UiT The Arctic University of Norway

BIO-AI LAB | ARCTIC LLM WORKSHOP 2023

Large Language Models

Day 1 - Session 4

LLM Application Development

Aaron Celeste

aaron.v.celeste@uit.no

27. Oct 2023

This Presentation

- Vector Embedding
- Vector Databases
- LLM API
- Langchain
  - Agents
  - Memory
- OpenAI Playground
- Demo

Images generated by Dall-E

Application Development

Other Frameworks besides Langchain:

FlowiseAI

AutoGPT

AgentGPT

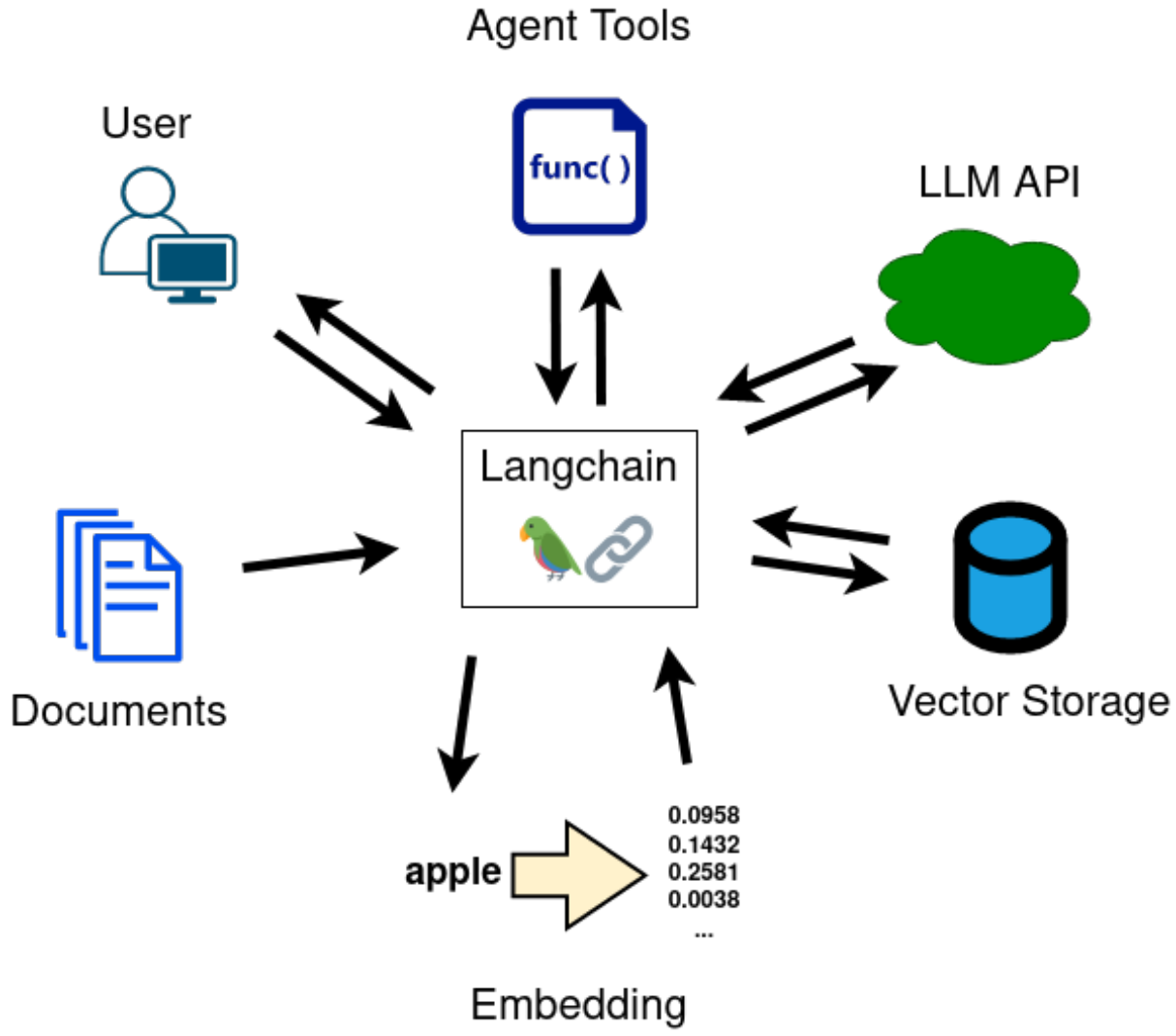
BabyAGI

Langdock

GradientJ

LlamaIndex

MetaGPT



Vector Embedding

Text is converted into a string of numbers (a vector) in a way that's specific to the particular llm

