UQLM: A Python Package for Uncertainty Quantification in Large Language Models

Dylan Bouchard[1] DYLAN.BOUCHARD@CVSHEALTH.COM
Mohit Singh Chauhan[1] MOHITSINGH.CHAUHAN@CVSHEALTH.COM
David Skarbrevik[1] DAVID.SKARBREVIK@CVSHEALTH.COM
Ho-Kyeong Ra[1] HOKYEONG.RA@CVSHEALTH.COM
Viren Bajaj[1] BAJAJV@AETNA.COM
Zeya Ahmad[1] ZEYA.AHMAD@CVSHEALTH.COM
[1]*CVS Health, Wellesley, MA*

Editor: Joaquin Vanschoren

Abstract

Hallucinations, defined as instances where Large Language Models (LLMs) generate false or misleading content, pose a significant challenge that impacts the safety and trust of downstream applications. We introduce `uqlm`, a Python package for LLM hallucination detection using state-of-the-art uncertainty quantification (UQ) techniques. This toolkit offers a suite of UQ-based scorers that compute response-level confidence scores ranging from 0 to 1. This library provides an off-the-shelf solution for UQ-based hallucination detection that can be easily integrated to enhance the reliability of LLM outputs.

Keywords: large language model, uncertainty quantification, hallucination detection, Python, AI safety

1. Introduction

Large language models (LLMs) have revolutionized the field of natural language processing, but their tendency to generate false or misleading content, known as hallucinations, significantly compromises safety and trust. LLM hallucinations are especially problematic because they often appear plausible, making them difficult to detect and posing serious risks in high-stakes domains such as healthcare, legal, and financial applications. As LLMs are increasingly deployed in real-world settings, monitoring and detecting hallucinations becomes crucial.

Traditional evaluation methods involve 'grading' LLM responses by comparing model output to human-authored ground-truth texts, an approach offered by toolkits such as Evals (OpenAI, 2024) and G-Eval (Liu et al., 2023). While effective during pre-deployment testing, these methods are limited in practice since users typically lack access to ground-truth data at generation time. This shortcoming motivates the need for generation-time hallucination detection methods.

Existing solutions to this problem include source-comparison methods, internet-based grounding, and uncertainty quantification (UQ) methods. Toolkits that offer source-comparison scorers, such as Ragas (Es et al., 2023), Phoenix (Arize AI, 2025), DeepEval (Ip and Vongthongsri, 2025), and others (Hu et al., 2024; UpTrain AI Team, 2024; Zha et al., 2023; Asai et al., 2023) evaluate the consistency between generated content and input prompts. However, these methods can mistakenly validate responses that merely mimic prompt phrasing without ensuring factual accuracy. Toolkits that leverage Internet searches for fact-checking, such as FacTool (Chern et al., 2023), introduce delays and risk incorporating erroneous online information, failing to address the inherent uncertainty in model outputs. Lastly, although numerous UQ techniques have been proposed in the literature, their adoption in user-friendly, comprehensive toolkits remains limited. For example, while SelfCheckGPT (Manakul et al., 2023) incorporates some UQ scorers, its set of techniques is narrow and does not integrate generation with evaluation, thus creating barriers for practitioners outside specialized AI research environments. LangKit (WhyLabs, 2025) and NeMo Guardrails (Rebedea et al., 2023) also offer UQ scorers but are similarly narrow in scope. LM-Polygraph (Fadeeva et al., 2023) provides a valuable collection of UQ-based approaches for LLMs and is built on the Hugging Face ecosystem, representing an important step toward more accessible UQ tools, though opportunities remain for enhancing accessibility for non-specialized practitioners.

We aim to bridge these gaps by introducing a comprehensive open-source Python package, `uqlm`, that democratizes advanced research in LLM uncertainty quantification. UQLM (Uncertainty Quantification for Language Models) implements a diverse array of uncertainty estimation techniques to compute generation-time, response-level confidence scores and uniquely integrates generation and evaluation processes. This integrated approach allows users to generate and assess content simultaneously, without the need for ground-truth data or external knowledge sources, and with minimal engineering effort. This democratization of access empowers smaller teams, researchers, and developers to incorporate robust hallucination detection into their applications for safer and more reliable AI systems.

