

DIGITIZATION AND ANALYZATION IN THE NATURAL GAS INDUSTRY

Romet Limited

Frederick J Maly MBA, MAM, BSEE

Senior Product Manager

May 15, 2026

1. Introduction

Digitization and analyzing data in natural gas measurement is transforming with Artificial Intelligence (AI). This has rapidly moved from research labs into everyday tools such as Microsoft Copilot, ChatGPT, Perplexity, and countless enterprise applications. Despite this widespread adoption, most people, including developers, do not have a clear understanding of how natural gas applications are run with large language models (LLMs) and/or how are created. In addition, why massive data centers and GPUs are required, how models are stored and deployed, how front-end tools handle user data, and how enterprise systems integrate AI safely and securely.

This white paper will first provide a clear, non-mathematical but technically accurate explanation of how digitization and analyzation work with natural gas meter data. But first we need to understand how modern AI works, from training to deployment, and how the next-generation energy intelligence platform uses AI in a secure, SOC-2-aligned architecture.

2. The Foundation of Digitizing and Analyzing Data

To start with, natural gas meter data can use LLMs. LLM's are neural networks trained to predict the next token (word or sub-word) in a sequence. Although the core task is simple, predict the next token, the scale of training produces emergent capabilities such as reasoning, writing, coding, and analysis.

Learning a language as an infant and training a large language model are similar on the surface. An infant spends years hearing words, phrases, and sentences in everyday life. Over time, the child begins to recognize patterns; which sounds go together, which words follow others, and what meanings those combinations tend to carry. Through constant exposure, the child gradually builds an internal model of the language.

An LLM goes through a comparable pattern-exposure process, but at an enormous scale and speed. Instead of hearing speech from parents and caregivers, the model is exposed to billions of sentences inside a data center. Instead of taking years, the training takes place over a few months using thousands of processors working in parallel. Just like the infant, the model doesn't memorize every sentence. It absorbs patterns: how words relate, how grammar works, and how ideas connect.

So, while the biology of a child and the mathematics of an LLM are completely different, the analogy holds in one important way: both learn language by being exposed to massive amounts of examples and internalizing the patterns that emerge.

Below are some LLMs currently being used by Microsoft Copilot and AWS Bedrock

Model Family	Used By	Parameter Count
GPT-4	Copilot	~1.76T (estimated)
GPT-4o / 4.1	Copilot	Not disclosed
Semantic Indexing Model	Copilot	Not disclosed

Re-ranking Model	Copilot	Not disclosed
Claude 3 / 3.5	AWS Bedrock	Not disclosed
Llama 2	AWS Bedrock	7B / 13B / 70B
Llama 3	AWS Bedrock	8B / 70B
Titan Text	AWS Bedrock	Not disclosed
Titan Embeddings	AWS Bedrock	Not disclosed
Jurassic-2	AWS Bedrock	Not disclosed
Cohere Command R / R+	AWS Bedrock	Not disclosed

Table 1

2.1 How LLMs Learn

During training, the model repeatedly reads text; and lots of it. OpenAI used posts from Reddit for the model to learn words. The object of learning is for the LLM to predict the next word, or token, from the text that it has been trained on. It “reads” one word at a time and then the model predicts the next word based on each word in the prompt. If the word is incorrect, a “weight” is added to adjust the model’s accuracy.

The LLM predicts facts by learning the language, how people write and speak, and identifying patterns within that data. For example, to predict that the capital of France is Paris, an AI model is trained on extensive text data containing information about countries, cities, and capitals.

Example: Predicting the Capital of France

1. **Training on Text Data:** The AI reads millions of sentences and documents where "Paris" is frequently mentioned alongside "France" and "capital."
2. **Understanding Context:** It learns the relationships between words and phrases, recognizing that "Paris" often appears in contexts related to "France" and "capital city."
3. **Calculating Probabilities:** When asked about the capital of France, the AI calculates the likelihood of various cities being the answer based on its training.
4. **Making a Prediction:** Since "Paris" has the highest probability given the context, the AI predicts "Paris" as the capital of France.

This process involves complex algorithms and statistical models that allow AI to make informed predictions based on learned language patterns and contextual clues.

- Reads text
 - Predicts the next word or token
 - Compares prediction vs. the correct token
 - Computes error (loss)
 - Updates billions of parameters through matrix multiplication
- This loop runs trillions of times across massive datasets.

