DSPy:
Not Your Average Prompt Engineering

jina.ai @JinaAI_ hello@jina.ai

WHY DONT YOU LIKE PROMPT ENGINEERING

- It is simple, effective and cost-efficient.
  - Most tricks can be explained in one or few-sentences, instead of 8-page
  - Take you 5 mins to validate the effectiveness via LLM UI/API
  - At almost no cost on GPU
- It is brittle and lacks of a systematic way to improve it.

DSPy:
Not Your Average Prompt Engineering

Basic Prompt Engineering Modules

jina.ai @JinaAI_ hello@jina.ai

ZERO-SHOT

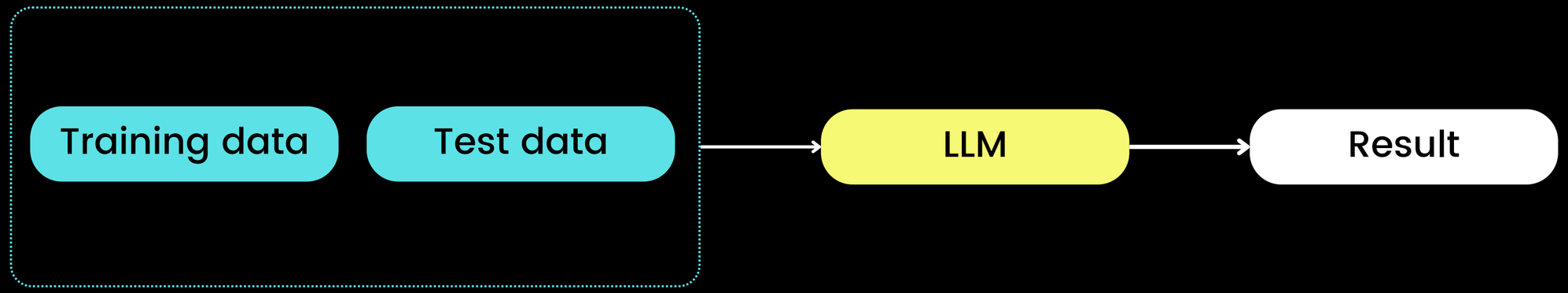
Zero-shot

```
Text: Oh, great. Another meeting. Just what I needed to make my day even more exciting.
Sentiment:
```

FEW-SHOT

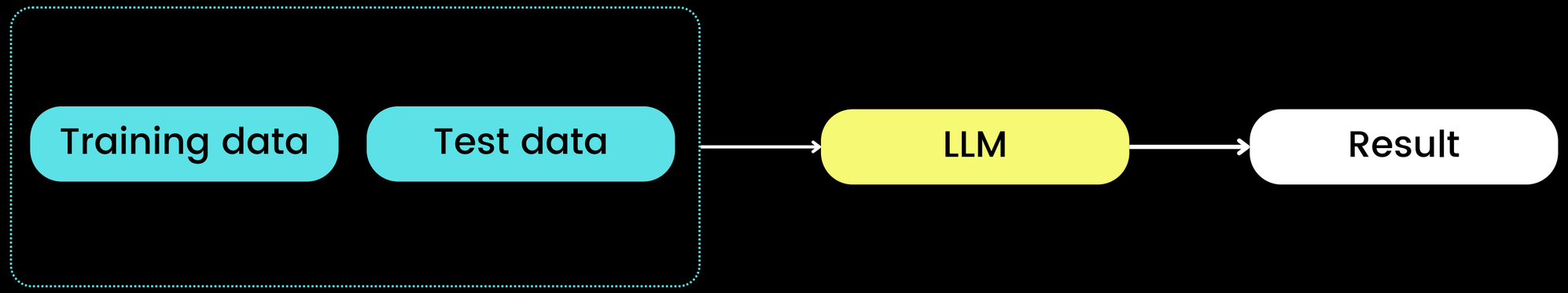
Few-shot examples

```
Text: Wow, thanks for leaving your dirty dishes in the sink for the tenth time this week. It really adds to the ambiance of our kitchen.
Sentiment: negative
Text: Oh, wonderful. Another software update. I'm thrilled to see what new bugs and glitches it brings.
Sentiment: negative
Text: Wow, you're so good at interrupting people. I wish I had that skill too.
Sentiment: negative
Text: Oh, fantastic! The traffic is absolutely amazing today. I can't wait to spend another hour sitting in my car.
Sentiment: negative
Text: I had an amazing time on my vacation. The sights, the food, and the people were all incredible.
Sentiment:
```



FEW-SHOT

- Presents a set of high-quality demonstrations, each consisting of both input and desired output, on the target task.
- Performance is influenced by
  - Training examples, and the order of the examples
  - Example string template