Each number represents a feature of the LLM model

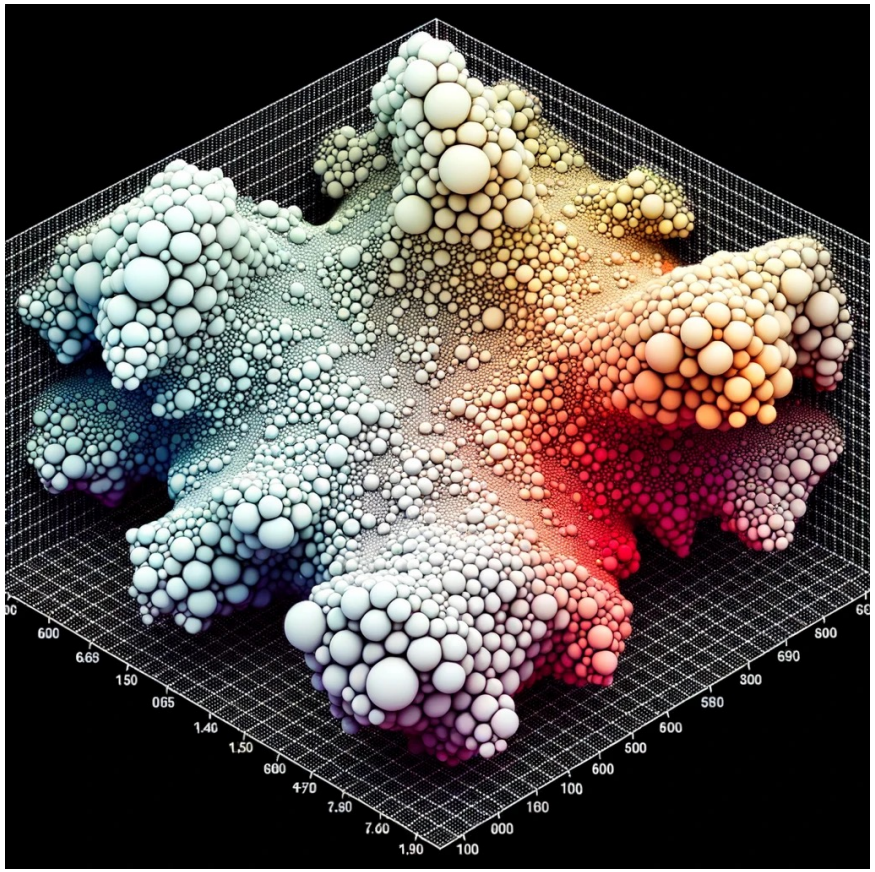
This group of features together encode concepts like positivity, gender, formality, and any other concepts that effect the meaning of a block of text in a way that the LLM model was trained to detect

LLMs process these vectors
LLMs are trained on vectors of texts
When you submit a query, it's converted into a vector embedding before it's fed to the model

Works also on images

Dall-E

Vector Databases



Dall-E

What if you want to ask an LLM about a large text which it has not been trained on such as private records?

Split it into chunks, vectorize, and plot them in high dimensions

Compare the chunks to your query and submit similar chunks to the LLM along with your query

These similarity searches are not new

Modern examples: Pinecone and Chroma

LLM API

Companies that make LLMs available offer APIs to embed text into vectors as well as query their LLMs

In python you can submit queries to specific OpenAI or HuggingFace models programmatically after installing a library easily accessible through pip install

```
from langchain.llms import OpenAI
import os

os.environ["OPENAI_API_KEY"] = "sk-XXX"

llm = OpenAI( model_name="text-davinci-
003" )

print( llm( "Tell me about tromso" ) )
```

Langchain

Middle man between user and LLM, helping to format prompts

Prebuilt chains, and rearrangeable modules

Chains package together commonly used functions like getting a summary of multiple texts and summarizing the combined summaries

Everything is customizable using the building blocks of these chains so you can do complicated things like nesting agents inside other agents

