Honors Philosophy Thesis
Arina Shah
Advisor: Ned Block

World Models

Large-language models (LLMs) have gained great traction over the past few years. ChatGPT leads as the most popular LLM with around 300 million weekly active users worldwide and 10 million paying subscribers to ChatGPT Plus (BackLinko, 2025). LLMs have also become increasingly integrated across various industries. In healthcare they are used for medical diagnosis assistance (Yang, 2024), patient record summarization (Madzime, 2024), personalized treatment recommendations (Benary, 2023), etc. In finance they are used for fraud detection and risk assessment (Korukanti, 2024), financial analysis and market predictions (Zhao, 2024), personalized financial advice (Lakkaraju, 2023), etc. In the legal industry, they are used for legal research and case law analysis (Izzidien, 2024), contract review and drafting (Kasundra, 2024), etc. In education, they are being used to personalize learning experiences (Abdelnabi, 2025). In HR departments they are being used for resume screening and candidate matching (Kavas, 2024), employing onboarding processes (Adejumo, 2024), etc.

Through the years LLMs like Claude 3.5 Sonnet, GPT-4o, and Llama 3.1 405b, have shown great achievement in intelligence. Scoring between 80-90% on a wide variety of benchmarks assessing commonsense reasoning around everyday events, multiple-choice questions in 57 subjects (professional/academic), and reading comprehension & arithmetic tasks (Kirkovska, 2024). However, these same LLMs struggle in a wide variety of areas. They struggle with gaining logical reasoning and rely on surface-level patterns (Wan, 2024). They frequently make errors in applying rules, even when they can infer them (Mu, 2023). They are prone to generating false or misleading information, especially when dealing with topics beyond their training data (Yao, 2023). While they seem to do well on the commonsense reasoning benchmark, they still struggle with complex reasoning tasks that involve questions at the graduate level (Rein, 2023). Further current literature has found that modern LLMs are poor at faithfully conveying their uncertainty (Kim, 2024), and that better alignment is necessary in order to improve their trustworthiness.

As LLMs become increasingly popular and integrated across industries, addressing their limitations becomes crucial. This paper will first examine the challenges LLMs face in complex reasoning and faithfully conveying uncertainty. Next, it will provide a high-level overview of how LLMs function. Following this, the concept of "world" models will be introduced as a potential solution to these challenges. The paper will then explore implementations of world models by Joshua Tenebaum and Yann LeCun, highlighting the key differences between their respective approaches.

Struggle to Reason and Plan

Reasoning evaluation benchmarks encompass various domains including arithmetic, symbolic reasoning, and commonsense reasoning. A study from the paper, "Large Language Models for Mathematical Reasoning: Progresses and Challenges" highlights how robustness and accuracy decline as problem complexity increases (Ahn, 2024). Further, LLMs struggle to generalize across different question formats. For instance, they show inconsistent performance in varying textual forms and tend to generate different answers for identical questions when queried multiple times. Recently Llama 2 -70B and GPT-4 were tested on their reasoning capabilities using the GPQA dataset. This dataset consists of 448 multiple-choice questions written by domain experts in biology, physics, and chemistry. The questions are highly challenging, with experts (who have or had a PhD) achieving an average accuracy of 65% and non-experts scoring around 34%.

GPT-4 and Llama 2-70B were tested using four learning approaches: zero-shot, few-shot, zero-shot chain-of-thought, and few-shot chain-of-thought learning. In zero-shot learning, the model is tested using only task instructions without any examples. For instance, ask the model “Let's solve this step by step: If John has 5 apples and gives 2 to Mary, who then eats 1, how many apples are left with Mary?” with the instruction “Answer the question”, would be considered zero-shot learning. Here, the model relies solely on its pre-existing knowledge base without any additional examples or context related to the specific question. In few-shot learning, the model is provided with a few examples to guide its response. For instance, before answering the previous question, the model would be a similar example. Question: “If Sarah has 8 candies and gives 3 to Tom, who then eats 2, how many candies does Tom have?” Answer: “Tom has 1 candy (received 3, ate 2, so 3 - 2 = 1).” In a zero-shot chain of thought learning the model is given an additional prompt, such as “Let’s solve this step by step” to encourage reasoning through the problem before providing an answer. In few-shot chain-of-thought learning, the model is first provided with examples that demonstrate step-by-step reasoning on similar questions before being presented with the target question.

The accuracy results for LlaMa and GPT-4 under these different learning approaches were as follows. On zero shot learning Llama achieved approximately 27.6% accuracy, while GPT-4 scored 32.1%. On few-shot learning, Llama had an accuracy of 26.8%, whereas GPT-4 performed better at 38.1% accuracy. On zero-shot chain-of-thought learning Llama reached about a 28.5% accuracy, while GPT-4 improved to 39.5%. On a few-shot chain-of-thought learning Llama showed about a 29.1% accuracy and GPT-4 showed a 28.7% accuracy[1]. These results highlight that the models not only underperformed compared to experts but in many cases

[1] I acknowledge that this benchmark has sparked controversy for having incorrect answers and therefore the trustworthiness of the dataset is questionable. However I still think that it reinforces the point that these models struggle to reason even after being trained on more data.

also performed worse than non-experts. This is an interesting result as the model is trained on a large corpus of data that includes relevant material to physics, chemistry, and biology. Further, even when provided with additional context, examples, and explicit prompts to break down their reasoning step by step, the models continued to struggle with achieving a high accuracy. It seems unclear as to why the models struggle to reason even after their extensive training in domain-specific knowledge. However, exploring the underlying causes of this reasoning difficulty is beyond the scope of this paper. Instead, this work will focus on how world models could offer a potential solution to this challenge.

Struggle to Convey Uncertainty

LLMs face a significant challenge in effectively conveying their uncertainty. Since they often make errors in complex reasoning tasks, it is crucial that they provide users with clear indications of uncertainty, much like humans do in conversation. When people express uncertainty, it prompts others to double-check information. However, LLMs typically lack this capability, often presenting responses with unwarranted confidence. Research by Sree Harsha Tanneru explored *verbalized uncertainty*, an approach where the model is prompted to output a confidence score (ranging from 0 to 100%) alongside its explanations. However, the study found that this method is unreliable, as LLMs tend to overestimate their confidence. Similarly, a study by Gal Yona reinforces the idea that LLMs struggle to faithfully convey uncertainty, highlighting the need for better alignment to improve their trustworthiness. Even advanced techniques such as *chain-of-thought (CoT) prompting*, which provides step-by-step reasoning to enhance transparency, fail to reliably express uncertainty. A study by Mile Turpin demonstrates that CoT explanations can systematically misrepresent the true reasoning behind a model's prediction.

These explanations are susceptible to biasing features introduced in the input, which the model often fails to acknowledge, further undermining their reliability as uncertainty indicators.

The inability of LLMs to accurately represent their intrinsic uncertainty can lead to *overreliance*, where users take actions based on incorrect outputs. As LLMs are increasingly integrated into various industries, overreliance can have serious consequences. A striking example occurred in 2023 when a lawyer unknowingly included fake judicial opinions generated by ChatGPT in a legal brief submitted to the court. To mitigate such risks, regulators have been developing new frameworks to curb overreliance. Article 14 of the Draft EU AI Act explicitly mandates the development and evaluation of measures to prevent this issue. In academic literature, extensive research has focused on fostering human-AI collaboration. Various interventions aim to improve joint decision-making by providing explanations of model behavior, displaying confidence scores to users, or, in some cases, withholding AI-generated outputs when uncertainty is too high. However, these methods are ineffective in preventing overreliance if LLMs' intrinsic uncertainty cannot be reliably quantified. One might argue that instead of relying on LLMs, we should defer entirely to human judgment. However, human decision-making is also prone to bias and uncertainty. Cognitive biases such as overconfidence, base rate neglect, and the conjunction fallacy frequently distort human reasoning, making blind reliance on human judgment just as problematic as reliance on AI-generated outputs. The solution to better decision-making is not choosing between LLMs and humans but rather finding ways to effectively integrate both while addressing their respective limitations.