FEW-SHOT BIASES

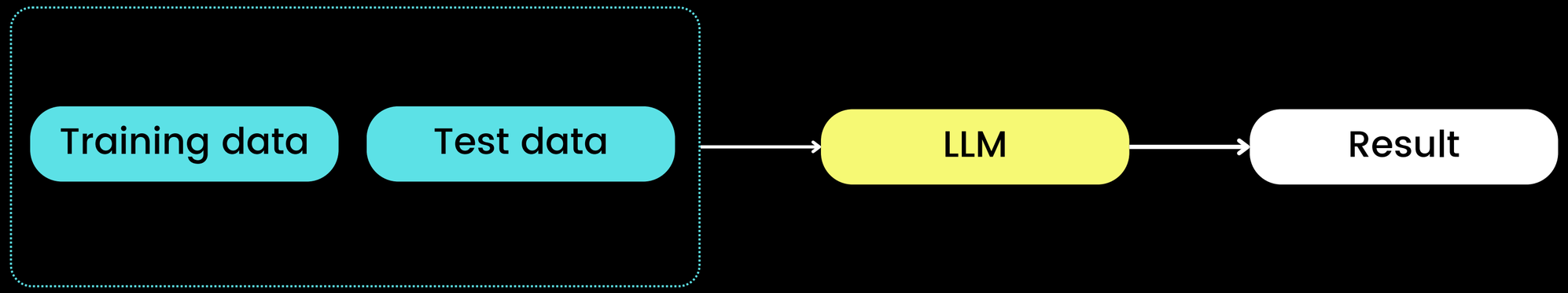
- **Majority label bias** exists if distribution of labels among the examples is unbalanced;
- **Recency bias** refers to the tendency where the model may repeat the label at the end;
- **Common token bias** indicates that LLM tends to produce common tokens more often than rare tokens.

In classic ML

→ Over-sampling

→ Data shuffling

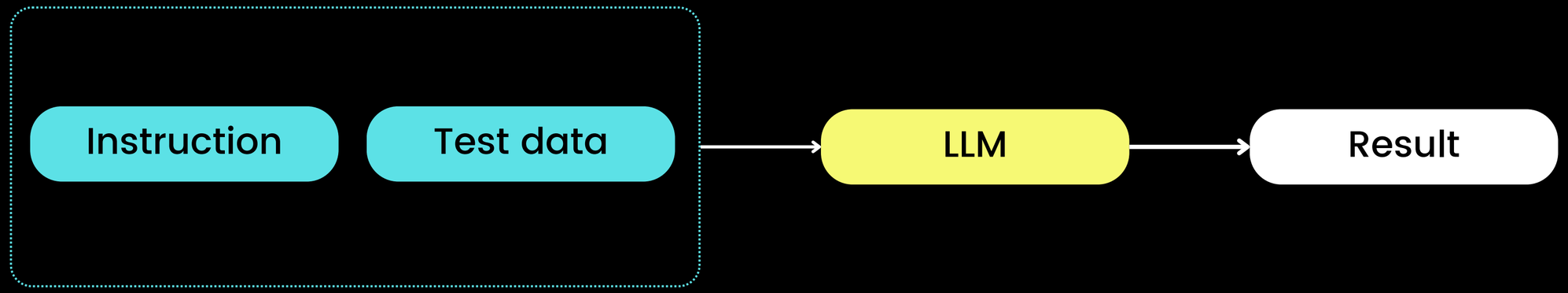
→ Paraphrasing



INSTRUCTION PROMPT

Instruction prompt

```
Please label sentiment of the sentence below into "positive", "negative" and "neutral".
Text: I had an amazing time on my vacation. The sights, the food, and the people were
all incredible.
Sentiment:
```



INSTRUCTION PROMPT

- Explicitly telling model what to do, instead of showing a set of demostrations (i.e. few-shot) and let model immitate.

Instruction few-shot prompting

Few-shot instruction prompting

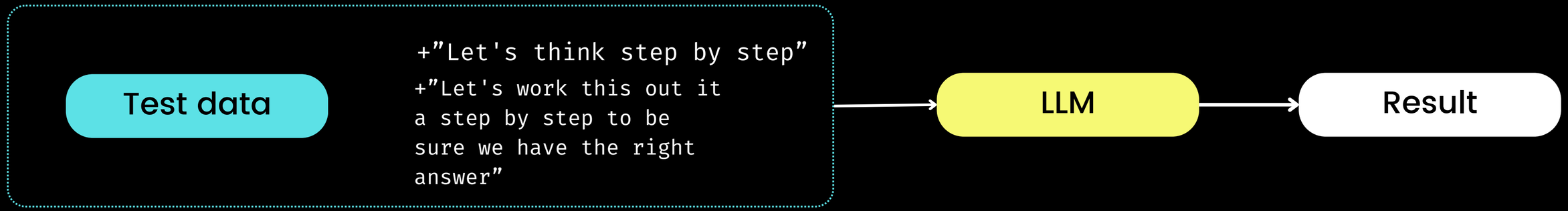
CHAIN-OF-THOUGHT (COT)