Dall-E

Langchain

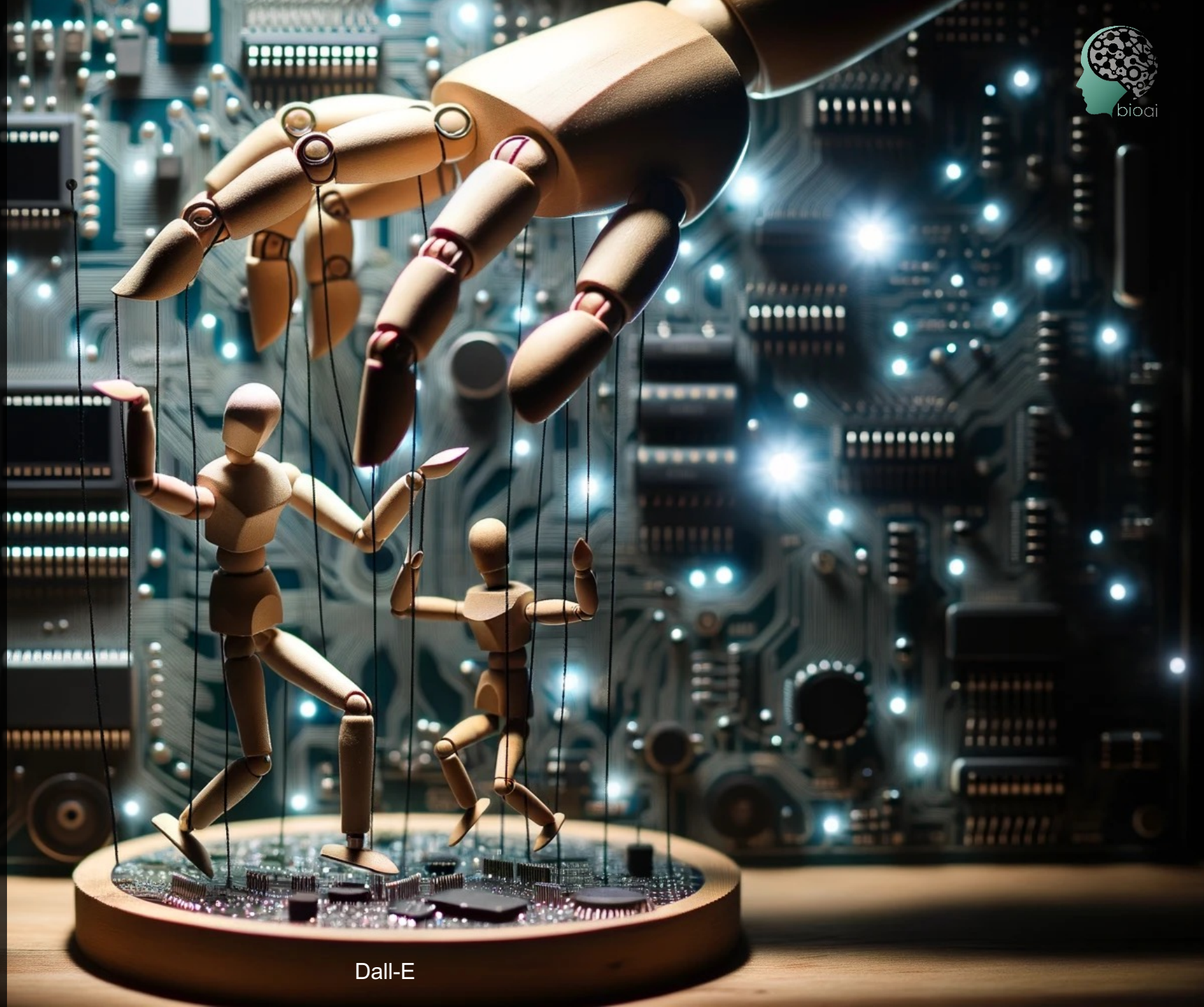
Agents

Prompt an LLM to make a step by step plan to accomplish the goal of the user's query

Define tools which the LLM can use

The agent sends the output of the tool back to the LLM, giving it the opportunity to change course based on the situation

Variations: strict plan adherence, specialized or general, chat capability to hold a conversation enabling user follow up



Dall-E

Langchain

Simplicity of setting up an agent

```
from langchain.agents.agent_toolkits import create_python_agent
from langchain.tools.python.tool import PythonREPLTool
from langchain.llms.openai import OpenAI
from langchain.agents.agent_types import AgentType
import os
os.environ["OPENAI_API_KEY"] = "sk-XXX"

query = input("What can I help you with?  ")

agent = create_python_agent(
    OpenAI(temperature=0.5),
    tool=PythonREPLTool(),
    verbose=True,
    agent_type=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    agent_executor_kwargs={"handle_parsing_errors": True},
)

print(agent.run("This is a linux machine. First, import any python libraries that you'll need.
" + str(query) + ". If you get the same error twice, try a different method. Before you finish,
please also print a summary of any results to the console with a print statement. Add a hashtag
# to the end of python code."))
```

