Agentics Tutorial

Lecture 3: Agentics

Guest lectures at Columbia University Class on Agentic AI, Fintech, and the Data Economy. - Prof. Agostino Capponi

Alfio Massimiliano Gliozzo

IBM research

gliozzo@us.ibm.com

Recap from Lecture 2 – Conversational Agents

- Classical vs. GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Multi Agent Systems: CrewAI
- Hands on: build simple agent with tools in CrewAI

Lecture 3: Agentics

- Challenges of using conversational agents on enterprise data
- Logical Transduction
- Agentics
- Hands on: Agentics Basics

Outline

- Challenges of using conversational agents for business analytics
- Logical Transduction
- Agentics
- Hands on: Agentics Basics

Challenges of working with enterprise data

Data Quality & Integration

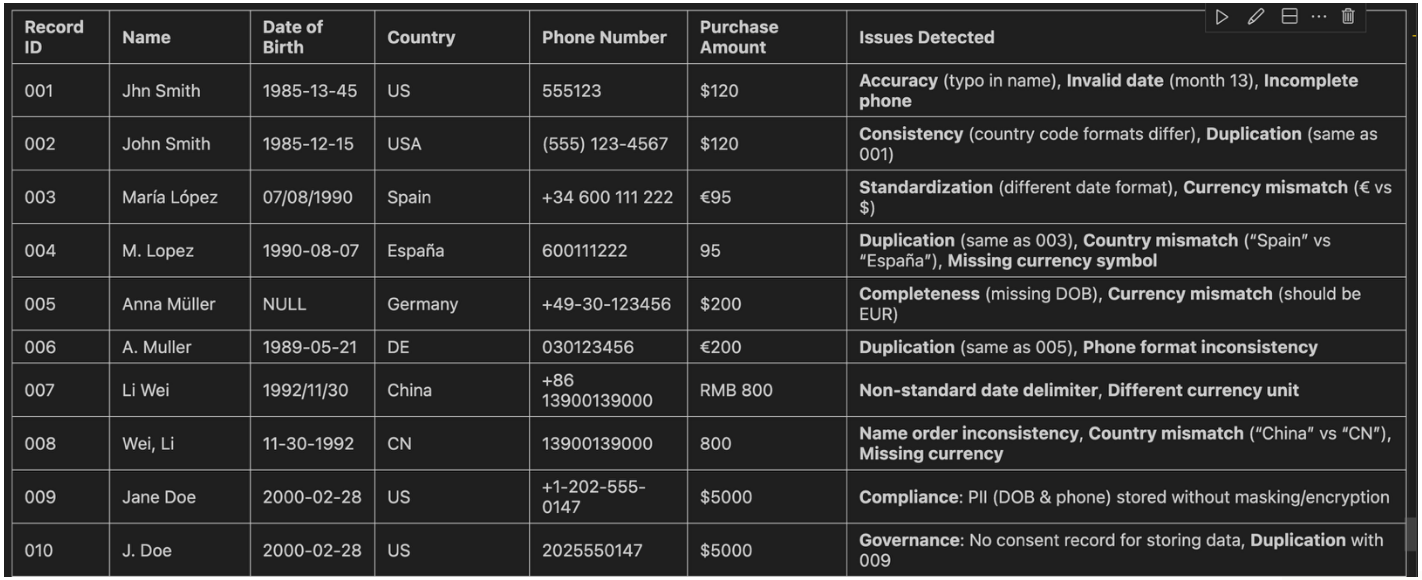
Incomplete, inconsistent, or siloed data reduces AI accuracy.

Difficulty connecting structured (SQL, OLAP) and unstructured (text, docs) sources.

Governance & Compliance

Sensitive financial, HR, or customer data must meet **privacy regulations** (GDPR, HIPAA, etc.).

Hard to ensure AI decisions remain within governance frameworks.



Record ID	Name	Date of Birth	Country	Phone Number	Purchase Amount	Issues Detected
001	Jhn Smith	1985-13-45	US	555123	$120	**Accuracy** (typo in name), **Invalid date** (month 13), **Incomplete phone**
002	John Smith	1985-12-15	USA	(555) 123-4567	$120	**Consistency** (country code formats differ), **Duplication** (same as 001)
003	María López	07/08/1990	Spain	+34 600 111 222	€95	**Standardization** (different date format), **Currency mismatch** (€ vs $)
004	M. Lopez	1990-08-07	España	600111222	95	**Duplication** (same as 003), **Country mismatch** ("Spain" vs "España"), **Missing currency symbol**
005	Anna Müller	NULL	Germany	+49-30-123456	$200	**Completeness** (missing DOB), **Currency mismatch** (should be EUR)
006	A. Muller	1989-05-21	DE	030123456	€200	**Duplication** (same as 005), **Phone format inconsistency**
007	Li Wei	1992/11/30	China	+86 13900139000	RMB 800	**Non-standard date delimiter**, **Different currency unit**
008	Wei, Li	11-30-1992	CN	13900139000	800	**Name order inconsistency**, **Country mismatch** ("China" vs "CN"), **Missing currency**
009	Jane Doe	2000-02-28	US	+1-202-555-0147	$5000	**Compliance**: PII (DOB & phone) stored without masking/encryption
010	J. Doe	2000-02-28	US	2025550147	$5000	**Governance**: No consent record for storing data, **Duplication** with 009

Explainability & Trust

Black-box AI outputs are hard to interpret for BI users.

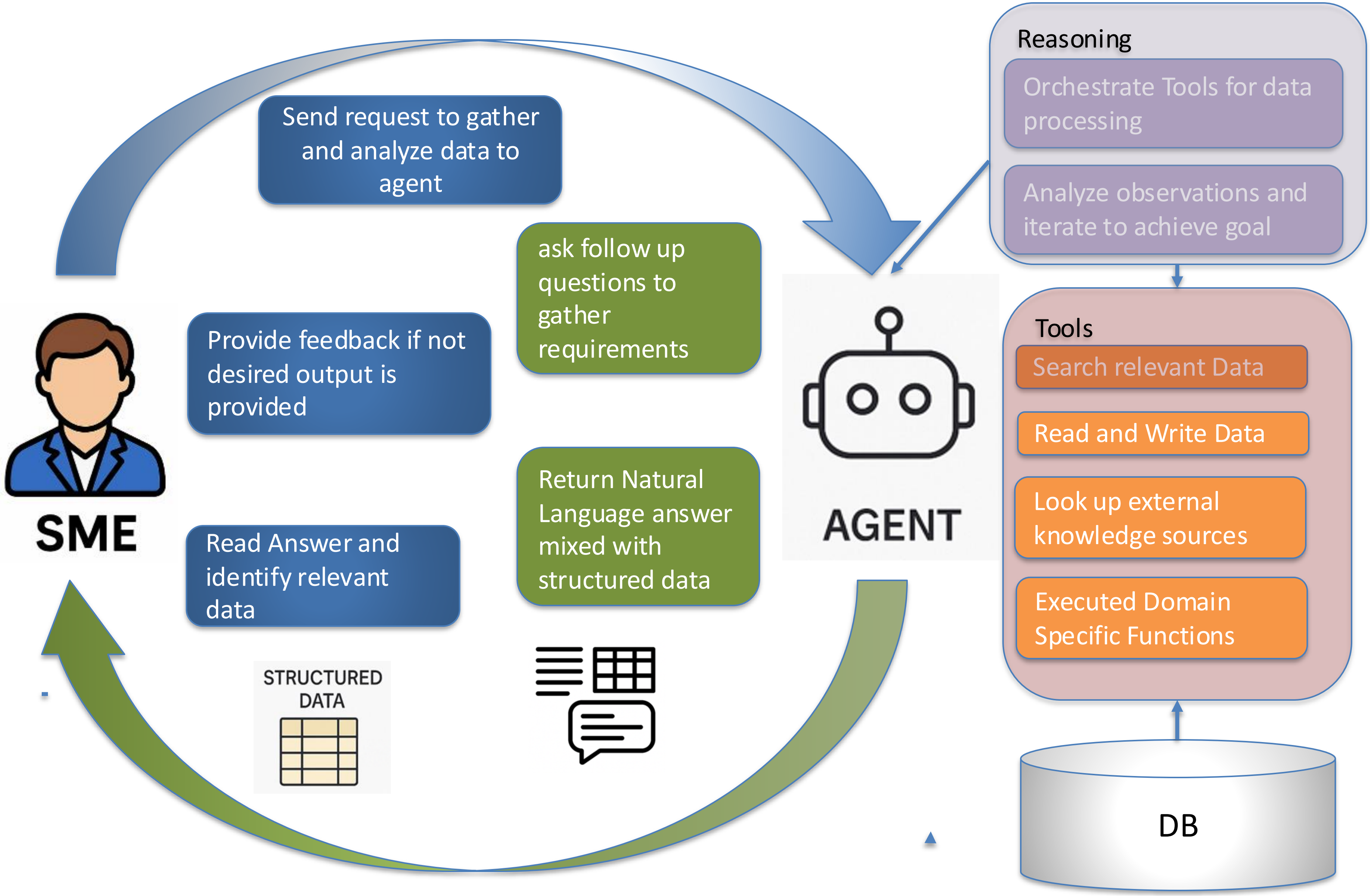
Business stakeholders demand **transparent, auditable reasoning**.

