04.03.2025

Hugging Face AI Agent Course

Step I: Understand What are Agents

Agent Nedir?: An AI model capable of reasoning, planning and interacting with its environment

Çevresiyle etkileşime girerek bilgi toplayan, karar veren ve belirli hedef doğrultusunda hareket eden otonom sistem

→ An agent is a system that leverages an AI model to interact with its environment in order to achieve a user defined objective. It combines reasoning (akıl yürütme), planning (planlama) and the executions (eylem gerçekleştirme) of actions (often via external tools) to fulfill tasks.

Agents have two main parts:

1. The brain: AI model ⇒ Where all the thinking happens. (Reasoning and planning is here)
2. The body: Everything agent is equipped with

Temel Özellikleri

✱ Agent workflow: Think → Act → Observe (using tools)

↳ Perception (Algılama): Çevresinden veri toplar

↳ Decision Making: Veriyi analiz ederek en iyi aksiyonu belirler

↳ Action / Execution (Etkileşim): Çevresini değiştirecek eylemde bulunur.

↳ Öğrenme Yeteneği: Değişikliklere uyum sağlar ve deneyimlerinden ders çıkarı

✱ LLM is the most common AI model in agents.

Input → text
Output → text
} Some of them creates images, how? ⇒ with tools.

An agent perform tasks via tools to complete actions.

Design of the tools is very important and has a great impact on the quality of agent.

✱ Specific tools or general tools can be useful according to the task

Agent interacts with its environment suitable for real-life usage

Some Examples

↳ Personal Virtual Assistants

↳ Customer Service Chatbots

↳ AI NPC

→ **To Summarize:** Agent is a system that uses AI model (generally LLM) as its core reasoning engine, to:

- Understand natural language: Interpret and respond
- Reason and plan: Analyze, make decisions, devise strategies for problem solving.
- Interact with its environment: Take actions and observe results

Step II - Learn what are LLMs and Messages System

Each agent needs an AI model and LLM is the most common one.

LLM: Type of AI model that excels at understanding and generating human natural language.

Trained with vast amounts of text data. ⟹ Learns patterns, structure, nuance

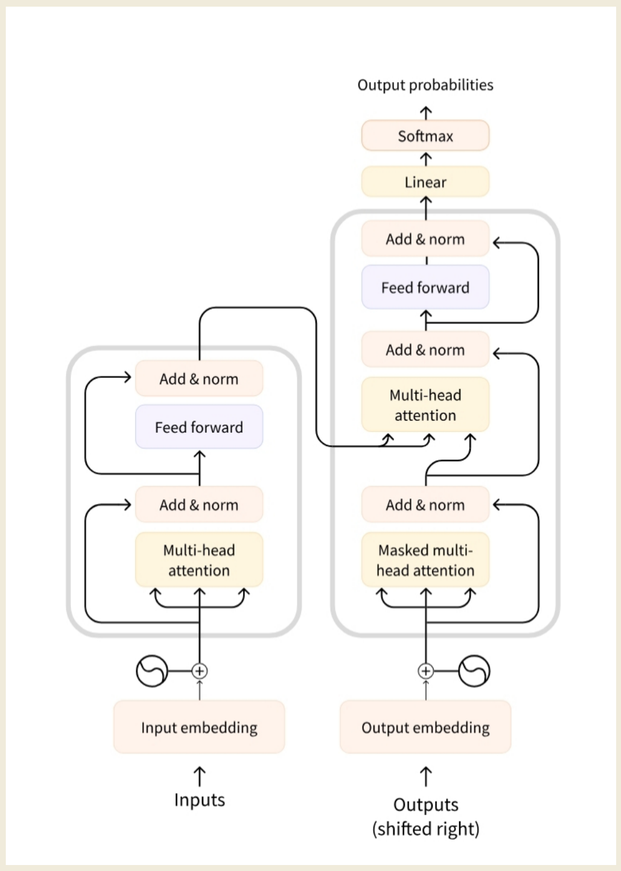
* With many millions of parameters

Transformer Architecture

A deep learning architecture based on the Attention algorithym

→ Has gained significant interest after BERT from Google in 2018

"Attention is all you need" 2017 Google research.



3 types of transformers:

1. **Encoders**: Input text (or data), Output dense representation
   Yoğunluk temsili: Verinin çok boyutlu vector olarak temsili. Metnin anlamı ve bağlamı modelin anlayacağı hale gelir.
   - Anlam, kelimeler arası ilişki ve bağlam
2. **Decoders**: Generates new tokens (anlamlı dil birimi) to complete a sequence, one token at a time.
3. **Seq2seq (Enc-Dec)**: Encoder first process the input into a context representation then decoder generates an output sequence.

7 Layers: Input embedding / Positional encoding / Self attention / Multi-head attention / Feed-forward Neural Network / Normalization / Residual Connections

***Principle of an LLM**: Predict the next token, given a sequence of previous tokens. Autoagressive ⟹ Output from one pass becomes input for the next one. Continues until the predicted token is EOS.

Token: Unit of information LLM works with.

