Synaptic Sidekick: Project Report

Name: Brandon Lewis
Class: Artificial Intelligence
Github Repo link: https://github.com/BranPLewis/synaptic_sidekick

I. Knowledge Added

This section details the content of the foundational papers selected for the agent's knowledge base.

- **CHESSQA: EVALUATING LARGE LANGUAGE MODELS FOR CHESS UNDERSTANDING (arXiv:2510.23948v1)**
  This paper introduces ChessQA, a new benchmark designed to test how well LLMs understand chess, evaluating everything from basic rules and tactical motifs to high-level semantic concepts.
- **A Unified Theory for Causal Inference: Direct Debiased Machine Learning via Bregman-Riesz Regression (arXiv:2510.26783v1)**
  This paper proposes a unified theoretical framework (DDML) that integrates and connects various methods for causal inference, such as Riesz regression, targeted maximum likelihood estimation (TMLE), and matching.
- **Victim as a Service: Designing a System for Engaging with Interactive Scammers (arXiv:2510.23927v1)**
  This work describes CHATTERBOX, an LLM-based system built to automatically engage in long-term conversations with online scammers (like "pig butchering" scams) to study their tactics at scale.
- **What's the next frontier for Data-centric AI? Data Savvy Agents! (arXiv:2511.01015v1)**
  This position paper argues that the next major step in data-centric AI is the creation of "data-savvy agents" that can autonomously and proactively acquire, process, validate, and adapt to data.
- **Greedy Sampling Is Provably Efficient for RLHF (arXiv:2510.24700v1)**
  This paper provides theoretical proof that a simple "greedy sampling" method is surprisingly efficient for Reinforcement Learning from Human Feedback (RLHF), challenging the need for more complex algorithms.
- **Filtering instances and rejecting predictions to obtain reliable models in healthcare (arXiv:2510.24368v1)**
  This paper introduces a two-step method to make healthcare ML models more reliable by first filtering "hard" instances from the training data and then rejecting low-confidence predictions during inference.
- **Anti-concentration is (almost) all you need (arXiv:2510.23719v1)**
  This paper demonstrates that for local random quantum circuits, the property of anti-concentration (a measure of randomness) is essentially equivalent to being a relative-error approximate state 2-design.
- **Methylator: A Modular Framework for DNA Methylation Analysis in Mammals and Plants Using Galaxy (A_Modular_Framework_for DNA.pdf)**
  This paper presents "Methylator," a user-friendly, end-to-end software framework integrated into the Galaxy platform to simplify the analysis of DNA methylation data for both mammals and plants.
- **Conduction velocity of intracortical axons in monkey primary visual cortex grows with distance: implications for computation (arXiv:2510.23391v2)**
  This neuroscience paper re-analyzes data to show that signal speed in V1 intracortical axons increases with distance, suggesting they are fast enough to be a key part of complex visual processing.
- **CONVOLUTION FEATURES OF UNIVALENT MEROMORPHIC FUNCTIONS GENERATED BY BARNES-MITTAG-LEFFLER FUNCTION (arXiv:2510.23452v1)**
  This pure mathematics paper investigates the convolution properties and geometric characteristics of a

specific class of complex functions (univalent meromorphic functions) related to the Barnes-Mittag-Leffler function.

- **Happiness as a Measure of Fairness (arXiv:2511.01069v1)**
  This paper proposes a novel fairness framework for ML grounded in "happiness"—a measure of utility a group gains from an outcome—which can be optimized efficiently via a linear program.
- **NO-RANK TENSOR DECOMPOSITION USING METRIC LEARNING (arXiv:2511.01816v1)**
  This work introduces a "no-rank" tensor decomposition method that uses metric learning (triplet loss) to optimize for semantic similarity rather than traditional rank-based reconstruction.
- **A Pragmatic Way to Measure Chain-of-Thought Monitorability (arXiv:2510.23966v1)**
  This paper proposes practical metrics, "legibility" and "coverage," to measure the monitorability of an LLM's Chain-of-Thought (CoT) reasoning, which can be implemented with an LLM-based autorater.
- **Privacy-Preserving Semantic Communication over Wiretap Channels with Learnable Differential Privacy (arXiv:2510.23274v1)**
  This work presents a secure semantic communication (SemCom) system that uses learnable Differential Privacy (DP) noise to protect sensitive semantic information from eavesdroppers in a wiretap channel.
- **TEST-TIME ALIGNMENT OF LLMS VIA SAMPLING-BASED OPTIMAL CONTROL IN PRE-LOGIT SPACE (arXiv:2510.26219v1)**
  This paper proposes AISP, a test-time alignment method that applies adaptive importance sampling to the pre-logits (penultimate layer outputs) of an LLM to maximize rewards without fine-tuning.
- **THE GORESKY-HINGSTON COPRODUCT ON BASED LOOP SPACES (arXiv:2510.23812v1)**
  This mathematics paper constructs a lift of the Goresky-Hingston coproduct for based loop spaces, providing formulas to compute its interaction with maps between manifolds.
- **Validating Open Cluster Candidates with Photometric Bayesian Evidence (arXiv:2510.23375v1)**
  This paper presents a Bayesian framework (MIMO) for validating open star cluster (OC) candidates by statistically comparing the evidence for a cluster+field model versus a pure field star model.