Real-Time Performance

BI often requires **low-latency dashboards**.

AI inference can be slow or computationally expensive at scale.

Use of conversational agents on structured data leads to error propagation and latency …



Reasoning

Orchestrate Tools for data processing

Analyze observations and iterate to achieve goal

Tools

Search relevant Data

Read and Write Data

Look up external knowledge sources

Executed Domain Specific Functions

Send request to gather and analyze data to agent

ask follow up questions to gather requirements

Provide feedback if not desired output is provided

Return Natural Language answer mixed with structured data

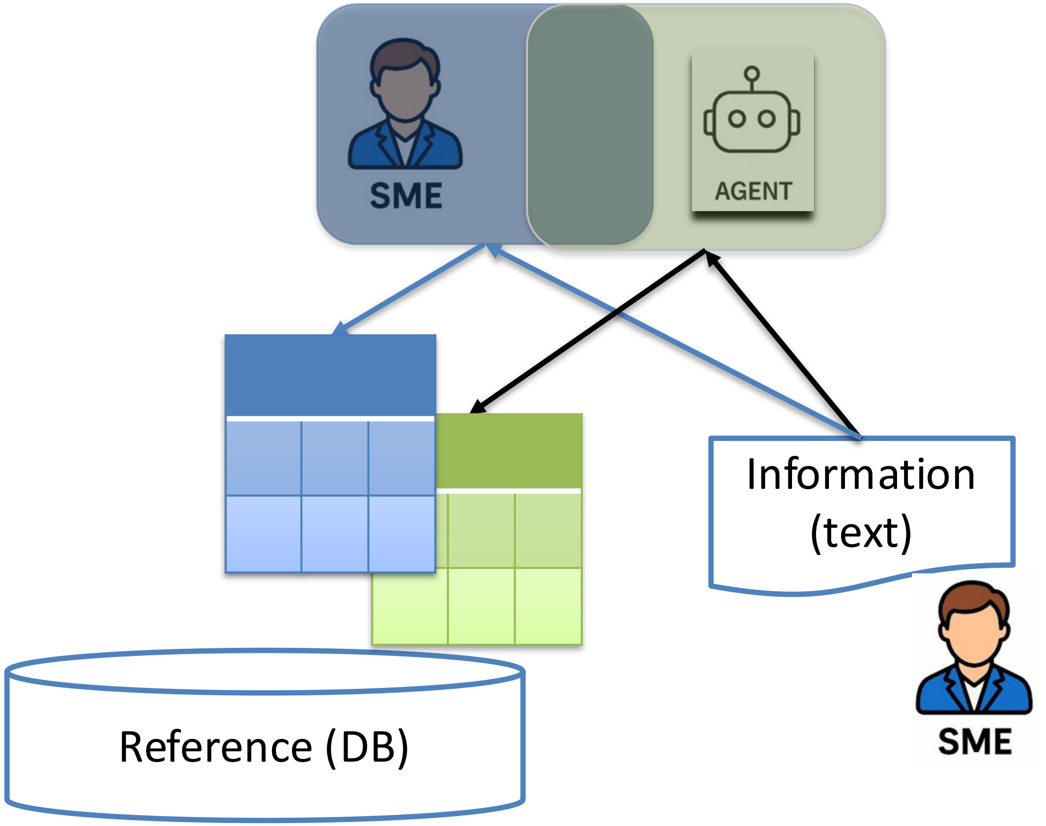
Read Answer and identify relevant data

Let’s look at the root cause: Semiotics of Business Analytics

Meaning

Symbolic (ontologies, DB schema)

Sub-symbolic(Embeddings, Transformers)



Text is interpreted into internal representations (meaning)

Internal Representations are grounded into the DB data

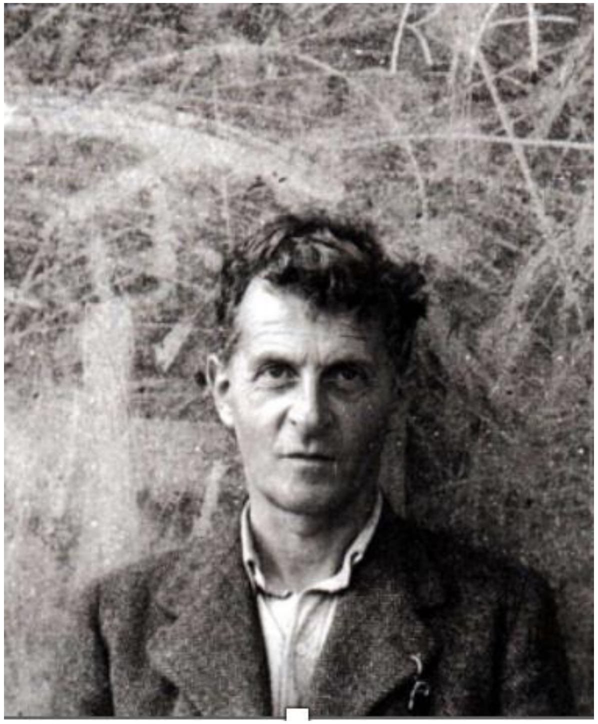
Text should be precise enough to refer DB structures

This is very unlikely, that’s what formal languages are for …

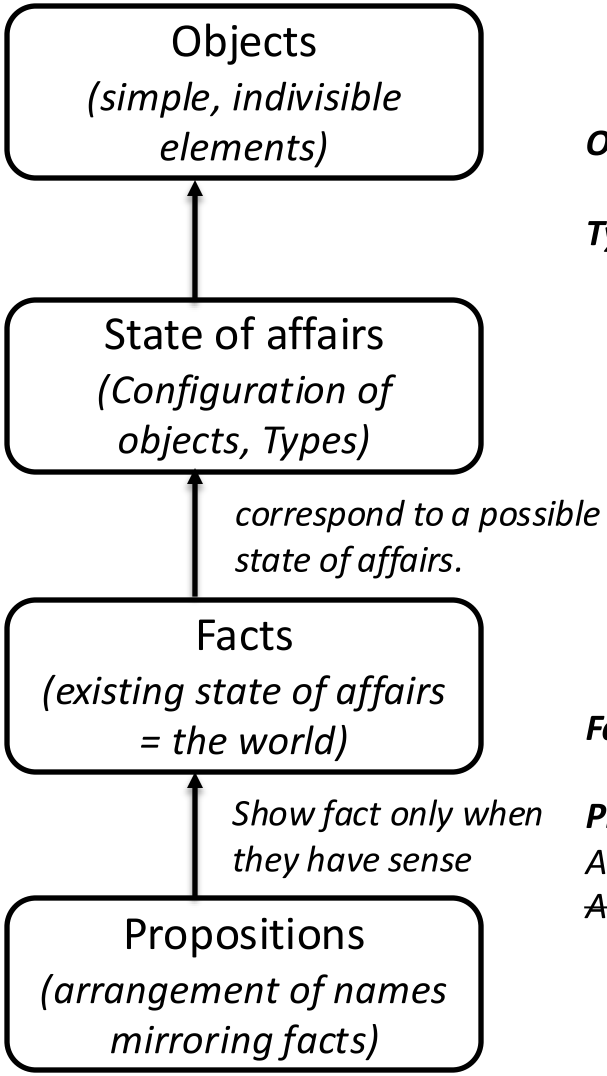
We need a better (intermediate) representation than language to instruct LLMs

Representation Theory of Meaning

Wittgenstein , Tractatus Logico-Philosophicus, 1921



all meaning ultimately rests on this pictorial relation between proposition and world.



Objects
(simple, indivisible elements)

State of affairs
(Configuration of objects, Types)

correspond to a possible state of affairs.