2.2 What the Model Actually Learns

Through this process, the model learns grammar, semantics, world knowledge, reasoning patterns, coding structures, and problem-solving steps. The model does not store documents; it stores patterns in the form of numerical weights.

3. How LLMs Are Trained: Data Centers, GPUs, and Matrix Multiplication

3.1 Why Training Requires Massive Compute

Training an LLM involves billions of parameters, trillions of matrix multiplications, thousands of GPUs running in parallel, and weeks or months of continuous computation. GPUs are used because they can perform thousands of parallel mathematical operations.

How does text become numbers inside an AI model? When documentation is fed into a LLM, the system does not read words the way humans do. Every token, a full word or a fragment of a word, is converted into a vector, which is simply a long list of numbers representing meaning. These vectors capture relationships such as similarity (“engineer” is closer to “technician” than to “banana”) and context (“meter” in electrical engineering differs from “meter” in natural gas).

The model compares and transforms these vectors using matrix multiplication, the core mathematical operation that lets the system detect patterns, relationships, and structure across billions of examples. Every layer of the model performs additional matrix multiplications, gradually refining the representation of the input until the model can predict the next token.

Training requires massive data centers. Training an LLM means running trillions of these matrix multiplications across enormous datasets. This is only possible with specialized hardware, tens of thousands of GPUs working in parallel, synchronized down to microseconds.

These training clusters require:

- Massive electrical power to run the GPUs continuously for weeks or months
 - Industrial-scale cooling systems, often liquid-cooled, to remove the heat generated
 - High-bandwidth networking to keep all GPUs communicating efficiently
 - Purpose-built data center infrastructure designed specifically for AI training workloads
- This is why companies build billion-dollar AI training facilities: the computational load is far beyond what a normal data center can support.

Once training is complete, the model’s knowledge is stored in its parameters and the training center is no longer needed. The numerical weights that encode everything the model has learned. At that point:

- The model becomes a fixed artifact, like a very large software file.
- It can be copied, deployed, and run on far smaller infrastructure.
- Inference (using the model) requires only a fraction of the compute needed for training.

Depending on the model size, it can run on:

- A small cluster of GPUs
- A single high-end server
- A laptop or even a mobile device (for smaller models)

The enormous training data center is only needed during the creation of the model. Afterward, the model can be hosted almost anywhere.

Below is Table 1 showing the size of the hard drive needed to store an LLM model.

Model Size	FP16 Storage	Practical Drive Size
7 Billion Parameters	14 GB	32-64 GB
13 Billion Parameters	26 GB	64-128 GB
70 Billion Parameters	140 GB	256-512 GB
175 Billion Parameters	350 GB	512 GB-1 TB
500 Billion Parameters	1,000 GB (1 TB)	2 TB
1 Terabyte Parameters	2,000 GB (2 TB)	4 TB

Table 2

One can see from Table 2 below the different LLM models and there relative size.

Parameter Size	Real or Representative Models
7B-15B	LLaMA-3 8B, Mistral 7B, Gemma 7B, Phi-3 Medium, Qwen 7B/14B
30B-70B	LLaMA-3 70B, Mixtral 80B, Qwen-72B, Yi-70B
175B	GPT-3, Jurassic-1 Jumbo, early PaLM variants
500B	Megatron-Turing 530B, PaLM-2 (largest variant)
1T+	GPT-4 (MoE), Gemini Ultra, Grok-2, DeepSeek-V3 (effective trillion-scale)

Table 3

It's safe to say that ChatGPT-4 could be hosted on a high end home server.

3.2 CPU vs. GPU vs. TPU

CPU, GPU, and TPU are three different types of processors, each built for a different style of computation. Additionally, there are two AWS chips that directly replace traditional GPUs. They are AWS Trainium and AWS Inferentia. The distinctions matter because modern AI workloads depend heavily on the right hardware. The sections below give definitions, grounded in current industry descriptions.

CPU — Central Processing Unit

What it is: The general-purpose processor found on every computer. It handles operating systems, applications, logic, branching, and orchestration.

What it's built for:

- Low-latency, sequential tasks
- Complex decision-making and control flow
- Running many different types of software

Why it matters: CPUs excel at versatility but are not optimized for the massive parallel math required for deep learning. They remain essential for preprocessing, data loading, and coordinating GPU/TPU workloads.

