DSPy Documentation

DSPy also offers a modular architecture, enabling you to mix and match pre-built modules for different natural language processing (NLP) tasks. This modularity makes it highly customizable to fit your specific needs, promoting flexibility and reusability.

Collect dataset ->write dspy program ->validate your logic -> compile dspy program→ point1(iterable)

DSPy programming consist:

1. Signature : A minimal description of the sub-task the LM is supposed to solve,a description of the input fields and description of the output fields.
2. Modules : Modules in DSPy are templated and parameterized to abstract these prompting techniques. This means that they are used to adapt DSPy signatures to a task by applying prompting, fine-tuning, augmentation, and reasoning techniques.
3. Teleprompting: Teleprompters act as optimizers for DSPy programs. They take a metric and, together with the DSPy compiler, learn to bootstrap and select effective prompts for a DSPy program’s modules.

Route code

import dspy

class RouteSignature(dspy.Signature):

query: dspy.InputField()

route: dspy.OutputField(desc="Route the query to either structured or unstructured")

class RouterModule(dspy.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.predict = dspy.Predict(RouteSignature)

def forward(self, query):

return self.predict(query=query)

SQL Generation Code

class sqlsingautre(dspy.Signature):

query:dspy.InputField()

table\_schema :dspy.InputField()

sql\_query:dspy.OutputField(desc:"Generate the sql code for given query based on given table scehma.")

class SQLGeneration(dspy.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.predict=dspy.Predict(sqlsignautre)

def forward(self,query,table\_schema):

return self.predict(query=query,table\_schema=table\_schema)

Unstructured Code

def retriever(query,vs):

## information retriever

class RAGResult(dspy.Signature):

"""Answer a question using the context."""

context = dspy.InputField()

question = dspy.InputField()

answer = dspy.OutputField(desc="A helpful and concise answer")

class ResultGeneration(dspy.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.retrieve = retriever

self.generate\_answer = dspy.ChainOfThought("context, question-> answer")

def forward(self, question):

context = self.retrieve(question)

return self.generate\_answer(context=context, question=question)

Questions : if i need to check →database access and data for teleprompter.

We need the complete pipeline | prompting

Can we make this in one signature the entire flow. In react(not multiple levels calling).