LUQ: Language Models Uncertainty Quantification Toolkit

Alexander Nikitin [1] **Martin Trapp** [1] **Pekka Marttinen** [1]

Abstract

Uncertainty quantification is a principled approach to ensuring the robustness, reliability, and safety of large language models (LLMs). However, progress in this field is hindered by the lack of a unified framework for benchmarking. Additionally, creating suitable datasets for uncertainty quantification is computationally demanding because it often requires sampling LLMs multiple times per each sample. In this work, we propose and describe a software framework that (i) unifies the benchmarking of uncertainty quantification methods for language models, and (ii) provides an easy-to-use tool for practitioners developing more robust and safer LLM applications.

1. Introduction

Large language models (LLMs) have rapidly become the primary tool in many real-world scenarios, such as in clinical applications (Thirunavukarasu et al., 2023), to extract key points from legal documents (Lai et al., 2024), or as research assistants (Yamada et al., 2025). Moreover, they are increasingly employed as autonomous or semi-autonomous agents (Yao et al., 2023; Schick et al., 2023). Projections indicate that LLMs are expected to become even more capable, assuming the correctness of existing scaling laws (Kaplan et al., 2020; Wu et al., 2024).

Given the increasing adoption of LLMs in real-world applications, including high-stakes applications, it is crucial to assess and ensure their reliability. However, as of today, LLM applications often suffer from hallucinations (Ji et al., 2023), meaning they "generate content that is nonsensical or unfaithful to the provided source content" (Ji et al., 2023). Moreover, Xu et al. (2024) argue that hallucinations are inevitable problem for LLMs. Thus, LLM hallucinations threaten reliability in safety-critical tasks, such as clinical documentation (Asgari et al., 2025) or data-to-text tasks, where the tolerance towards unfaithful content generation is very low or privacy concerns exist (Carlini et al., 2021).

A promising avenue to mitigating hallucinations is quantifying the uncertainty of the model. High predictive uncertainty has been shown to be associated with incorrect responses (Kadavath et al., 2022). Thus, knowing the model's uncertainty can help detect and prevent hallucinations and allows LLM providers to decide when to return the generated response and whether to forward a question to a human expert for detailed assessment. Consequently, several techniques for uncertainty quantification (UQ) in LLMs have been proposed, including measuring the maximum predicted probability (Hendrycks & Gimpel, 2017), estimating the predictive entropy (Malinin & Gales, 2020), approximating the semantic uncertainty of the model (Farquhar et al., 2024; Nikitin et al., 2024b), or use the Bayesian framework (Yang et al., 2024; Li et al., 2025). Moreover, uncertainty quantification has also recently been investigated in instruction-following tasks (Heo et al., 2025).

However, despite the importance of mitigating hallucinations, software tooling that supports the development and benchmarking of mitigation techniques is still in its infancy. Consequently, many research works re-implement previous methods, but do not provide a coherent toolkit. Moreover, implementing and testing probabilistic tools for uncertainty quantification can be particularly challenging due to their inherent stochasticity. The current lack of software not only hinders the use of existing uncertainty quantification methods in LLM applications but also impedes the development and advancement of novel techniques.

Contribution. We propose the *Language Models Uncertainty Quantification Toolkit* (`LUQ`) for uncertainty quantification experimentation and benchmarking in LLMs. Our software framework aims to democratise and advance research on uncertainty quantification in LLMs by providing: (i) implementations of popular uncertainty quantification methods, (ii) readily available datasets for uncertainty quantification experimentation, and (iii) evaluation metrics for benchmarking the quality of the uncertainty estimates.

The LUQ toolkit available on GitHub: `https://github.com/alexandervnikitin/luq`

[1]Department of Computer Science, Aalto University, Espoo, Finland. Correspondence to: Alexander Nikitin <alexander.nikitin@aalto.fi>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

with documentation available under: https://alexandervnikitin.github.io/luq.

