

What is worth of your data to LLM?

Abhishek Tiwari 

Citation: *A. Tiwari*, "What is worth of your data to LLM?", Abhishek Tiwari, 2024. [doi:10.59350/p9kx-11w84](https://doi.org/10.59350/p9kx-11w84)

Published on: September 08, 2024

When a machine learning model is trained on a dataset, not all data points contribute equally to the model's performance. Some are more valuable and influential than others. Unfortunately value of data for training purposes is often nebulous and difficult to quantify. Applying data valuation to large language models (LLMs) like GPT-3, Claude 3, Llama 3.1 and their vast training datasets has faced significant scalability challenges to date. We need principled and scalable ways to answer data valuation related questions.

This isn't just an academic concern. As AI systems integrate further into our lives and economy, the issue of fairly compensating data providers has gained urgency. Recent legal challenges against big tech companies highlight the tension between data creators and those who profit from it. The New York Times' lawsuit against OpenAI and Microsoft over copyright infringement directly relates to the issue of data valuation in AI. Without a clear method for valuing data contributions, we risk a system where AI benefits mainly accrue to big tech companies, leaving individual data creators uncompensated.

In 2019 in a seminal research paper, Zou and Ghorbani introduced concept of equitable valuation in the context of data for machine learning models. Their Data Shapley approach outlined key principles of equitable valuation i.e. assigning value to individual data points or sources based on their contribution to the model's performance. Data Shapely provided a solid foundation for thinking about fair data valuation. However, implementing them in practice, especially for large-scale models like LLMs, is where things get tricky.

A new paper (currently under review) from researchers at Carnegie Mellon University, University of Toronto, and other institutions addresses this challenge. It presents a significant advance in quantifying the value of training data for large AI models. The researchers developed an algorithm called LOGRA (Low-rank Gradient Projection) that improves the scalability of influence functions, a technique for estimating data value. LOGRA tackles a fundamental question: given a specific AI model output, which pieces of training data were most influential in producing it?