Langchain

Memory

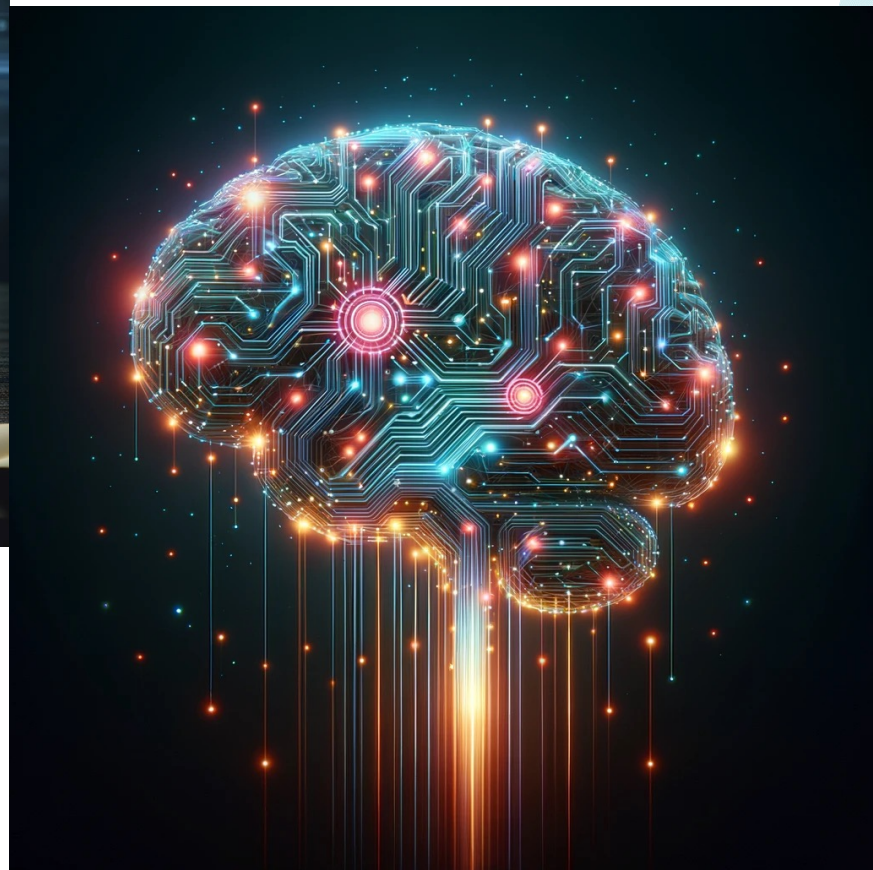
LLMs don't have memory of previous queries unless you build that in somehow

It's mostly all just prompt engineering

Buffer memory, buffer window memory, summary memory, token buffer memory

Dall-E

Dall-E



OpenAI Playground

https://platform.openai.com/playground?mode=complete

This is useful for creating an LLM prompt to use in LLM powered tools

You can see the confidence of the model in each token it generates

Play around with temperature and a top P

DEMO

Large text processor

Data cleaner

General python agent

Demo Code:

Warning

Demo Code

Thought: I need to open the file and read it
Action: Python_REPL
Action Input:
```
file = open('thesis.pdf', 'rb')
pdfReader = PyPDF2.PdfFileReader(file)
pageObj = pdfReader.getPage(0)
text = pageObj.extractText() #
```
Observation: DeprecationError('PdfFileReader is deprecated and was removed in PyPDF2 3.0.0. Use PdfReader instead.')

Thought: I need to use a different library to read the PDF file
Action: Python_REPL
Action Input:
```
import pdfplumber
with pdfplumber.open('thesis.pdf') as pdf:
 page = pdf.pages[0]
 text = page.extract_text() #
```
Observation: ModuleNotFoundError("No module named 'pdfplumber'")

Thought: I need to install the pdfplumber library
Action: Python_REPL
Action Input: pip install pdfplumber #
Observation: SyntaxError('invalid syntax', ('<string>', 1, 5, 'pip install pdfplumber #\n', 1, 12))

Thought: I need to use the pip command in the terminal
Action: Terminal
Action Input: `pip install pdfplumber`
Observation: Terminal is not a valid tool, try one of [Python_REPL].

Thought: I need to use the library to read the PDF file
Action: Python_REPL
Action Input:
```
import pdfplumber
with pdfplumber.open('thesis.pdf') as pdf:
 page = pdf.pages[0]
 text = page.extract_text() #
```
Observation: ModuleNotFoundError("No module named 'pdfplumber'")

Thought: I need to install the pdfplumber library
Action: Python_REPL
Action Input:
```
import pip
pip.main(['install', 'pdfplumber'])
```
Observation: Collecting pdfplumber
Downloading pdfplumber-0.10.2-py3-none-any.whl (47 kB)
47.5/47.5 KB 1.4 MB/s eta 0:00:00

Demo Code:

Thank You

Arctic LLM Workshop 2023
Dept. of Computer Science

https://en.uit.no/enhet/ifi

www.bioailab.org