2. LUQ: Language Models Uncertainty Quantification Toolkit

Framework overview. **LUQ** is a comprehensive toolkit that simplifies every stage of uncertainty estimation for LLM responses. It provides a unified API for generating samples, curated high-quality datasets for LLM experimentation, user-friendly uncertainty quantification (UQ) methods, and a robust evaluation framework for comparing different UQ techniques. Additionally, **LUQ** includes educational tutorials to help LLM developers better understand and apply UQ methods. The following sections explore each of these components in more detail.

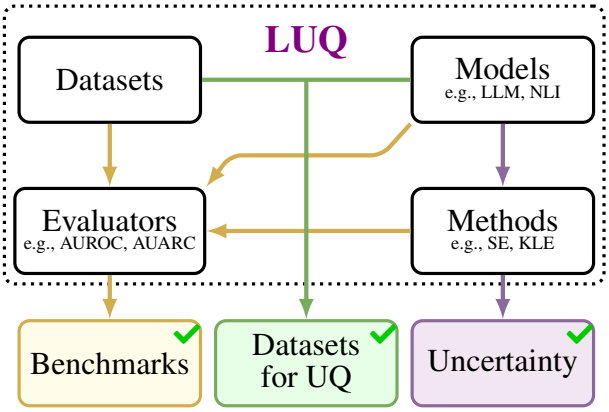

Figure 1: **LUQ** delivers three *key outputs*: datasets for uncertainty quantification, benchmark evaluation procedures, and tools for measuring uncertainty. These outputs are built using **LUQ**'s *main components*: models (wrappers around LLMs, natural language inference models, and others), evaluators (modules that assess UQ performance), and methods (the core of **LUQ**, various techniques for quantifying uncertainty in LLMs). See the corresponding paragraphs below for details on each building block.

Methods `luq.methods`. **LUQ** incorporates a variety of methods for uncertainty quantification (UQ) in LLMs, including maximum predicted probability (as in Hendrycks & Gimpel (2017)), top-k probability gaps, predictive entropy (PE)(Malinin & Gales, 2020), P(True) (Kadavath et al., 2022), semantic entropy (SE) (Farquhar et al., 2024), and kernel language entropy (KLE) (Nikitin et al., 2024b). These methods generally involve sampling multiple responses from an LLM and assessing the model's uncertainty based on these samples. Some techniques, such as SE and KLE, analyse the semantic similarity between sampled responses, while others, like PE, focus on the distribution of predicted probabilities across the samples.

Let the set $\mathcal{T}$ be a vocabulary of tokens, $S \in \mathcal{T}^N$ be a sequence of length N, consisting of tokens, $s_i \in \mathcal{T}$. Then the probability of a sentence for an input x is given as

$$p(S = s \mid x) = \prod_i p(s_i | s_{<i}, x), \tag{1}$$

where $s_{<i}$ denotes all tokens preceding s_i.

`luq.models.max_probability`. This estimator computes the maximum probability of the outputs as

$$\mathrm{MP}(x) = 1 - \max_s p(s \mid x), \tag{2}$$

where the maximum is taken over all possible output sequences s. **LUQ** approximates this value by considering only a finite set of sampled sequences.

`luq.models.top_k_gap`. Analogous to MP, the Gap_k estimator analyses the probability distribution over output sequences. It quantifies uncertainty by measuring the difference between the highest probability and the k-th highest probability, defined as

$$\mathrm{Gap}_k(x) = 1 - (\max_{s'} p(s' \mid x) - \max_s{}^k p(s \mid x)), \tag{3}$$

where $\max^k$ denotes the k-th largest value of $p(s \mid x)$ over all sequences s.

`luq.model.predictive_entropy`. The predictive entropy for an input x and a random output sequence S is defined as

$$U(x) = H(S \mid x) = -\sum_s p(s \mid x) \log p(s \mid x), \tag{4}$$

where the summation is over all possible output sequences s. As in previous cases, this score is approximated using a finite number of sampled outputs.

`luq.model.p_true`. Multiple responses are generated by an LLM using a high temperature setting to encourage diversity. Separately, a single response is generated at a low temperature, which is treated as the final answer. The LLM is then prompted to assess the correctness of this final answer. The model's uncertainty is quantified based on the token probability distributions associated with the generated response.