Chain of thoughts prompt

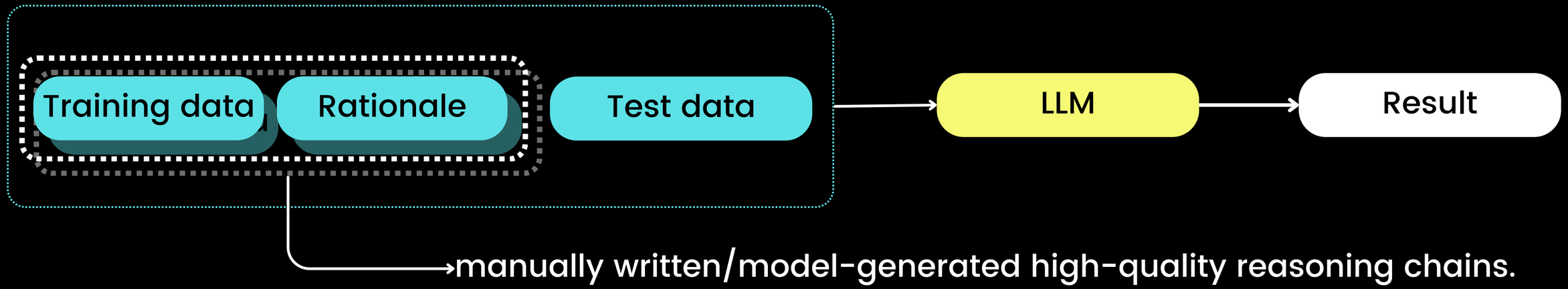
```
Question: Tom and Elizabeth have a competition to climb a hill. Elizabeth takes 30 minutes to climb the hill. Tom takes four times as long as Elizabeth does to climb the hill. How many hours does it take Tom to climb up the hill?
Answer: It takes Tom 30*4 = <<30*4=120>>120 minutes to climb the hill.
It takes Tom 120/60 = <<120/60=2>>2 hours to climb the hill.
So the answer is 2.
===
Question: Jack is a soccer player. He needs to buy two pairs of socks and a pair of soccer shoes. Each pair of socks cost $9.50, and the shoes cost $92. Jack has $40. How much more money does Jack need?
Answer: The total cost of two pairs of socks is $9.50 x 2 = $<<9.5*2=19>>19.
The total cost of the socks and the shoes is $19 + $92 = $<<19+92=111>>111.
Jack need $111 - $40 = $<<111-40=71>>71 more.
So the answer is 71.
===
Question: There are three birds on the tree, shot one down, how many are left on the tree?
Answer:
```

CHAIN-OF-THOUGHT (COT)

Zero-shot CoT

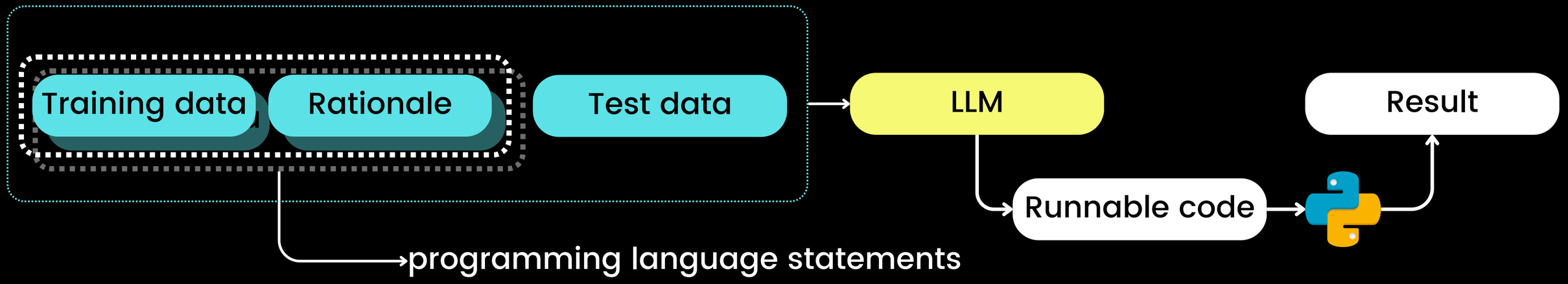


Few-shot CoT



PROGRAM-OF-THOUGHT (POT)

Few-shot PoT



Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

CoT

The first number is 0, the second number is 1, therefore, the third number is 0+1=1. The fourth number is 1+1=2. The fifth number is 1+2=3. The sixth number is 2+3=5. The seventh number is 3+5=8. The eighth number is 5+8=13.
..... (Skip 1000 tokens)
The 50th number is 32,432,268,459.