To effectively implement methods that foster human-AI collaboration, it is crucial to develop reliable ways to quantify the uncertainty of LLMs. Existing research on this topic can be broadly categorized into perturbation-based methods, gradient-based methods, attention-based methods, example-based methods, and natural language explanations. However, uncertainty quantification in LLMs is particularly challenging due to their immense complexity, with billions of parameters. The lack of transparency in their internal mechanisms makes it difficult to identify the sources of uncertainty and measure it effectively. Most of the existing approaches require *white-box access*—a setup where the model's internals can be observed but not altered. However, access to such subsystems is often restricted and not transparent, further complicating efforts to quantify uncertainty. Without reliable uncertainty estimates, decision-makers who rely on LLMs risk being misled, potentially leading to severe consequences. On the other hand, if uncertainty cannot be effectively addressed, AI systems may be abandoned in certain applications, resulting in the loss of valuable predictive insights that could otherwise enhance decision-making.

High-Level Overview of The LLM Architecture

Many people conflate Artificial Intelligence (AI) with LLMs, however, while LLMs are a kind of AI, they are not the only form. AI is an enormous paradigm encompassing any machine that exhibits "intelligent" behavior. AI can be categorized into four paradigms based on functionality. Reactive machine AI: These are systems with no memory so they cannot use past experiences to influence their decision-making process. They operate solely based on the current input they receive. They also follow a set of predefined rules or instructions to determine their actions and are designed to perform very specific tasks. An example is Netflix's recommendation engine, which suggests content based on predefined patterns without learning from past interactions. Limited memory AI: These systems can recall past events and outcomes, allowing them to monitor specific objects or situations over time. LLMs like GPT fall into this category since they leverage previous data to generate responses. Theory of Mind AI: This is a theoretical form of AI that, if realized, would be capable of understanding the thoughts and emotions of individuals. However, no existing AI system possesses these capabilities. Self-aware AI: This is another theoretical category that would possess super AI capabilities. Like Theory of Mind AI, this paradigm remains purely speculative.

Not all AI paradigms rely on machine learning, meaning they do not all use the same architectural principles as LLMs. Machine learning is a subfield of AI focused on pattern recognition and data-driven decision-making. Deep learning, a specialized branch of machine learning, deals with processing text, images, and code using artificial neural networks (ANNs). Multimodal LLMs, which process multiple data types, fall under the deep learning category. This AI hierarchy raises an important question: Can the architecture that LLMs possess lead us to Artificial General Intelligence (AGI)? There are many definitions of AGI, but in an interview with the *HardFork* podcast by *The New York Times*, DeepMind CEO Demis Hassabis described AGI as a system that is "generally capable and able to do any cognitive task that humans can do." Later in the paper, we will see how Tenebaums' strategy to implement world models utilizes a hybrid structure, not just relying on deep learning, but combining symbolic reasoning, probabilistic inference, and deep learning. However, LeCun's approach will fall squarely within the deep learning paradigm, as his architecture uses self-supervised learning. Self-supervised learning is where a model generates its own labels from unlabeled data to train itself, which is what LLMs do.

Now that I have characterized where LLMs fall in the greater scheme of AI let's dive into how they work. LLMs are built using deep neural network (DNN) architectures that consist of multiple layers of transformers. DNNs are a type of artificial neural network (ANN). ANNs are computational models that consist of interconnected units or nodes called neurons, which are organized in layers. An ANN consists of an input layer, one or more hidden layers, and an output layer. DNNs are considered deep ANNs because they have multiple hidden layers between the input and output layers. In these models, each neuron receives an input, processes it, and passes its output to the next layer of neurons. Each hidden layer has a fixed set of weights and these weights are multiplied by the input vectors producing an output vector that encodes increasingly complex and abstract information about the input sequence. Essentially, each layer transforms its input data into a higher-level representation. The transformation involves multiplying the original input by the learned weights of the hidden layer, producing a raw score that determines how much each element should contribute to the final output. An activation function is then applied to capture more complex patterns, allowing the network to capture complex patterns beyond simple relationships. Transformer models introduce a critical mechanism called self-attention. Self-attention enables the model to weigh the importance of different parts of the input sequence differently for each output element. This allows the model to capture long-range dependencies and contextual relationships within the data. LLMs are predictive models trained on vast amounts of raw data to learn optimal weight values for each layer. The training process involves several iterative stages including tokenization, embedding, forward propagation, loss calculation, backpropagation, and optimization. This process allows for the model to gradually refine its weight parameters and is outlined in more detail below:

1) The input sentence or statement is tokenized. Tokenization is the process of dividing a text into smaller units known as tokens. This is important because the text data can be represented numerically which is the format appropriate for statistical and computational analysis.
2) Each token is then mapped to a corresponding vector that combines word embeddings and positional encoding. Word embeddings capture the semantic and

syntactic properties of words. Positional encoding preserves the order of tokens in the sequence. This step ensures that words with similar meanings have similar representations while also preserving the sentence structure.

3) The corresponding vectors of these tokens then enter the first hidden layer, where they are multiplied by a fixed set of learned weights within the hidden layer. These weights are determined during training through a method of backpropagation. Backpropagation is an algorithm by which the model updates weights to minimize the error in its predictions. (It is important to note that the weights start out random and then are updated in the backpropagation process) This results in three different vectors: queries (Q), keys (K), and values (V). Q represents the token's role in retrieving relevant information. K represents how much attention a token should receive. V represents the actual information contained in the token.

4) The attention scores are calculated by taking the dot product of the query vector with all key vectors. This produces an attention score that indicates how much focus each token should have on other tokens in the sequence when generating the output.

5) To normalize the raw attention scores into probabilities, a softmax function is applied. The softmax function transforms the scores into values between 0 and 1, ensuring they sum to 1 across all tokens. This step re-scales the scores into a probability distribution, making them interpretable as attention weights. The model assigns higher probabilities to more relevant tokens to determine the relative importance of each value in the sequence.

6) The attention weights are then used to compute a weighted sum of the value vectors, generating the final output of the attention mechanism for each position in the sequence. This process enables each output element to integrate information from all positions in the input, with each contribution scaled according to its relevance, as determined by the attention scores.
7) The attention output is passed through a fully connected feedforward network, which applies an additional transformation to improve the model's understanding of the context. This step introduces more complexity and refines the representation before predicting the next token.
8) The output from one transformer layer is passed to multiple subsequent layers, where similar attention and feedforward processing occur at deeper levels of abstraction. The deeper layers capture more abstract relationships between words in the sequence.
9) The final output from the last transformer layer is passed to a linear layer, which maps it to a vocabulary-sized probability distribution using another softmax function. If the model has a vocabulary of 50,000 words the output will be a vector of size 50,000 with each value representing a raw score for a word in the vocabulary. The token with the highest probability is selected as the predicted next token.
10) In models like GPT, the prediction process is repeated iteratively. The newly predicted token is appended to the sequence, and the model runs through the entire process again to generate the next token. This continues until the end of the prediction is reached.