`luq.model.semantic_entropy`. For an input x and a set of semantic clusters $C_j \in \Omega$, where each semantic cluster is a set of semantically equivalent texts, Semantic Entropy (SE) is defined as

$$\mathrm{SE}(x) \approx -\sum_{j=1}^{M} p'(C_j \mid x) \log p'(C_j | x), \tag{5}$$

where C_j is one of M clusters extracted from the generations and $p'(C_j \mid x)$ is a normalized semantic probability, i.e., $p'(C_j \mid x) = {}^{p(C_j \mid x)}/_{\sum_i p(C_i \mid x)}$.

`luq.model.kernel_language_entropy`. One limitation of SE is that it relies on coarse semantic clusters and ignores the semantic distances between individual samples. KLE generalises SE by replacing the entropy calculation over semantic clusters with a semantic kernel—that is, a positive semi-definite matrix that captures the semantic similarities between generated samples. It then uses the von Neumann Entropy of this kernel to measure uncertainty.

Models `luq.models`. **LUQ** provides wrappers for LLMs and NLI models to standardise the APIs across different providers. For example, `HFLLMWrapper` can be used for integration with HuggingFace (Wolf et al., 2019) or `ClaudeWrapper` for integration with Claude[1].

Evaluators `luq.evals`. Uncertainty quantification methods are commonly assessed based on how well they predict the correctness of a model's outputs. Following the approach of Farquhar et al. (2024) and Nikitin et al. (2024b), for each sample in the NLG datasets, we generate multiple responses by an LLM and a final low-temperature response. The correctness of the final response is evaluated by another language model acting as a judge (Gu et al., 2024). We also plan to incorporate additional methods to enhance the accuracy assessment. We use two standard evaluation metrics: the Area Under the Receiver Operating Characteristic Curve (AUROC) and the Area Under the Accuracy-Rejection Curve (AUARC) (Nadeem et al., 2009). **LUQ** implements these metrics along with user-friendly APIs for ease of use. A challenge in applying UQ methods to LLMs for mitigating hallucinations is ensuring that the model is calibrated for the specific task. The `luq.evals.CalibrationEval` tool helps evaluate an LLM's calibration on a given dataset by computing the expected calibration error and visualising calibration curves.

Datasets `luq.datasets`. A key challenge in evaluating the above methods on natural language generation (NLG) benchmarks is the creation of high-quality datasets. This process is resource-intensive: for each example in an existing NLG dataset and for each LLM under evaluation, multiple responses must be generated. Additionally, the final responses for each initial sample must be manually assessed for correctness to compute evaluation metrics such as AUROC and AUARC, as described earlier. To address this challenge, we provide pre-generated datasets for several popular LLMs, helping to democratise and streamline research on uncertainty estimation in language models. **LUQ** has connectors for CoQA (Reddy et al., 2019), BioASQ (Krithara et al., 2023), and NQ (Kwiatkowski et al., 2019) datasets, and provides a step-by-step tutorial on adding new datasets.

[1] `https://claude.ai/`

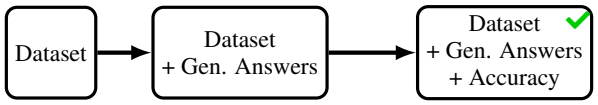

Figure 2: Dataset for UQ Preparation. Step 1: Augment the initial NLG dataset with outputs generated by various large language models (LLMs). Step 2: Evaluate each sample in the augmented dataset using an LLM-based judge to assess the accuracy of the generated content.

```
2  import torch
3  import luq
4  from luq.methods import *
5  from luq.models import HFLLMWrapper
6  from transformers import AutoTokenizer,
       AutoModelForCausalLM

8  model_id = "gpt2"
9  tokenizer = AutoTokenizer.from_pretrained(
       model_id)
10 model = AutoModelForCausalLM.
       from_pretrained(
       model_id, torch_dtype=torch.float32)

11 llm = HFLLMWrapper(tokenizer=tokenizer,
       model=model)
12 samples = luq.models.
       generate_n_samples_and_answer(
       llm,
       prompt="A, B, C, or D"
   )

13 mp_estimator = MaxProbabilityEstimator()
14 print(mp_estimator.estimate_uncertainty(
       samples))
```

Listing 1: **LUQ** code example: evaluating uncertainty of a model on synthetic prompt with MP.