II. Tools Added

This section describes the custom tools implemented for the agent, their purpose, and their usage.

1. Document Retrieval (retrieve_documents)

- **Files:** retrieval_tool.py, vector_store.py, ingest_pdfs.py
- **Purpose:** This is the primary tool for accessing the agent's knowledge base. It provides a semantic search interface over the PDF documents (described in Section I) that have been processed and stored in a vector database.
- **Usage:** The tool is implemented as a RetrieveDocumentsTool class, which wraps a ChromaRetriever. The agent can call this tool with a natural-language query and a top_k parameter (defaulting to 4) to find the most relevant document chunks. The underlying ChromaRetriever uses a SentenceTransformer model (all-MiniLM-L6-v2) to generate embeddings and a persistent ChromaDB database to store and search them.

2. Query Reformulator (query_reformulator)

- **File:** query_reformulator.py
- **Purpose:** This tool acts as a "helper" to improve the effectiveness of the retrieve_documents tool. It takes a single query and generates a list of deterministic, research-focused variations to broaden the search recall.

- **Usage:** The agent can pass a query to this tool. It does not call an LLM; instead, it uses simple heuristics to append terms like "overview," "survey," "recent advances in," and "site:arxiv.org." The agent can then use this list of reformulated queries to perform multiple retrievals.

3. Machine Learning Web Search (MachineLearningWebSearchTool)

- **File:** web_tools.py
- **Purpose:** This tool provides the agent with the ability to search the live web for information, but constrains the search to a curated list of high-quality machine learning websites. This ensures that retrieved information is from authoritative sources.
- **Usage:** The tool internally uses the DuckDuckGoSearchTool. When the agent provides a query, the tool automatically constructs a more complex search string that filters results to a specific list of sites, including kaggle.com, kdnuggets.com, deeplearning.ai, huggingface.co, and arxiv.org. *(Note: The tool's class name is MachineLearningWebSearchTool, which accurately reflects its function, while its internal name attribute is Utah_Tech_University_website_search.)*

III. Prompts Modified

This section outlines the primary system prompt used to instruct the agent, as well as the specific prompts (descriptions) provided to the agent for tool selection.

1. Main System Prompt

This is the high-level instructions prompt provided to the ToolCallingAgent in run_agent.py to define its core persona and objective.

You are an agent programmed
simply to help users learn about Machine Learning
concepts, topics, tools, and libraries.

2. Tool Prompts (Descriptions)

These descriptions are provided to the model to help it decide when and how to use each tool.

- **retrieve_documents:**
  Use semantic search over the machine_learning dataset to gather relevant evidence.

- **Utah_Tech_University_website_search (MachineLearningWebSearchTool):**
  Searches through machine learning related websites listed for information, URL's.

  (Note: The name attribute of this tool (Utah_Tech_University_website_search) appears to be mismatched with its implementation and description, which both focus on a curated list of ML websites.)
- **query_reformulator:**
  Generate a small list of reformulations / paraphrases for a natural-language query. Useful to expand retrieval recall when searching academic sources.

IV. Example Results

This section provides sample queries and the agent's corresponding final responses to demonstrate its functionality.

Example 1: General Knowledge Query

Query:
"Tell me about Machine Learning, what it is and what its used for?"
Response:
"Based on the available information, Machine Learning (ML) is a field of study focused on creating models that can learn from data. These models identify patterns and relationships within a training dataset to make predictions or decisions on new, unseen data.
The primary use of Machine Learning is to solve complex problems where traditional programming is difficult or infeasible. Some key applications and concepts found in the documents include:

- **High-Stakes Domains:** ML models are widely used in critical fields like healthcare for tasks such as predicting clinical deterioration, aiding in diagnoses, and recommending treatments.
- **Scientific Computing:** In areas like computational neuroscience, where data can be scarce, techniques like metric learning are used. Metric learning helps create meaningful representations from limited data, making it a valuable tool for scientific applications.
- **Data-Centric AI:** There is a focus on developing reliable machine learning systems by emphasizing the quality of the data used for training. This is often referred to as "data-centric AI."
- **Managing Uncertainty:** An important aspect of machine learning is understanding and managing uncertainty in predictions, which is crucial for building reliable systems.

