `much_segmenter`) and UQ, and reported both in absolute terms and as a proportion of LLM generation (2,758s). We report ROC-AUC and Precision–Recall (PR) AUC. The numbers appended to the baseline names refer to the hyperparameters. For CCP, the first number corresponds to δ and the second to σ . For SAR, the number corresponds to σ , and for Token Entropy, it corresponds to δ .

Annotation Instructions given to both human annotators and GPT annotation pipeline

Task

Given the original **question**, the model's **answer**, a **segmented version** of that answer, and some **reference knowledge**, assign a factuality score to each segment.

Scoring Convention

-1 → Factually incorrect / contradicts previous ideas / unverifiable / approximative

1 → Factually correct

Be **tough and rigorous**: Only assign "1" when the content is clearly factually correct and relevant to the question.

Edge Cases

1. Denial of answer:

If the model explicitly states it cannot answer or denies having information (e.g., "I did not find any information..."), assign label 1. The denial itself is factually correct.

2. Targeted labels:

When a segment is incorrect, apply the label to the specific factual claim, not the surrounding phrasing. Example: Question = "Who was the first Carolingian king?"

Answer = "The first Carolingian king was Clovis."

Here, the incorrect part is "Clovis," not the framing "The first Carolingian..." .

3. Numbers:

3a. *Exact values required* (e.g., dates, rank in a competition, small quantities <100):

- Exact match → 1
- Otherwise → -1

3b. *Approximate values acceptable* (e.g., population, area):

- Exact match → 1
- Within 10% error with "around", "approximately" or similar context → 1
- Within 10% error without such context → -1
- Larger error → -1

4. Chunk granularity:

If a chunk contains any factually false statement, the entire chunk should be labeled -1.

5. Overage generation mistakes and false details:

If the answer contains overgeneration that adds details to an incorrect claim (e.g., stating the citizenship or birthdate of the wrong person, or providing details about an event that did not occur), assign -1 to both the incorrect claim and any overgenerated or unverifiable details associated with it.

6. EOS token

End-of-sequence tokens such as <eos>, </s> or <| eot_id |> should be labeled 1, except if they appear in the same chunk as incorrect information (see rule 4).

Figure 5: Annotation instructions

Prompt used for the generation of MUCH dataset

System: You are a helpful assistant. Always answer questions directly. Your answers should be very concise and precise.

User: <question>

Assistant:

Figure 6: Generation prompt

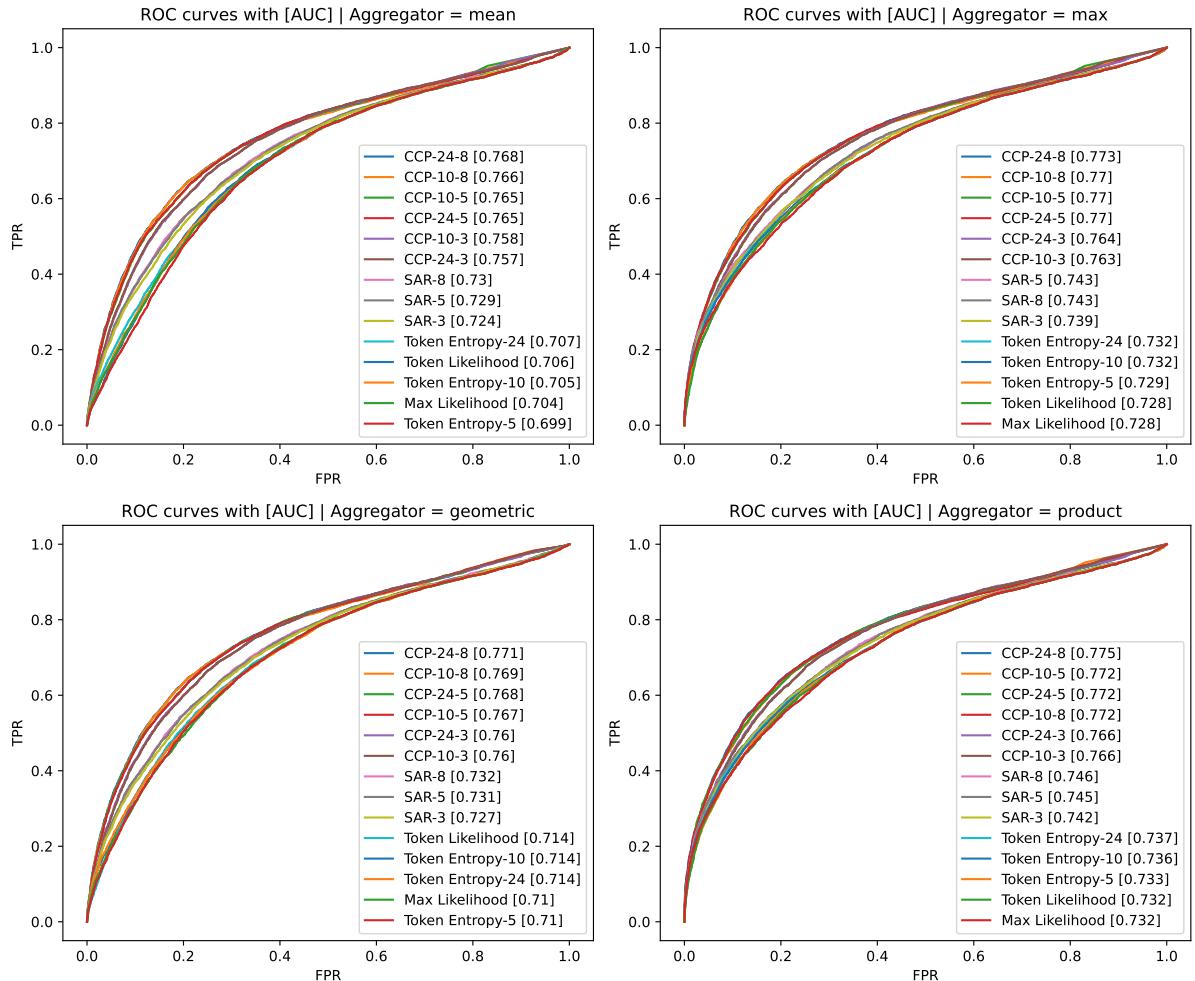


Figure 7: ROC curves for all baselines, depending on the aggregator used for evaluation.