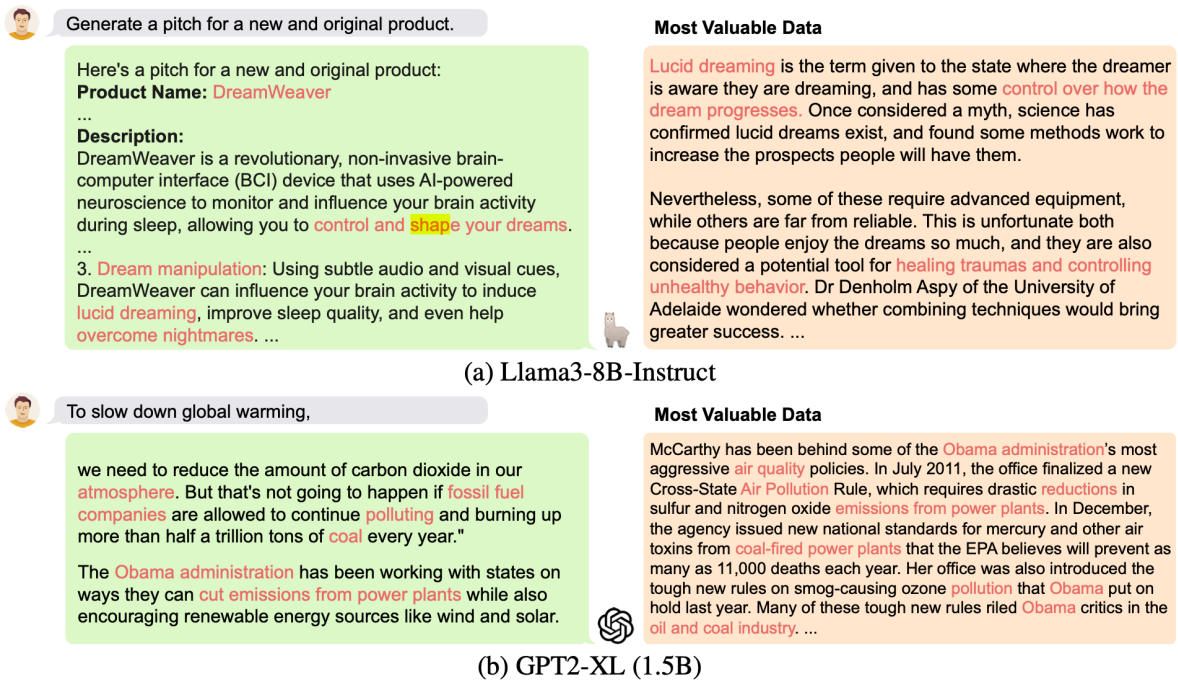


Figure 1: Most valuable data identified by LOGRA for a given query or prompt

Computational cost

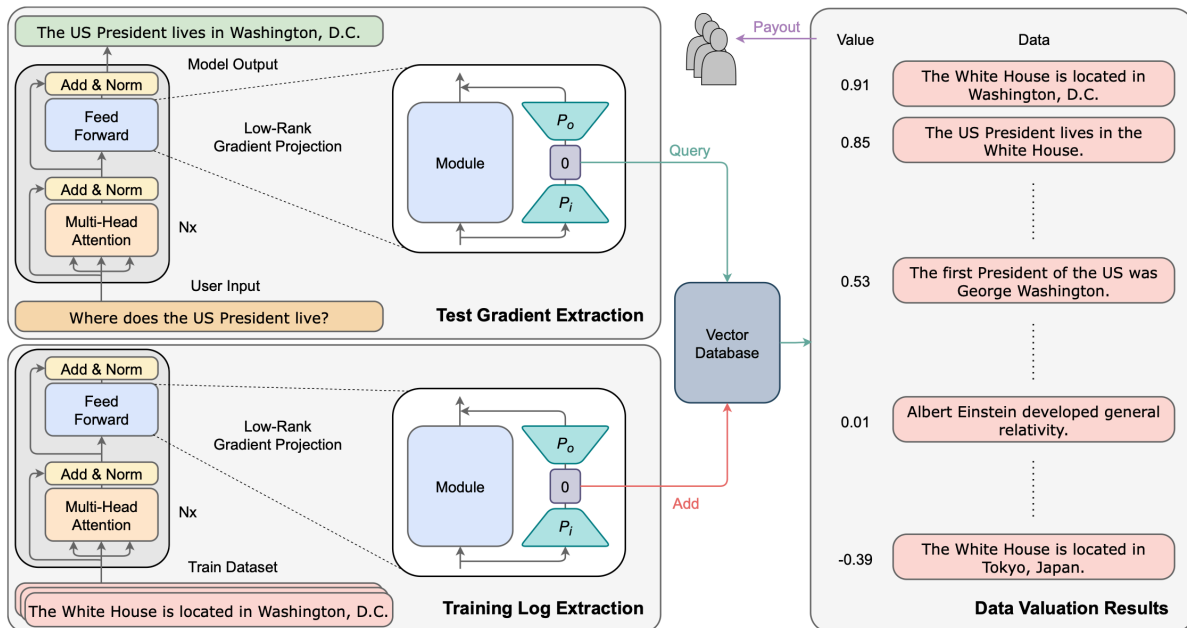
Existing data valuation methods face significant limitations when applied to large language models (LLMs) and vast training datasets, especially when compared to the LOGRA approach. Many traditional methods, such as Data Shapley or leave-one-out approaches, require retraining the model multiple times with different subsets of data. This becomes computationally infeasible for large models like GPT-3 or Claude 3, where a single training run can cost millions of dollars. Other methods like TRAK or Arnoldi IF use large projection matrices that consume significant GPU memory, limiting their applicability to billion-parameter models.