To demonstrate how an LLM would make an inference, imagine I asked, “How many states are there in America?” The sentence would first be tokenized into “["How", "many", "states", "are", "there", "in", "America", "?"]. Each token would be mapped to a word embedding and a positional encoding, "states" might be represented as [0.12, 0.85, 0.63, ...], "America" might be [0.45, 0.32, 0.77, …]. The model would assign queries (Q), keys (K), and values (V) for each token. It may recongize that “states” and “America” are strongly related and should be given higher attention weights when predicting the answer. So when attention scores are converted into a probability distribution using the softmax function and we focus on the word “states” the model would assign attention weights like: "America" → 0.45, "many" → 0.30, "are" → 0.15, "there" → 0.10. This shows that “America” is the most relevant word when thinking about “states”. The processed information passes through multiple transformer layers, where the model refines its understanding by stacking attention outputs. After several layers, the model learns that the phrase “How many states” is a common way to ask for a number. The word “America” suggests the model should recall facts about the United States. Then the model’s final layer would generate a vocabulary-sized probability distribution over possible next tokens. In our example the word 50 would have a 92% probability, 51 would have 6%, 49 would have 1%, Washington would have 0.5%, Canada would have 0.1%, etc. Since 50 has the highest probability the model would select it as the predicted next token. If this were a long-form answer, then the model would repeat the process, generating the next word based on “50”, and continuing until a stopping condition, like punctuation, is met.

Large language models also incorporate “reinforcement learning from human feedback” (RLHF) to improve their alignment with human values. While the core task of next-token prediction enables models to generate text, it does not inherently ensure outputs that align with normative goals. To address this, RLHF refines the model by fine-tuning it on human-preferred responses. There are three stages of RLHF: 1) Human crowd workers rank multiple different responses produced by the model according to qualities such as helpfulness, harmlessness, and honesty. 2) The collected rankings are used to train a reward model, which assigns a numerical score to a given response based on its perceived quality. 3) The LLM undergoes a reinforcement learning process where it generates responses, receives scores from the reward model, and updates its parameters to maximize the predicted reward.

What is a World Model?

LLMs' main objective is to predict the next token but it is unclear whether this objective allows for them to understand the world. Thinkers like Yann LeCun, Raphael Milliere, and Joshua Tenenbaum believe that world models are necessary for AI systems to reach human-like intelligence. However, what exactly a world model is and ought to look like has been a difficult concept to tease out as recent machine learning papers proposing new architectures often struggle to clearly articulate what constitutes such a model. There seems to be agreement that in general a world model is an internal representation that simulates aspects of the external world. An internal representation, "refers to the way our minds store and process information about the external world, essentially creating a mental model of reality based on our perceptions, memories, and beliefs, which can include visual images, sounds, feelings, and other sensory modalities" (Harvey, 2001). Creating a mental model means developing a representation in your mind of how something works, based on the data that you acquire through experiences and observations. It is kind of like building an internal picture of a system or concept to guide one's thinking and actions. World models take inspiration from the mental models of the world that humans develop naturally. Humans develop their "world models" by continuously gathering sensory information from their environment, engaging in interactions, and forming mental models that evolve and adapt over time based on new experiences. This allows them to make predictions and navigate new situations based on learned patterns and casual relationships. A system has a world model if it can understand, interpret, and predict phenomena in a way that reflects real-world dynamics, including causality and intuitive physics. Having an internal representation of the external world enables systems to simulate possible future scenarios more effectively because it provides a structured way to reason about cause and effect, track changes over time, and anticipate outcomes based on past experiences. A world model would be able to encode causal mechanisms that allow it to simulate outcomes of different actions, rather than just picking up on statistical correlations. Further, the model can anticipate long-term outcomes better because it would be able to internally simulate different possible futures using its mental model.

Some researchers argue LLMs lack world models because their primary objective is next-token prediction rather than building structured, casual representations of the world. LLMs process language by breaking down the text into discrete tokens and predicting the most likely next token based on statistical patterns in vast training data. Granularizing inputs into tokens inherently loses information that could be crucial for understanding the deeper causal mechanism behind events. World models aim to construct coherent, structured representations of the environment, forming casual models of how things work rather than just predicting sequences. Further, because LLMs are trained to recognize and generate patterns in text, they excel at correlational learning - predicting likely words based on context. However, it seems that causal reasoning requires more than recognizing statistical association, it necessitates understanding the underlying mechanisms that drive events. This may demand a structured model that explicitly encodes cause-and-effect dynamics. Moreover, world models are meant to explicitly simulate environments, mental states, and counterfactuals, and LLMs do not explicitly simulate mental states. These researchers also argue that world models require dynamic updating. This means updating beliefs when encountering new data, revising prior knowledge when contradictions arise, and tracking evolving contexts such as world events, scientific discoveries, or shifting social norms. This would make the system more similar to how humans learn as they constantly review their mental models when they acquire new experiences. However, a core limitation of LLMs is that their parameters are fixed after training. For instance, if an LLM trained in 2021 is asked, "Who is the current President of the United States" it would give an incorrect answer as it could not update with new information. Humans are adaptive learners, continuously updating both their beliefs and their confidence levels based on new information and experience. In contrast, LLMs are static learners, as their knowledge remains fixed after training, preventing them from dynamically adjusting their understanding over time.

Other researchers argue that LLMs actually do have internal representations that resemble a world model, even if they do not have explicitly structured world models. Through training on massive data corpora, LLMs implicitly capture statistical patterns that can appear as rudimentary causal reasoning. This suggests that rather than lacking a world model altogether, LLMs may be learning an implicit, data-driven approximation of one. This perspective suggests that models do not require a separate architecture to explicitly encode a world model; rather, the world model emerges naturally from the model's learning process. This idea is highlighted by Milliere in his paper."A Philosophical Introduction to Language Models Part II: The Way Forward." Othello-GPT is a model trained solely on sequences of moves from Othello games without being explicitly provided the rules or board structure. It is fundamentally based on a transformer model, similar to the GPT-style architecture. The input is a sequence of Othello game moves and the goal is to predict the next move in a sequence of Othello moves. The model was able to learn patterns of legal moves and strategic play without ever seeing the explicit game board. The claim Milliere makes is that Othello-GPT learned an internal representation of the board. Researchers used non-linear probes to analyze the model's activations (neuron outputs). They discovered that Othello-GPT implicitly reconstructs the board state within its internal neuron activations. This means that the model wasn't just memorizing sequences but had learned to infer and maintain a world model of the board. Othello-GPT was able to form a coherent, structured representation of its environment, the Othello board, from just training on legal move sequences. It seems plausible that being able to reconstruct the board state and rules means that the model developed a mental model of Othello's environment. In this example, world models do not need to be explicitly structured within the architecture itself. A transformer model, trained in an autoregressive fashion, can still implicitly develop a world model without being designed for it. This suggests that word models could be emergent and that LLMs could develop a world model without completely altering their architecture. However, a major challenge in this idea is the fact that the Othello-GPT "world model" is not actually a model of the world but the Othello game board. LLMs would have to establish a mental model of the world which may be difficult to achieve due to the scale and nature of what the emergent internal representation would have to look like.

