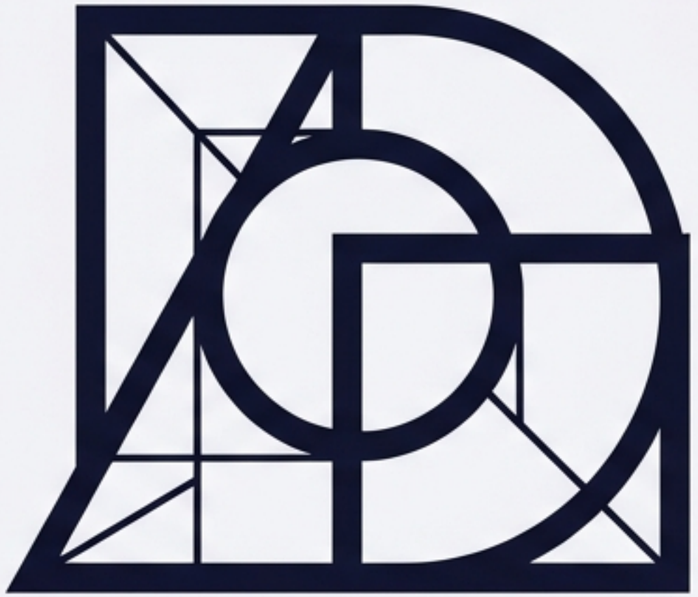
Move beyond manual string manipulation.
JetBrains Mono



Iterate fast.
Optimize weights.
Build modular systems.

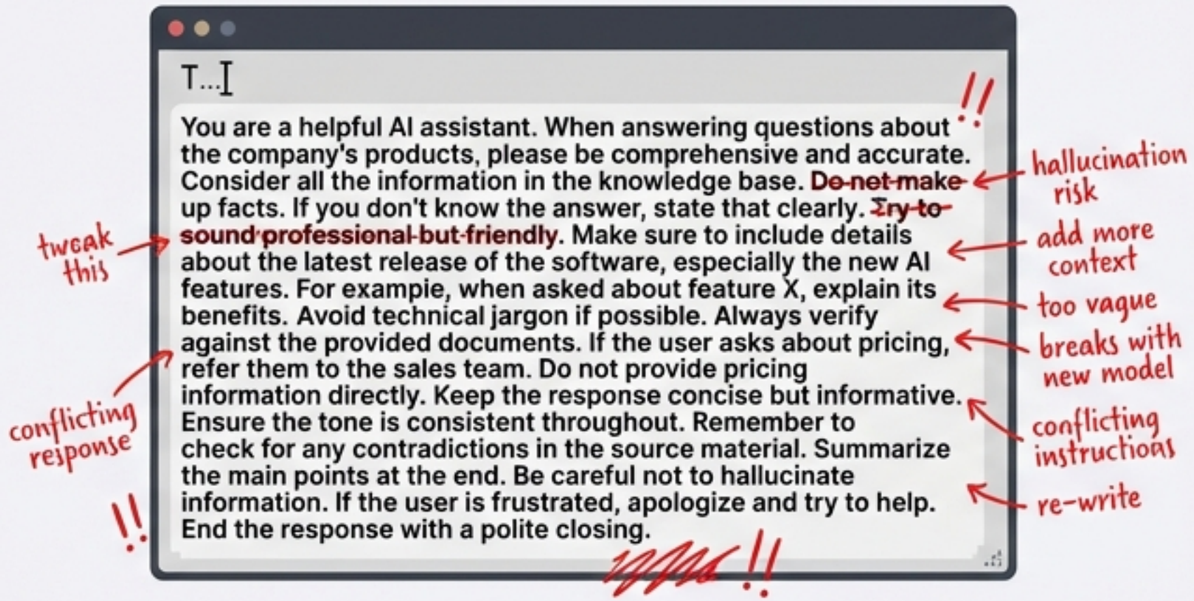
Programming—Not Prompting—Foundation Models

DSPy: The framework for building modular, self-improving AI systems.

A StanfordNLP Project

Manual prompt engineering is brittle, unscalable, and imprecise.

The Problem: Prompt Engineering



Fragile strings that break when models change.

The Solution: Modular Programming

```python
import dspy

class Summarize(dspy.Signature):
    """Summarizes the provided text concisely."""

    text: str = dspy.InputField(desc="Text to summarize")
    summary: str = dspy.OutputField(desc="Concise summary")

# Compile the signature into an optimized program
summarize_program = dspy.ChainOfThought(Summarize)

def generate_summary(input_text: str) -> str:
    """Generates a summary using the compiled program."""
    response = summarize_program(text=input_text)
    return response.summary
```

Typed, optimized, and compiled architecture.

The Reality: Developers spend hours manually tweaking text rather than engineering robust systems.

DSPy replaces string manipulation with modular Python code.

DSPy = Declarative Self-improving Python

A framework for **programming**—rather than prompting—language models. Instead of writing paragraphs of text, you write compositional Python code. DSPy uses this code to teach your LM to deliver high-quality outputs.

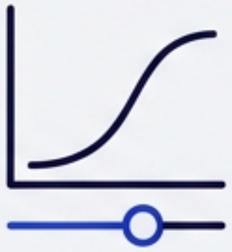
```python
class GenerateAnswer(dspy.Signature):
    """Answer questions with short factoid answers."""

    context = dspy.InputField(desc="may contain relevant facts")
    question = dspy.InputField()
    answer = dspy.OutputField(desc="often between 1 and 5 words")
```

Treat prompts and weights as optimizable hyperparameters.