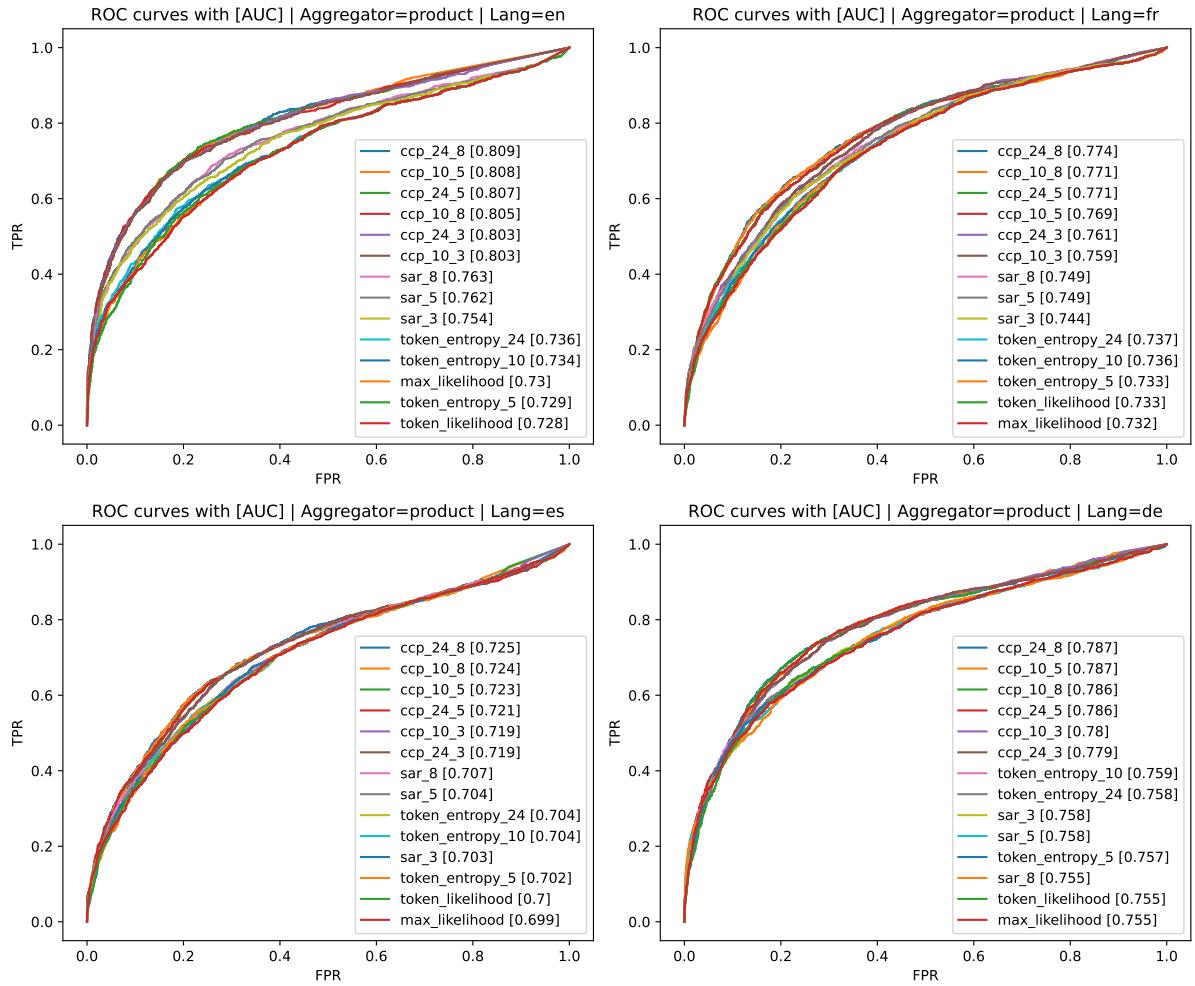


Figure 8: ROC curves for all baselines, grouped by language.

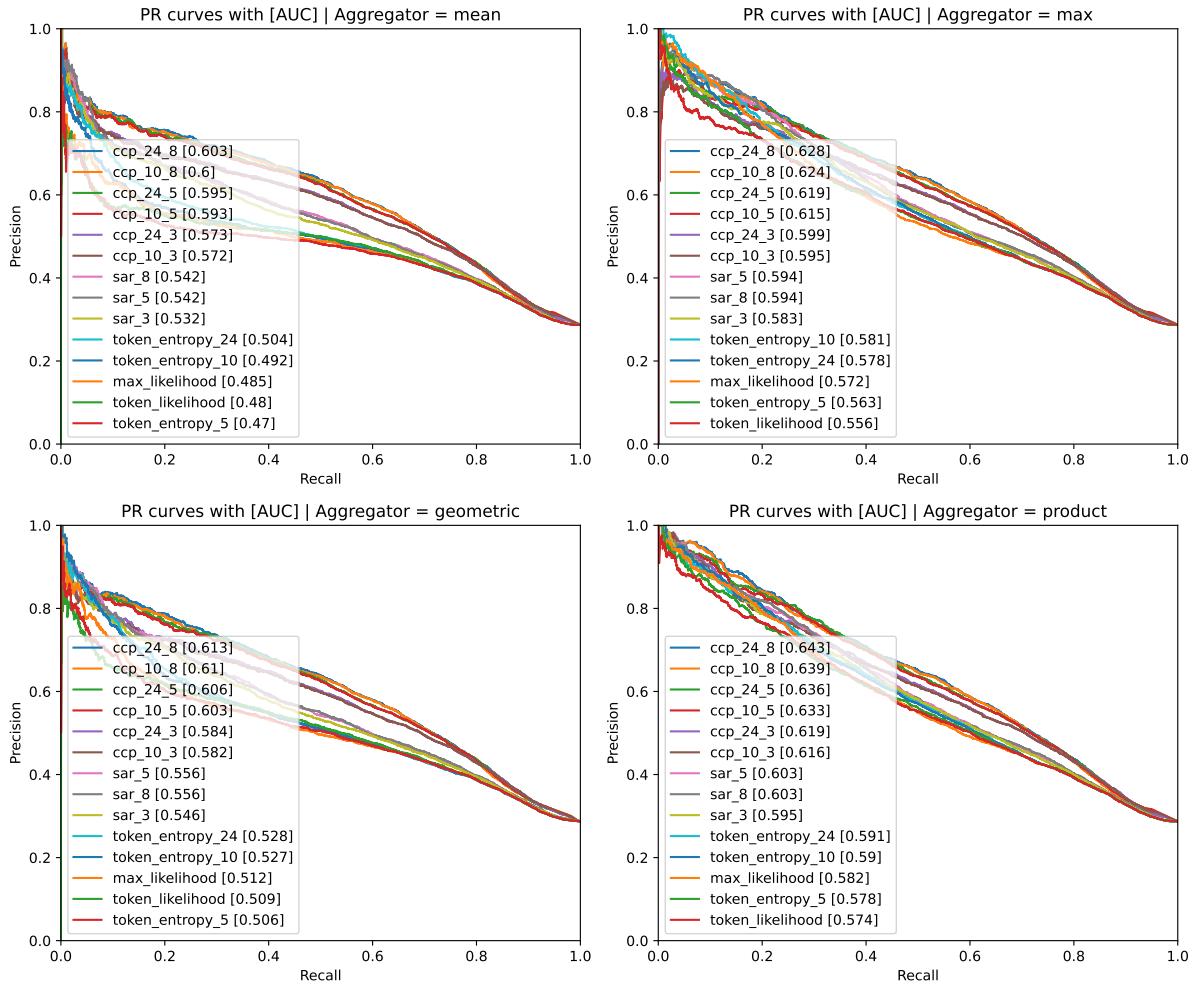


Figure 9: Precision-Recall curves for all baselines, depending on the aggregator used for evaluation.

