Member-only story

Evaluating LLMs: Beyond Simple Metrics

Ritesh Gupta · Follow
4 min read · Oct 26, 2024

From Metrics to Meaning: Effective Strategies for Large Language Model Evaluation

Evaluating machine learning models can often be straightforward: for structured tasks, metrics like AUC (for classification) and MSE (for regression) provide a clear performance view. But evaluating large language models (LLMs) is an entirely different game. For tasks like flagging spam, evaluation is simpler; either the email is spam or it isn't. However, LLMs generate complex, open-ended responses, making evaluation challenging due to:

☒ The high cost of manual reviews
☒ The open-ended nature of input and output
☒ Generic benchmarks like MMLU falling short for custom, specific cases

Photo by ThisIsEngineering: https://www.pexels.com/photo/code-projected-over-woman-3861969/

So, how do you evaluate LLMs effectively and at scale? Here are three proven strategies to help:

1. Semantic Similarity 📏

For LLM tasks where the model should respond in line with a desired meaning, comparing the semantic similarity of the generated output with an ideal output is effective. By representing both the model-generated and the ideal responses as embedding vectors, we can use **cosine similarity** to calculate how close they are in meaning. The higher the cosine similarity, the better the response!

Example:

For a customer support LLM trained to assist users with refund policies, you can create a sample ideal answer, embed both this answer and the model's response, and compute the cosine similarity between the vectors. If the LLM response scores over 0.85 in cosine similarity, it's considered "very good."

Why it works 👍:

- **Objective measurement of meaning** across complex responses
- Allows rapid, automated scoring based on meaningfulness rather than exact match

> **Pro Tip:** Tools like Sentence-BERT or OpenAI's embeddings API are highly effective for semantic similarity calculations.

2. LLM-as-a-Judge 🧑⚖️

Hiring humans to evaluate every response is costly and time-consuming. Instead, consider creating a secondary LLM model to act as a **reviewer agent**. This "LLM-as-a-Judge" approach involves programming an agent to assess the quality of main LLM responses based on a predefined grading rubric. The grading rubric could cover aspects like relevance, coherence, completeness, and tone.

Example:

For an LLM tasked with providing medical information, create an LLM reviewer that scores the output based on criteria like accuracy, empathy, and clarity. By using carefully crafted prompts and scoring rubrics, this reviewer can grade responses on a scale from 1 to 10 and return a detailed evaluation of each response's strengths and weaknesses.

Why it works 👍:

- **Automates evaluation** of nuanced qualities like coherence and helpfulness
- Saves time and cost by reducing the need for human reviewers
- Scales efficiently to large datasets

> **Pro Tip:** Prompt your reviewer agent with specific examples to make it as close to a human evaluator as possible. Make sure it understands the grading rubric and expected quality level.

3. Explicit Feedback from Users 📊

When your LLM is deployed in a real-world environment, leveraging **explicit user feedback** is invaluable. By integrating options for users to rate responses (such as thumbs up/down) or request a re-generation, you gather direct insights on performance from the end-users themselves. This feedback can serve as a real-world quality gauge and provides a dataset for further fine-tuning.

Example:

For a chatbot that helps users with product troubleshooting, add a UI element with "thumbs up" 👍 and "thumbs down" 👎 options after each response. If users click "thumbs down," ask for specific reasons such as "not relevant" or "confusing response." Use this data to refine the model's prompt or improve training data.

Why it works 👍:

- Provides **direct insight** into what users find helpful or unhelpful
- **Closes the feedback loop** by identifying specific areas for improvement
- Allows ongoing optimization for real-world effectiveness

> **Pro Tip:** Encourage users to provide feedback with minimal friction. The easier it is for them to rate a response, the more feedback you'll receive.

Putting It All Together 🔄

Once you have this feedback loop in place, you're ready to optimize your LLM further. By using metrics like semantic similarity, LLM-as-a-Judge, and explicit feedback, you can regularly evaluate and improve the model. Here are some methods to enhance the system over time:

1. **Prompt Engineering** — Refine prompts based on observed feedback, making them more specific to reduce ambiguity.
2. **Fine-tuning —** Use gathered feedback as a dataset to fine-tune your model for improved performance.
3. **Retrieval-Augmented Generation (RAG)** — Integrate factual retrieval to make responses more accurate and relevant.
4. **Model Updates and Scaling** — Regularly update model versions and test them using the same feedback loop to measure improvements.

> **Example Outcome:** An LLM providing legal advice can be refined with RAG, allowing it to pull from verified legal resources in response to queries. User feedback helps identify the most trusted sources, ensuring the LLM remains accurate and authoritative.

By combining these methods, you'll have a robust framework for evaluating and continuously improving LLMs in open-ended and complex tasks.