2. Usage

The `uqlm` library, available at `https://github.com/cvs-health/uqlm`, provides a collection of UQ-based scorers spanning four categories: black-box UQ, white-box UQ, LLM-as-a-Judge, and ensembles.[1] The corresponding classes for these techniques are instantiated by passing a LangChain `BaseChatModel` (LLM) instance to the constructor.[2] Each of these classes contains a `generate_and_score` method, which generates LLM responses to a user provided list of prompts and computes response-level confidence scores, which range from 0 to 1.

2.1 Black-Box Uncertainty Quantification

Black-box uncertainty quantification exploits the stochastic nature of LLMs and measures the consistency of multiple responses to the same prompt. These consistency measurements can be conducted with various approaches, including discrete semantic entropy (Farquhar et al., 2024), number of semantic sets (Lin et al., 2024), non-contradiction probability (Chen and Mueller, 2024), entailment probability (Lin et al., 2024), BERTScore (Manakul et al., 2023), exact match rate (Cole et al., 2023), and cosine similarity (Shorinwa et al., 2024). Black-box UQ scorers are compatible with any LLM, but increase latency and generation

[1]For a detailed overview of available scorers and associated experiment results, we refer the reader to this project's companion paper, Bouchard and Chauhan (2025).

[2]UQLM is compatible with all LangChain Chat Models. More details are provided in Appendix B.

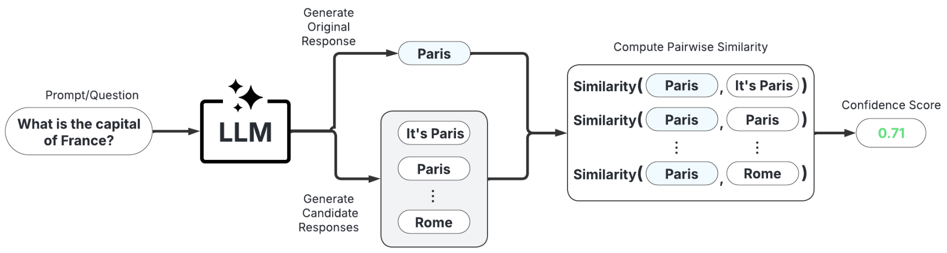
Prompt/Question
What is the capital of France?
LLM
Generate Original Response
Paris
Generate Candidate Responses
It's Paris
Paris
Rome
Compute Pairwise Similarity
Similarity(Paris , It's Paris)
Similarity(Paris , Paris)
Similarity(Paris , Rome)
Confidence Score
0.71

Figure 1: Illustration of a Black-Box Scorer Workflow

costs. The `BlackBoxUQ` class uses the user-provided LLM to generate, for each prompt, an original response and additional candidate responses, then computes consistency scores using the specified scorers (see Figure 1).[3] See below for a minimal `BlackBoxUQ` example.[4]

```python
from uqlm import BlackBoxUQ
bbuq = BlackBoxUQ(llm=llm, scorers=["exact_match", "noncontradiction"])
results = await bbuq.generate_and_score(prompts=prompts, num_responses=5, use_best=True)
```

2.2 White-Box Uncertainty Quantification

White-box uncertainty quantification leverages token probabilities to compute uncertainty, as in Figure 2. UQLM's `WhiteBoxUQ` class includes several categories of methods. Single-generation methods, which offer the advantage of no additional latency or generation costs, include minimum token probability (Manakul et al., 2023), length-normalized token probability (Malinin and Gales, 2021), sequence probability (Vashurin et al., 2025a), likelihood margin (Farr et al., 2024), mean top-K token entropy (Scalena et al., 2025), and maximum top-K token entropy (Scalena et al., 2025). Sampling-based methods include semantic entropy (Farquhar et al., 2024), semantic density (Qiu and Miikkulainen, 2024), Monte Carlo predictive entropy (Kuhn et al., 2023), and CoCoA (Vashurin et al., 2025b). Additionally, the class implements the $P(\text{True})$ method (Kadavath et al., 2022), which requires one extra generation per response. Note that WhiteBox UQ will only work with model APIs that expose token probabilities. See below for example `WhiteBoxUQ` usage.

```python
from uqlm import WhiteBoxUQ
wbuq = WhiteBoxUQ(llm=llm, scorers=["min_probability"])
results = await wbuq.generate_and_score(prompts=prompts)
```

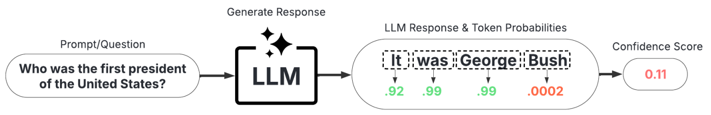

Figure 2: Illustration of a Single-Generation White-Box Scorer Workflow

[3]Users may skip generation by calling the `score` method with pre-generated responses. The workflow depicted in Figure 1 does not apply to semantic entropy, which does not designate an 'original response'.