Facts
(existing state of affairs = the world)

Show fact only when they have sense

Propositions
(arrangement of names mirroring facts)

Objects: *Alfio, IBM*

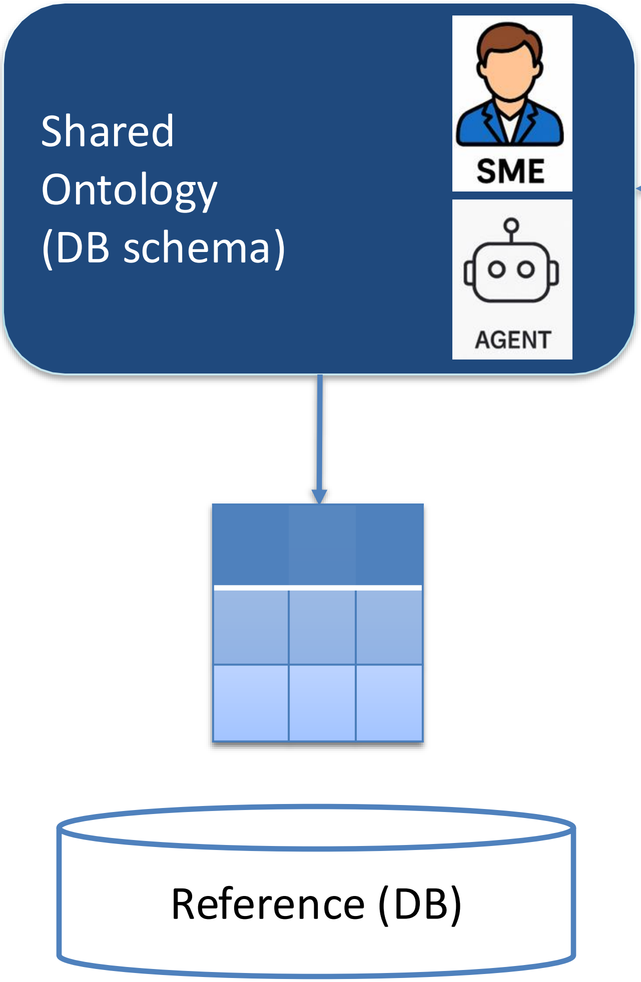
Types:

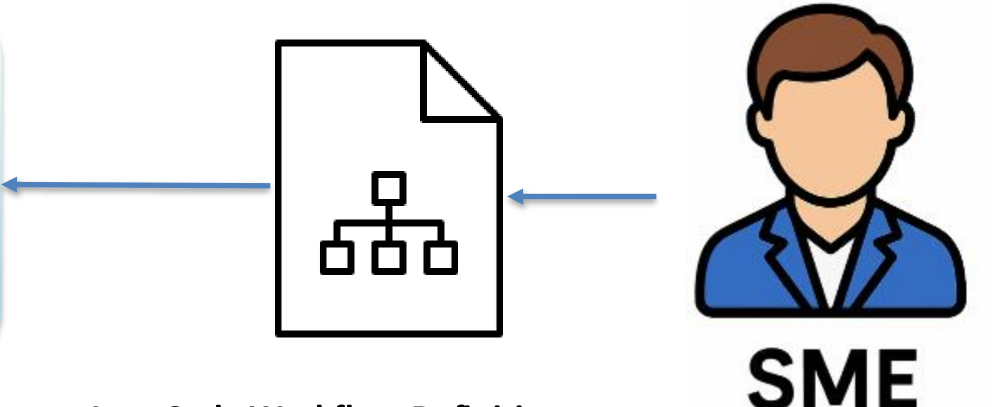
```
Person:
        name: str
        employee_id: int
Company
        name: str
        ceo: Person
Work_for:
        employee: Person
        employer: Company
```

Facts*: work_for(Alfio, IBM)*

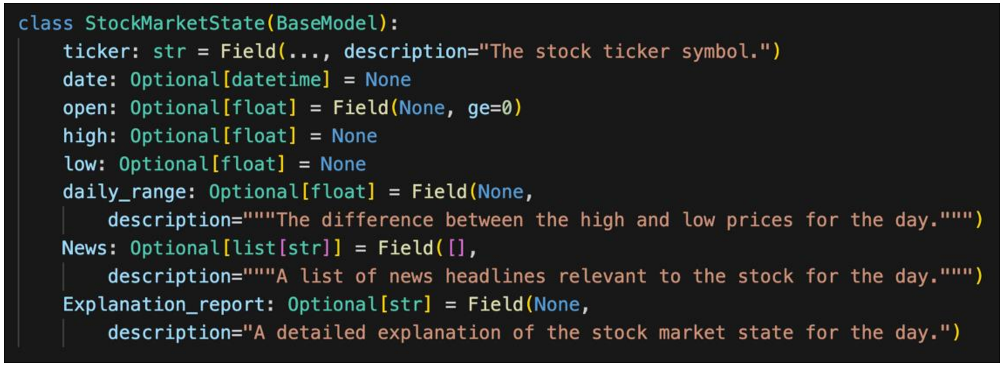
Propositions*:*
Alfio work for IBM
~~*Alfio lives in IBM*~~

What if we use **Pydantic** to instruct and communicate with LLMs?





Low Code Workflow Definition
Agents are instructed by SMEs by defining Pydantic Models, connecting them to the data to design business workflows



```
class StockMarketState(BaseModel):
    ticker: str = Field(..., description="The stock ticker symbol.")
    date: Optional[datetime] = None
    open: Optional[float] = Field(None, ge=0)
    high: Optional[float] = None
    low: Optional[float] = None
    daily_range: Optional[float] = Field(None,
        description="""The difference between the high and low prices for the day.""")
    News: Optional[list[str]] = Field([],
        description="""A list of news headlines relevant to the stock for the day.""")
    Explanation_report: Optional[str] = Field(None,
        description="A detailed explanation of the stock market state for the day.")
```

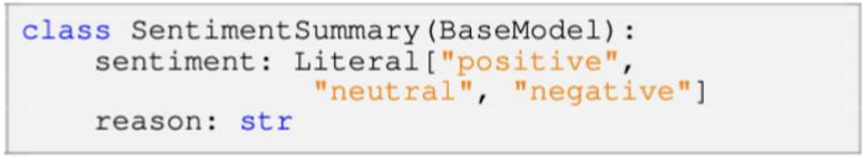
Outline

- Challenges of using conversational agents for business analytics
- Logical Transduction
- Agentics
- Hands on: Agentics Basics

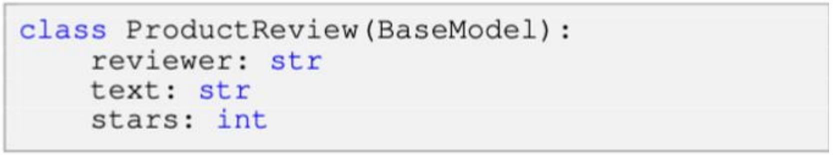
Logical Transduction

The inference-driven transformation of an object *x* of type *X* into an object *y* of type *Y*, such that for all predicted slot values of *y* it is possible to provide a logical explanation supported by the slot values of x

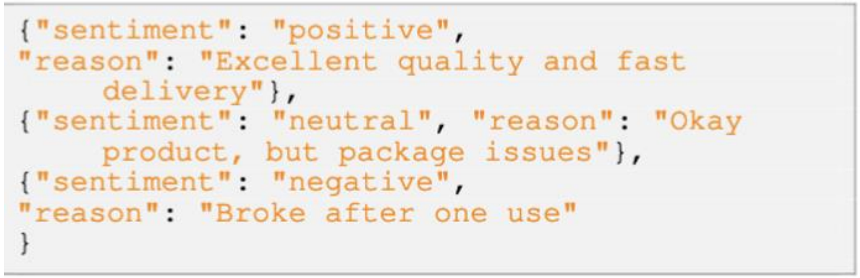
$$\mathbf{y} = Y \ll \mathbf{x}$$



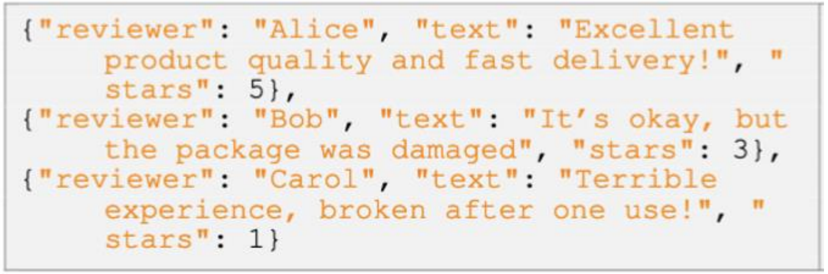
```
class SentimentSummary(BaseModel):
    sentiment: Literal["positive",
                "neutral", "negative"]
    reason: str
```



```
class ProductReview(BaseModel):
    reviewer: str
    text: str
    stars: int
```