Implementing an architecture that incorporates a world model could address several key limitations of current LLMs. Unlike standard LLMs, which rely purely on next-token prediction, world models enable the anticipation of future outcomes and the ability to update knowledge dynamically based on new experiences. This would enhance an LLM's ability to reason through complex tasks, plan ahead, and engage in long-term coherent thinking rather than just producing text based on short-term context. Further world model architectures would allow for systems to track their own uncertainty by maintaining an internal model of reality that can be queried and adjusted dynamically. This would prevent over-reliance by users, as the system could quantify and communicate uncertainty rather than producing overconfident but potentially misleading outputs. Moreover, it is crucial to establish what it truly means for an AI system to possess a world model and to establish how such models should be designed and implemented. This understanding will help determine whether world models must be explicitly built into the systems architecture or if they can emerge naturally through the learning process. For the remainder of this paper, I will examine the explicit world model architectures proposed by Yann LeCun and Joshua Tenenbaum to explore how world models should be implemented. By analyzing their approaches, I aim to identify key differences in their perspectives on how a world model ought to function.

Yann LeCun's Notion of a World Model

According to Yann LeCun, a world model is an internal model that an AI system uses to simulate and predict the dynamics of the environment it operates within. LeCun's architecture integrates the word model with several other modules to enhance its functionality. It includes a perception module that senses and interprets the environment, a configurator that adjusts the system's level of detail based on task requirements, and task and guardrail objectives that guide decision-making while ensuring safety and compliance. LeCun formalizes this idea in more concrete terms by defining it as a system that integrates past observation, internal state tracking, action proposal, and uncertainty modeling. The model is given 4 inputs. It is given current observations through data x(t) and a state estimate of the world (s(t)). This is a representation of what the model believes to be true about the current state of the world/environment at a given time. The estimate is derived through "memory" of important past information that isn't directly observable in the current moment. For instance, when playing chess we would say that the current position is the observation. Previous information like what moves were played, patterns or strategies the opponent seems to be following, and general rules of the games would provide context to the current observation. The model is also given an action proposal that considers what actions might be taken (a(t)). Yann LeCun doesn't explicitly specify how the action proposal is determined, but usually, they could be determined by search algorithms and direct inputs from users. However, it would seem that the selection of these actions is typically handled by a separate component of the system, which is out of the scope of the paper. Lastly, the model is given a latent variable (z(t)) which accounts for uncertainty, by introducing noise or bias. This uncertainty represents all the unknown factors that could influence what happens next. For instance, if I was traveling to Spain it would account for weather delays. This variable is determined through sampling from a distribution or varied over a set. It gives parameters for the set of plausible predictions.

With these variables, the model encodes the raw observation into representations, (h(t) = Enc(x(t)). Representations are just transformed versions of raw data that are more suitable for processing by the model. For instance, an observation of an image may just have pixel values but the representation might encode edges, object locations, texture patterns, and other features that are useful for prediction. Then it computes a prediction about the next state based on the given variables (Pred(h(t), s(t), z(t), a(t))). Pred() is a trainable determinist function that predicts the next state of the world given current information. This function is typically implemented as a neural network that can learn from data. The model learns from sequences of triplets, so it gets an observation, generates an action, and then the next observation. Imagine a scenario where a robot needs to move object A to location X, but object B is in the way. Through the world model process the model would observe positions of A, B, and X. Then it would encode these in representations h(t). Then it would simulate multiple action sequences. For instance, try to move B first, then A, or try to find a path around B, or try to push both objects together. For each simulation, the model would use latent variables to consider uncertainties. Predict outcomes and potential problems. Then produce estimates of the likelihood of success. It would choose the action sequence with the best-predicted outcome and continuously update predictions as actions are taken.

LeCun proposes using a joint embedding predictive architecture (JEPA) to produce the above schema. It is the core mechanism for learning and predicting abstract representations of the environment. JEPA is designed to learn abstract representations of the world in a self-supervised manner and predict missing or future information without directly reconstructing raw sensory input. Unlike generative models that rely on probabilistic distribution to explicitly model uncertainty, JEPA learns deterministic embeddings that capture abstract representations of the world. Deterministic embeddings mean that JEPA would always produce the same embedding, there is no randomness or probability involved in how it encodes information. It directly learns fixed representations that describe the world in a predictable way, because according to LeCun, "probabilistic models are intractable in high-dimensional continuous domains". JEPA operates in an abstract space where relationships between states are learned directly. For instance, instead of breaking text into discrete tokens, JEPA would make entire sentences or phrases in a continuous, structured embedding space. These embeddings would aim to capture the sentence's abstract meaning and relationships between different concepts. For instance, given the sentence, "The cat jumped over the fence because it saw a bird" the model would learn an embedding that captures the relationships between the words. In LLMs, each word is mapped to a vector based on how statistically often they occur. “Cat” and “jumped” might have a strong learned relationship because they frequently appear together.“Because” would not necessarily indicate a causal relationship since the model just predicts the most likely words based on patterns. This is why if we removed “saw a bird”, an LLM may struggle to infer what triggered the jump unless it had been trained on enough similar experiences. However with JEPA, “because” would allow the model to infer a causal structure. Instead of just encoding the individual word, JEPA constructs an internal graph-like representation. This means that if we removed the word “saw a bird” JEPA would still be able to infer that some external stimulus caused the cat to jump. The system would then save the abstract state representations of such embeddings. For instance state 1: “cat on ground”, action: “jump”, state 2: “cat over the fence”.

JEPA is integral for the representation learning component of the world model where $(h(t)) = Enc(x(t))$. It specializes in learning these structured representations efficiently because it maps the input and output into the same abstract embedding space. The idea is to map both the input x (current observation) and the target y (future observation) into a shared representation space and then perform predictions in that space, rather than in the raw data space. By performing predictions in the shared representation space the model learns to capture only the essential structures needed for prediction, eliminating the need to store unnecessary details. The embeddings also enforce consistency between related inputs and their logical continuations. Let's imagine that the input is “the glass fell off the table” and the output is “the glass broke”. JEPA would learn that the transition between “glass fell” and “glass broke” is a structured, predictable relationship. This is powerful because even if the raw wording or format of the inputs differed, JEPA would still be able to recognize semantically equivalent inputs. So if the input was “there was a loud noise” or “the glass cracked” JEPA would understand they are all valid representations of the next state. This allows for the system to make high-level inferences rather than memorizing patterns which is helpful in complex reasoning tasks. Essentially the way JEPA is structured the embeddings capture casual relationships within the data, which allows the model to reason about cause-and-effect relationships and dependencies, and predict future states. This approach still utilizes a large amount of training data however what is done with the training data diverges from the LLMs approach. Rather than focusing on generating data distributions and patterns, JEPA serves to learn a structured representation of the world by predicting in an abstract representation space.

JEPA differs from traditional LLMs in that its primary learning objective is to predict future or missing states in an abstract representation space. Instead of processing language through sequential token prediction, JEPA constructs and reasons through structure relationships in its internal world model. Rather than viewing words as isolated units within a probability distribution, JEPA learns abstract, structured knowledge representations that allow it to directly infer answers rather than simply generating the most probable next token. The model organizes relationships between concepts in a structured graph-like manner, enabling it to retrieve explicitly stored facts rather than relying on probabilistic word associations. For instance, if JEPA has previously encountered the fact that “the U.S has 50 states”, it can directly retrieve this structured knowledge from its internal representation. However, if the exact fact is not stored explicitly, JEPA does not default to probabilistic guessing like a traditional LLM. Instead, it infers the answer using related knowledge, such as the “U.S is a country that has internal divisions”, “Canada has 10 provinces”, and “The European Union has 27 member states”. Using this internal knowledge structure, JEPA can estimate a reasonable range for the number of U.S.

states based on comparative reasoning. It is important to note that JEPA does not explicitly store knowledge as a graph. Instead, it learns structured relationships between concepts through its embedding process, where related concepts are positioned close together in a high-dimensional space. This structure allows JEPA to infer missing information by aligning embeddings rather than relying on explicit symbolic rules or pre-defined retrieval mechanisms. Moreover, due to the nature of how it predicts, JEPA is aware of the internal representations it possesses and those it lacks. This allows it to convey uncertainty more effectively to the user, reducing the risk of hallucination or misleading outputs.