Build Modular AI Systems

Move away from monolithic prompt blocks. Compose your application using reusable, logical modules.



Optimize Prompts & Weights

DSPy offers algorithms for automatically optimizing the underlying instructions and model weights to maximize a metric.

Iterate Fast

Change your logic without rewriting your prompts. The framework handles the adaptation for you.

Modern Engineering Manifesto

Compiling declarative calls into self-improving pipelines

DSPy acts as a compiler for LLM calls. It systematically optimizes the prompt strategy (demonstrations and instructions) based on the data you provide, eliminating manual “prompt hacking”.

From simple classifiers to sophisticated RAG pipelines.

Simple Classifiers

Automate basic categorization tasks with higher accuracy.

RAG Pipelines

Build Retrieval-Augmented Generation systems that are modular and easy to debug.

Agent Loops

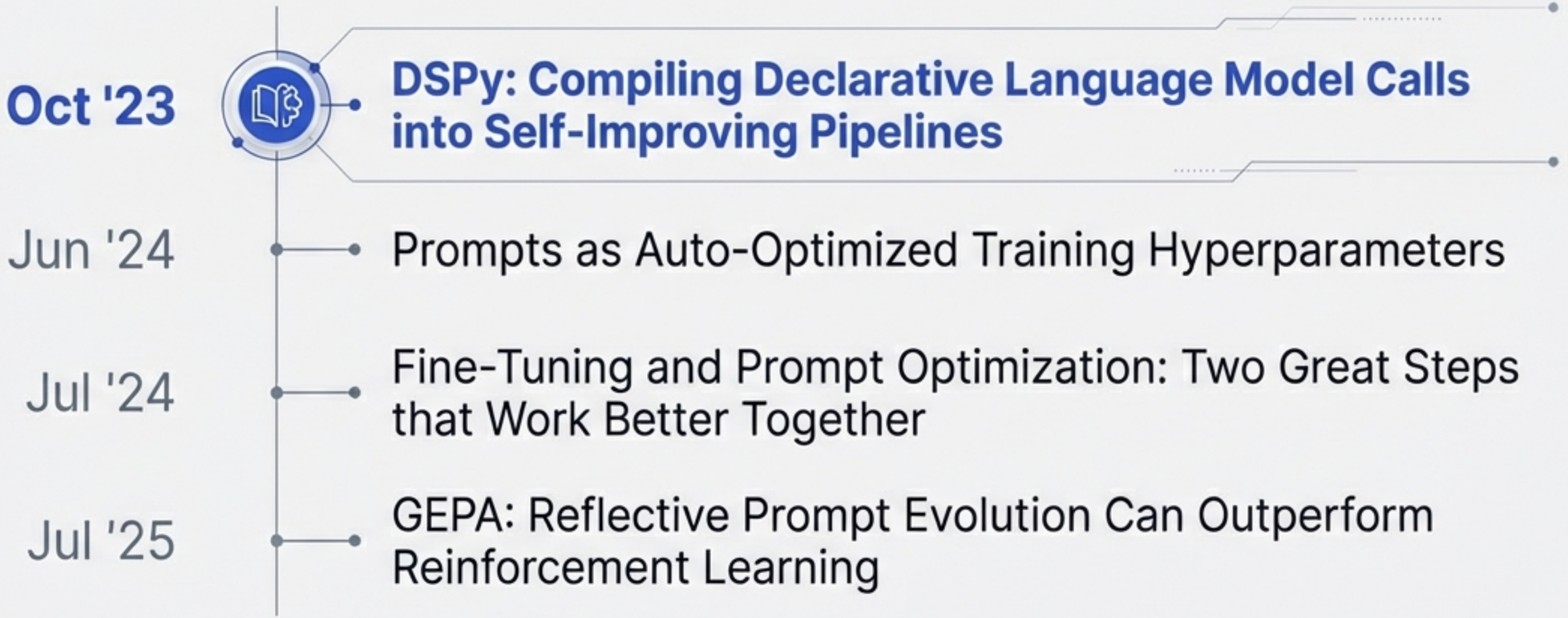
Construct complex workflows where the LM improves its own behavior over time.

“Whether you’re building simple classifiers, sophisticated RAG pipelines, or Agent loops.”

Modern Engineering Manifesto

Built on rigorous research from StanfordNLP.

Developed by Omar Khattab and the StanfordNLP team.



Primary Citation: Khattab et al., ICLR 2024.

An open-source standard with massive developer traction.

32.1k
GitHub Stars

2.6k
Forks

1.4k
Projects using DSPy

99.2%
Python Codebase

License: MIT License (Open & Permissive). 350+ Contributors.

Start building modular AI systems today.

```
# Standard Installation
pip install dspy

# Install from Main Branch
pip install git+https://github.com/stanfordnlp/dspy.git
```

Next Steps

- → Read the docs: **dspy.ai**
- → Join the community: Discord & GitHub Issues.
- → Follow updates: @DSPyOSS on Twitter / LinkedIn.

Stop Prompting. Start Programming.

DSPy offers the algorithms to optimize your AI systems so you can focus on the architecture, not the wording.

github.com/stanfordnlp/dspy
dspy.ai