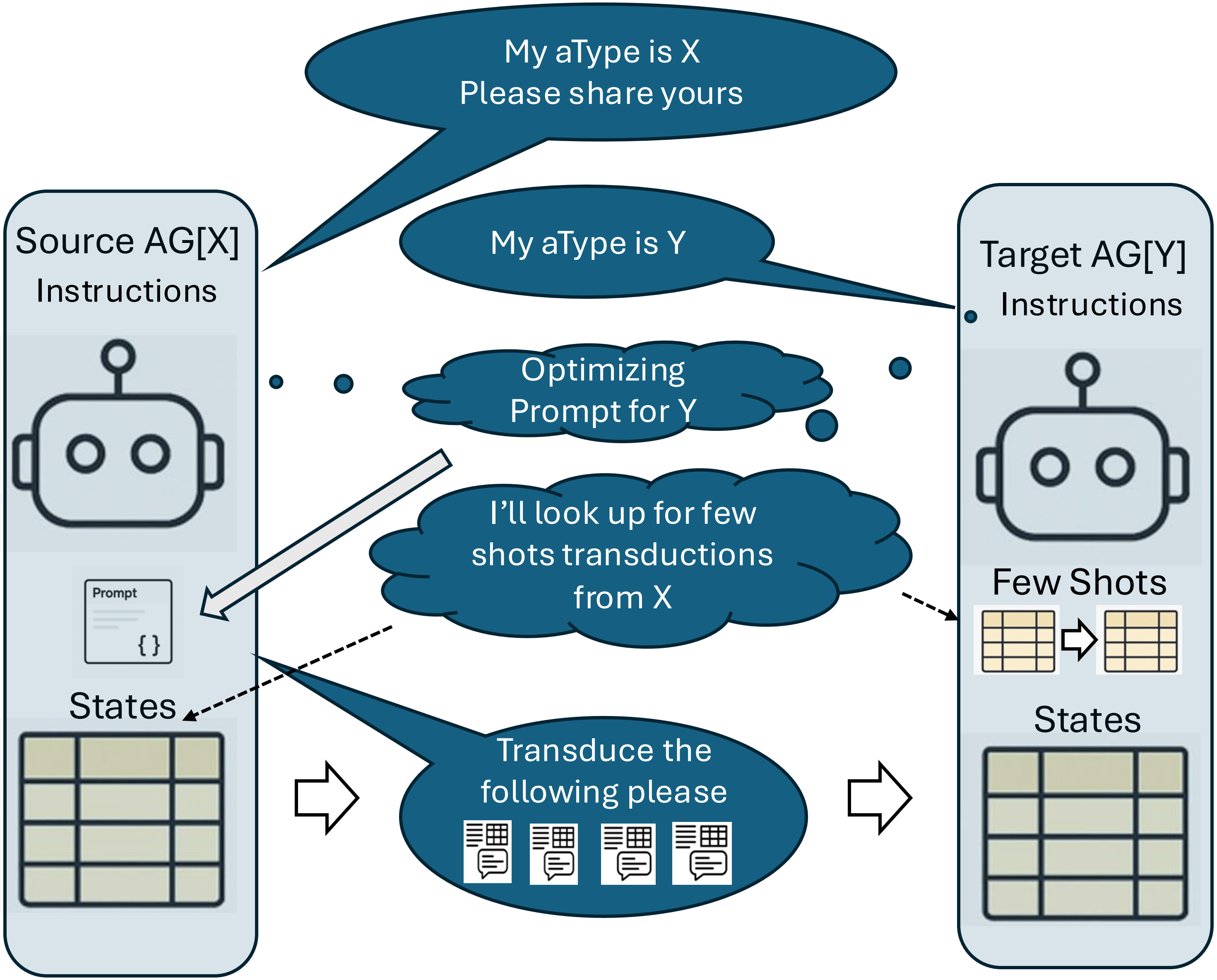


```
{"sentiment": "positive",
"reason": "Excellent quality and fast
    delivery"},
{"sentiment": "neutral", "reason": "Okay
    product, but package issues"},
{"sentiment": "negative",
"reason": "Broke after one use"
}
```

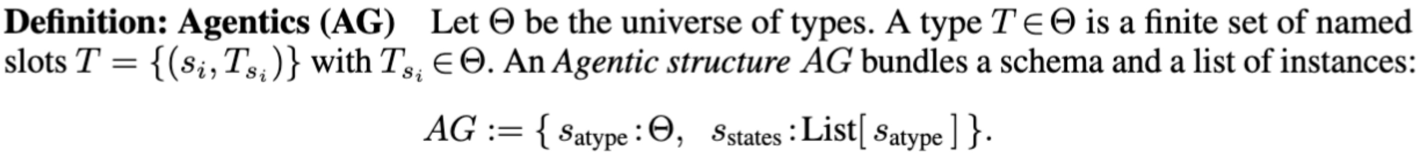


```
{"reviewer": "Alice", "text": "Excellent
    product quality and fast delivery!", "
    stars": 5},
{"reviewer": "Bob", "text": "It’s okay, but
    the package was damaged", "stars": 3},
{"reviewer": "Carol", "text": "Terrible
    experience, broken after one use!", "
    stars": 1}
```

Logical transduction is a stateless negotiation of meaning between agents

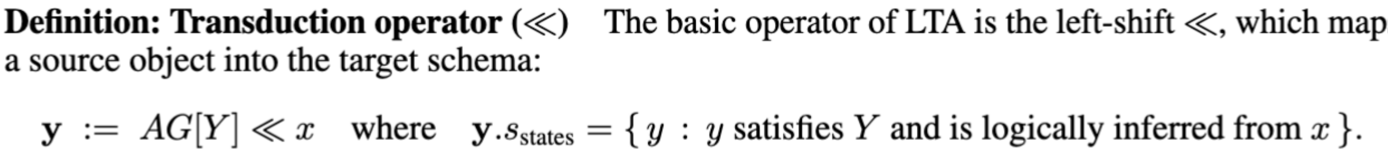


Logical Transduction Algebra (LTA)



Definition: Agentics (AG) Let Θ be the universe of types. A type $T \in \Theta$ is a finite set of named slots $T = \{(s_i, T_{s_i})\}$ with $T_{s_i} \in \Theta$. An *Agentic structure AG* bundles a schema and a list of instances:

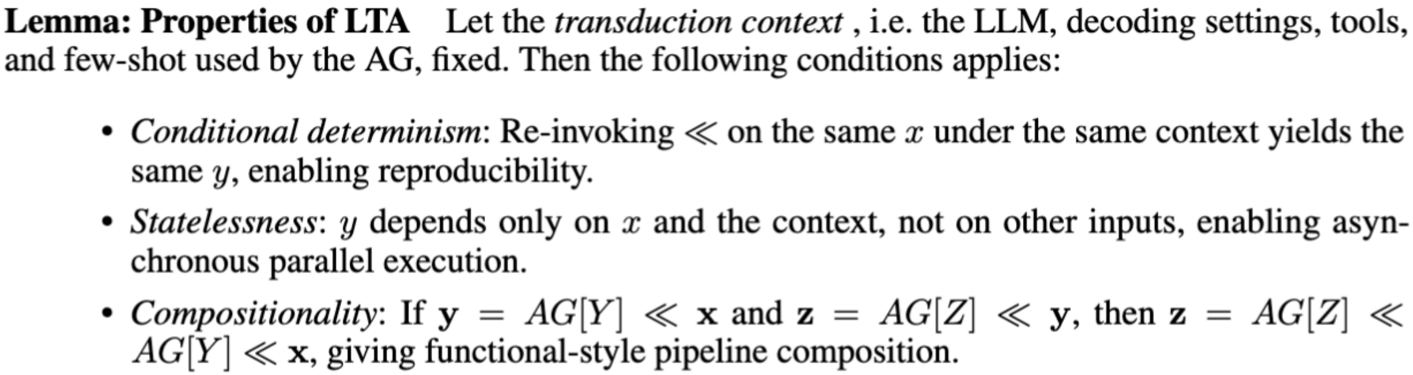
$$AG := \{\, s_{\text{atype}} : \Theta, \quad s_{\text{states}} : \text{List}[\, s_{\text{atype}} \,] \,\}.$$



Definition: Transduction operator ($\ll$) The basic operator of LTA is the left-shift $\ll$, which maps a source object into the target schema:

$$\mathbf{y} \; := \; AG[Y] \ll x \quad \text{where} \quad \mathbf{y}.s_{\text{states}} = \{\, y \; : \; y \text{ satisfies } Y \text{ and is logically inferred from } x \,\}.$$

Atype Operands: Merge, Subset, Product, Quotient, Rebind



Lemma: Properties of LTA Let the *transduction context* , i.e. the LLM, decoding settings, tools, and few-shot used by the AG, fixed. Then the following conditions applies:

- *Conditional determinism*: Re-invoking $\ll$ on the same x under the same context yields the same y, enabling reproducibility.
- *Statelessness*: y depends only on x and the context, not on other inputs, enabling asynchronous parallel execution.
- *Compositionality*: If $\mathbf{y} = AG[Y] \ll \mathbf{x}$ and $\mathbf{z} = AG[Z] \ll \mathbf{y}$, then $\mathbf{z} = AG[Z] \ll AG[Y] \ll \mathbf{x}$, giving functional-style pipeline composition.

