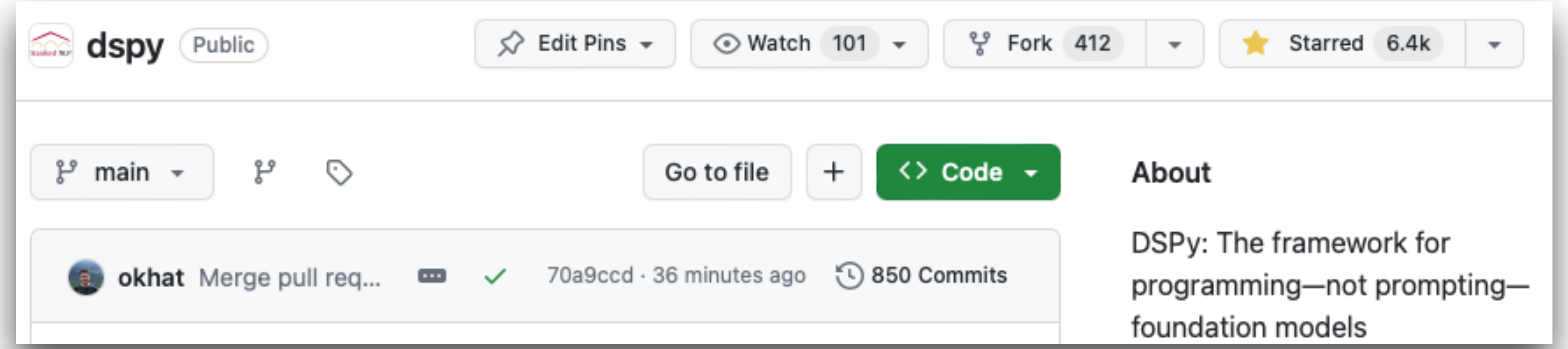
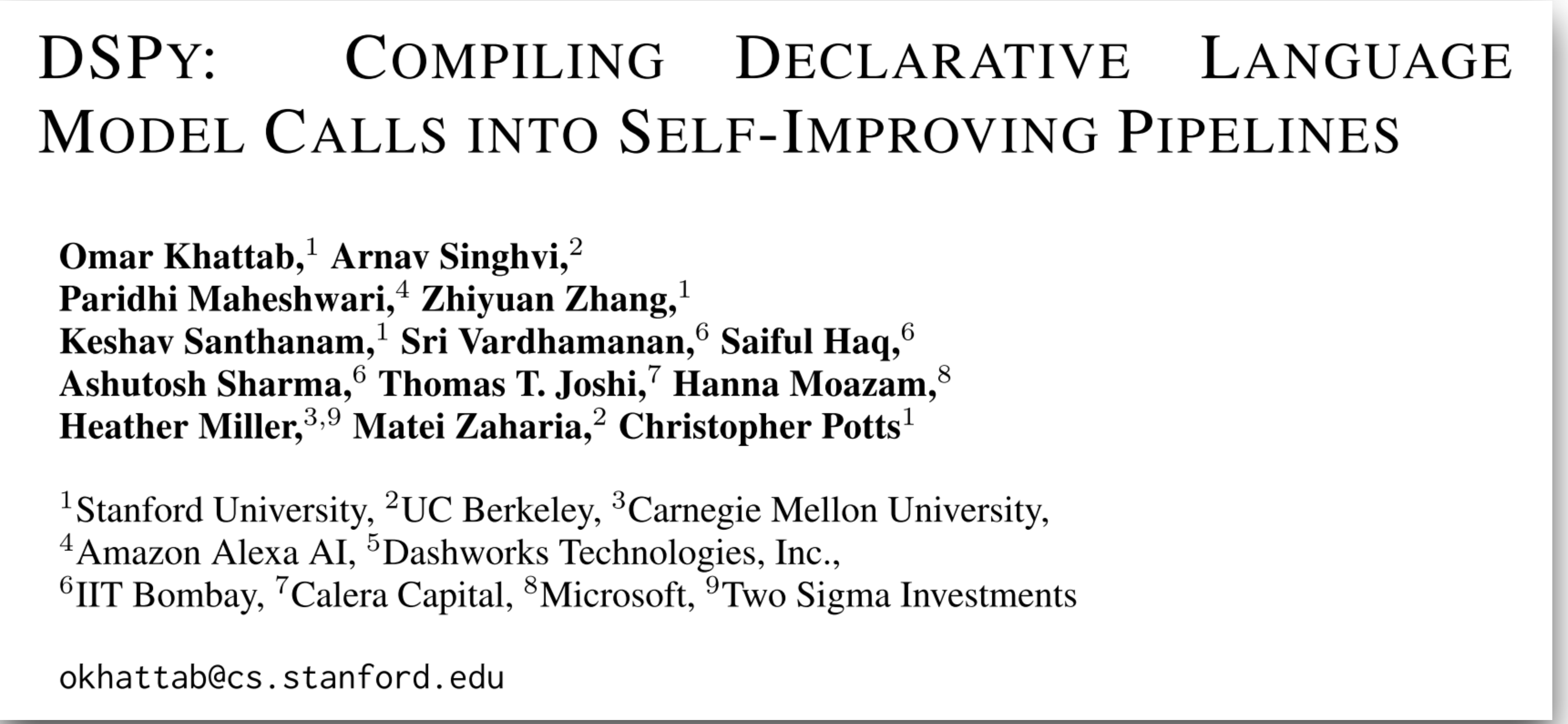
# DSPy overview