32,432,268,459 ✗

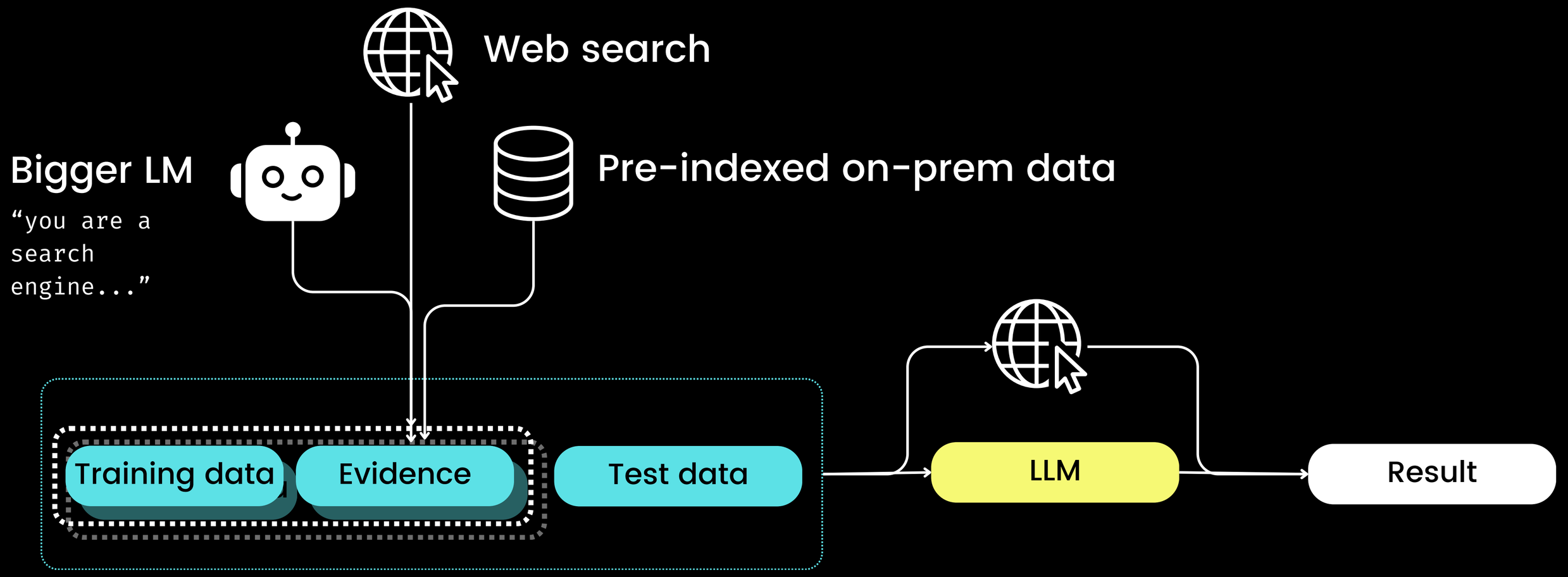
PoT

```
length_of_fibonacci_sequence = 50
fibonacci_sequence = np.zeros(length_of_)
fibonacci_sequence[0] = 0
fibonacci_sequence[1] = 1
For i in range(3, length_of_fibonacci_sequence):
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +
    fibonacci_sequence[i-2]
ans = fibonacci_sequence[-1]
```

python

12,586,269,025 ✓

RETRIEVER


Web search
Bigger LM
"you are a search engine..."
Pre-indexed on-prem data
Training data
Evidence
Test data
LLM
Result

READER

Convert any URL to an LLM-friendly input with a simple prefix https://r.jina.ai/

READER

Enter your URL
https://arxiv.org/abs/2310.19923

Click below to fetch the source code of the page directly

Reader URL
https://r.jina.ai/https://arxiv.org/abs/2310.19923

Stream Mode Click below to obtain the content through our Reader API

FETCH CONTENT

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>  <title>[2310.19923] Jina Embeddings 2: 8192-Token General-Purpose Text Embeddings for Lo
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="apple-touch-icon" sizes="180x180" href="/static/browse/0.3.4/images/icons/apple-tou
  <link rel="icon" type="image/png" sizes="32x32" href="/static/browse/0.3.4/images/icons/favico
  <link rel="icon" type="image/png" sizes="16x16" href="/static/browse/0.3.4/images/icons/favico
  <link rel="manifest" href="/static/browse/0.3.4/images/icons/site.webmanifest">
  <link rel="mask-icon" href="/static/browse/0.3.4/images/icons/safari-pinned-tab.svg" color="#5
  <meta name="msapplication-TileColor" content="#da532c">
  <meta name="theme-color" content="#ffffff">
  <link rel="stylesheet" type="text/css" media="screen" href="/static/browse/0.3.4/css/arXiv.cs
  <link rel="stylesheet" type="text/css" media="print" href="/static/browse/0.3.4/css/arXiv-prin
  <link rel="stylesheet" type="text/css" media="screen" href="/static/browse/0.3.4/css/browse_se
```

```
Title: Jina Embeddings 2: 8192-Token General-Purpose Text Embeddings for Long Documents