[4]`use_best=True` selects the mode from the highest-probability semantic cluster (Farquhar et al., 2024).

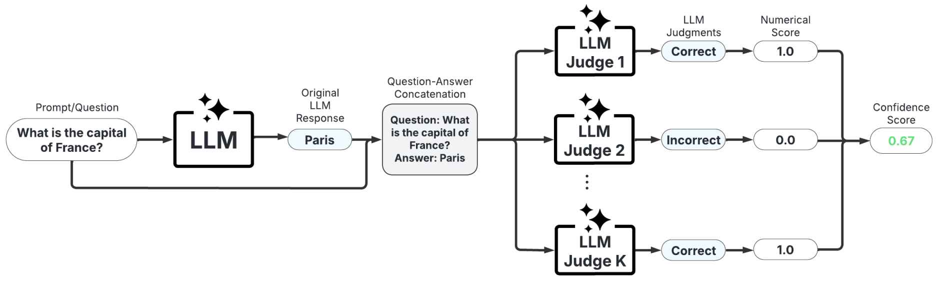

Figure 3: Illustration of LLM-as-a-Judge Workflow

2.3 LLM-as-a-Judge

LLM-as-a-Judge uses an LLM to evaluate the correctness of a response to a particular question. To achieve this, a question-response concatenation is passed to one or more LLMs along with instructions to score the response's correctness using the `LLMPanel` class (see Figure 3). In the constructor, users pass a list of LLM objects to the `judges` argument and specify one of four scoring templates for each judge with the `scoring_templates` argument. These four scoring templates are as follows: binary ($\{\text{incorrect}, \text{correct}\}$ as $\{0, 1\}$), ternary ($\{\text{incorrect}, \text{uncertain}, \text{correct}\}$ as $\{0, 0.5, 1\}$), continuous (any value between 0 and 1), and a 5-point Likert scale $(0, 0.25, ..., 1)$. Instructions are customized using the `additional_context` argument. Implementing the `generate_and_score` method (minimal example below) returns the score from each judge along with aggregations such as average, minimum, and median.

```
from uqlm import LLMPanel
panel = LLMPanel(llm=llm1, judges=[llm2, llm3], scoring_templates=["continuous", "likert"])
results = await panel.generate_and_score(prompts=prompts)
```

2.4 Ensemble Approach

Lastly, `uqlm` offers both tunable and off-the-shelf ensembles that leverage a weighted average of any combination of black-box UQ, white-box UQ, and LLM-as-a-Judge scorers. Similar to the aforementioned classes, `UQEnsemble` enables simultaneous generation and scoring with a `generate_and_score` method. Using the specified scorers, the ensemble score is computed as a weighted average of the individual confidence scores, where weights may be default weights, user-specified, or tuned (refer to Appendix A for tuning details). If no scorers are specified, the off-the-shelf implementation follows an ensemble of exact match, non-contradiction probability, and self-judge proposed by Chen and Mueller (2024).

3. Conclusions

This paper introduced `uqlm`, an open-source Python toolkit offering a collection of state-of-the-art uncertainty quantification techniques for LLM hallucination detection. We believe `uqlm` democratizes uncertainty quantification techniques from the literature, empowering practitioners to effectively detect hallucinations at generation time with minimal engineering effort. To get started with `uqlm`, we refer readers to the Github repository and Documentation site.

Author Contributions

Dylan Bouchard was the principal developer and researcher of the UQLM project, responsible for conceptualization, methodology, and software development of the UQLM library. Mohit Singh Chauhan helped lead research and software development efforts. David Skarbrevik, Ho-Kyeong Ra, Viren Bajaj, and Zeya Ahmad contributed to software development.