In essence, Machine Learning is a powerful tool for building predictive models from data, with applications spanning from scientific research to critical real-world systems."

Example 2: Comparative Query

Query:
"What is machine learning and how does it differ from traditional programming?"
Response:
"Machine learning is a subset of artificial intelligence (AI) that enables systems to learn and improve from experience without being explicitly programmed. It focuses on developing algorithms that can access data and use it to learn for themselves.
Traditional programming, on the other hand, relies on explicitly programmed instructions. A programmer writes a set of rules or a specific algorithm that tells the computer exactly how to solve a problem. The program will then follow these instructions to produce an output.

Here are the key differences between machine learning and traditional programming:

1. Approach:

- **Traditional Programming:** You write the rules. You provide the program with a set of instructions on how to process input and generate output.
- **Machine Learning:** The machine learns the rules from data. You provide the system with a large amount of data (examples of inputs and desired outputs), and the algorithm learns the patterns and relationships in the data to make predictions or decisions.

2. Data Dependency:

- **Traditional Programming:** The program's logic is independent of the data it processes.

- **Machine Learning:** The model is highly dependent on the data used for training. The quality and quantity of the data directly impact the performance and accuracy of the model.

3. Output:

- **Traditional Programming:** The output is a program that performs a specific task.
- **Machine Learning:** The output is a "model," which is a set of rules and patterns learned from the data. This model can then be used to make predictions on new, unseen data.

4. Adaptability:

- **Traditional Programming:** To change the program's behavior, a programmer must manually update the code.
- **Machine Learning:** Models can be retrained on new data to adapt and improve their performance over time without explicit reprogramming.

In summary, traditional programming is about writing explicit instructions for a computer to follow, while machine learning is about creating systems that can learn from data and make their own rules."

Example 3: Technical Comparison Query

Query:
"How does a convolutional neural network (CNN) differ from a recurrent neural network (RNN)?"
Response:
"Based on my knowledge, here is a breakdown of the key differences between Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs):

Core Distinction

The fundamental difference lies in their architecture and the type of data they are designed to process.

- **CNNs** are designed to process **spatial data**, like images, where the proximity of data points (e.g., pixels) is important. They excel at recognizing patterns and hierarchies of features within a grid-like structure.
- **RNNs** are designed to process **sequential data**, like text or time series, where the order of data points is crucial. They excel at tasks involving memory and understanding context over time.

Detailed Comparison

Feature	Convolutional Neural Network (CNN)	Recurrent Neural Network (RNN)
Architecture	Primarily a **feed-forward** network. Information flows in one direction from input to output through a series of layers (convolutional, pooling, fully connected).	Contains **feedback loops**. The output from a previous step is fed back as input to the current step, creating a form of memory.
Data Processing	Processes data in **spatial hierarchies**. It uses filters (kernels) to scan over the input and detect features, from simple edges to	Processes data **sequentially**. It iterates through the sequence one element at a time, maintaining a "hidden state" that captures

	complex shapes.	information from previous elements.
Key Operations	**Convolution**: Applying filters to input data to create feature maps. **Pooling**: Down-sampling feature maps to reduce dimensionality and create translational invariance.	**Recurrent Loop**: A connection that allows information to persist from one step to the next in the sequence.
Memory	Has **no inherent memory** of past inputs. Each input is processed independently. (It only learns and remembers features during the training phase).	Has **internal memory** (the hidden state) that allows it to remember information from past elements in the sequence. This is why it's suited for ordered data.
Input Data	Typically takes a **fixed-size input**, such as an image of a specific dimension (e.g., 224x224 pixels).	Can handle **variable-length sequences** of input, such as sentences of different lengths.
Use Cases	• **Image Recognition & Classification** • **Object Detection** • **Computer Vision** • **Medical Image Analysis**	• **Natural Language Processing (NLP)** • **Speech Recognition** • **Time Series Analysis** • **Machine Translation**

Analogy

- A **CNN** is like your **visual cortex**. It processes what it sees right now, identifying objects and patterns in a static image without needing to know what you saw a minute ago.
- An **RNN** is like your **short-term memory** when reading. To understand the current word in a sentence, you need to remember the words that came before it.

In summary, the choice between a CNN and an RNN depends entirely on the nature of the problem and the data. If the task involves spatial patterns in grid-like data, a CNN is the appropriate tool. If the task involves understanding sequences where order and context matter, an RNN is the right choice."