URL Source: https://arxiv.org/abs/2310.19923

Markdown Content:
Authors:[Michael Günther](https://arxiv.org/search/cs?searchtype=author&query=G%C3%BCnther,+M),

[View PDF](https://arxiv.org/pdf/2310.19923) [HTML (experimental)](https://arxiv.org/html/2310.1

> Abstract:Text embedding models have emerged as powerful tools for transforming sentences into
> To address these challenges, we introduce Jina Embeddings 2, an open-source text embedding mod

Submission history
-----------------

From: Han Xiao \[[view email](https://arxiv.org/show-email/11a0a836/2310.19923)\]
```

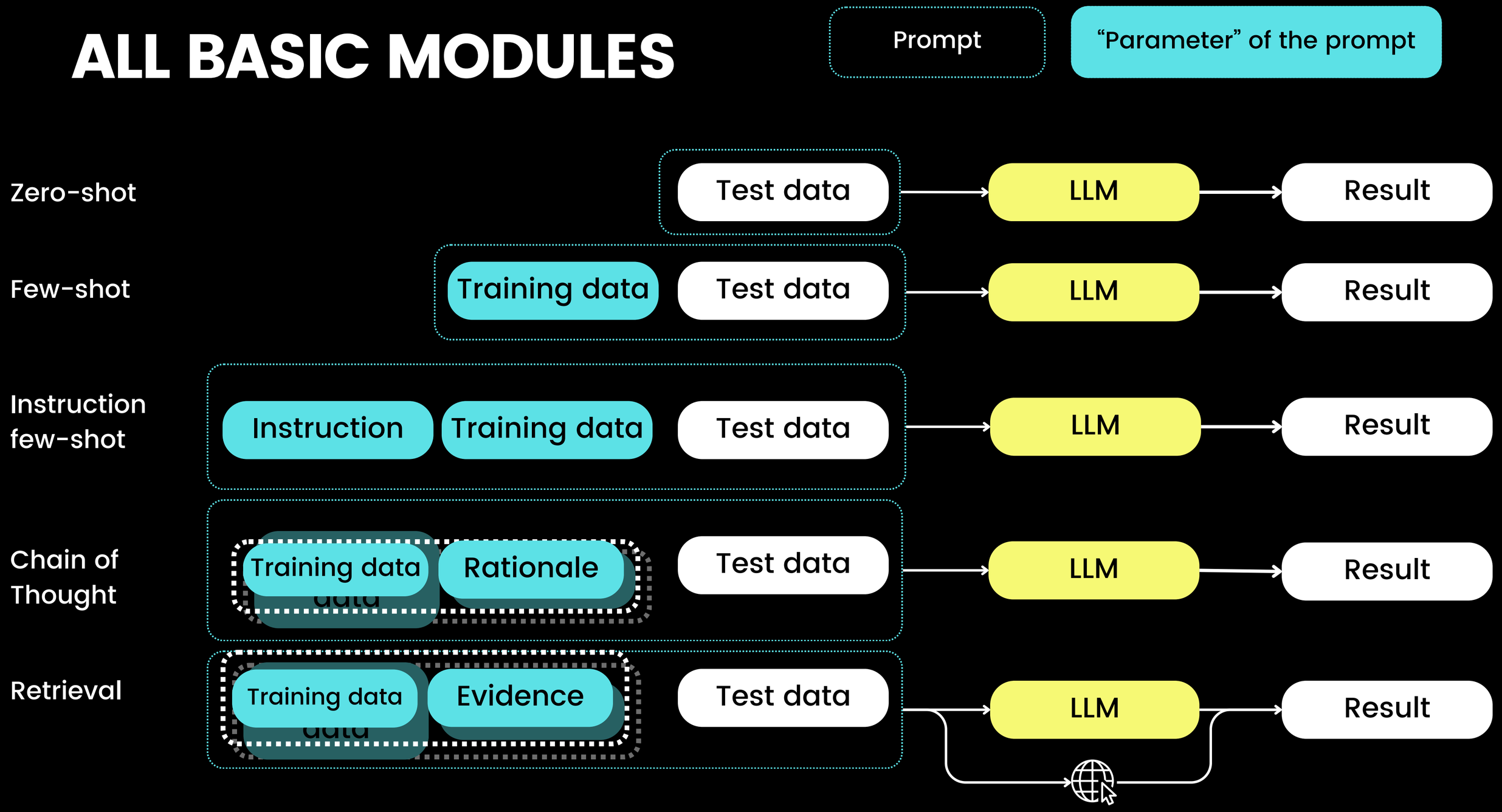
Pose a Question
Summarize the content in two sentences.

Input a question and combine it with the fetched content for LLM to generate an answer