## Programming—not prompting —Foundation Models



DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines


**Omar Khattab,[1] Arnav Singhvi,[2]**
**Paridhi Maheshwari,[4] Zhiyuan Zhang,[1]**
**Keshav Santhanam,[1] Sri Vardhamanan,[6] Saiful Haq,[6]**
**Ashutosh Sharma,[6] Thomas T. Joshi,[7] Hanna Moazam,[8]**
**Heather Miller,[3,9] Matei Zaharia,[2] Christopher Potts[1]**

[1]Stanford University, [2]UC Berkeley, [3]Carnegie Mellon University,
[4]Amazon Alexa AI, [5]Dashworks Technologies, Inc.,
[6]IIT Bombay, [7]Calera Capital, [8]Microsoft, [9]Two Sigma Investments

okhattab@cs.stanford.edu


Insop, 9/9/2024

# Agenda

Motivation

DSPy overview

Resources

# Motivation

# LM usage

- Zero-shot: ask a question and LM answers.
- Few-shot: ask a question and provide examples, then LM answers. This is also called in-context learning.
- Retrieval-augmented generation (RAG): retrieve relevant contexts first, then use the contexts as part of the query.

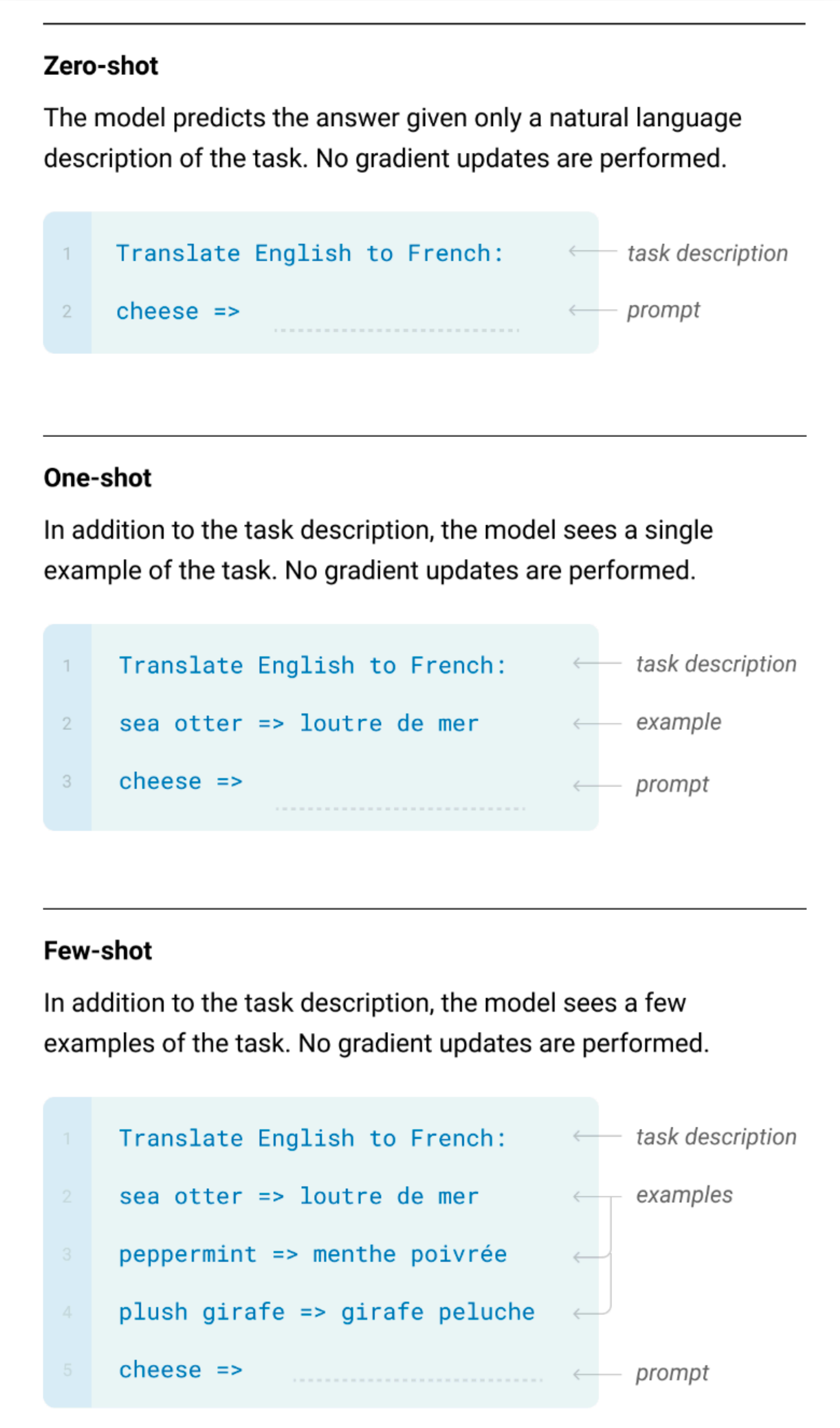



Image source: T. Brown et. al, Language Models are Few-Shot Learners

# Development with LLMs

- Test prompts using the playground.
- Put together prompts and use the API to generate text.
- As the prompt and parsing get more complicated, use LLM orchestration software, such as Langchain or llamaindex.
- These tools allow you to perform advanced operations and chain operations, but they also hide details, so they are hard to debug.
- When the models are updated, you need to update your prompt.

- Links: Langchain, llamaindex

# Why DSPy?

- DSPy introduces a small set of much more powerful and *general-purpose modules* that can learn to prompt your LM within your pipeline on your data.
- When you *change your data*, make tweaks to your program's control flow, or change your target LM, DSPy compiler can map your program into a new set of prompts that are optimized specifically for this pipeline.
