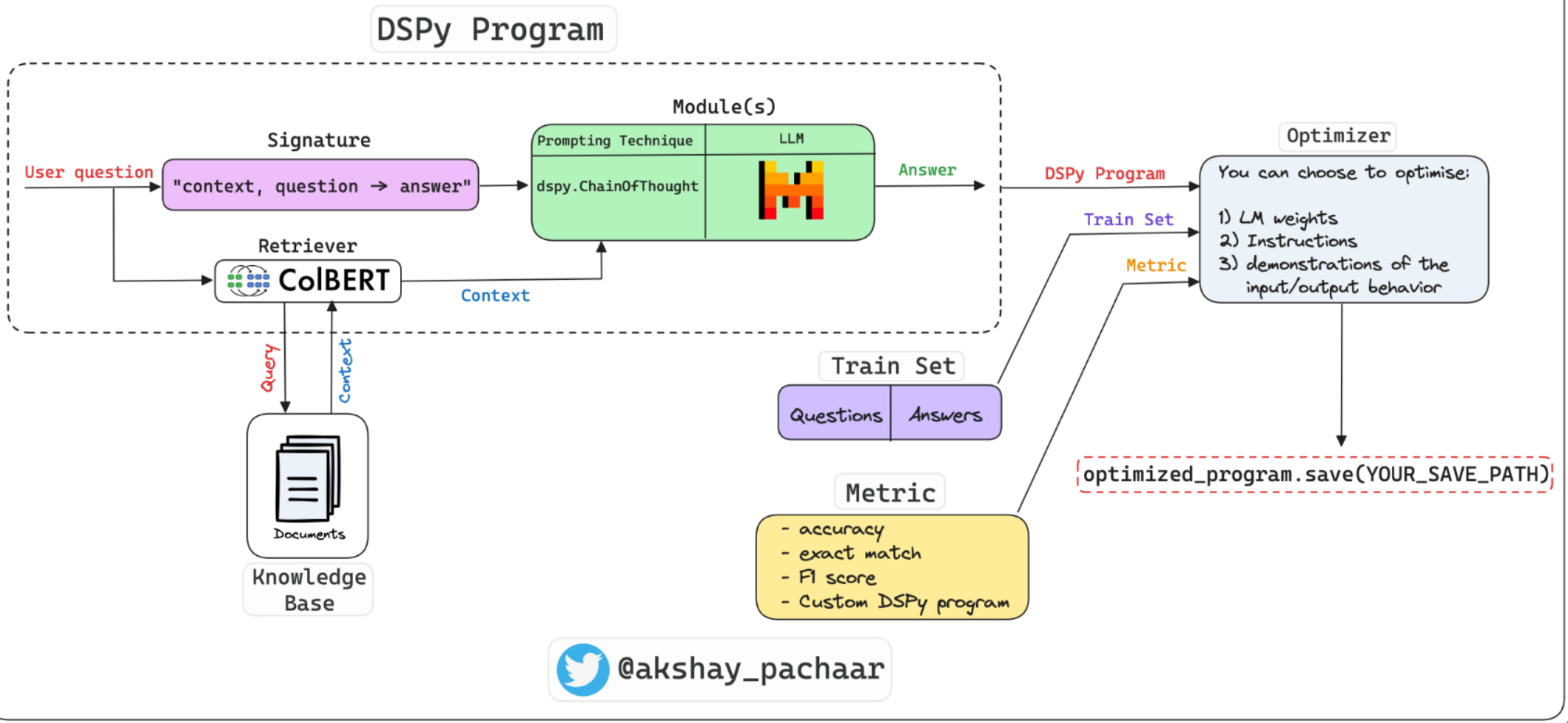
DSPy: Programming not prompting LMs



1️⃣ Signature

We use a signature to tell DSPy what to do instead of how to do; they save us from writing gigantic & complex prompts.

DSPy supports inline short strings as signatures; however, you can always write a class for it if necessary!

