DSPy Introduction

- Enables to build language model applications using modular and declarative programming.
- Shifts the focus from traditional prompt engineering to a structured approach, allowing for the creation of robust and scalable AI-powered solutions.
- Simpler and requires less prompt engineering skills than LangChain
- Has many built in capabilities to ease the development of AI applications

DSPy Main components

Core Elements:

1. **Signatures**: Define the interface for LLM tasks
2. **Modules**: Implement reusable components with specific functionality
3. **Optimization**: Fine-tune prompts automatically based on examples
4. **Evaluation**: Measure and track system performance

NUFAR GASPAR | ng@ngai.ai | +972-54-7884031 |

Signatures

Signatures

- Define the **input and output fields** for LLM tasks
- Act as a **contract** between components
- Example: `question -> answer` or `context, question -> reasoning, answer`

```python
class QASignature(dspy.Signature):
    """Answer questions accurately based on the question."""
    question = dspy.InputField()
    answer = dspy.OutputField()
```

Modules

Modules

- **Reusable components** that implement signatures
- Can be composed into larger systems
- Examples: `Predict`, `ChainOfThought`, custom modules

```python
class SimpleQA(dspy.Module):
    def __init__(self):
        self.qa = dspy.Predict(QASignature)

    def forward(self, question):
        return self.qa(question=question)
```

Optimization & Evaluation

Optimization

- Automatically **improve prompts** using example data
- Examples: `BootstrapFewShot`, `PromptOptimizer`
- Teaches models how to approach tasks with minimal examples

Evaluation

- **Measure performance** of DSPy programs
- Create custom metrics for specific tasks
- Iterate on design based on evaluation results

Pre-Requisites to be able to work on DSPy

- Python 3.8+
- Required packages: dspy, datasets, pandas, numpy, openai
- API key for OpenAI (in the code)
- Jupyter notebook environment/ other Python-friendly env.

Example code with exercises, Implementations and examples

- This code guides you through the core concepts of DSPy, building progressively from basic principles to implementing their own RAG system.
- **Learning Objectives**
  - Understand DSPy's core abstractions: Signatures, Modules, and Programs
  - Learn how to use language models effectively through DSPy
  - Implement and optimize retrieval-augmented generation systems
  - Evaluate and iteratively improve prompt programs
  - Create a complete RAG application from scratch
- Use the given code and augment to assimilate learning, continue the work in HW

DSPy Materials to dig dipper

- 35 min tutorial vides
- 1 hr intro video
- DSPy on Github
- IBM Watson tutorial
- More complex RAG with DSPy tutorial
- Simple DSPy intro on Kaggle
- RAG and Vector DB training