

E-BOOK · OSNEY SOUZA

AI That Never Makes Up Numbers

Separate what AI reads from what code calculates to build trustworthy systems in production

FREE SAMPLE

Osney A. de Souza

AI Engineer · osneyandrade.github.io

AI That Never Makes Up Numbers

Separate what AI reads from what code calculates to build trustworthy systems in production

Osney A. de Souza · AI Engineer

Edition: October 2026

© 2026 Osney A. de Souza. All rights reserved.

Single-user license: please do not redistribute this file. For team or company use, ask for a license.

Examples use fictitious data. This material contains no code, data or screens from the author's clients or employers.

<https://osneyandrade.github.io>
juniorthesouza017@gmail.com

Contents

1	The boundary between reading and calculation	5
	Understand why language models cannot generate financial values and how to structure a flow where the AI reads and code calculates.	
2	Reading unstructured documents	<i>Continues in the full version</i>
	Implement data extraction from scanned PDFs, photos, and spreadsheets using OCR and language models to locate fields without fabricating content.	
3	Deterministic oracle and secrecy gate	<i>Continues in the full version</i>
	Create a validation layer that recalculates values and blocks incorrect mappings before data enters the destination system.	
4	RAG with citation and local embeddings	<i>Continues in the full version</i>
	Build search systems for metadata and documents using embeddings and local models to prevent leakage of sensitive data.	
5	Agents with tool-calling for verification	<i>Continues in the full version</i>
	Develop agents that call tools to verify dates, registrations, and totals, delegating business logic to code.	
6	Cost cascade and continuous evaluation	<i>Continues in the full version</i>
	Organize the use of cloud and local models by cost and establish error metrics for production monitoring.	
7	Integration with destination systems	<i>Continues in the full version</i>
	Connect the reading and validation pipeline to the management system, generating exception panels for human review.	

Before you start

Who it is for

- Developers building AI solutions for finance, tax, and operations.
- Technical leaders responsible for deploying AI in regulated environments.
- Analysts needing to automate document reading without compromising data integrity.
- Teams managing unstructured data like invoices, contracts, and scanned reports.

What you will be able to do

- Understand why language models cannot generate financial values and how to structure a flow where the AI reads and code calculates.
- Implement data extraction from scanned PDFs, photos, and spreadsheets using OCR and language models to locate fields without fabricating content.
- Create a validation layer that recalculates values and blocks incorrect mappings before data enters the destination system.
- Build search systems for metadata and documents using embeddings and local models to prevent leakage of sensitive data.
- Develop agents that call tools to verify dates, registrations, and totals, delegating business logic to code.
- Organize the use of cloud and local models by cost and establish error metrics for production monitoring.
- Connect the reading and validation pipeline to the management system, generating exception panels for human review.

CHAPTER 1

The boundary between reading and calculation

GOAL *Understand why language models cannot generate financial values and how to structure a flow where the AI reads and code calculates.*

The Danger of Free Value Generation

Language models are trained to predict the next word in a sequence. They learn linguistic patterns, not exact mathematics. When a model receives a prompt like "what is the total on the invoice?", it attempts to estimate the value based on what it read in the text. This generates numerical hallucinations. The model may invent a number that does not exist in the original document, or round inconsistently. In auditing, accounting, and financial operations, a typo or rounding error is not a detail; it is a critical failure that can lead to fines, loss of trust, and direct loss.

The fundamental boundary in AI systems for documents and numbers is clear: AI reads, code calculates. AI is excellent at extracting text from a scanned PDF, identifying that a field is "Total Value", and understanding the context of the sentence. However, AI should never be the primary source of a monetary value or a logical calculation. The final value must always come from a structured read or a recalculation performed by deterministic code.

How to Structure the Read and Validation Flow

To build a reliable system, you must separate responsibilities. The ideal flow begins with the extraction of unstructured text. AI reads the document and identifies fields. Next, the system extracts the raw numerical values. These

raw values are passed to a code layer that performs validation and calculation.

Consider the following hypothetical scenario:

example A digitized invoice contains three item lines:

- Item A: Quantity 10, Unit Price 100.00
- Item B: Quantity 5, Unit Price 200.00
- Item C: Quantity 2, Unit Price 150.00

The document also has a "Subtotal" field filled in manually as 2,000.00.

A language model could read the document and say: "The subtotal is 2,000.00". This seems correct, but if there is a typo in the original document (for example, the subtotal written as 2,005.00), the model may accept the wrong value if it is not specifically trained to detect inconsistencies.

Deterministic code, on the other hand, reads the item lines, multiplies the quantity by the unit price for each, and sums the results. The code result is 3,500.00. The system compares this calculated result with the document subtotal. If there is a difference, the system generates an exception for human review. The final value accepted by the system is the one calculated by the code, not the one read by the model.