The content discusses the challenges faced by existing open-source text embedding models in representing lengthy documents and introduces Jina Embeddings 2 as a solution to address these challenges.

The paper introduces Jina Embeddings 2, an open-source text embedding model that can handle up to 8192 tokens, addressing the limitation of existing models in representing long documents. The model achieves state-of-the-art performance on various embedding-related tasks and matches the performance of OpenAI's ada-002 model, with experiments showing that extended context improves performance in tasks like NarrativeQA.

ALL BASIC MODULES



DSPy:
Not Your Average Prompt Engineering

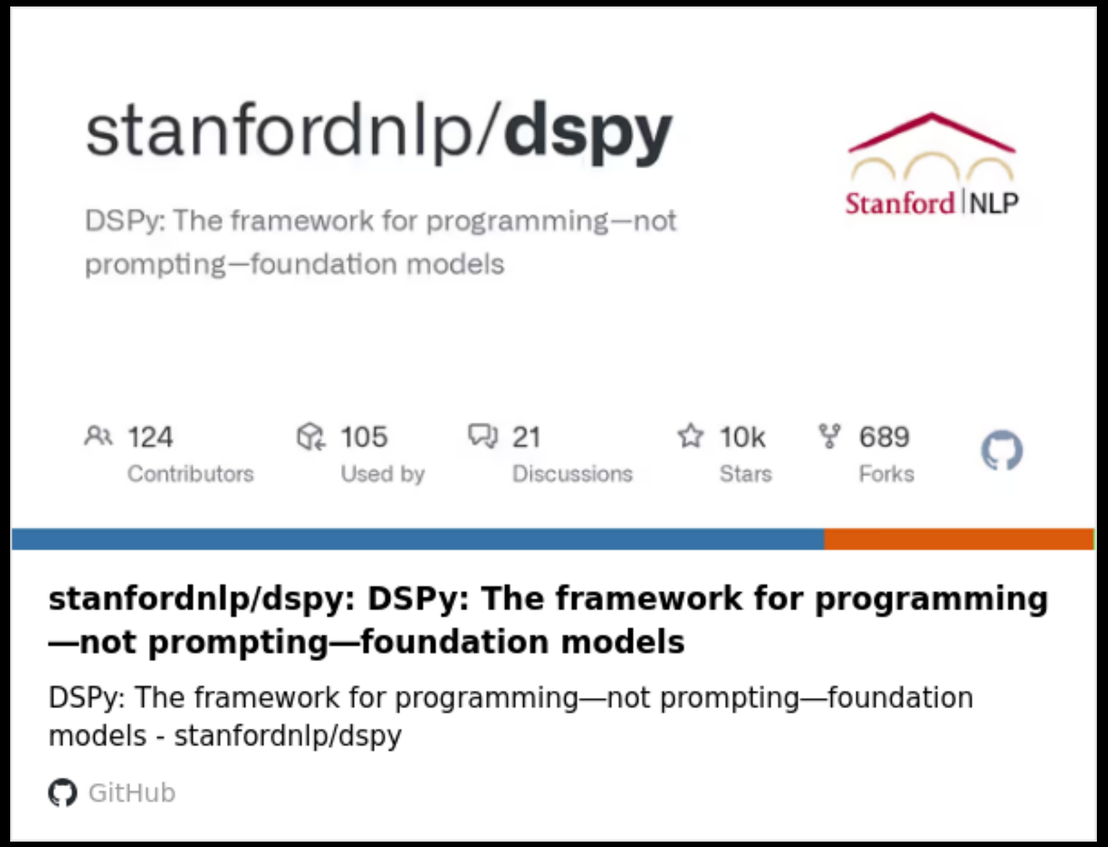
What is DSPy

DSPY