Swipe 👉

@akshay_pachaar

DSPy: Signatures

Defined as a short string, with argument names that define semantic roles for inputs/outputs:

- Question Answering: "question → answer"
- Summarization: "document → summary"
- Sentiment Classification: "sentence → sentiment"
- RAG Question Answering: "context, question → answer"
- MCQs with Reasoning: "question, choices → reasoning, selection"

@akshay_pachaar

2. Module

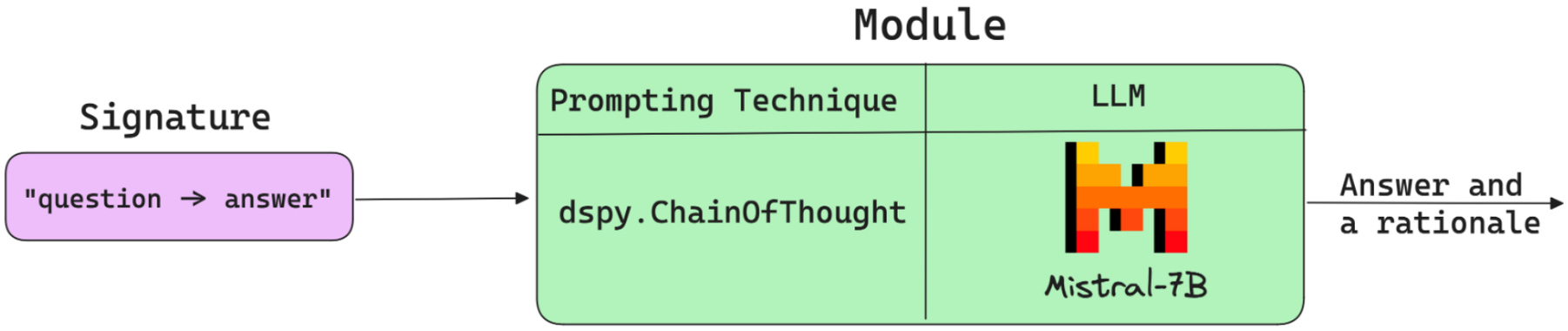
A module takes a signature & expresses it in form of a sophisticated prompt or whatever your signature specifies, based on the specified technique & the LLM configured.

It can be thought of as a parametrised layer in PyTorch that can learn from the data (input/output)

Swipe 👉

@akshay_pachaar

DSPy: Modules




```python
question = 'What is the nationality of the chef and restaurateur featured in Restaurant: Impossible?'

# define a CoT predictor & pass in the signature
generate_answer_with_chain_of_thought = dspy.ChainOfThought("question → answer")

# Now pass in the predictor your question
pred = generate_answer_with_chain_of_thought(question=question)

# Print the question, the chain of thought, and the prediction.
print(f"Question: {question}")
print(f"Thought: {pred.rationale.split('.', 1)[1].strip()}")
print(f"Predicted Answer: {pred.answer}")

# Outputs
# Question: What is the nationality of the chef and restaurateur featured in Restaurant: Impossible?
# Thought: We know that the chef and restaurateur featured in Restaurant: Impossible is Robert Irvine.
# Predicted Answer: British
```

@akshay_pachaar

Now, before delving into the concept of an optimizer, let's define a DSPy program that we want to optimise.

Here's a basic DSPy RAG program & it's similar to PyTorch where we have an __init__() function for module definitions and a forward function for module interactions.