GPU — Graphics Processing Unit

What it is: A massively parallel processor originally designed for rendering graphics. It contains thousands of smaller cores that can perform the same operation on many pieces of data at once.

What it's built for:

- Highly parallel workloads
- Matrix and vector operations
- Deep learning training and inference

Why it matters: GPUs became the backbone of modern AI because neural networks rely on repeated matrix multiplications across huge datasets. Their architecture allows them to train models like CNNs (Convolutional Neural Networks) and Transformers far faster than CPUs.

TPU — Tensor Processing Unit

What it is: A custom AI accelerator designed by Google. It is an ASIC (Application-Specific Integrated Circuit) built specifically for tensor operations—the mathematical core of neural networks.

What it's built for:

- Large-scale matrix multiplication
- High-throughput training and inference
- Cloud-scale AI workloads

Why it matters: TPUs deliver extremely high performance-per-watt for deep learning. They are optimized for Google's TensorFlow ecosystem and excel at hyperscale training where efficiency and speed are critical.

These distinctions help explain why AI training requires specialized data centers. CPUs alone cannot handle the parallel math required for LLMs. GPUs (and TPUs in Google's ecosystem) provide the parallel compute needed

to run trillions of matrix multiplications during training. After training, inference can run on far smaller hardware because the heavy learning phase is complete.

3.3 Where the Model “Belongs” During Training

During training:

- Weight matrices belong in GPU memory
 - Optimizer state belongs in GPU/CPU memory
 - Data batches stream continuously
- Nothing is saved to disk unless the training script writes a checkpoint.

3.4 How the Model Is Saved

At intervals, the training system copies weights from GPU to CPU, serializes them, and saves them in formats such as .pt, .safetensors, .bin, .onnx, or .gguf. This file is the model.

4. After Training: How LLMs Are Deployed

Once training is complete, the model is “frozen” and can be loaded onto cloud GPUs, small data centers, edge servers, laptops, or even phones. At this point the training is complete. There is no TRAINING performed in the model or anywhere else. The word “training” is often confused with configuring the agent (discussed later).

4.1 Model Size Examples

Model	Parameters	Deployment
GPT-2	1.5B	Laptop / small GPU
Llama-3 8B	8B	Laptop / single GPU
GPT-3	175B	Cloud only
GPT-4 class	1T+ (estimated)	Large data centers only

Table 4

5. How Front-End AI Tools Work (Copilot, ChatGPT, Perplexity)

Front-end tools (Copilot, ChatGPT, etc.) are interfaces, not LLM models. They accept your request, send it to a backend model, and display the response. The interfaces determine which models to use, how to route the request, and what additional context or system instructions to apply, so the same prompt can behave differently depending on the interface.

Why this matters

- The “interface” decides which model family to call (GPT-4, GPT-4o, Claude, Gemini, etc.).
 - The “interface” may add grounding (search, documents, enterprise data).
 - The “interface” may apply safety rules, formatting rules, or domain-specific logic.
 - The “model” itself only generates text based on the input it receives.
- This separation explains why the same model behaves differently across platforms and why front-end tools can evolve independently of the underlying LLM.

5.1 Where Uploaded Documents Go

Stored only in your session (not used for training):

- Microsoft Copilot
- Azure OpenAI
- AWS Bedrock
- IBM watsonx
- Cohere

May be used for training new models if you opt in:

- OpenAI (consumer ChatGPT)
- Google Gemini
- Anthropic Claude

Example: Cursor AI:

- Privacy Mode ON → no training
- Privacy Mode OFF → data may be used for training

This means that if the privacy mode is turned off, Cursor will use your documentation and any interactions a person has with Cursor, to be used in training a new model. Once the model is trained the information provided is part of the model.

5.2 “AI made my document public” a misconception

If AI returns a document, it means it was already public. AI cannot publish private documents unless privacy mode is turned off.

6. How Enterprise AI Works. A 40,000 foot view: The next-generation energy intelligence platform Architecture

The next-generation energy intelligence platform uses a secure, multi-layered architecture built on AWS Bedrock, AWS Lambda, and AWS DynamoDB, with strict tenant isolation and schema-aware orchestration. The system converts a natural-language question into a structured database query and returns formatted results without exposing raw infrastructure.