It is worth mentioning that LeCun believes that LLMs are a simplified version of the JEPA model. LLMs work directly with raw inputs however these inputs are not encoded into a representation. In Lecun's world model, there is complex state tracking and in an LLM this state tracking is just a window of past tokens. World models consist of full prediction with actions, current state, past state, and latent variables, they explicitly model actions and their effects. However LLMs only predict the next token using the past state and latent variables, they have no concept of actions. World models can handle continuous values and physical states, however LLMs only work with discrete tokens. World models can learn complex state encodings but LLMs just use raw token history. World models predict full state transitions however LLMs only predict the probability distribution over the next token. World models can learn complex memory structures but LLMs have a fixed-size window of past tokens. This shows why LLMs may be limited compared to full-world models, they can't truly simulate physical systems, can't model continuous state spaces, and are limited to pattern matching in token space.

Tenenbaum's Notion of a World Model

Tenebaum's proposal is intended to create a new computational framework, rational meaning construction, which includes building a hybrid architecture that explicitly encodes world models into the architecture. The program then performs inference on these models to predict and plan, explain and evaluate, learn, and teach, conditioned on goals, observations, and background knowledge. The world model involves a probabilistic generative framework that not only encodes existing knowledge but also allows for the structured expansion of this knowledge based on new information received through language. Let's imagine an example of dynamic physical scenes. We want to answer questions like, "How fast does the blue ball move after collision?". The first step would be to create a generative world model listing background information relevant to the domain, such as mass, shape, friction, and kinematic states of the balls and blocks on the tabletop. Each object's physical properties such as mass and friction are modeled as stochastic variables (random). This allows for a predefined distribution but also for the properties to vary given a new condition. When a specific query is included in natural language, "imagine that the red object is a ball, and is pretty heavy," the descriptors are translated into conditional expressions with the probabilistic program. For instance "pretty heavy" may translate to "mass 2* greater than initial mass". The world model is then updated to represent the distribution of the relevant physical properties. For instance, the original world model indicated that the red ball has a mass of 140 grams. Now that mass would be updated to 280 grams. Using a physics engine integrated within the probabilistic framework, the model would simulate physical interactions over time, considering the initial states and forces derived from probabilistic conditions. The engine would then calculate the resulting velocities of the objects involved in the collision, updating these values step-by-step through the simulation timeline. After the simulation, the model would extract the velocity of the ball immediately following the collision, which is the direct answer to the query. However since the model needs to account for uncertainties and variability in the initial conditions, rather than making a single deterministic prediction, it assigns probabilities to different future scenarios.

Tenenbaum's generative word models integrate the probabilistic language of thought (PLoT). According to a paper called "Concepts in a Probabilistic Language of Thought" written by Goodman, Tenebaum, and Gertstenbert, "PLoT is a system in which representations are built from a language-like composition of concepts, and the content of those representations is a probability distribution on world states." It is an extension of the Language of Thought (LoT) hypothesis, which posits that human cognition relies on structured, symbolic representations similar to the grammar and rules of a language. PLoT incorporates probability theory to allow the AI system to infer, generalize, and make decisions in uncertain environments. PLoT defines how knowledge is structured in the AI's internal world model. Instead of memorizing pixel-level information, PLoT enables the model to represent concepts as structured, compositional symbols with probabilistic relationships. For instance, given the statement "a cup is a container that holds liquid" the model encodes the object "cup", then the properties, "container", "holds liquid", then the relationship, "cup -> contains -> liquid". The symbolic structure is compositional, meaning it can be reused and extended to other concepts like, "bowl", "bottle", or "teapot". It translates the raw data into structured symbolic representations that follow logical and grammatical rules. Further, instead of memorizing that "cups hold liquid" as a fixed fact, PLoT adds a probabilistic exception to the general rule, "cups hold liquid (with probability 75%)" in the case that the cup is actually holding markers. Unlike LLMs that tend to memorize exceptions explicitly and struggle with outliers, PLoT dynamically applies structured rules with probabilities, allowing it to reason about edge cases without retraining. PLoT also assigns probabilistic relationships to uncertain attributes. This allows for reasoning under uncertainty and making flexible predictions. For instance the probability of an animal flying given it is a bird is 95%. However the probability of an animal flying given it is a penguin is 0%. Now, if PLoT encounters a new bird species it has never seen before, it can infer a likely probability for its ability to fly based on its internal representations of other birds that can and cannot fly. Moreover, PLoT models cause-and-effect relationships explicitly rather than relying on statistical correlation. For instance, it has the general representation given from an input, that "if you drop a cup, it will break": "Drop(object) -> Impact (surface) -> Break(Object) (with probability 90%). Now, if asked "What if the cup were made of rubber?" PLoT would adjust the primary general representation to "Drop(object) -> Impact (surface) -> Break(Object) (with probability 5%). This enables counterfactual reasoning, meaning that the model can test alternative realities rather than just predicting based on past data. PLoT differs from deep learning because it learns from few examples, uses symbolic, compositional structures, uses probability to make flexible, uncertain predictions, and simulates casual interactions and counterfactuals.

Tenenbaum's architecture integrates PLoT with LLMs to translate natural language utterances into code expressions in a probabilistic programming language. Probabilistic programming is a family of mathematical languages and actual programming languages that allow researchers to take the conceptual idea of rational world modeling, inference, and decision and turn it into practical engineering terms that can be models of human minds as well as more human-like AI. To do this, the programs build on symbolic languages to express abstract knowledge for modeling the world. They use those symbolic languages to express probabilistic models where you can be uncertain about everything: the state of the world, how the world works, how the different kinds of data you are receiving are connected to the underlying state of the world, and then the ability to make joint inference about all those different sources of uncertainty as the basis of perception for reasoning, planning, and learning. Probabilistic programs allow for this to happen by producing thoughts that the system can internalize and externalize through probabilistic inference. Imagine a scenario where a model observes the following human behavior, “Alice reaches for an umbrella before stepping outside”. An LLM might predict text-based associations, such as: “People carry umbrellas when it rains”. However, PLoT translates the observation into a structured probabilistic program, “let’s [2]assume a 70% chance of rain”. If it is raining there is a 90% chance someone takes an umbrella and if it is not raining there is a 10% chance they take an umbrella for another reason (sunshade). Then the system performs Bayesian inference to update beliefs given new data, like if Alice does take an umbrella. Once the model internalizes this reasoning, it can use it in future interactions, for instance, if Bob later asks, “Should I bring an umbrella” the model would re-run the probabilistic program that was run on Alice. Through this schema Tenenbaums model can infer hidden causes from observed effects, model uncertainty explicitly, and generalize learned knowledge instead of memorizing past data.