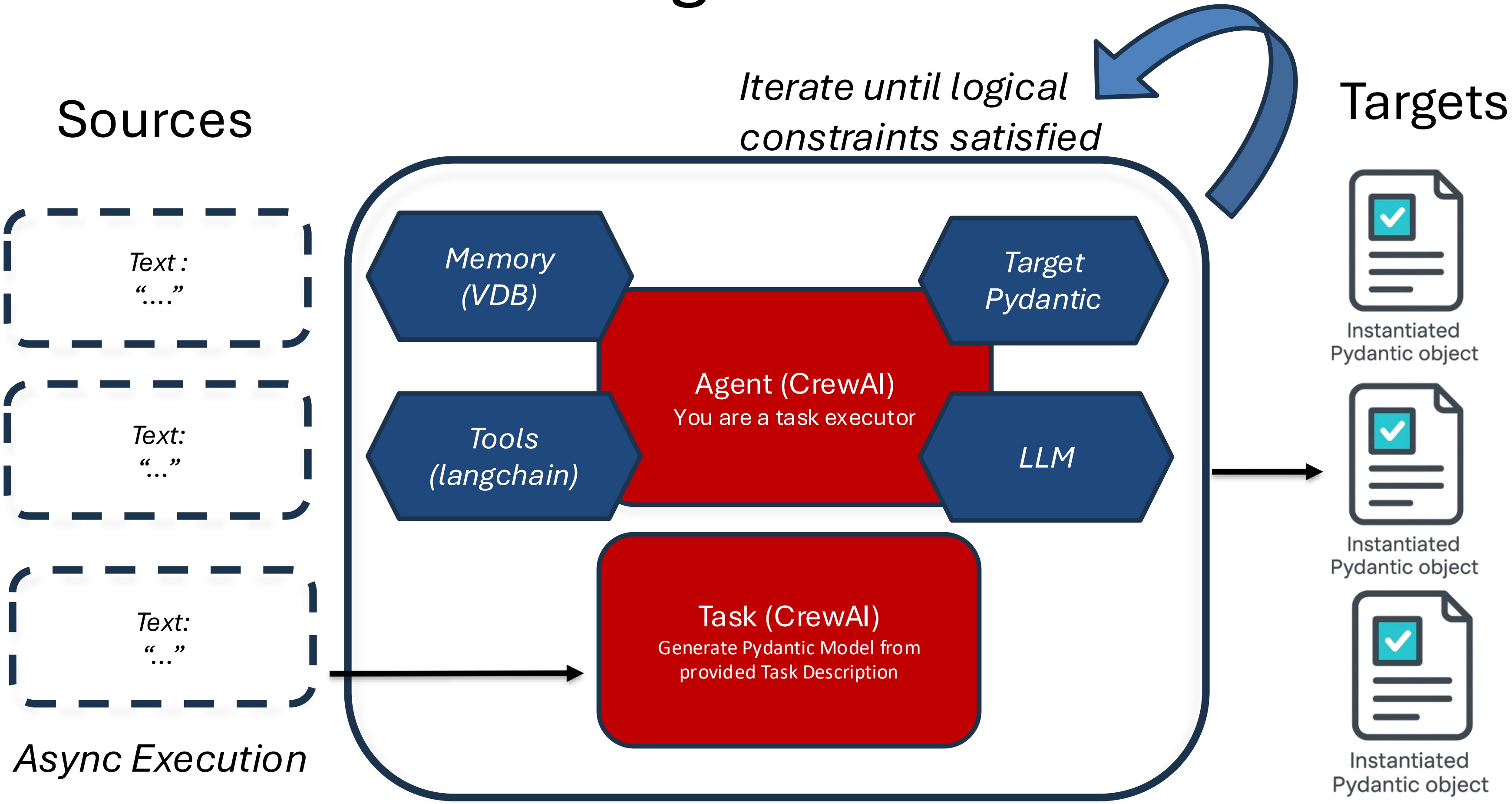
Outline

- Challenges of using conversational agents for business analytics
- Logical Transduction
- Agentics
- Hands on: Agentics Basics

What is Agentics?

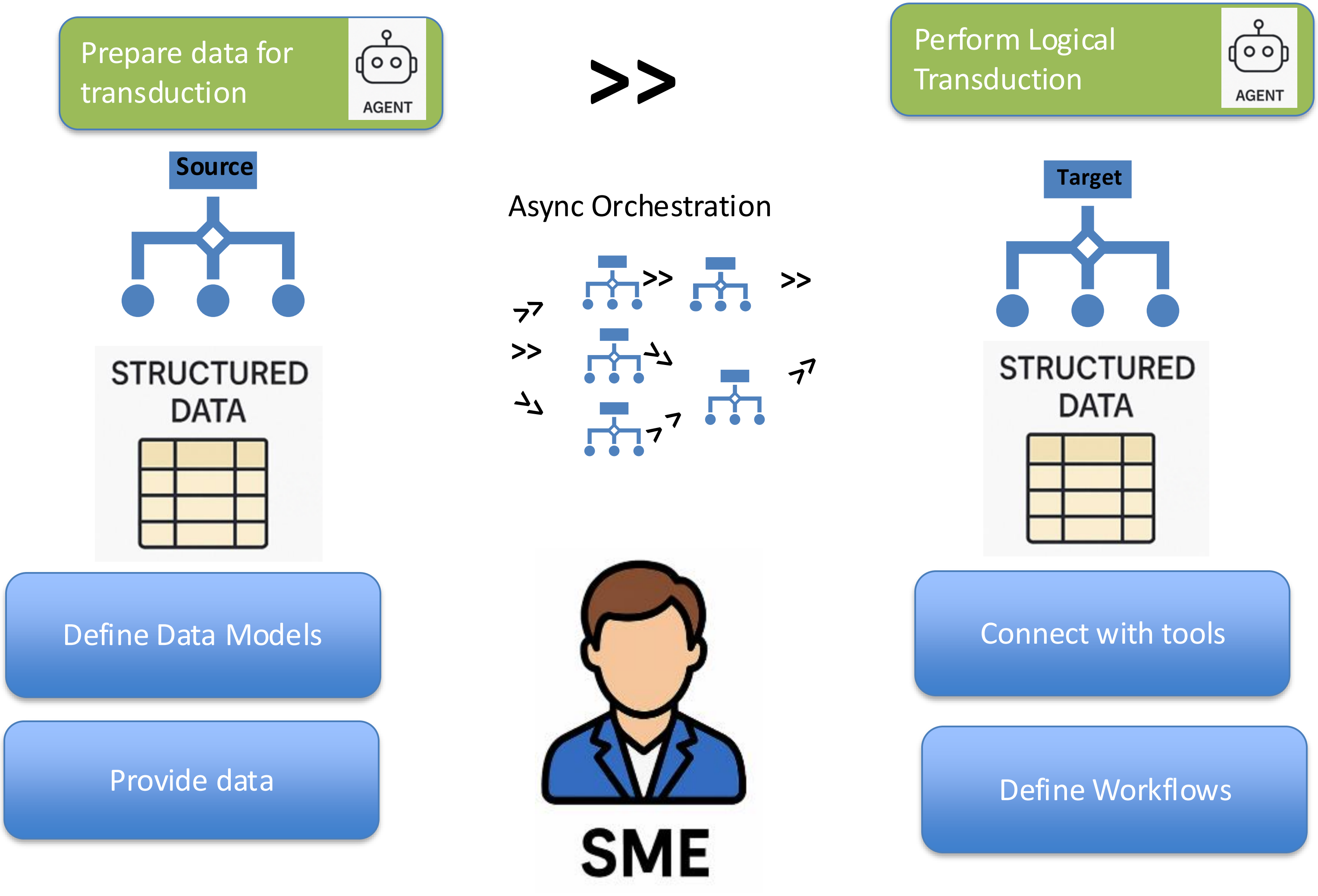
- An **Open Source Python implementation** of Transduction Algebra, Builds Upon Pydantic and CrewAI https://ibm.github.io/Agentics/
- **Typed Interfaces to LLMs:** Agentics bridges unstructured LLM output and structured data by enforcing Pydantic schemas (atypes).
- **Logical Transduction as a Primitive:** Transduction is built-in and composable, not ad-hoc prompt engineering. You can chain transformations, self-transduce, and merge AGs easily.
- **Async & Parallel Execution:** supports asynchronous scaling
- **Tool Integration:** Agentics integrates with MCP (and by extension external tools)
- **Beyond Anthropomorphic** view: no need for message history and memory

Logical Transduction is implemented by stateless agents with tools


Iterate until logical constraints satisfied
Sources
Targets
Text : "...."
Text: "..."
Text: "..."
Async Execution
Memory (VDB)
Target Pydantic
Agent (CrewAI)
You are a task executor
Tools (langchain)
LLM
Task (CrewAI)
Generate Pydantic Model from provided Task Description
Instantiated Pydantic object
Instantiated Pydantic object
Instantiated Pydantic object

Pydantic Transducers are implemented by **stateless REACT agents with tools** whose goal is to generate an object of the **target pydantic type** which is logically inferred by their source and additional observations from tools, None otherwise. Reasoning and planning can be optionally used.