Step-by-Step Flow of a Query

1. User enters a prompt

A user types a natural-language question into the next-generation energy intelligence platform interface (e.g., “Show me the meters with the highest gas usage today.”).

2. Bedrock Agent receives the request

The prompt is sent to an AWS Bedrock Agent, which orchestrates the reasoning steps needed to answer the question.

3. Next-generation energy intelligence platform’s schema and rules are loaded into the Agent

The Agent is provided with the next-generation energy intelligence platform’s schema, allowed tables, tenant filters, and business rules.

This ensures the model only generates queries that respect utility boundaries and approved data structures.

4. Bedrock Agent generates an executable “script”

Using the schema and the user’s question, the Agent produces a structured output—typically Python snippet, DynamoDB query, or SQL-like expression—depending on the next-generation energy intelligence platform’s configuration.

This script is not executed inside Bedrock; it is handed off to Lambda.

5. AWS Lambda executes the script

Lambda receives the generated script and securely executes it against the next-generation energy intelligence platform's DynamoDB tables.

Lambda enforces:

- tenant isolation
- access control
- query validation
- logging and audit requirements

6. Lambda returns structured data to Bedrock

Lambda sends the query results back to the Bedrock Agent in a structured such as a JSON format.

7. Bedrock formats the final response

The Agent converts the raw data into a human-readable answer, tables, summaries, charts, or narrative explanations, and returns it to the user.

6.1 Why This Architecture Is Secure

- Bedrock never stores prompts
- Lambda is sandboxed
- Database credentials are isolated
- No training occurs on Next-generation energy intelligence platform data
- All data stays within the utility's tenant

7. Automate the prompt for AI to “observe” and “Notify” of situations.

Suppose a person wants “AI” to observe and notify if the proportional band of a regulator has changed indicating an issue with the regulator.

1. The equation or formula to determine the proportion bands are created.
2. The equation or formula is loaded into the AI Agent (entering the equation in a Copilot prompt and pressing enter loads the equations into Copilot's agent.) along with Next-generation energy intelligence platform's schema and rules.
3. A “prompt” is created and loaded into a scheduler (such as a Cron) and the prompt is scheduled to run at predetermined time.
4. The request is then sent to the agent at predetermine times.
If the scheduler runs the “prompt” once a day or every hour, and if any regulator is out of specifications the user will be notified of the anomaly.

8. Summary

This white paper explained how the natural gas industry will digitize and analyze data with AI and LLMs. It explained how LLMs are trained, how they are deployed, how front-end tools handle data, how user documents are stored or forgotten, how training on user data works, why AI cannot make private documents public, and how Energy Array's AI architecture works end-to-end.

Author BIO



Frederick J. Maly has been engaged in the design, deployment, and administration of computer systems in the natural gas industry since 1985. His career spanning multi-terminal Prime minicomputer platforms and large-scale Unix and Linux-based telemetry infrastructures.

He has designed and managed systems serving thousands of major natural gas consumers across the Midwestern United States, analyzing meter data, integrating various databases, data acquisition, communications, and automation in mission-critical environments.

He has worked with various operating systems and software languages developing applications ranging from natural gas utility measurement, shop floor management and even medical field services.

These skills provide a practical foundation for applying artificial intelligence and advanced analytics to complex, safety-critical operational domain.

References:

1. LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. Nature, 521(7553), 436–444. <https://doi.org/10.1038/nature14539>
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. In Advances in Neural Information Processing Systems (NeurIPS). <https://arxiv.org/abs/1706.03762>
3. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). *Language models are few-shot learners*. In Advances in Neural Information Processing Systems (NeurIPS). <https://arxiv.org/abs/2005.14165>
4. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. In Proceedings of NAACL-HLT. <https://arxiv.org/abs/1810.04805>
5. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI Technical Report. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. <https://www.deeplearningbook.org/>
7. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press. <http://incompleteideas.net/book/the-book-2nd.html>

8. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). *Concrete problems in AI safety*. <https://arxiv.org/abs/1606.06565>
9. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). *On the opportunities and risks of foundation models*. Stanford Institute for Human-Centered AI. <https://arxiv.org/abs/2108.07258>
10. Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). *Transformers in vision: A survey*. ACM Computing Surveys, 54(10), 1–41. <https://arxiv.org/abs/2101.01169>