OpenAI in PHP

How AI works in theory & how to use it in PHP

How LLMs Work

Weekly Topics

1. **How Generative AI & LLMs Work**
2. Prompt Engineering
3. Retrieval-Augmented Generation (RAG)
4. Fine-Tuning a Model
5. Agents
6. Using VUMC's OpenAI API in PHP

Key Concepts

EDGE FOR SCHOLARS

- LLM = Large Language Model
- Tokenization
- Probabilistic Computation
- Technological Limitations
- Plugins & Functions
- Output: Streaming
- Language Support
- Costs
- Need for Review

LLMs

- Convert text into tokens (more coming)
- Turn tokens into vectors & matrices
- Quick computing on Graphical Processing Units (GPUs)
  - Other AI-specific processing units under development
- Energy-intensive calculations on GPUs

Generativity of LLMs

- Based on weights, the model predicts/generates appropriate feedback for prompt
- OpenAI, used by Microsoft Azure, with ChatGPT-4 as the current “gold standard” of AI
- Other LLMs, including open-source implementations have become more common
  - Not supported by VUMC for security/privacy purposes

Tokenization

- For analysis, AI turns text into “tokens” that represent about ¾ of a word.
- Each token represents a meaningful concept
  - *E.g.*, running becomes “run” and “ing”
  - *E.g.*, basketball becomes “basketball”
- Tokens are in turn turned into huge vectors and matrices for analysis (Linear Algebra)

Probabilistic Computation

- With the input of the prior prompt & the previous token, matrix math returns a matrix of recommended items for the next token with a list of probabilities
- Randomization is used, with a random seed & temperature for willingness to accept less-recommended options
- You can also specify a minimum rank in the list to avoid the LLM choosing a “wacky” item

Temperature & Seed: Indices of Determinism

- Temperature = float [0.0 – 2.0]
  - Higher numbers = more creative & less brittle
  - Lower numbers = more deterministic & more brittle
  - 0.7 is a common value for chatbots to avoid becoming brittle
  - 2.0 would cause some crazy responses
  - Specified in the input parameters
- Seed = Integer [1 – *MaxInt*]
  - This is same seed used to feed the randomization code, just as with any random number generator
  - To make items less random, you can specify a seed in the input parameters

Technological Limitations

- Hallucinations
- Mathematics & Reasoning
- Training Data
- One Prompt at a Time

Hallucinations

- Every once in a while, AI will provide a wildly inaccurate response
  - *E.g.*, presented with cancer symptoms, it diagnoses a cold
- Unknown cause – subject of much investigation
  - Bad probability choice?
  - Bad training data?
  - Bad reasoning?
- Gets worse as input gets larger (more tokens)
- Necessitates keeping a human in the loop for review

Mathematics & Reasoning

- AI is terrible at math, especially with larger numbers
- Why? It predicts the next token (*i.e.*, one digit) from the prior digit instead of doing the math algorithm
- *E.g.*, "What is 432587 * 574123?" "432587 × 574123 = 248332228101."
  - Calculator: 248358146201
- Workaround: Functions & plugins
- Better reasoning by adding "Show your work." But still not the right answer!
  - This technique can help any reason-oriented task, not just math!

Mathematics & Reasoning

To manually calculate (432587 \times 574123), we will break it down step by step.
Step 1: Write the numbers vertically for multiplication

```
  432587
×  574123
----------
```

Step 2: Multiply each digit of the second number by the first number
We multiply (432587) by each digit of (574123), starting from the rightmost digit.

Multiply by 3 (units place of 574123):
[432587 \times 3 = 1297761] Place this under the line, aligned with the units place:

```
  432587
×  574123
----------
  1297761
```

...

Step 3: Add everything together
Now, sum up all the rows:

```
   1297761
+   8651740
+  43258700
+ 173034800
+3028109000
+21629350000
--------------
 248332228101
```

Final Answer:
[432587 \times 574123 = 248332228101]

Training Data

- LLMs are only as good as their training data
- No fact-checking capabilities!
- “Garbage in, garbage out”
- Must avoid bad training data or at least to drown it out
- Manual data curation helps
- True for custom input information in prompts, RAG, and fine-tuning (*cf.* coming weeks)

One Prompt at a Time

- AI has no inherent means of storing the history of a conversation (i.e., no memory!)
- It simply replies to the previous prompt
- Workaround: Store summaries of past conversation & provide in background
- More coming in Prompt Engineering presentation...

Plugins & Functions

- LLMs can call functions or plugins to perform certain tasks
  - *E.g.*, check the temperature in a city, look up data in a database, use a calculator
- The LLM decides when to call the plugin/function based on the prompt and the code's description
- Effect: Injects some determinism into probabilism
- Not yet expected to be used in EFS projects

Output: Streaming

- Because AI bases next word on prior token, output is conducive to streaming
  - AI is trying to predict next token sequentially, and outputs them token-by-token onto an output stream
- Thus, chat tools output data real-time
- APIs wait until full response completed

Language Support

- Generally, LLMs are good with code because coders developed it for their own purposes

- Great: Python integrations
- JSON, CSVs, YAML, Markdown
- Query languages (*e.g.*, MySQL, NoSQL)

- Must watch out for language commentary
  - Can request “Without commentary…” to LLM

Cost

- Costs calculated per token based on the processing required
- Costs are minimal per prompt (cents per prompt), but can accrue in high-volume applications
- Costs must be paid attention to throughout these presentations
- Costs: ChatGPT-4 > ChatGPT-3.5-Turbo

Need for Review

- Always need to keep a human “in the loop” for review to avoid hallucinations
- AI cannot be an excuse to turn off your brain
- You can use AI to check itself; however, it’s expensive to check an entire section
  - Still, this automation in ChatGPT-4 remains the gold standard for evaluation
  - Use another instance/session if possible
- Most important lesson for software design: Review AI’s output

Recap

- LLMs
- Tokenization
- Probabilistic Computation
- Technological Limitations
- Plugins & Functions
- Output: Streaming
- Language Support
- Costs
- Need for Review

Weekly Topics

1. ~~How Generative AI & LLMs Work~~
2. Prompt Engineering
3. Retrieval-Augmented Generation (RAG)
4. Fine-Tuning a Model
5. Agents
6. Using VUMC's OpenAI API in PHP

Next Week: Prompt Engineering

OpenAI in PHP

How AI works in theory & how to use it in PHP

How LLMs Work