- **D**eclarative **S**elf-improving Language **P**rograms, pythonically.

- DSPy is a framework for algorithmically optimizing prompts and LM weights, especially in a prompt pipeline.

- However, it is hard to learn.
  - *"Yeah man, I have been seeing DSPy everywhere but haven't found time to check it out yet"* - almost everyone I talk to about the project.



UNDERSTANDING DSPY

- DSPy closes the loop of prompt engineering;
- DSPy separates the logic (what) from textual representation (how).

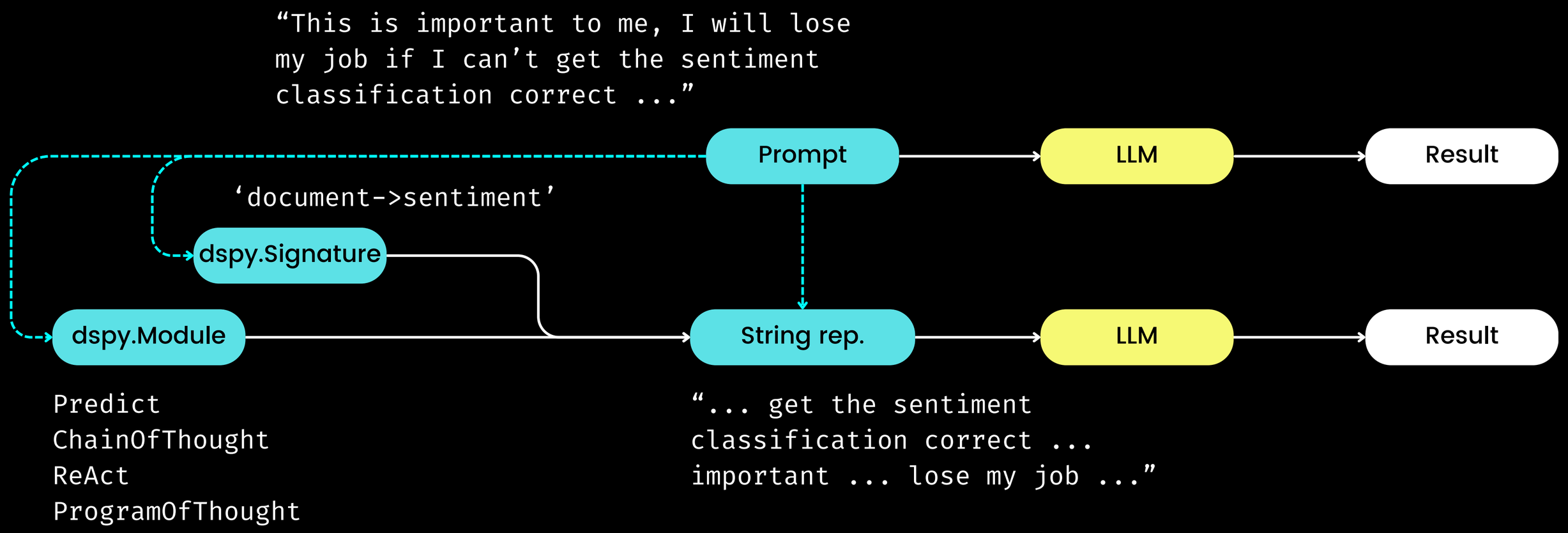
UNDERSTANDING DSPY

- DSPy closes the loop of prompt engineering;

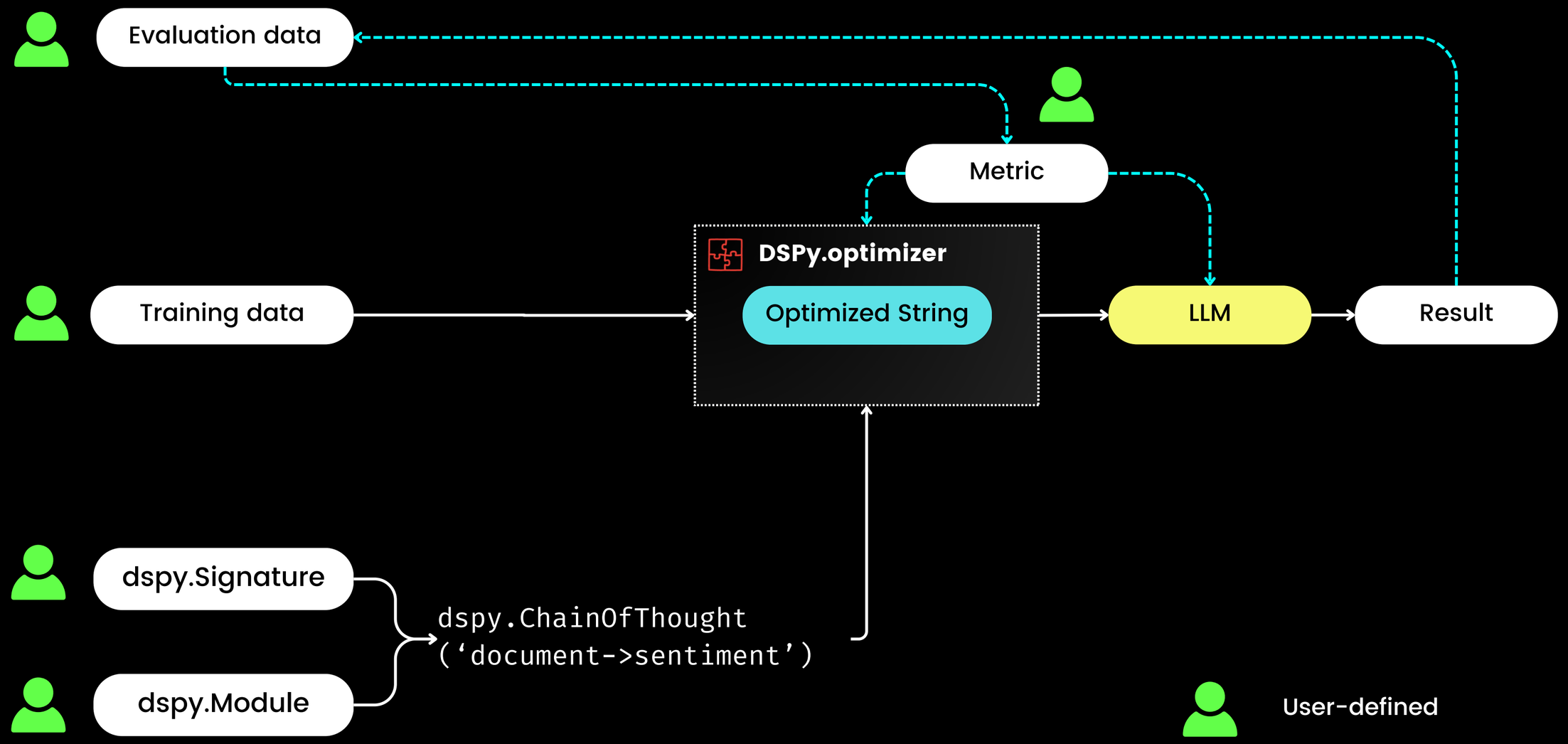
Transforming prompt engineering from what is often a *manual, handcrafted process* into a *structured, well-defined machine