From Documents to Data: extracting value with LLMs, Embeddings, and Human-in-the-Loop without draining your wallet

29 July 2026 - 6 min. read

Data and Analytics | Large Language Models

Everyone wants to show off with Generative AI, but then they clash with reality: every company sits on a mountain of PDFs, crooked scans, and utility bills that hide precious data. Extracting them at an *Enterprise* scale raises two major problems: how to do it accurately and, above all, how to do it without the CFO passing out when the AWS bill arrives at the end of the month?

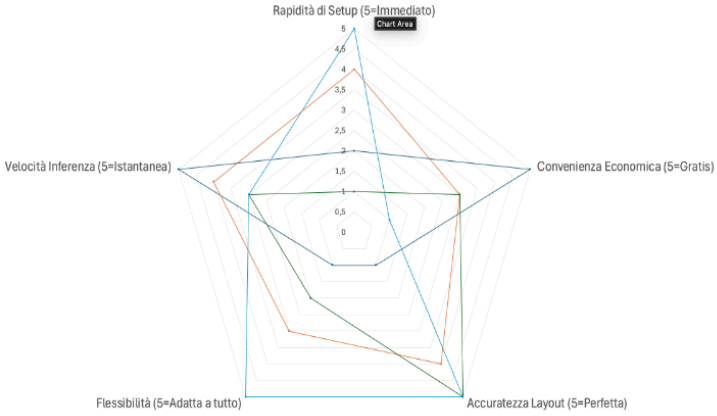
Together with NeN, an innovative, 100% green and digital Italian energy provider, we tackled this exact challenge. Here is how we transitioned from a "Pure LLM" PoC (beautiful, but expensive) to an intelligent, hybrid, and self-learning architecture

The Initial Challenge: the charm (and limits) of "Pure LLM"

When you need to extract data from a document, the market presents you with four options:

1. **Traditional OCR:** Cheap and fast. However, if the layout changes by a millimeter, it panics. Zero semantic understanding.
2. **Managed Services (Feature extraction):** Ready to use, but inflexible if you have specific edge cases (and watch out for vendor lock-in).
3. **Fine-tuned Hybrid Models:** Excellent for spatial analysis, but slow on massive workloads and reluctant to digest unseen formats.

4. **Large Language Models (LLMs):** The magic wand. Total flexibility, they understand context and catch the data even if the scan is poorly done. The only flaw? They are expensive.



In our early stage, we bet everything on setup speed and maximum versatility. Therefore, we chose the "all LLM" route. We built a *ground truth* of 20 utility bills (electricity, gas, dual, native PDFs, and scans) and put several models in the ring.

LLM Selection: no guessing allowed

Choosing a model based on a "gut feeling" is not an acceptable option in engineering. It is the rigorous, data-driven method that transforms a gamble into a solid architectural decision. Here is the process we followed:

- **Validation Set:** Without a reference truth (*ground truth*), you are measuring absolutely nothing. We used our 20 heterogeneous bills to cover real-world scenarios: different providers, multiple versions of the bill for the same supplier, mixed formats (image and PDF), and supply types (gas, electricity, dual).
- **Prompt Engineering:** We wrote the prompt to provide clear and unambiguous instructions to the model.
- **Selecting the Contenders:** We identified a subset of 4 generative models of the same "caliber", ideal for the task. Specifically, we brought into the testing arena **Amazon Nova 2 Lite**, **Amazon Nova Pro**, **Claude 3 Haiku**, and **Claude 4.5 Sonnet**.
- **The Ruthless Benchmark:** We ran the models on the validation set to extract objective results.

The absolute winner for the price/performance ratio? **Claude 4.5 Sonnet**, orchestrated naturally through **Amazon Bedrock**.

All very nice. But how much does it cost?

The PoC was set up in no time: decoupling with **Amazon API Gateway**, storage on **Amazon S3**, **AWS Lambda** acting as the glue, and Bedrock working the magic. The system extracted data with frightening accuracy. But there was a problem: using a high-end LLM for *every single bill*, especially on production volumes, is not financially sustainable. It is like using a Ferrari to go grocery shopping around the corner.

We had to optimize. And to do so, we looked our data in the face. Bills have two key characteristics:

1. The input formats are few (native PDF or image).
2. The templates are repetitive: bills coming from the same provider will reasonably have a common layout (unless there is a version change).

Why pay the LLM to struggle to understand a structure that we have actually already seen?

The pivot: Embeddings and the return of Classical OCR

We needed a way to instantly understand if a document belonged to a known template. Pure computer vision models are fast but fragile, while multimodal models are too heavy. The solution? **The Embedding algorithm**. We transform the bill template into a numerical vector (an objective digital fingerprint). To understand if a new bill matches a known layout, we just need to calculate the *cosine similarity* between vectors. Fast, painless, and cheap.

We thus designed a two-speed **Version 2.0** architecture:

- **Routing:** An **AWS Lambda** takes the file, validates it and, if necessary, performs pre-processing on the images.
- **Orchestration:** Here **AWS Step Functions Express** comes into play, handling the vector comparison logic by looking up the "signatures" of templates already saved on **Amazon DynamoDB**.

From here, the flow splits:

- **The Fast Track (Known Layout):** The template already exists. Instead of waking up the LLM, we divert the traffic to a *self-contained* Lambda that uses good old

classical OCR. Already knowing where to look (the spatial coordinates are saved in the DB), the OCR extracts the data in milliseconds at a ridiculous cost.

- **The Discovery Track (Unknown Layout):** The template is new. We call **Amazon Bedrock**. The LLM extracts the info semantically, but it doesn't just do that: we map the *bounding boxes* (the coordinates of the keys) and calculate the embedding of the new document. We save this "new signature" on DynamoDB. From the next bill onward, this exact layout will go into the Fast Track!

The result? A self-learning ecosystem. Costs plummet as the system stores new templates, and the LLM works only when gray matter is truly needed.

Human-in-the-Loop: trust is good, control is better

Delegating everything to an algorithm in production is the fastest way to lose sleep at night. For this reason, we inserted a **Human-in-the-Loop (HITL)** pattern as the linchpin of governance:

- **Traceability:** Every extracted piece of data has its spatial digital footprint and a rigorous *confidence score*.
- **Serverless Backoffice:** We set up a Single Page Application in React (hosted on **S3** and globally distributed via CDN with **CloudFront**). Here, operators can visually validate the data overlaid on the original bill image. The backend APIs? Lambda, of course.
- **Proactive Alerting:** What happens if Eni suddenly changes its invoice layout? The average confidence score drops. **Amazon CloudWatch** notices, triggers an alarm, and **Amazon SNS** floods the engineering team with emails before the business even realizes it.

Takeaways

Generative AI is not the solution to everything, it is a tool. Architectural maturity in the Cloud is not demonstrated by firing Foundation Models at every single task hoping it works. True Engineering (with a capital E) lies in orchestration: use **Large Language Models** only for pure cognitive capability, rely on **classical OCR** to push efficiency on deterministic tasks, and use **Embeddings** for intelligent routing.

This way you will have an architecture that not only scales amazingly, but learns on its own, leaves you in full control of your data, and makes your FinOps smile.

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

Paolo Serale

As a Senior Data & AI Project Manager at beSharp, I guide companies and teams in adopting advanced Cloud and AI solutions.With solid experience in managing complex projects, I transform large volumes of data into strategic value and concrete innovation.As a tech speaker and evangelist, I share my passion for the future of technology through industry events and communities.