Code Examples **LUQ** provides a concise API and is easy to integrate with large LLM providers, such as HuggingFace, and APIs such as OpenAI. We provide an example of source code in Listing 1, which we explain in detail below.

In Listing 1, we use HuggingFace's `transformers` library to instantiate a GPT-2 (Radford et al., 2019) model and wrap it with **LUQ**'s HuggingFace wrapper `HFLLMWrapper` (lines 8 – 11). Once instantiated, we can use **LUQ**'s HuggingFace wrapper to generate a dataset, i.e., multiple samples from the model for a given prompt (line 12). Finally, we use the `MaxProbabilityEstimator` to compute the maximum probability of the outputs, c.f., Equation (2).

Software development practices. **LUQ** follows modern software development practices. It is implemented in

Python 3 and supports all versions from 3.10 onward. The framework uses PyTorch (Paszke, 2019) as its core deep learning library, and supports integration with HuggingFace (Wolf et al., 2019), the OpenAI API[2], and the Claude API[3]. Additionally, it provides a generic wrapper for compatibility with custom LLM implementations.

The framework includes comprehensive unit tests, and all changes to the main branch are automatically checked with a linter[4] and unit tests. Code coverage statistics are measured for each pull request using codecov[5] displayed on the project's front page, and for each pull request. Pre-commit hooks ensure that tests pass and PEP8 compliance is maintained across the entire codebase history. The project follows semantic versioning for all releases. The documentation is deployed on a separate web page ANONYMIZED. The package can be installed from PyPI[6] or from the sources.

Tutorials. **LUQ** offers a series of tutorials on best practices for using **LUQ**, with a particular focus on uncertainty quantification in LLMs. We plan to expand these tutorials to include new methods and use cases.

3. Related Work

Open source software for ML. Open-source software (OSS) has been instrumental in the recent advancements of machine learning and deep learning. Frameworks such as PyTorch (Paszke, 2019), TensorFlow (Abadi et al., 2016), and Jax (Bradbury et al., 2018) have significantly simplified the development of complex neural architectures while enabling efficient use of hardware accelerators. In parallel, numerous tools have been developed to improve the safety and reliability of machine learning systems, including TrueLens (Datta et al., 2022), TSGM (Nikitin et al., 2024a), ART (Nicolae et al., 2018), Foolbox (Rauber et al., 2017), and Torch-Uncertainty. Building on the best practices from large amount of OSS for ML, our work advances the field of trustworthy AI by focusing on uncertainty quantification for LLMs. The concurrent UQLM (Bouchard & Chauhan, 2025) library focuses on hallucination detection in LLMs. **LUQ** not only focuses on methods for uncertainty quantification and hallucination detection, but also addresses datasets and evaluation metrics, providing a framework to advance research on uncertainty quantification in LLMs.

Safety and reliability of LLMs. Safety and reliability of LLMs are crucial for the successful adoption of LLMs in practical scenarios (Achiam et al., 2023). A key challenge in the field of LLM safety is preventing hallucinations (Ji et al., 2023). Various approaches have been proposed to address this issue, including uncertainty quantification (Farquhar et al., 2024), training-time techniques (Kang et al., 2024), and the use of external information for fact-checking (Lewis et al., 2020). Uncertainty quantification methods that help to mitigate hallucinations in LLMs include P(True) (Kadavath et al., 2022), spectral graph metrics (Lin et al., 2023), semantic entropy (SE) (Farquhar et al., 2024), and kernel language entropy (KLE) (Nikitin et al., 2024b). **LUQ** progresses towards more reliable LLMs by implementing uncertainty quantification techniques and providing a coherent framework for uncertainty quantification and benchmarking of uncertainty quantification techniques.

[2] https://platform.openai.com
[3] https://docs.anthropic.com
[4] https://flake8.pycqa.org/en/latest/
[5] https://codecov.io
[6] https://pypi.org/

4. Conclusion

Ensuring the safety and reliability of LLMs is essential for their successful adoption in real-world applications. A key component to achieving reliability in LLMs is minimising hallucinations—or alternatively, clearly communicating the model's uncertainty in its responses to users.