Conflict of Interest

The authors are employed and receive stock and equity from CVS Health® Corporation.

Acknowledgments

We wish to thank Piero Ferrante, Blake Aber, Matthew Churgin, Erik Widman, Robert Enzmann, Xue (Crystal) Gu and Huiwen Hu for their helpful suggestions as well as Dimitris Tsapetis, Vipin Gyanchandani, Drew Ross, Jinesh Mehta, Namrata Walanj, and Joshua Mabry for their contributions to UQLM.

Appendix A. Ensemble Tuning

In order to tune the ensemble weights prior to using the `generate_and_score` method, users must provide a list of prompts and corresponding ideal responses to serve as an ‘answer key’. The LLM’s responses to the prompts are graded with a grader function that compares against the provided ideal responses. If a grader function is not provided by the user, the original LLM is used as an LLM-based grader to evaluate the correctness of the responses relative to the answer key.

Once the binary grades (‘correct’ or ‘incorrect’) are obtained, an optimization routine solves for the optimal weights according to a specified classification objective. The objective function may be threshold-agnostic, such as ROC-AUC, or threshold-dependent, such as F1-score. After completing the optimization routine, the optimized weights are stored as class attributes to be used for subsequent scoring. Below is a minimal example illustrating this process.

```python
from uqlm import UQEnsemble
## ---Option 1: Off-the-Shelf Ensemble (Chen & Mueller, 2023)---
# uqe = UQEnsemble(llm=llm)
# results = await uqe.generate_and_score(prompts=prompts, num_responses=5)

## ---Option 2: Tuned Ensemble---
scorers = [ # specify which scorers to include
    "exact_match", "noncontradiction", # black-box scorers
    "min_probability", # white-box scorer
    llm # use same LLM as a judge
]
uqe = UQEnsemble(llm=llm, scorers=scorers)

# Tune on tuning prompts with provided ground truth answers
tune_results = await uqe.tune(
    prompts=tuning_prompts, ground_truth_answers=ground_truth_answers
)
# ensemble is now tuned - generate responses on new prompts
results = await uqe.generate_and_score(prompts=prompts)
results.to_df()
```

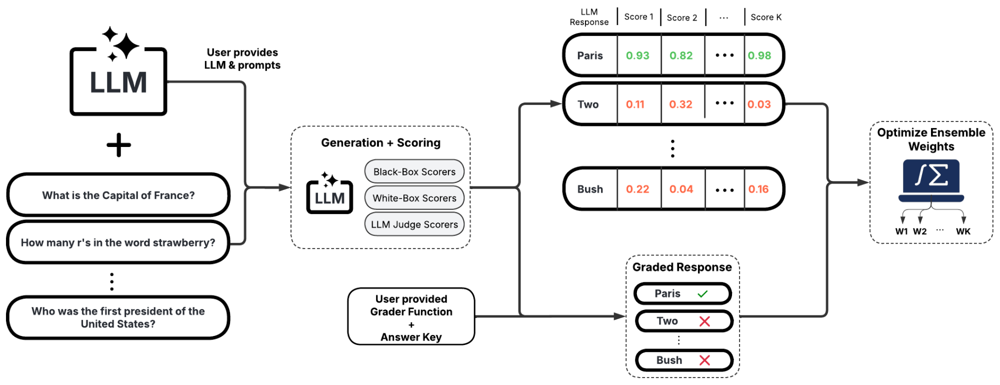

Figure 4: Illustration of Ensemble Tuning

Appendix B. Compatible Models

UQLM integrates with the LangChain Chat Model interface, making it compatible with a large and evolving ecosystem of providers and models; an exhaustive list is infeasible. In practice, users access models through LangChain wrappers for commercial API providers such as OpenAI, Anthropic, Google, Cohere, and AWS Bedrock, as well as for locally deployed open-source models like Llama 3, Mistral, and Qwen using tools such as Ollama. For an up-to-date catalog and provider-specific options, we refer readers to the LangChain documentation for supported chat models.[5] Note that white-box uncertainty methods require access to token log probabilities, which some wrappers do not expose. LangChain documents how to request these log probabilities and how to verify whether a specific wrapper returns them.[6] Table B provides detailed compatibility information for each scorer family.