Technical Implementation with Python

Below, a snippet of Python code that illustrates this separation. The script reads the extracted data (simulated) and calculates the total. It does not use AI to generate the value, but rather to validate whether the value extracted by OCR or AI is consistent with the calculation.

```

import pandas as pd

# Simulation of data extracted from a PDF or OCR
# The language model provided the fields, but not the final value
extracted_data = {
    "item_a_qty": 10,
    "item_a_price": 100.00,
    "item_b_qty": 5,
    "item_b_price": 200.00,
    "item_c_qty": 2,
    "item_c_price": 150.00,
    "document_subtotal": 2000.00 # Value read from the document, possibly
incorrect
}

# Deterministic calculation function
def calculate_total(data):
    total = (data["item_a_qty"] * data["item_a_price"]) + \
            (data["item_b_qty"] * data["item_b_price"]) + \
            (data["item_c_qty"] * data["item_c_price"])
    return total

# Calculation of the actual value
calculated_total = calculate_total(extracted_data)

# Validation: The calculated value must be equal to the document value
if abs(calculated_total - extracted_data["document_subtotal"]) > 0.01:
    print(f"ALERT: Discrepancy detected.")
    print(f"Calculated value: {calculated_total}")
    print(f"Value in document: {extracted_data['document_subtotal']}")
    # Here a human review exception gate would be triggered
else:
    print(f"Validation approved. Total: {calculated_total}")

```

In this example, the code ignores the document value if it does not match the calculation. The system operates based on mathematical truth, not the model's perception.

Implementation Checklist

To ensure your system does not invent numbers, follow these guidelines:

- ☐ Never use LLM output as a primary source for monetary values.
- ☐ Always extract raw values from the document before processing them.
- ☐ Implement a deterministic calculation function in code to validate values.
- ☐ Compare the calculated value with the value extracted from the document.
- ☐ Automatically raise an exception if the difference exceeds a minimum tolerance (e.g., 0.01).
- ☐ Use a deterministic oracle to veto incorrect mappings before saving.
- ☐ Keep language models localized for sensitive data, but use code for financial logic.

Next Steps

Now that we understand that AI reads and code calculates, we need to address how to handle unstructured documents, such as scanned PDFs or photos. Reading these documents requires specific techniques to ensure text is extracted accurately before being processed.

```

import pandas as pd

# Simulation of data extracted from a PDF or OCR
# The language model provided the fields, but not the final value
extracted_data = {
    "item_a_qty": 10,
    "item_a_price": 100.00,
    "item_b_qty": 5,
    "item_b_price": 200.00,
    "item_c_qty": 2,
    "item_c_price": 150.00,
    "document_subtotal": 2000.00 # Value read from the document, possibly
incorrect
}

# Deterministic calculation function
def calculate_total(data):
    total = (data["item_a_qty"] * data["item_a_price"]) + \
            (data["item_b_qty"] * data["item_b_price"]) + \
            (data["item_c_qty"] * data["item_c_price"])
    return total

# Calculation of the actual value
calculated_total = calculate_total(extracted_data)

# Validation: The calculated value must be equal to the document value
if abs(calculated_total - extracted_data["document_subtotal"]) > 0.01:
    print(f"ALERT: Discrepancy detected.")
    print(f"Calculated value: {calculated_total}")
    print(f"Value in document: {extracted_data['document_subtotal']}")
    # Here an exception gate would be triggered for human review
else:
    print(f"Validation approved. Total: {calculated_total}")

```

About the author



Osney A. de Souza is an AI Engineer based in Joinville, Brazil, with 5 years of software development experience. He builds and runs in production an LLM-assisted audit platform: reading unstructured documents (scanned PDFs, Excel, OFX, CSV), RAG with embeddings and pgvector, tool-calling agents, MCP, Structured Outputs, and local and cloud models in a cost-based cascade, backed by about 3,100 automated tests. His core rule: the model never produces the final number. Before that he worked on computer vision and license plate recognition at scale, back-end, mobile apps and systems integration. He is studying Software Engineering at Univille.

Website osneyandrade.github.io
GitHub github.com/OsneyAndrade
Contact juniorthesouza017@gmail.com

FREE SAMPLE

Enjoyed the first chapter?

The full version also includes:

- Reading unstructured documents
- Deterministic oracle and secrecy gate
- RAG with citation and local embeddings
- Agents with tool-calling for verification
- Cost cascade and continuous evaluation
- Integration with destination systems

Get the E-book Now

<https://osneyandrade.github.io/ia-que-nao-inventa-numero/en/>