Agentics puts the agents directly into your data for more precise and scalable execution



Prepare data for transduction

Source

Define Data Models

Provide data

>>

Async Orchestration

SME

Perform Logical Transduction

Target

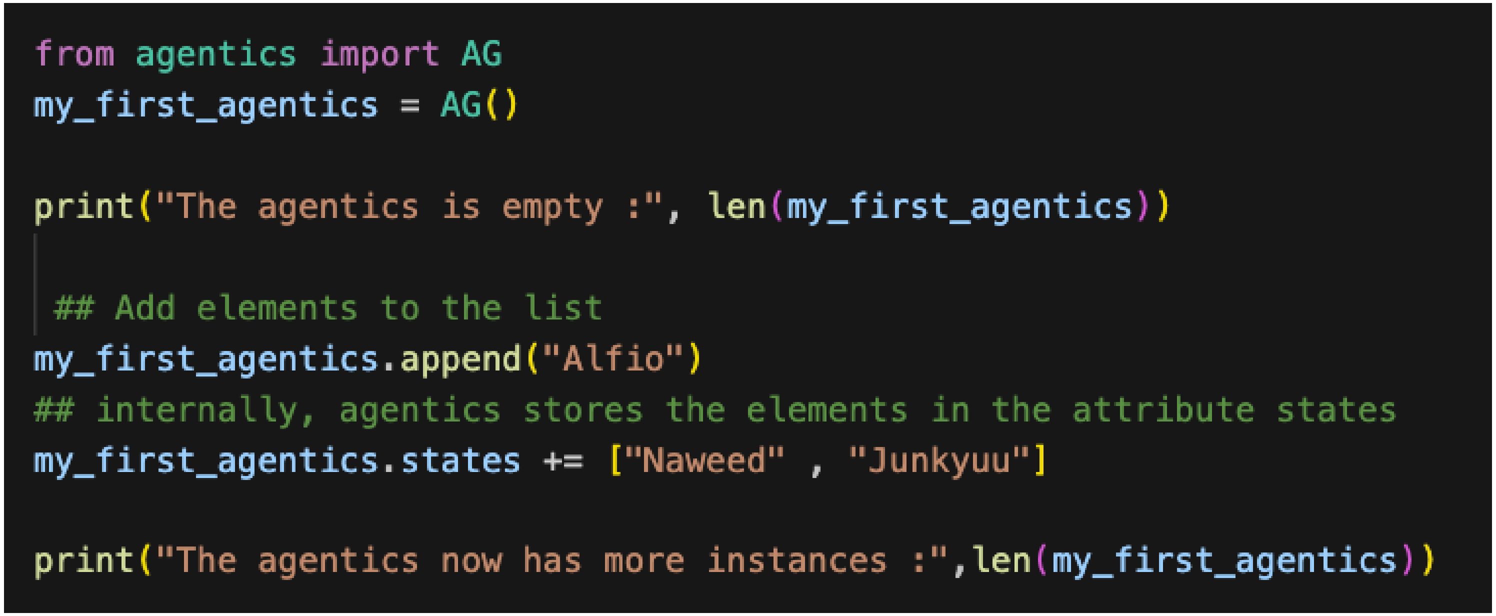
Connect with tools

Define Workflows

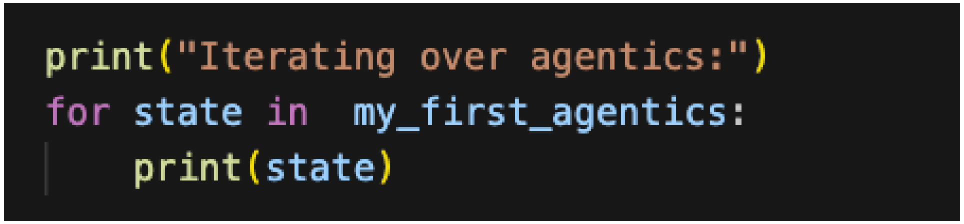
Outline

- Challenges of using conversational agents for business analytics
- Logical Transduction
- Agentics
- Hands on: Agentics Basics

Use Agentics as Lists

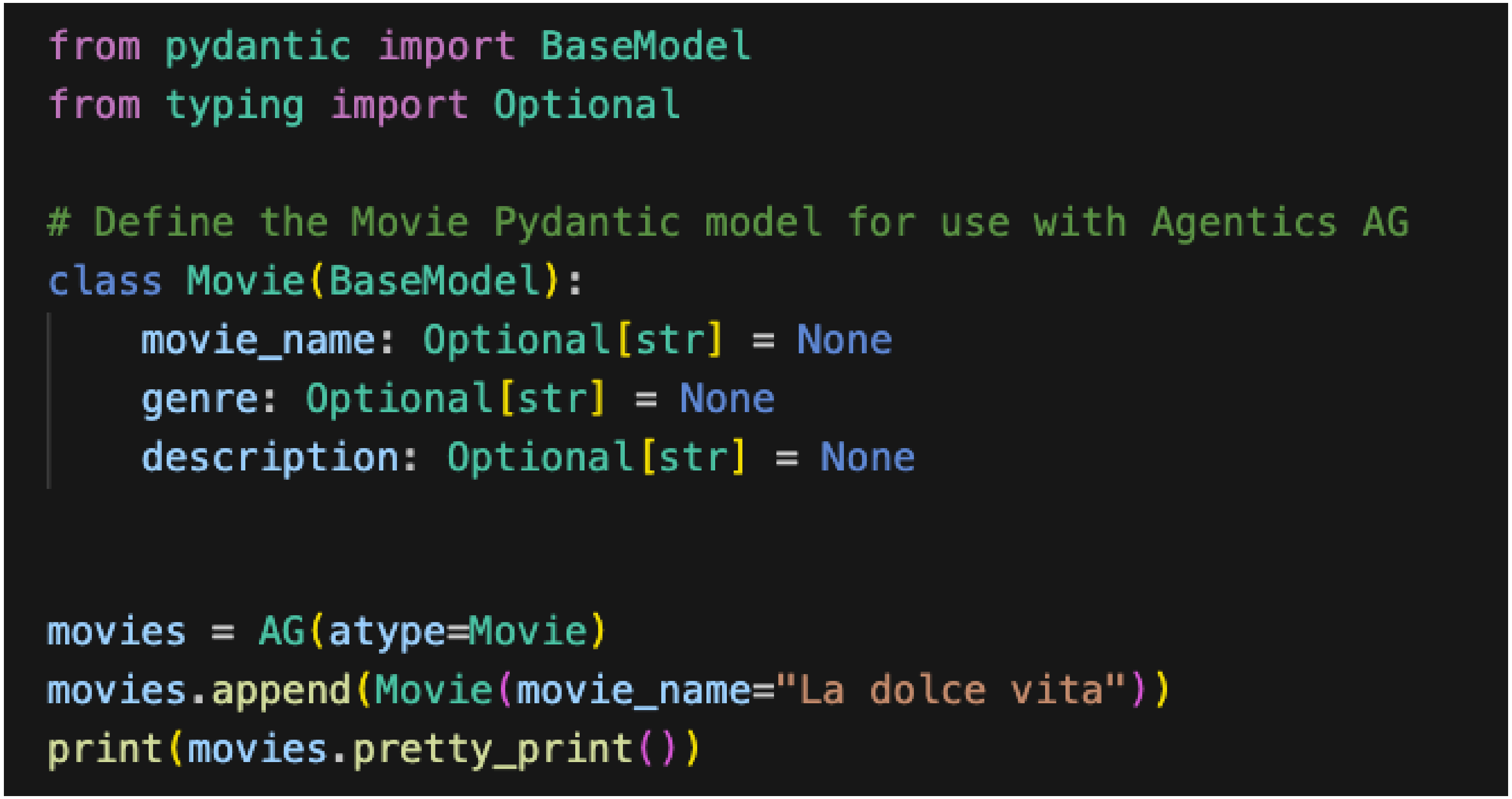


```
from agentics import AG
my_first_agentics = AG()

print("The agentics is empty :", len(my_first_agentics))

 ## Add elements to the list
my_first_agentics.append("Alfio")
## internally, agentics stores the elements in the attribute states
my_first_agentics.states += ["Naweed" , "Junkyuu"]

print("The agentics now has more instances :",len(my_first_agentics))
```