LUQ addresses this challenge by democratizing uncertainty quantification (UQ) methods for developers building LLM-based applications. It offers user-friendly abstractions that support both research and development of new UQ techniques. The framework includes datasets, model wrappers covering a wide range of LLMs, evaluation metrics, and a broad collection of UQ methods. In the future, we plan to enhance the framework by adding benchmarks of current methods and providing seamless access to UQ datasets and evaluation metrics.

LUQ is rigorously tested and follows software engineering best practices to ensure robustness and usability.

Broader Impact. As LLMs are increasingly adopted in real-world settings, effectively detecting and reducing hallucinations in LLMs is key to improving the trustworthiness of modern machine learning applications. An important step to achieve this is providing a concise framework to quantify uncertainties and benchmark technologies, thus democratising uncertainty quantification research on LLMs. Thus, we anticipate that our work will positively impact fields applying LLMs in real-world settings while also supporting LLM researchers in developing new techniques by providing an open-source framework for experimentation and benchmarking of uncertainty quantification in LLMs.

Limitations. While uncertainty quantification enhances the robustness of LLMs, it does not offer an absolute measure of reliability or guarantee the correctness or incorrectness of a given answer.

Acknowledgements

We acknowledge the computational resources provided by the Aalto Science-IT project. MT acknowledges funding from the Research Council of Finland (grant number 347279).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. TensorFlow: a system for Large-Scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Asgari, E., Montaña-Brown, N., Dubois, M., Khalil, S., Balloch, J., Yeung, J. A., and Pimenta, D. A framework to assess clinical safety and hallucination rates of LLMs for medical text summarisation. *npj Digital Medicine*, 8 (1):1–15, 2025.

Bouchard, D. and Chauhan, M. S. Uncertainty quantification for language models: A suite of black-box, white-box, llm judge, and ensemble scorers. *arXiv preprint arXiv:2504.19254*, 2025.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security Symposium*, pp. 2633–2650. USENIX Association, 2021.

Datta, A., Fredrikson, M., Leino, K., Lu, K., Sen, S., Shih, R., and Wang, Z. Exploring conceptual soundness with trulens. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 302–307. PMLR, 2022.

Farquhar, S., Kossen, J., Kuhn, L., and Gal, Y. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Heo, J., Xiong, M., Heinze-Deml, C., and Narain, J. Do LLMs estimate uncertainty well in instruction-following? In *International Conference on Learning Representations (ICLR)*, 2025.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

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

Kang, K., Wallace, E., Tomlin, C., Kumar, A., and Levine, S. Unfamiliar finetuning examples control how language models hallucinate. *arXiv preprint arXiv:2403.05612*, 2024.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Krithara, A., Nentidis, A., Bougiatiotis, K., and Paliouras, G. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170, 2023.

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

Lai, J., Gan, W., Wu, J., Qi, Z., and Yu, P. S. Large language models in law: A survey. *AI Open*, 2024.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, R., Klasson, M., Solin, A., and Trapp, M. Streamlining prediction in Bayesian deep learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Lin, Z., Trivedi, S., and Sun, J. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*, 2023.

Malinin, A. and Gales, M. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*, 2020.

Nadeem, M. S. A., Zucker, J.-D., and Hanczar, B. Accuracy-rejection curves (arcs) for comparing classification methods with a reject option. In *Machine Learning in Systems Biology*, pp. 65–81. PMLR, 2009.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

Nikitin, A., Iannucci, L., and Kaski, S. Tsgm: A flexible framework for generative modeling of synthetic time series. *Advances in Neural Information Processing Systems*, 37:129042–129061, 2024a.

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

Paszke, A. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

Reddy, S., Chen, D., and Manning, C. D. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., and Ting, D. S. W. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Xu, Z., Jain, S., and Kankanhalli, M. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*, 2024.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Yang, A. X., Robeyns, M., Wang, X., and Aitchison, L. Bayesian low-rank adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.