- In short, DSPy is for when you need a *lightweight* but automatically-optimizing programming model — not a library of predefined prompts and integrations.

Text from: https://dspy-docs.vercel.app/docs/faqs

# Overview

# LM usage

- LM can be defined and used.
- Multiple LMs can be set.
    - OpenAI APIs.
    - OSS LMs using APIs from together, Anyscale.
    - OSS LMs from locally hosted server.
    - OSS LMs running on a local machine.

```python
# OpenAI
model_name = "gpt-3.5-turbo"
lm = dspy.OpenAI(model=model_name)

# Azure OpenAI
model_arg = {"engine": model_name, "deployment_id": model_name,
             "api_version": openai.api_version,
             "api_base": openai.api_base,
}
provider_name = "azure"
lm = dspy.OpenAI(model=model_name,
                    api_key=openai.api_key,
                    api_provider=provider_name,
                    **model_arg)

# Together.ai
model="mistralai/Mistral-7B-v0.1"
lm = dspy.Together(model=model)

# local model using Ollama
lm = dspy.OllamaLocal(model='mistral')

# OR use recent Ollama's OpenAI API support
lm = dspy.OpenAI(api_base='http://localhost:11434/v1/',
    api_key='anything',
    model='mistral:7b-instruct-v0.2-q6_K',
    stop='\n\n',
    model_type='chat')
```

Link for Multi LM: https://github.com/stanfordnlp/dspy/tree/main/docs
ollama openai API example: https://twitter.com/JR_Knox1977/status/1756018720818794916

# LM example

- Simple text request
- Request text with configurations
  - temperature: high temp. generates creative output
  - n: number of calls
- inspect_history: useful to check the text request and output
  - n: number of history
  - white text: prompt sent to LM
  - green text: text generated by LM

```python
lm("Which award did Gary Zukav's first book receive?")
```

✓ 0.8s Python

```
['Gary Zukav\'s first book, "The Dancing Wu Li Masters: An Overview of the N
```

```python
lm("Which U.S. states border no U.S. states?", temperature=0.9, n=4)
```