↳ There is also some special tokens (like <|endoftext|>)

A Single Decoding Loop

~ When input text tokenized, model computes a representation of the sequence that captures information about the meaning and position of each token

~ When this representation goes to the model, outputs scores that rank of each token as being the next one

Based on the scores there is multiple strategies to select tokens

↳ Always take token with maximum score

* Want to learn more about decoding? NLP course.

Attention is all you need: When predicting next word not every word in a sentence is equally important. ⟹ Key aspect of Transformer Architecture

The capital of France is ⟹ Identifying the most relevant words

(capital, France → Carry the most meaning)

Context lenght → Refers to maximum number of tokens the LLM can process, and the maximum attention span it has.

Predicting the next token by looking at every input token → So → your first input sentence, is very important

PROMT

How are LLMs Trained?

Trained on large text datasets, through a self-supervised (GPT) or masked language modeling (BERT) objective.

↳ There isn't a correct answer (label). Model learns patterns and structure of language.

Dil yapısı, anlam bütünlüğü, bağlam öğrenilir.

Fine Tuning: After initial pre-training, LLMs fine tuned with a supervised learning to perform specific tasks.

LLM is the brain of the agent.

Messages and Special Tokens

Behind the scenes messages are concatenated and formatted into a prompt that the model can understand

Models use special tokens to delimit where the user and assistant turns start and end.

↳ They use different formatting rules and delimiters for the messages in the conversation.

System Messages: Define how the model should be. Persistent instructions, guiding every subsequent interaction.
(System prompts)

System Message also:

- Gives information about available tools
- Provides instructions to the model on how to format the actions to take
- Includes guidelines on how the thought process should be segmented.

→ Chat templates

Preserve conversation history
Store previous exchanges between the user and the assistant } This maintains context and leads to more coherent multi-turn conversations

! We always concatenate all the messages in the conversation and pass it to the LLM as a single stand alone sequence.

Chat template converts all the messages inside a python list into a prompt (A string input with tokens that contains all the messages).

```
conversation = [
   {"role": "user", "content": "I need help with my order"},
   {"role": "assistant", "content": "I'd be happy to help. Could you provide your order
number?"},
   {"role": "user", "content": "It's ORDER-123"},
]
```

Python list

→

```
<|im_start|>system
You are a helpful AI assistant named SmolLM, trained by Hugging Face<|
im_end|>
<|im_start|>user
I need help with my order<|im_end|>
<|im_start|>assistant
I'd be happy to help. Could you provide your order number?<|im_end|>
<|im_start|>user
It's ORDER-123<|im_end|>
<|im_start|>assistant
```

A prompt (SmolLM2 chat template)

↳ Base Model vs. Instruct Model

- A base model is trained on raw text data to predict next token
- An instruct model is fine-tuned to follow instructions and engage in conversations

ChatML is one chat template format with clear role indicators (system, user, assistant)

✳ In transformers, chat templates include Jinja2 code. (JSON messages —Jinja2→ Textual representation that the model can understand

↳ The transformers library takes care of chat templates as part of the tokenization process.

What are Tools?

AI agents have ability to take actions, this happens via tools.

A tool is a function given to the LLM. (Should fulfill a clear objective)

↳ Should be something that complements the power of an LLM

✳ If the agent needs up-to-date data we must provide it through some tool.

- A tool should contain:
  - ↳ A textual description of function
  - ↳ A callable
  - ↳ Arguments with typings
  - ↳ Outputs with typings (Optional)

✳ LLMs can only recieve and generate text. So, providing a tool for an agent means teaching an LLM the existance of a tool and ask to generate a text that will invoke tools.

↳ It is agents responsibility to recognise that a tool call is required and invoke the tool

↳ Output of a tool is another type of message in the conversation (Tool calling steps are not shown to the user)

↳ In fact agent uses the tool not the LLM.

✳ Tools description is in the system message and have to be precise and accurate about what the tool does and what is the exact input it expects.

↳ It is generally in JSON. But it is not necessary.

Our tool called Calculator →

```
def calculator(a: int, b: int) ->
int:
    """Multiply two integers."""
    return a * b
```

—Textual description for LLM→ Tool Name: calculator, Description: Multiply two integers., Arguments: a: int, b: int, Outputs: int

- To provide additional tools we must be consistent and use the same format. → A fragile process Try to not overlook some details.
  (One of the solutions is auto-formatting tool sections)

Auto-formatting Tool Sections

- Tools way doesn't matter for LLM, tools name, what the tool does, inputs and outputs are all that matters.

↳ We can use a **generic tool class** whenever we need to use a tool.

↓ includes

name (str), description (str), function (callable), arguments (list), outputs (str / list), __call__(), to_string()

✳ @tool decorator retrieve all the information for us via pythons inspect module. We can use tool's to_string method.

✳ Finally it is injected in the system prompt ⟶

```
system_message="""You are an AI assistant designed to help users efficiently and accurately. Your
primary goal is to provide helpful, precise, and clear responses.

You have access to the following tools:
Tool Name: calculator, Description: Multiply two integers., Arguments: a: int, b: int, Outputs: int
"""
```