Traditional influence function methods often involve expensive operations like computing inverse Hessian-vector products, which scale poorly with model size. Due to these computational constraints, some methods are forced to use very small projection dimensions, which can limit their ability to capture complex relationships in the data. Many existing approaches also require computing and storing full gradients for all training data, which is memory-intensive and computationally expensive for large datasets.

LOGRA addresses these limitations in several innovative ways. It uses an efficient gradient projection algorithm that leverages the structure of backpropagation, reducing both computational and memory costs. This allows for higher projection dimensions without incurring prohibitive memory costs,

potentially leading to more expressive and accurate valuations. LOGRA can compute projected gradients without materializing full gradients, saving memory and improving efficiency.

In experiments, it demonstrated up to 6,500 times higher throughput and 5 times lower GPU memory usage compared to previous methods when applied to an 8 billion parameter language model. This improvement makes data valuation techniques feasible for today’s largest AI models.



Integration with LLM Ecosystem

Another significant limitation of existing data valuation methods is the lack of compatibility with the complex ecosystems and optimisation techniques used in training large language models. This makes it difficult to integrate these methods into existing LLM training pipelines.

LOGRA is designed to be compatible with various scalability tools in the LLM ecosystem, making it more practical for real-world large-scale AI development environments. These integrations are primarily facilitated through LOGIX, the software package developed alongside LOGRA.

```
# Running data valuation with large scale language models like Llama3
logix.setup({"grad": ["log"]})
logix.eval()
merged_test_logs = []
for idx, batch in enumerate(tqdm(data_loader)):
    data_id = tokenizer.batch_decode(batch["input_ids"],
                                     skip_special_tokens=True)
    targets = batch.pop("labels")
    with run(data_id=data_id, mask=batch["attention_mask"]):
```

```
model.zero_grad()
logits = model(**batch)

shift_logits = logits[..., :-1, :].contiguous()
shift_labels = targets[..., 1:].contiguous()
loss = F.cross_entropy(
    shift_logits.view(-1, shift_logits.size(-1)),
    shift_labels.view(-1),
    reduction="sum",
    ignore_index=-100,
)
accelerator.backward(loss)

test_log = logix.get_log()
merged_test_logs.append(copy.deepcopy(test_log))

if idx == args.batch_size - 1:
    break

merged_test_log = merge_logs(merged_test_logs)
result = run.influence.compute_influence_all(merged_test_log,
log_loader)
```

LOGIX is designed to work seamlessly with popular deep learning frameworks and LLM-specific tools. It's compatible with PyTorch, one of the most widely used frameworks for developing and training large language models. This compatibility extends to PyTorch's advanced features like automatic mixed precision (autocast) and just-in-time compilation (torch.compile), which are crucial for efficient LLM training.

I appreciate the non-intrusive design of LOGIX. Too often, adding new capabilities to an AI pipeline means rewriting half your codebase. Use of PyTorch hooks to intercept and collect necessary information without requiring significant modifications to the training loop. This makes it easier to add data valuation capabilities to existing LLM training pipelines.

More importantly, LOGIX integrates well with distributed training paradigms, which are essential for training LLMs across multiple GPUs or machines. It supports various forms of data parallelism, including Fully Sharded Data Parallelism (FSDP), a technique commonly used to train very large models by sharding model parameters across multiple devices.

Lastly, LOGIX provides support for efficient storage and retrieval of projected gradients, turning the data valuation problem into a vector similarity search problem. As a result, LOGIX achieves significantly higher throughput and lower GPU memory usage compared to existing methods when applied to large models. While the current implementation uses memory-mapped files, the authors note that it could be extended to use more advanced vector databases, which are commonly used in LLM deployment for efficient retrieval of relevant information.

Conclusion

Overall, I believe LOGRA and LOGIX represent a significant step forward in making data valuation practical for large-scale AI. The authors clearly understand the LLM ecosystem and have created a tool that fits seamlessly into existing workflows. It's not perfect, but it's a step in the right direction. For anyone working on LLMs, this could be a valuable addition to their toolkit.