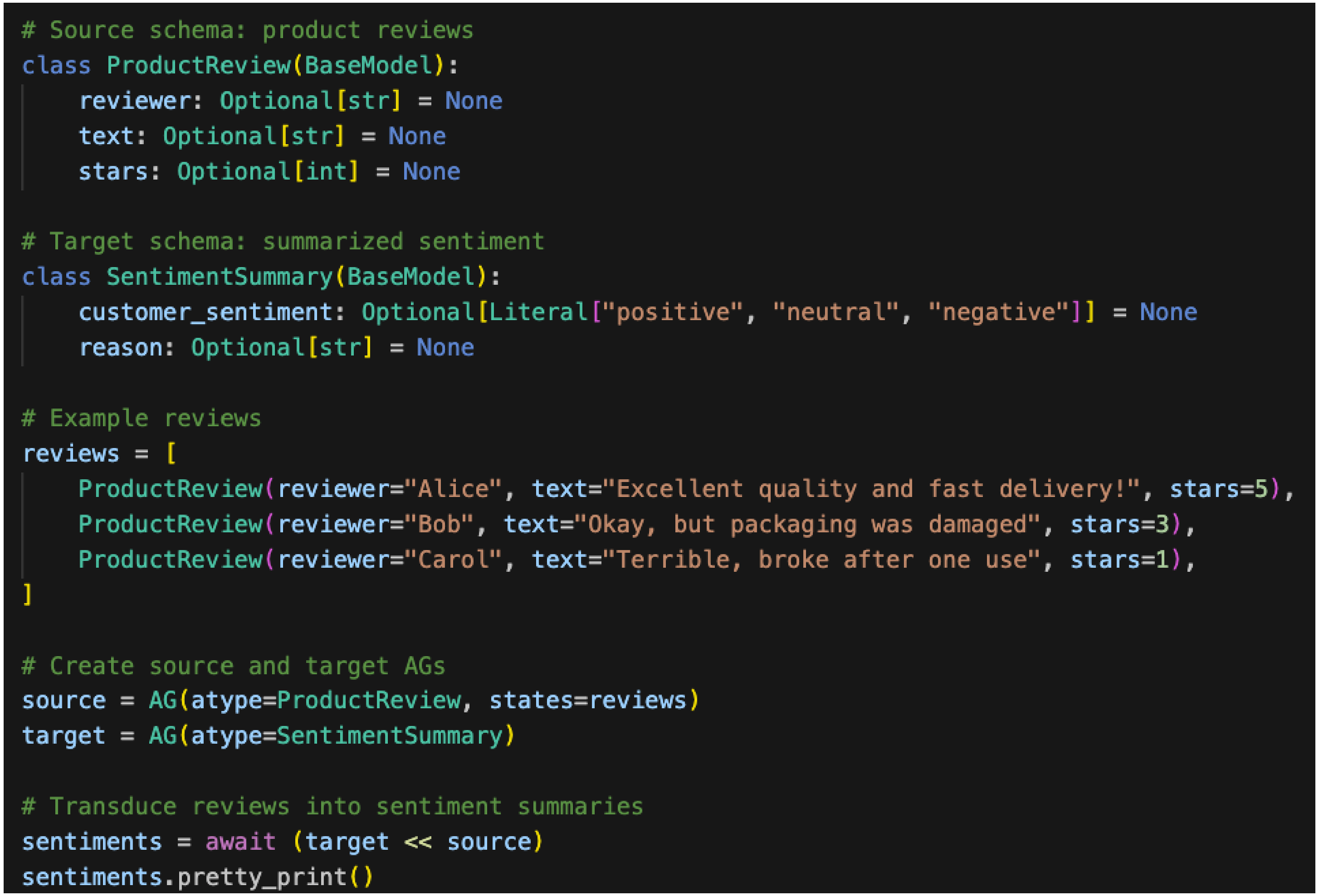
```
print("Iterating over agentics:")
for state in  my_first_agentics:
    print(state)
```

Atypes



```
from pydantic import BaseModel
from typing import Optional

# Define the Movie Pydantic model for use with Agentics AG
class Movie(BaseModel):
    movie_name: Optional[str] = None
    genre: Optional[str] = None
    description: Optional[str] = None


movies = AG(atype=Movie)
movies.append(Movie(movie_name="La dolce vita"))
print(movies.pretty_print())
```

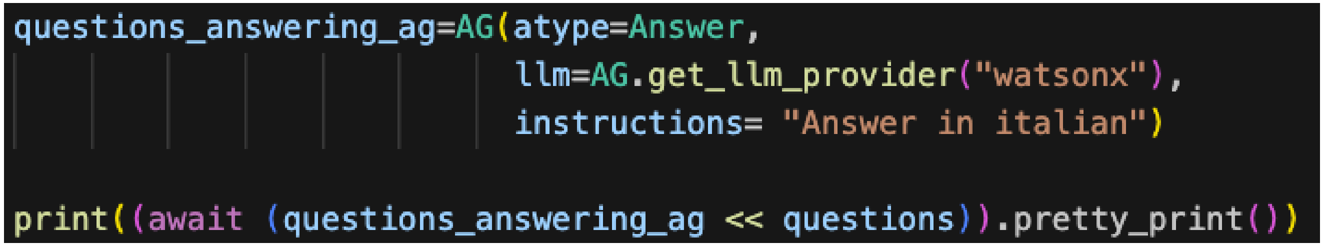
Logical Transduction



```python
# Source schema: product reviews
class ProductReview(BaseModel):
    reviewer: Optional[str] = None
    text: Optional[str] = None
    stars: Optional[int] = None

# Target schema: summarized sentiment
class SentimentSummary(BaseModel):
    customer_sentiment: Optional[Literal["positive", "neutral", "negative"]] = None
    reason: Optional[str] = None

# Example reviews
reviews = [
    ProductReview(reviewer="Alice", text="Excellent quality and fast delivery!", stars=5),
    ProductReview(reviewer="Bob", text="Okay, but packaging was damaged", stars=3),
    ProductReview(reviewer="Carol", text="Terrible, broke after one use", stars=1),
]

# Create source and target AGs
source = AG(atype=ProductReview, states=reviews)
target = AG(atype=SentimentSummary)

# Transduce reviews into sentiment summaries
sentiments = await (target << source)
sentiments.pretty_print()
```

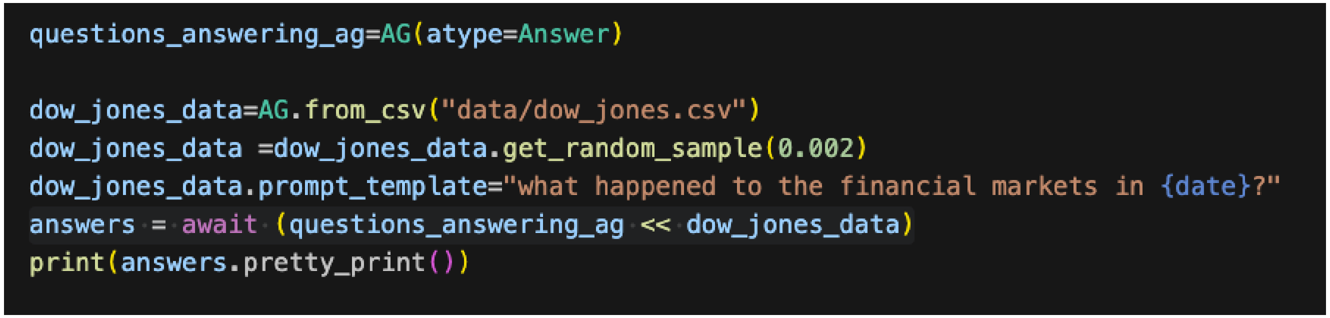
Customizing Transduction

Instructions



```
questions_answering_ag=AG(atype=Answer,
                          llm=AG.get_llm_provider("watsonx"),
                          instructions= "Answer in italian")

print((await (questions_answering_ag << questions)).pretty_print())
```