Tenebaum is proposing a unified model framework that incorporates various aspects of language processing. However, while the framework is unified, it is likely designed to be modular, where specific sub-models or components handle different aspects of the cognitive processes. The model would determine which sub-model is relevant by using natural language to provide clues about the relevant domain. The model would have generative world models encoded with a set of assumptions, rules, and representations based on theoretical knowledge, empirical data, and expert input over the most relevant and common domains. Further, as the

[2] All of these statements would be written in a high level programming language like Python. However for the sake of clarity I wrote the statements as natural language expressions.

system interacts with users it would gather new information that would refine its existing world models. If the interaction is not well-documented, the system notes this as a gap and may be updated. Tenenbaum's proposal includes taking new information and adapting it into the model's architecture. Further, it would seem intractable to hard code generative world models for all domains, however, Tenebaum also gives a proposal for the architect's ability to create new world models given the existing ones. This approach signifies a shift from scaling up (LLM's approach) to growing up. Rather than inputting vast amounts of training data the model uses explicit world models and then uses probabilistic programming to update the world models to match the scenario.

Key Differences Amongst Two Architectures

Both LeCun and Tenebaum argue that world models are essential components of AI systems, as they are fundamental to human cognition. At their core, world models serve as internal representations of reality, a concept on which both researchers agree. They also share a common stance on moving beyond traditional deep learning techniques—rather than relying solely on pattern matching, they advocate for models that can learn structured relationships between raw inputs. However, their approaches to world models diverge in key ways. LeCun proposes an architecture that he describes as a more advanced and robust extension of what LLMs do in theory. In contrast, Tenenbaum emphasizes the integration of explicit world knowledge and reasoning capabilities into AI systems. His work focuses on combining neural language models with cognitively inspired symbolic modules, such as physics simulators, graphics engines, and planning algorithms. For the remainder of this paper, I will highlight two notable differences between their proposed world model architectures. 1) LeCun takes an empiricist approach, favoring learning from experience, while Tenenbaum adopts a nativist perspective, advocating for the inclusion of innate structures and prior knowledge. 2) LeCun moves away from probabilistic approaches, whereas Tenenbaum heavily incorporates probabilistic modeling to capture uncertainty and support reasoning.

Nativism Vs Empiricism

In the debate over how intelligence emerges—both in humans and artificial systems—Joshua Tenenbaum and Yann LeCun represent two distinct philosophical perspectives. Tenenbaum's approach aligns with nativism, which posits that certain cognitive structures and knowledge are innate rather than acquired purely through experience. LeCun, on the other hand, follows an empiricist approach, which argues that knowledge is entirely learned from experience without the need for pre-existing structures. These contrasting views shape their respective AI architectures: Tenenbaum's PLoT model integrates symbolic, structured knowledge from the outset, whereas LeCun's JEPA architecture emphasizes self-supervised learning from raw data.

Nativism holds that certain abilities, cognitive structures, or core knowledge are pre-wired into the brain rather than acquired solely through experience. Tenenbaum's probabilistic and symbolic approach mirrors this idea, as his architecture incorporates cognitively motivated symbolic modules that enable structured reasoning. Like nativists, he believes that the human mind comes pre-equipped with fundamental concepts and learning mechanisms, allowing for rapid generalization from minimal data. Within the nativist tradition, several theories have been proposed to explain innate cognition. Some nativists argue that domain-specific knowledge is built into the brain at birth, helping infants navigate the world from their earliest moments. For example, Elizabeth Spelke's Core Knowledge Theory suggests that babies are born with innate knowledge in physics (such as object permanence), number sense, and social cognition. Others, like Noam Chomsky, propose that while humans may not be born with explicit knowledge, they possess built-in mental structures that facilitate learning. His Universal Grammar (UG) theory suggests that humans are born with a Language Acquisition Device (LAD)—a cognitive structure that provides a framework for learning language. This explains why children can rapidly acquire complex languages without systematic training, a phenomenon that would be difficult to explain under a purely empiricist view. Tenenbaum's proposal follows this nativist line of thinking. He argues that in the first year or two of life, humans exhibit intelligence even before they acquire language, demonstrating abilities such as helping others, recognizing intentions, and interacting meaningfully with their environment. This suggests that humans possess built-in cognitive structures that facilitate learning from minimal data and allow for robust generalization. In an interview, Tenenbaum explicitly states, *"There's a key intelligence that's built in from the beginning, not what emerges at the end. (...) Our minds are built from the very beginning to have models of the physical and social world."* This view directly contrasts with LeCun's empiricist stance.

Empiricism, as defined by the Stanford Encyclopedia of Philosophy, asserts that *"we have no source of knowledge in S or for the concepts we use in S other than experience."* This means that knowledge is not innate but instead acquired entirely through sensory experiences and interaction with the environment. This is the perspective that LeCun, LLMs, and Millière's architectural ideas align with. The empiricist tradition has deep philosophical roots. John Locke famously argued that the human mind is a tabula rasa (blank slate), gaining knowledge only through experience. Later, B.F. Skinner and John Watson developed behaviorist theories, proposing that all behavior is learned through conditioning. For instance, in language acquisition, children learn words through reinforcement—hearing speech and receiving positive feedback. This is strikingly similar to how LLMs improve performance through Reinforcement Learning from Human Feedback (RLHF), where models are refined based on human preference rankings. It is important to note that empiricism does not reject the idea that the brain has structures that support learning. Rather, it denies that these structures contain innate knowledge. Instead, empiricists argue that the brain has general learning mechanisms—such as memory, pattern recognition, and neural plasticity—that allow knowledge to be built entirely from experience. This philosophy is directly reflected in neural networks, which start with randomized weights and only gain knowledge by processing training data. Unlike Tenenbaum's PLoT model, which incorporates structured symbolic modules, neural networks do not have built-in functions like a physics engine or an innate conceptual framework. Instead, their knowledge emerges through data-driven learning, much like an empiricist would argue for human cognition. LeCun's world model architecture exemplifies this empiricist approach by emphasizing self-supervised learning. Rather than integrating explicit symbolic reasoning systems or predefined physics engines, his model observes massive amounts of raw data and utilizes the JEPA encoding structure to capture causal relationships. Instead of explicitly defining the rules of physics, LeCun's model learns these rules purely from data, discovering patterns in a different representational space than traditional LLMs. This aligns with empiricism, which holds that the mind starts as a blank slate, and knowledge is constructed solely from experience.

JEPA vs PLOT

JEPA and PLoT are key components of both LeCun and Tenenbaums architectures respectively. However, they differ greatly with respect to how they form internal representations make predictions, and handle uncertainty. JEPA's primary goal is to learn structured abstract representations and predict missing/future states in a high-dimensional space. PLoT's primary goal is to use probabilistic reasoning and symbolic structures to model knowledge, uncertainty, and casual relationships. Notably, LeCun’s architecture does not rely on probability distributions, whereas Tenenbaums architecture explicitly incorporates probabilistic methods to handle uncertainty. One of the fundamental differences between these models is how they represent knowledge internally. JEPA learns abstract, compressed embeddings, encoding knowledge in continuous vector representations where similar concepts are positioned close together in high-dimensional space. In contrast, PLoT organizes knowledge into discrete symbolic structures, representing information through logical rules, objects, properties, and relationships. This means that while JEPA generalizes knowledge through embeddings, PLoT explicitly defines it using compositional, rule-based frameworks. Let's take the sentence “If it rains, then you should take an umbrella”. Instead of encoding the word separately, JEPA would embed the entire relationship between “rain” and “taking an umbrella” as a structured concept in its learned representation space. It would store “rain” and “umbrella” as related abstract states, forming a structured transition like: State 1: “Raining”, Action: “Take Umbrella”, State 2: “Stayed dry.” These representations would exist in a high-dimensional abstract space. Whereas, PLoT would explicitly encode the relationship as a conditional rule: P(Take Umbrella| Rain) ~ 95%. This means that “given it is raining, there is a 95% probability that carrying an umbrella is the correct action.” It would also break down the statement into modular components, event: “it is raining (weather = rain)”, consequence: “carrying an umbrella prevents getting wet.”, casual rule: “if rain, then carry an umbrella.”