Swipe 👉

@akshay_pachaar

```python
import dspy

# setup lm & retriever model
turbo = dspy.OpenAI(model='gpt-3.5-turbo')
colbertv2_wiki17_abstracts = dspy.ColBERTv2(url='http://wiki17_abstracts')

dspy.settings.configure(lm=turbo, rm=colbertv2_wiki17_abstracts)

class RAG(dspy.Module):

    def __init__(self, num_passages=3):
        super().__init__()

        self.retrieve = dspy.Retrieve(k=num_passages)
        self.generate_answer = dspy.ChainOfThought(GenerateAnswer)

    def forward(self, question):
        context = self.retrieve(question).passages
        prediction = self.generate_answer(context=context, question=question)
        return dspy.Prediction(context=context, answer=prediction.answer)
```

3️⃣ Optimizer

A DSPy optimizer can optimize 3 things:

- LM Weights
- Instructions (prompts/signatures)
- Demonstrations of the input output behaviour

Here's how we use a train set & custom metric
to optimize the RAG program we defined earlier:

Swipe 👉

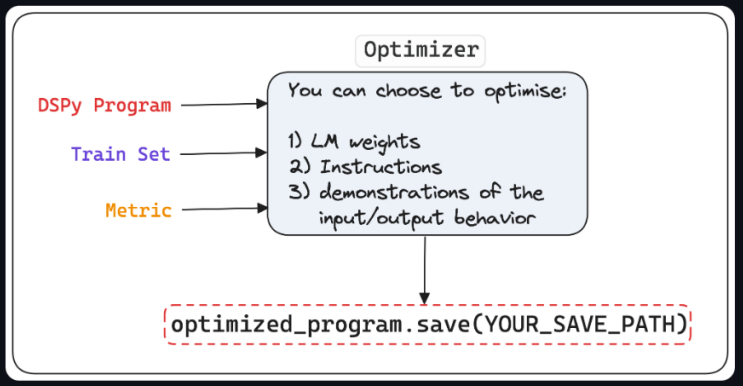
@akshay_pachaar

DSPy: Optimizer

```python
from dspy.teleprompt import BootstrapFewShot


# define a train set with 5-10 examples of your choice
trainset = [('In what year was the star of To Hell and Back born?', '1925'),
        ('Which author is English: John Braine or Studs Terkel?', 'John Braine')]

trainset = [dspy.Example(question=question, answer=answer).with_inputs('question') for question, answer in train]

# Validation logic: check that the predicted answer is correct.
# Also check that the retrieved context does actually contain that answer.
def validate_context_and_answer(example, pred, trace=None):
    answer_EM = dspy.evaluate.answer_exact_match(example, pred)
    answer_PM = dspy.evaluate.answer_passage_match(example, pred)
    return answer_EM and answer_PM

# Set up a basic optimizer, which will optimize/compile our RAG program.
optimizer = BootstrapFewShot(metric=validate_context_and_answer)

# Optimize/Compile!
compiled_rag = optimizer.compile(RAG(), trainset=trainset)
```

Automatically generates a few shot examples(demonstrations) based on trainset for each stage/module & can further optimise them based on validation logic if necessary.

DSPy Program



Automatic Evaluations ✨

Now that we have optimised our program, DSPy offers automatic evaluation where you need to provide:

- A dev set of questions & expected answers
- An evaluation metric (supports predefined & custom metric)

Swipe 👉

@akshay_pachaar

DSPy: Automatic Evaluation

```
# Gather a few examples for eval set, more the merrier.
devset = [
       ('Kyle Moran was born in the town on what river?', 'Castletown River'),
       ('What year was the father of the Princes in the Tower born?', '1442'),
       ('What river is near the Crichton Collegiate Church?', 'the River Tyne'),
       )]

devset = [dspy.Example(question=question, answer=answer).with_inputs('question')
          for question, answer in dev]

evaluate = Evaluate(devset=devset, metric=validate_context_and_answer)

# Displays the eval results for unoptimized RAG
evaluate(RAG())

# Displays the eval results foroptimized RAG
evaluate(cot_compiled)
```

Defined earlier , last tweet

Note: We've shared all the code & results in the studio below!

That's a wrap!

If you interested in:

- Python 🐍
- Data Science 📈
- Machine Learning 🤖
- MLOps 🛠️
- NLP 🗣️
- Computer Vision 📽️
- LLMs 🧠

Follow me on LinkedIn ✔️
Everyday, I share tutorials on above topics!

Cheers!! 🙂

@akshay_pachaar