✓ 1.3s Python

```
['There are two U.S. states that do not share borders with any other U.S. st
 "There are two U.S. states that do not border any other U.S. states:\n\n1.
 'There are two U.S. states that do not border any other U.S. states:\n\n1.
 'There are two U.S. states that do not border any other U.S. states. They a
```

```python
lm.inspect_history(n=1)
```

✓ 0.0s Python

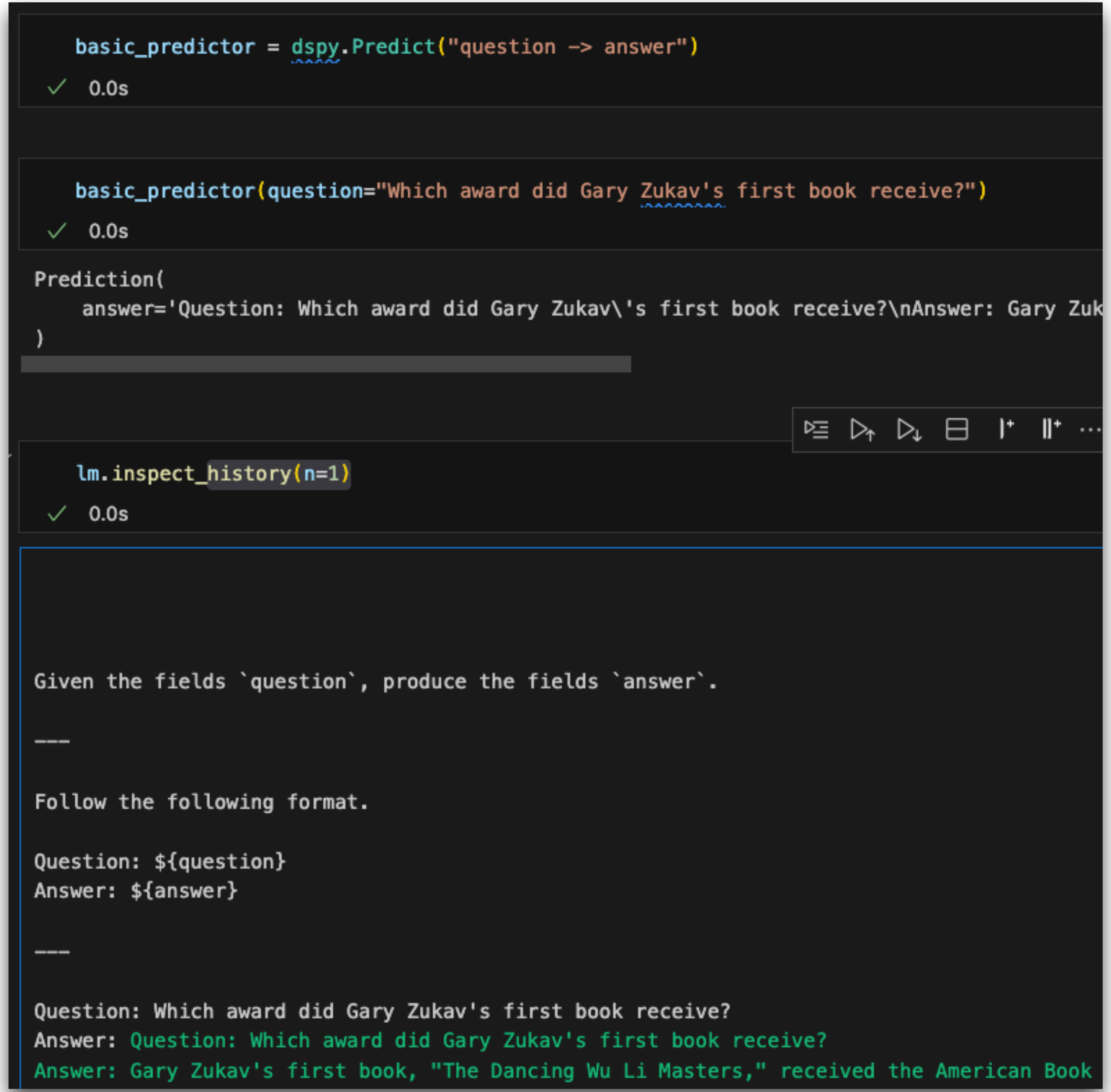
```
Which U.S. states border no U.S. states? There are two U.S. states that do n

1. Alaska: Located in the northwest extremity of the North American continen

2. Hawaii: Located in the central Pacific Ocean, Hawaii consists of a group
```

Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# Signature

## Basics

- Declarative statements that *what* we want to the model to do.
- “question -> answer” signature and dspy.Predict turns this into a QA system




Text from https://dspy-docs.vercel.app/docs/building-blocks/signatures ,
Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# Signature example

- Sentiment analysis: “sentence -> sentiment”
- In-line signature examples:
  - Summarization: "document -> summary"
  - Retrieval-Augmented Question Answering: "context, question -> answer"
  - Multiple-Choice Question Answering with Reasoning: "question, choices -> reasoning, selection”

```python
sentence = "it's a charming and often affecting journey."

classify = dspy.Predict('sentence -> sentiment')
classify(sentence=sentence).sentiment
```

✓ 0.6s

```
'Sentiment: positive'
```

```python
lm.inspect_history(n=1)
```

✓ 0.0s

```
Given the fields `sentence`, produce the fields `sentiment`.

---

Follow the following format.

Sentence: ${sentence}
Sentiment: ${sentiment}

---

Sentence: it's a charming and often affecting journey.
Sentiment: Sentiment: positive
```

Example from https://dspy-docs.vercel.app/docs/building-blocks/signatures ,

# Signature

## Use dspy.Signature class

- We can add more descriptions and tweak the initial instruction using dspy.Signature class
- Added description of the task using __doc__ = """Answer questions with short factoid answers."""
- Specify output to be "often between 1 and 5 words"
- Notice that the answer is 1-5 words as specified in signature, this was different when we did not specify outputField signature (see previous slide)

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

    question = dspy.InputField()
    answer = dspy.OutputField(desc="often between 1 and 5 words")

sig_predictor = dspy.Predict(BasicQASignature)

sig_predictor(question="Which U.S. states border no U.S. states?")
```

✓ 0.6s

```
Prediction(
    answer='Alaska, Hawaii'
)
```

```
Answer questions with short factoid answers.

---

Follow the following format.

Question: ${question}
Answer: often between 1 and 5 words

---

Question: Which U.S. states border no U.S. states?
Answer: Alaska, Hawaii
```

Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# Review PyTorch

- Since DSPy adapts design patterns from PyTorch, let's review the PyTorch pattern using a simple example
- __init__(): defines the nodes of the computational graph
- forward(): executes the graphs, i.e. forward pass

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Example from https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html

# Modules

- The DSPy module follows the pytorch design pattern. This allows DSPy to compose multiple tasks.
- Notice that the answer is 1-5 words, as specified in the signature. This was different when we did not specify the outputField signature (see previous slide).

```python
class BasicQA(dspy.Module):
    def __init__(self):
        super().__init__()
        self.generate_answer = dspy.Predict(BasicQASignature)

    def forward(self, question):
        return self.generate_answer(question=question)

basic_qa_model = BasicQA()

basic_qa_model(question="Which award did Gary Zukav's first book receive?")
```

✓ 0.6s

```
Prediction(
    answer='Oprah'
)
```

```
Answer questions with short factoid answers.

---

Follow the following format.

Question: ${question}
Answer: often between 1 and 5 words

---

Question: Which award did Gary Zukav's first book receive?
Answer: Oprah
```

Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# Modules

## Built-in modules

- dspy.Predict: Basic predictor. Does not modify the signature. Handles the key forms of learning (i.e., storing the instructions and demonstrations and updates to the LM).
- dspy.ChainOfThought: Teaches the LM to think step-by-step before committing to the signature's response.
- dspy.ProgramOfThought: Teaches the LM to output code, whose execution results will dictate the response.
- dspy.ReAct: An agent that can use tools to implement the given signature.
- dspy.MultiChainComparison: Can compare multiple outputs from ChainOfThought to produce a final prediction.
- dspy.majority: Can do basic voting to return the most popular response from a set of predictions.

Text from https://dspy-docs.vercel.app/docs/building-blocks/modules

# Optimizer
## Teleprompter

- Teleprompter can run the DSPy programs. It updates prompts and/or LM weights based on optimization methods.
- To use teleprompters, we need
  - A metric that evaluates output
  - A few training examples
- LabeledFewShot teleprompter add three demonstrations, which will be sampled from the training examples
- With 3 examples provided, notice that the answer is more akin to the question asked compared to the output from the previous slide

```python
from dspy.teleprompt import LabeledFewShot
fewshot_teleprompter = LabeledFewShot(k=3)
basic_fewshot_qa_model = fewshot_teleprompter.compile(basic_qa_model,
                                                      trainset=squad_train)
basic_fewshot_qa_model(
    question="Which award did Gary Zukav's first book receive?")
```

```
Answer questions with short factoid answers.