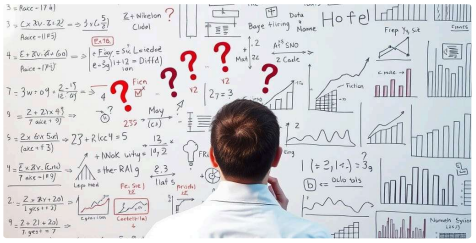
Written by Ritesh Gupta

3.6K Followers · 28 Following

Follow

Data Scientist, I write Article on Machine Learning| Deep Learning| NLP | Open CV | AI Lover ❤️

More from Ritesh Gupta



Ritesh Gupta

Can You Handle These 25 Toughest Data Science Interview Questions?

The role of a Data Scientist demands a unique blend of skills, including statistics, machine...

Sep 25 · 211 · 7

In Python in Plain English by Ritesh Gupta

7 GitHub Repos to Transform You into a Pro ML/AI Engineer

Hands-On Guides, Tools, and Frameworks to Fast-Track Your AI Journey

Nov 5 · 143 · 1

In Artificial Intelligence in Plain En... by Ritesh Gu...

From Jupyter to Production: Deploying Machine Learning...

Turn your Jupyter Notebook experiments into production-ready applications with this...

Oct 24 · 123

Ritesh Gupta

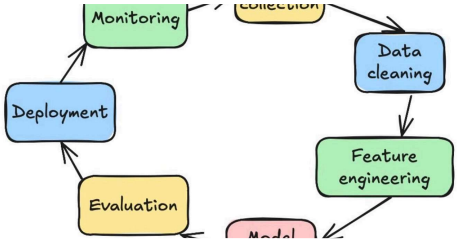
10 Must-Try LLM Projects to Boost Your Machine Learning Portfolio

🚀 10 Exciting LLM Projects to Elevate Your Machine Learning Skills! 🧠

Oct 6 · 124 · 1

See all from Ritesh Gupta

Recommended from Medium

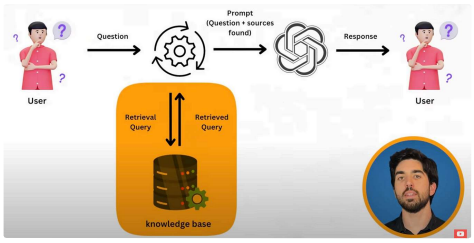


In Towards Data Science by Haden Pelletier

Every Step of the Machine Learning Life Cycle Simply Explained

A comprehensive guide to the ML life cycle with examples in Python

3d ago 193 3



In Towards AI by Louis-François Bouchard

Advanced RAG Evaluation Techniques for Optimal LLM...

Why RAG Evaluation Matters and Techniques to Leverage

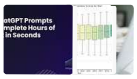
6d ago 7

Lists

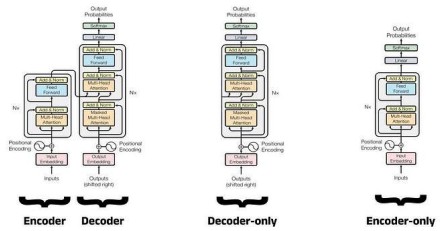
Predictive Modeling w/ Python
20 stories · 1691 saves

Practical Guides to Machine Learning
10 stories · 2057 saves

Natural Language Processing
1839 stories · 1461 saves



ChatGPT prompts
50 stories · 2295 saves



Vipra Singh

LLM Architectures Explained: BERT (Part 8)

Deep Dive into the architecture & building real-world applications leveraging NLP...

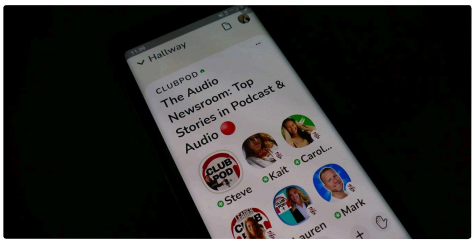
Nov 17 131

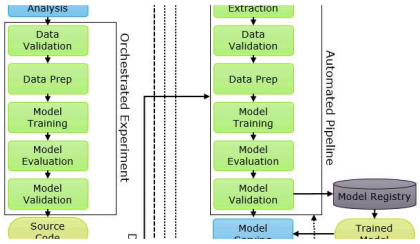
In Artificial Intelligence in Plain En... by Ritesh Gu...

From Jupyter to Production: Deploying Machine Learning...

Turn your Jupyter Notebook experiments into production-ready applications with this...

Oct 24 123





Pierre DeBois

Podcasts That Will Elevate Your Data Science Skills and Tech...

Like radio, podcasts offer a window into the experiences of others. Use that to build your...

4d ago 25

Manralai

MLOps & LLMOps Power Tools: 16 Must-Haves & Nice-to-Haves for...

Comprehensive version of the MLOps and LLMOps toolbelt

Jul 2 184

See more recommendations