Their approaches to prediction also diverge significantly. JEPA is deterministic, meaning it learns a prediction function that outputs a single, structured embedding for the next state. PLoT, on the other hand, is probabilistic, predicting a distribution over possible outcomes using Bayesian inference and probabilistic programming. This fundamental difference affects how they handle missing information. When encountering incomplete data, JEPA does not resort to probabilistic guessing but instead aligns learned embeddings to generate the most consistent representation based on prior knowledge. In contrast, PLoT explicitly models uncertainty, using logical rules and probability distribution to infer missing information rather than making a single deterministic prediction. Let's consider the following scenario, "yesterday, Alice left her house with an umbrella. A few hours later, she came home with wet clothes, what happened?" Since the model does not have an explicit encoding of this situation it would compare the current incomplete input to previously learned structured relationships such as, "rain leads to wet clothes unless protected by an umbrella", "an umbrella is meant to keep someone dry", and "if someone is wet despite having an umbrella, it likely means they didn't use it or rain was too strong". Based on this representation JEPA would predict that most likely, "it rained, and Alice either didn't use the umbrella or it wasn't effective enough to keep her dry." However, PLoT would generate multiple possible explanations and assign probabilities to each: P(Rained & Alice didn't use the umbrella | Wet Clothes) = 0.6 and P(Umbrella broke|Wet clothes) = 0.2. If additional information were provided like, "Alice's umbrella was inside her bag the whole time" then PLoT would update these probabilities. JEPA seems to prioritize structured, determinist inference and aims for a single internally consistent representation rather than maintaining uncertainty while PLoT explicitly represents multiple possibilities and models uncertainty using probability distributions. Additionally, JEPA accounts for unknown factors by introducing a latent variable, ensuring that missing information is structured within its learned representations rather than treated as an explicit probability. In contrast, PLoT directly represents uncertainty using probability distribution, allowing it to reason about multiple possible outcomes. Ultimately, LeCun's JEPA and Tenenbaum's PLoT represent two fundamentally different approaches to incorporating world models into AI. While JEPA seeks to develop flexible, structured embeddings for predictive learning, PLoT relies on explicit symbolic reasoning to model uncertainty and causal structures. Understanding these differences is crucial for advancing AI systems that require robust world representations, predictive reasoning, and the ability to handle incomplete or uncertain information effectively.

Conclusion

In this thesis, I examined the current limitations of large language models (LLMs), particularly their challenges in expressing uncertainty, reasoning, and planning in complex situations. To provide context for the advancements in AI, I first outlined the fundamental structure of LLMs and their limitations before introducing the concept of world models. I then explored the ongoing debate over whether world models need to be explicitly built into AI architectures or whether they can emerge naturally from learning processes. Following this, I examined how the two explicit world model architectures proposed by Yann LeCun and Joshua Tenenbaum align with two major cognitive theories—nativism and empiricism—and their implications for AI development Finally, I highlighted their key differences in approach, with specific emphasis on JEPA and PLoT.

At this stage, I remain uncertain about which approach is best suited for replicating true world models in AI systems. However, I strongly believe that a shift in architecture beyond traditional LLMs is necessary if we aim to achieve human-level intelligence. In particular, I argue that the development of explicit world model architectures is crucial due to their ability to express and manage uncertainty effectively. Both humans and AI systems will always face uncertainty, as the world itself is inherently unpredictable. However, the ability to convey and account for uncertainty transparently is essential for fostering human-AI collaboration and trust.

Thus, moving toward world models in any capacity is a critical step—not only for addressing the shortcomings of current LLMs but also for developing AI systems that can genuinely assist humans in better decision-making. By integrating structured world models, AI can move beyond mere statistical pattern matching to become more adaptive, interpretable, and aligned with real-world reasoning and planning.

Works Cited

Abdelnabi, Ahmad A Bany, et al. “Usefulness of Large Language Models (Llms) for Student Feedback on H&P during Clerkship: Artificial Intelligence for Personalized Learning.” *ACM Transactions on Computing for Healthcare*, 23 Jan. 2025, dl.acm.org/doi/abs/10.1145/3712298. Accessed 04 Feb. 2025.

Adejumo, Elijah Kayode, et al. “Towards Leveraging Llms for Reducing Open Source Onboarding Information Overload: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering.” *ACM Conferences*, 27 Oct. 2024, dl.acm.org/doi/abs/10.1145/3691620.3695286. Accessed 04 Feb. 2025.

Ahn, Janice, et al. “Large Language Models for Mathematical Reasoning: Progresses and Challenges.” *arXiv.Org*, 16 Sept. 2024, arxiv.org/abs/2402.00157. Accessed 04 Feb. 2025.

Author links open overlay panelAndrás Prékopa, et al. “Probabilistic Programming.” *Handbooks in Operations Research and Management Science*, Elsevier, 6 Apr. 2005, www.sciencedirect.com/science/article/abs/pii/S0927050703100059. Accessed 04 Feb. 2025.

Author links open overlay panelL.O. Harvey Jr., and At a minimum. “Vision, Psychology Of.” *International Encyclopedia of the Social & Behavioral Sciences*, Pergamon, 2 Nov. 2002, www.sciencedirect.com/science/article/abs/pii/B0080430767014686. Accessed 04 Feb. 2025.

Backlinko. “CHATGPT Statistics 2025: How Many People Use Chatgpt?” *Backlinko*, 31 Jan. 2025, backlinko.com/chatgpt-stats. Accessed 04 Feb. 2025.

Benary , Manuela, et al. "Leveraging Large Language Models for Decision Support in Personalized Oncology | Oncology | Jama Network Open | Jama Network." *Jama Network Open*, Jama Newtork Open , 2023, jamanetwork.com/journals/jamanetworkopen/fullarticle/2812097. Accessed 05 Feb. 2025.

GeeksforGeeks. "NLP: How Tokenizing Text, Sentence, Words Works." *GeeksforGeeks*, 31 Jan. 2024, www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/. Accessed 04 Feb. 2025.

Goodman, N. D. "Concepts in a Probabilistic Language of Thought." *Causality in Cognition Lab*, 1 Jan. 2015, cicl.stanford.edu/publication/goodman2015concepts/#:~:text=We%20may%20think%20of%20this,probability%20distribution%20on%20world%20states. Accessed 04 Feb. 2025.

Grow, Jacob. "The Strengths and Limitations of Large Language Models in Reasoning, Planning, and Code Integration." *Medium*, 23 Dec. 2024, medium.com/@Gbgrow/the-strengths-and-limitations-of-large-language-models-in-reasoning-planning-and-code-41b7a190240c. Accessed 04 Feb. 2025.

Ha, David. "World Models." *World Models*, arXiv, 2018, arxiv.org/pdf/1803.10122. Accessed 05 Feb. 2025.

Harnad, Stevan. "LLM Understanding: 15 J. TENENBAUM 'Word Models to World Models: Probabilistic Language of Thought.'" *YouTube*, 8 June 2024, www.youtube.com/watch?v=mvDxzmMpvl8&t=4527s. Accessed 04 Feb. 2025.

Harvard CMSA. “Yann Lecun | Objective-Driven AI: Towards AI Systems That Can Learn, Remember, Reason, and Plan.” *YouTube*, 28 Mar. 2024, www.youtube.com/watch?v=MiqLoAZFRSE. Accessed 04 Feb. 2025.