learning workflow*: i.e. preparing datasets, defining the model, training, evaluating, and testing. In my opinion, this is the most revolutionary aspect of DSPy.

UNDERSTANDING DSPY

- DSPy separates the logic (what) from textual representation (how).



DSPY.OPTIMIZER.COMPILE



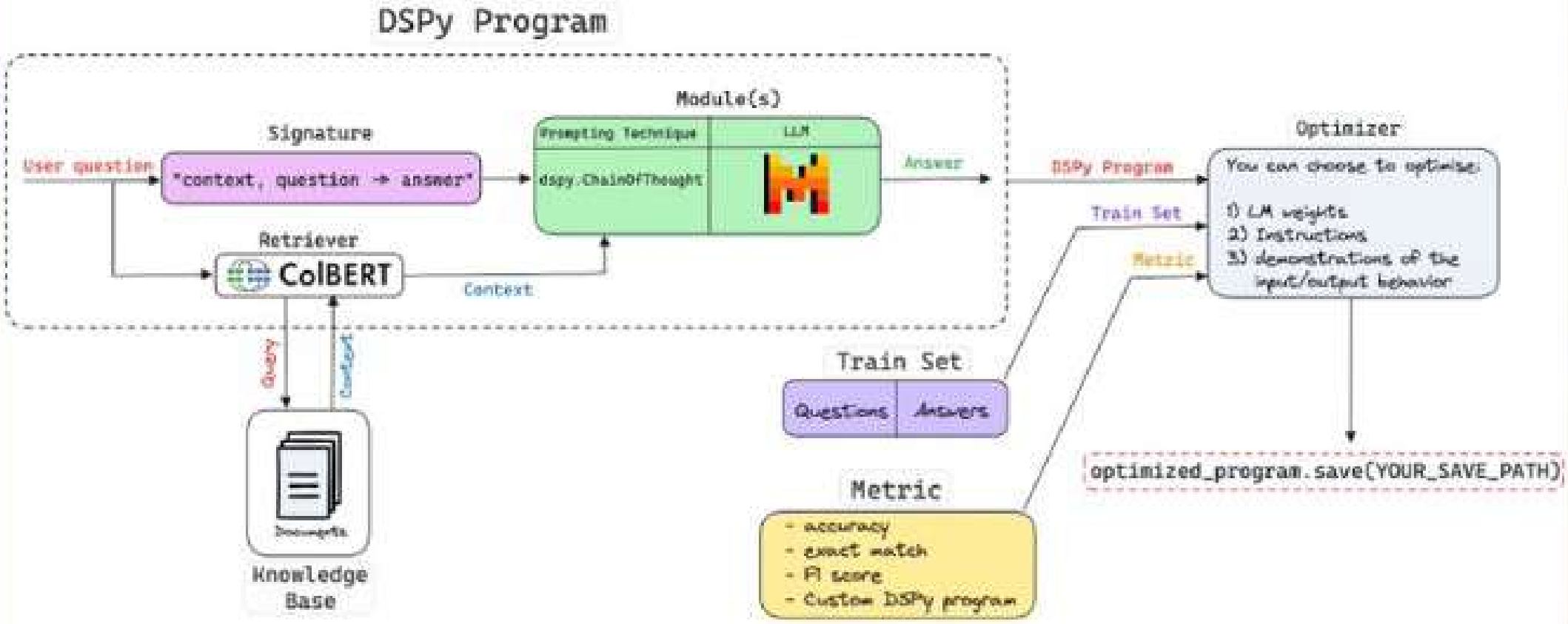
WHAT EXACTLY DSPY.COMPILE OPTIMIZE

The compile function acts at the heart of this optimizer, akin to calling optimizer.optimize(). Think of it as the DSPy equivalent of training. This compile() process aims to tune:

- the few-shot demonstrations
- the instructions
- the LLM weights

You can imagine DSPy as a toolbox of **discrete optimization methods.**

DSPy: Programming not prompting LMs



DSPy:
Not Your Average Prompt Engineering

Demo