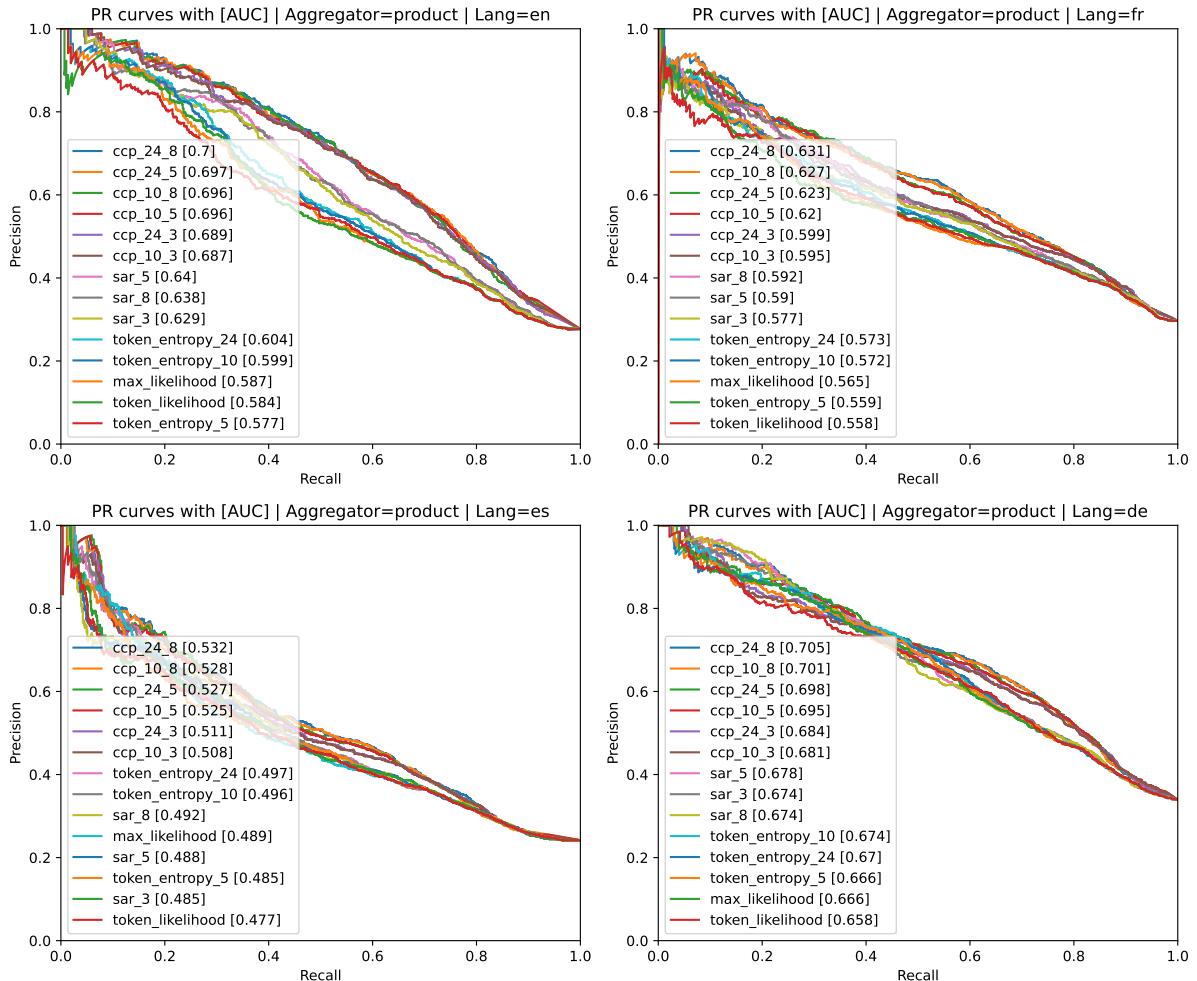


Figure 10: Precision-Recall curves for all baselines, grouped by language.