Prompt Templates



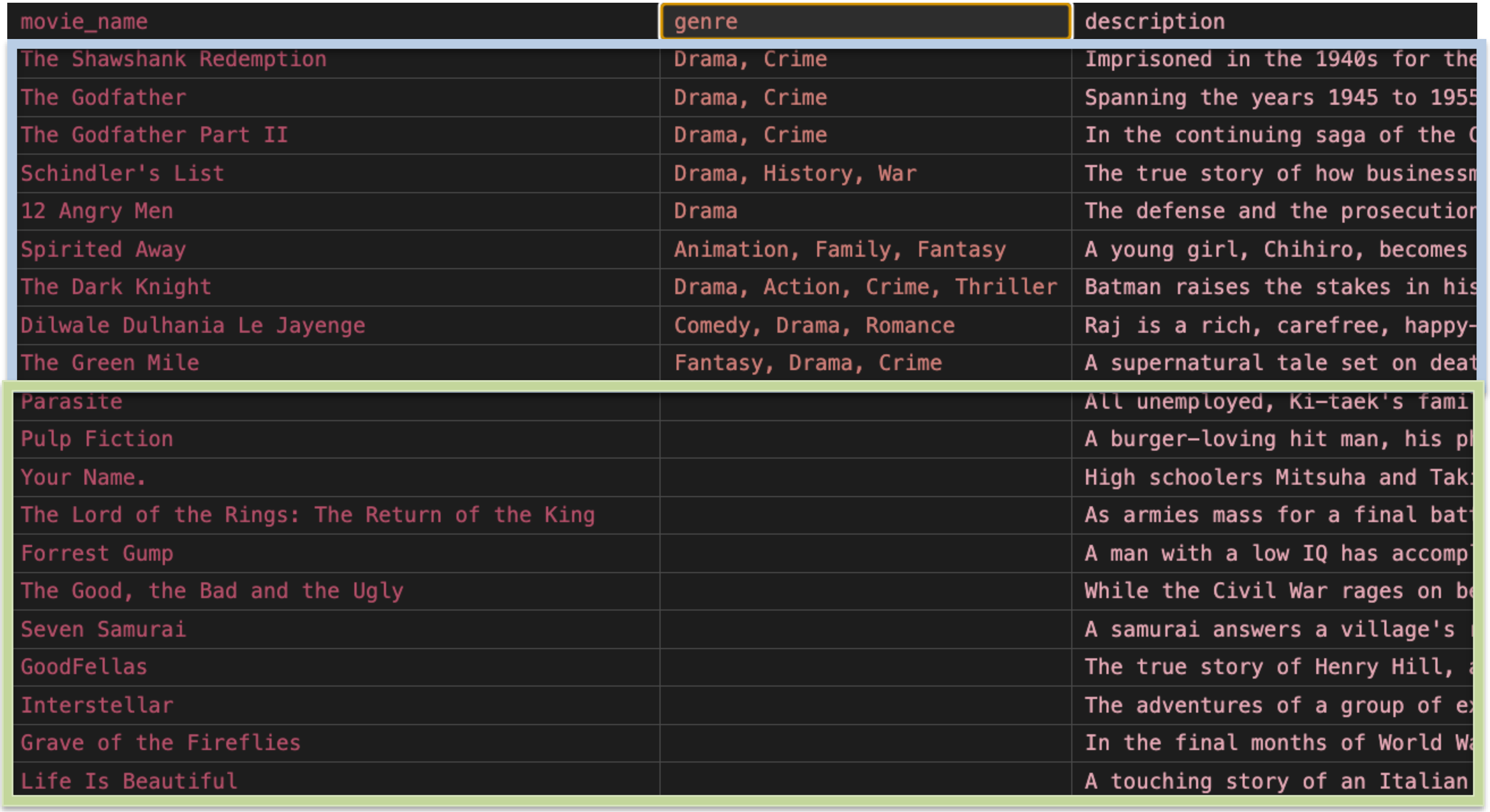
```
questions_answering_ag=AG(atype=Answer)

dow_jones_data=AG.from_csv("data/dow_jones.csv")
dow_jones_data =dow_jones_data.get_random_sample(0.002)
dow_jones_data.prompt_template="what happened to the financial markets in {date}?"
answers = await (questions_answering_ag << dow_jones_data)
print(answers.pretty_print())
```

Few Shots Learning

Target Attribute

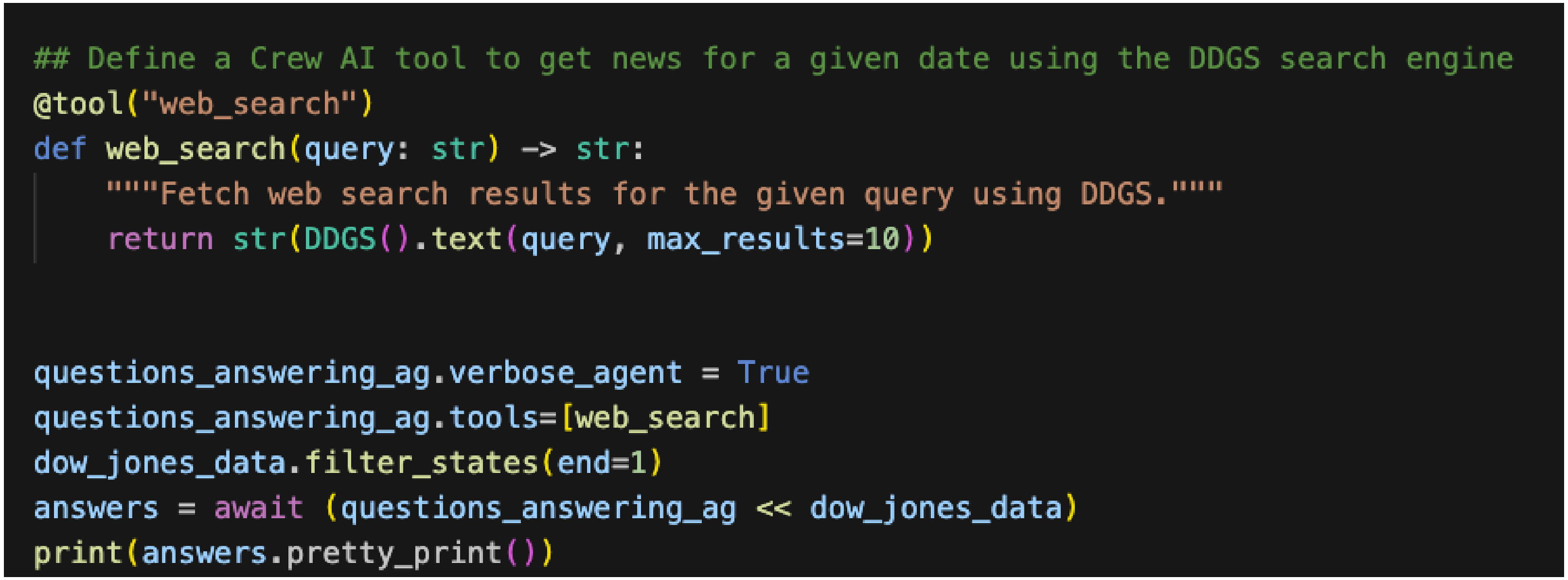
Training Few shots

Test Data



movie_name	genre	description
The Shawshank Redemption	Drama, Crime	Imprisoned in the 1940s for the
The Godfather	Drama, Crime	Spanning the years 1945 to 195
The Godfather Part II	Drama, Crime	In the continuing saga of the
Schindler's List	Drama, History, War	The true story of how businessm
12 Angry Men	Drama	The defense and the prosecutio
Spirited Away	Animation, Family, Fantasy	A young girl, Chihiro, becomes
The Dark Knight	Drama, Action, Crime, Thriller	Batman raises the stakes in hi
Dilwale Dulhania Le Jayenge	Comedy, Drama, Romance	Raj is a rich, carefree, happy-
The Green Mile	Fantasy, Drama, Crime	A supernatural tale set on dea
Parasite		All unemployed, Ki-taek's fami
Pulp Fiction		A burger-loving hit man, his p
Your Name.		High schoolers Mitsuha and Tak
The Lord of the Rings: The Return of the King		As armies mass for a final bat
Forrest Gump		A man with a low IQ has accomp
The Good, the Bad and the Ugly		While the Civil War rages on b
Seven Samurai		A samurai answers a village's
GoodFellas		The true story of Henry Hill,
Interstellar		The adventures of a group of e
Grave of the Fireflies		In the final months of World W
Life Is Beautiful		A touching story of an Italian

Using Tools



```python
## Define a Crew AI tool to get news for a given date using the DDGS search engine
@tool("web_search")
def web_search(query: str) -> str:
    """Fetch web search results for the given query using DDGS."""
    return str(DDGS().text(query, max_results=10))


questions_answering_ag.verbose_agent = True
questions_answering_ag.tools=[web_search]
dow_jones_data.filter_states(end=1)
answers = await (questions_answering_ag << dow_jones_data)
print(answers.pretty_print())
```

Any MCP tool can be used

Play with agentics

https://github.com/IBM/Agentics/blob/main/tutorials/lesson3.ipynb

Outline

- Challenges of using conversational agents for business analytics
- Logical Transduction
- Agentics
- Hands on: Agentics Basics

Homework

- Think about the application of GenAI to a financial dataset of your interest
- Build an agentic workflow that read data from csv file, elaborate them with transductions (possibly using tools) to gather information needed to make a decision.
- Push it to your Agentics Fork
- Discuss it with other students and at next lecture.
- Do not use autogenerated code if not to build small utils.
- For Example
  - Task: generate buy/sell/hold recommendation on a stock and simulate on historical data
  - Get stock historical stock prices
  - Extend dataset with additional data derived from financial news (e.g. sentiment) using tools
  - Use transduction to get the final recommendation
  - Validate your prediction w.r.t. the status quote
- Be creative and minimalistic, apply to your task of interest

References

Paper:

Alfio Gliozzo, Naweed Khan, Christodoulos Constantinides, Nandana Mihindukulasooriya, Nahuel Defosse, Junkyu Lee , **Transduction is All You Need for Structured Data Workflows,** arXiv:2508.15610

Code:

https://github.com/IBM/agentics/