---

Follow the following format.

Question: ${question}
Answer: often between 1 and 5 words

---

Question: What group did Paul VI address in New York in 1965?
Answer: United Nations

---

Question: What did Sander's study show in terms of black law students rankings?
Answer: half of all black law students rank near the bottom of their class afte

---

Question: What problems does linguistic anthropology bring linguistic methods t
Answer: anthropological

---

Question: Which award did Gary Zukav's first book receive?
Answer: Oprah's Book Club
```

Text from https://dspy-docs.vercel.app/docs/building-blocks/modules,
Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# Available teleprompters (optimizers)

- Automatic Few-Shot Learning
  - LabeledFewShot: Simply constructs few-shot examples.
  - BootstrapFewShot: Uses your program to self-generate complete demonstrations for every stage of your program. Will simply use the generated demonstrations (if they pass the metric) without any further optimization.
  - BootstrapFewShotWithRandomSearch: Applies BootstrapFewShot several times with random search over generated demonstrations, and selects the best program.
- Automatic Instruction Optimization
  - SignatureOptimizer: Generates and refines new instructions for each step, and optimizes them with coordinate ascent.
  - BayesianSignatureOptimizer: Generates instructions and few-shot examples in each step. The instruction generation is data-aware and demonstration-aware. Uses Bayesian Optimization to effectively search over the space of generation instructions/demonstrations across your modules.

Text from https://dspy-docs.vercel.app/docs/building-blocks/modules

# Evaluation

- Use automatic metrics to evaluate the generated text.
- Built-in exact matches can be useful
- LM can be used as judge for the evaluator

See more details from https://dspy-docs.vercel.app/docs/building-blocks/metrics

# Retrieval

- DSPy supports multiple retriever backend, which can be used in RAG (retrieval augmented generation) pattern.
- Here are the supported retrievers
  - ColBERT
  - ChromaDB
  - Pinecone
  - Qdrant
  - Weaviate
  - you.com

See more details from the repo, https://github.com/stanfordnlp/dspy/tree/main/dspy/retrieve

# RAG

- RAG: retrieve context based on the query and use it as part of the prompt
- We can update our signature with context, which is retrieved
- We use Colbert as a retriever in this example

```python
class ContextQASignature(dspy.Signature):
    __doc__ = """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")
```

```python
class RAG(dspy.Module):
    def __init__(self, num_passages=1):
        super().__init__()
        self.retrieve = dspy.Retrieve(k=num_passages)
        self.generate_answer = dspy.Predict(ContextQASignature)

    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)
```

Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# RAG

- With 3 retrieved contexts, LM was able to answer correctly this time. Note the the answer in the previous slide is different

```
rag_model = RAG(num_passages=3)
```

✓ 0.0s

```
rag_model(question="Which award did Gary Zukav's first book receive?")
```

✓ 0.0s

```
Prediction(
    context=['Gary Zukav | Gary Zukav Gary Zukav (born October 17, 1942) i
    answer='U.S. National Book Award'
)
```

```
Answer questions with short factoid answers.

---

Follow the following format.

Context: may contain relevant facts
Question: ${question}
Answer: often between 1 and 5 words

---

Context:
[1] «Gary Zukav | Gary Zukav Gary Zukav (born O
[2] «The Dancing Wu Li Masters | The Dancing Wu
[3] «Markus Zusak | a runner-up for the Printz
Question: Which award did Gary Zukav's first bo
Answer: U.S. National Book Award
```

Example from https://github.com/cgpotts/cs224u/blob/main/hw_openqa.ipynb

# Resources

- [Multi-hop search example](https://github.com/stanfordnlp/dspy/blob/main/intro.ipynb)
- [Why Im excited for DSPy](https://substack.stephen.so/p/why-im-excited-about-dspy)
- [DSPy doc](https://dspy-docs.vercel.app)
- [Getting Started with RAG in DSPy!](https://www.youtube.com/watch?v=CEuUG4Umfxs)
- [DSPy discord](https://discord.com/invite/HZwtqNzKCu)