Scorer family	**Needs logprobs**	**Compatibility**
Black-Box UQ	No	Compatible with all chat models
White-Box UQ	Yes	Chat models that expose logprobs
LLM-as-a-Judge	No	Compatible with all chat models
UQ Ensemble	Depends	Depends on included scorers

Table 1: UQLM method families and interface requirements.

[5] `https://python.langchain.com/docs/integrations/chat/`

[6] `https://python.langchain.com/docs/how_to/logprobs/`

References

Arize AI. Phoenix, 2025. URL `https://github.com/Arize-ai/phoenix`.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL `https://arxiv.org/abs/2310.11511`.

Dylan Bouchard and Mohit Singh Chauhan. Uncertainty quantification for language models: A suite of black-box, white-box, LLM judge, and ensemble scorers. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=WOFspd4lq5`.

Jiuhai Chen and Jonas Mueller. Quantifying uncertainty in answers from any language model and enhancing their trustworthiness. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5186–5200, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.283. URL `https://aclanthology.org/2024.acl-long.283/`.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*, 2023. doi: 10.48550/arXiv.2307.13528.

Jeremy Cole, Michael Zhang, Daniel Gillick, Julian Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. Selectively answering ambiguous questions. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 530–543, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.35. URL `https://aclanthology.org/2023.emnlp-main.35/`.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation, 2023. URL `https://arxiv.org/abs/2309.15217`.

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. LM-polygraph: Uncertainty estimation for language models. In Yansong Feng and Els Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 446–461, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.41. URL `https://aclanthology.org/2023.emnlp-demo.41`.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, Jun 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07421-0. URL `https://doi.org/10.1038/s41586-024-07421-0`.

David Farr, Nico Manzonelli, Iain Cruickshank, and Jevin West. Red-ct: A systems design methodology for using llm-labeled data to train and deploy edge classifiers for computational social science, 2024. URL https://arxiv.org/abs/2408.08217.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. Refchecker: Reference-based fine-grained hallucination checker and benchmark for large language models, 2024. URL https://arxiv.org/abs/2405.14486.

Jeffrey Ip and Kritin Vongthongsri. deepeval, March 2025. URL https://github.com/confident-ai/deepeval.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. URL https://arxiv.org/abs/2302.09664.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models, 2024. URL https://arxiv.org/abs/2305.19187.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction, 2021. URL https://arxiv.org/abs/2002.07650.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

OpenAI. Evals, 2024. URL https://github.com/openai/evals.

Xin Qiu and Risto Miikkulainen. Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space, 2024. URL `https://arxiv.org/abs/2405.13845`.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In Yansong Feng and Els Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.40. URL `https://aclanthology.org/2023.emnlp-demo.40`.

Daniel Scalena, Leonidas Zotos, Elisabetta Fersini, Malvina Nissim, and Ahmet Üstün. Eager: Entropy-aware generation for adaptive inference-time scaling, 2025. URL `https://arxiv.org/abs/2510.11170`.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions, 2024. URL `https://arxiv.org/abs/2412.05563`.

UpTrain AI Team. UpTrain, 2024. URL `https://github.com/uptrain-ai/uptrain`.

Roman Vashurin, Ekaterina Fadeeva, Artem Vazhentsev, Lyudmila Rvanova, Daniil Vasilev, Akim Tsvigun, Sergey Petrakov, Rui Xing, Abdelrahman Sadallah, Kirill Grishchenkov, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, Maxim Panov, and Artem Shelmanov. Benchmarking uncertainty quantification methods for large language models with lm-polygraph. *Transactions of the Association for Computational Linguistics*, 13:220–248, 2025a. ISSN 2307-387X. doi: 10.1162/tacl_a_00737. URL `http://dx.doi.org/10.1162/tacl_a_00737`.

Roman Vashurin, Maiya Goloburda, Albina Ilina, Aleksandr Rubashevskii, Preslav Nakov, Artem Shelmanov, and Maxim Panov. Uncertainty quantification for llms through minimum bayes risk: Bridging confidence and consistency, 2025b. URL `https://arxiv.org/abs/2502.04964`.

WhyLabs. langkit, 2025. URL `https://github.com/whylabs/langkit`.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. AlignScore: Evaluating factual consistency with a unified alignment function. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.634. URL `https://aclanthology.org/2023.acl-long.634/`.