Harvey , L.O. “Internal Representation.” *Internal Representation - an Overview | ScienceDirect Topics*, Science Direct, 2001, www.sciencedirect.com/topics/computer-science/internal-representation. Accessed 04 Feb. 2025.

Huynh, Duy. “State of LLM in 2023: A Quick Recap on Latest Advancements.” *Medium*, 27 Jan. 2024, medium.com/@vndee.huynh/state-of-llm-in-2023-a-quick-recap-on-latest-advancements-46a55dfe1fe5. Accessed 04 Feb. 2025.

Izzidien, Ahmed, et al. “LLM Vs. Lawyers: Identifying a Subset of Summary Judgments in a Large UK Case Law Dataset.” *arXiv.Org*, 4 Mar. 2024, arxiv.org/abs/2403.04791. Accessed 04 Feb. 2025.

Kasundra, Jaykumar, et al. “Adapting Open-Source Llms for Contract Drafting and Analyzing Multi-Role vs. Single-Role Behavior of CHATGPT for Synthetic Data Generation: Proceedings of the Third International Conference on AI-ML Systems.” *ACM Other Conferences*, 17 May 2024, dl.acm.org/doi/abs/10.1145/3639856.3639888?casa_token=8Fo1-AutuIUAAAAA%3AAGQp1xWmvFZm2hHeV_7aStJZXM4rgP5BF1kT8R4MwyxUGYJNuVcmW21p2mUfD3lgpTnIlxobPHlAnQ. Accessed 04 Feb. 2025.

Kavas, Hamit, et al. “Using Large Language Models and Recruiter Expertise For ...”

International Joint Conferences on Artificial Intelligence Organization, 2024, www.ijcai.org/proceedings/2024/1011.pdf. Accessed 05 Feb. 2025.

Kirkovska, Anita. “LLM Benchmarks: Overview, Limits and Model Comparison.” *LLM Benchmarks in 2024: Overview, Limits and Model Comparison*, Vellum, 11 Sept. 2024, www.vellum.ai/blog/llm-benchmarks-overview-limits-and-model-comparison. Accessed 04 Feb. 2025.

Korkanti, Sukanth. “Enhancing Financial Fraud Detection Using LLMs and Advanced Data Analytics.” *IEEE Xplore*, IEEE, 2024, ieeexplore.ieee.org/abstract/document/10760895?casa_token=CueGsX5fKJIAAAA A:bZ6oXLbf5rjTjD-NMwiAmZqq1uLnbfFIxctL1dyKjH1WGUMnL1RtvKIpdNR 3FIZIM_Fxm3p8BHY. Accessed 2025.

Lakkaraju, Kausik, et al. “LLMS for Financial Advisement: A Fairness and Efficacy Study in Personal Decision Making: Proceedings of the Fourth ACM International Conference on AI in Finance.” *ACM Other Conferences*, 25 Nov. 2023, dl.acm.org/doi/abs/10.1145/3604237.3626867. Accessed 04 Feb. 2025.

LeCun, Yann. “A Path towards Autonomous Machine Intelligence Version ...” *OpenReview.Net*, OpenReview, 2022, openreview.net/pdf?id=BZ5a1r-kVsf. Accessed 05 Feb. 2025.

LeCun, Yann. “What Is a World Model?: Yann LeCun Posted on the Topic.” *LinkedIn*, 20 Feb. 2024, www.linkedin.com/posts/yann-lecun_lots-of-confusion-about-what-a-world-model-activity-7165738293223931904-vdgR/. Accessed 04 Feb. 2025.

"LLM Leaderboard 2025." *2025*, Vellum, 4 Feb. 2025, www.vellum.ai/llm-leaderboard. Accessed 04 Feb. 2025.

Madzime, Ruvarashe, and Clement Nyirenda. "Enhanced Electronic Health Records Text Summarization Using Large Language Models." *arXiv.Org*, 12 Oct. 2024, arxiv.org/abs/2410.09628. Accessed 04 Feb. 2025.

Markie, Peter, and M. Folescu. "Rationalism vs. Empiricism." *Stanford Encyclopedia of Philosophy*, Stanford University, 2 Sept. 2021, plato.stanford.edu/entries/rationalism-empiricism/. Accessed 04 Feb. 2025.

META. "Yann LeCun on a Vision to Make AI Systems Learn and Reason like Animals and Humans." *AI at Meta*, 23 Feb. 2022, ai.meta.com/blog/yann-lecun-advances-in-ai-research/. Accessed 04 Feb. 2025.

Millière, Raphaël, and Cameron Buckner. "A Philosophical Introduction to Language Models - Part II: The Way Forward." *arXiv.Org*, 6 May 2024, arxiv.org/abs/2405.03207. Accessed 04 Feb. 2025.

Millière, Raphaël, and Cameron Buckner. "A Philosophical Introduction to Language Models -- Part I: Continuity with Classic Debates." *arXiv.Org*, 8 Jan. 2024, arxiv.org/abs/2401.03910. Accessed 04 Feb. 2025..

Mu, Norman, et al. "Can Llms Follow Simple Rules?" *arXiv.Org*, 8 Mar. 2024, arxiv.org/abs/2311.04235. Accessed 04 Feb. 2025.

Oliviarandall. "Community Perspective - Josh Tenenbaum." *AI2050*, 21 Jan. 2025, ai2050.schmidtsciences.org/community-perspective-josh-tenenbaum/. Accessed 04 Feb. 2025.

OpenAI, et al. "GPT-4 Technical Report." *arXiv.Org*, 4 Mar. 2024, arxiv.org/abs/2303.08774. Accessed 04 Feb. 2025.

Orgad, Hadas, et al. "Llms Know More than They Show: On the Intrinsic Representation of LLM Hallucinations." *arXiv.Org*, 28 Oct. 2024, arxiv.org/abs/2410.02707. Accessed 04 Feb. 2025.

Rein, David, et al. "GPQA: A Graduate-Level Google-Proof Q&A Benchmark." *arXiv.Org*, 20 Nov. 2023, arxiv.org/abs/2311.12022. Accessed 04 Feb. 2025.

Samet, Jerry, and Deborah Zaitchik. "Innateness and Contemporary Theories of Cognition." *Stanford Encyclopedia of Philosophy*, Stanford University, 13 Sept. 2017, plato.stanford.edu/entries/innateness-cognition/#EmpFinThe. Accessed 04 Feb. 2025.

St&ouml;ffelbauer, Andreas. "How Large Language Models Work." *Medium*, Data Science at Microsoft, 24 Oct. 2023, medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f. Accessed 04 Feb. 2025.

Tanneru, Sree Harsha, et al. "Quantifying Uncertainty in Natural Language Explanations of Large Language Models." *arXiv.Org*, 6 Nov. 2023, arxiv.org/abs/2311.03533. Accessed 04 Feb. 2025.

Turpin, Miles, et al. "Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting." *arXiv.Org*, 9 Dec. 2023, arxiv.org/abs/2305.04388. Accessed 04 Feb. 2025.

Wan, Yuxuan, et al. "Logicasker: Evaluating and Improving the Logical Reasoning Ability of Large Language Models." *arXiv.Org*, 8 Oct. 2024, arxiv.org/abs/2401.00757. Accessed 04 Feb. 2025.

Wong, Lionel, et al. “From Word Models to World Models: Translating from Natural Language to the Probabilistic Language of Thought.” *arXiv.Org*, 23 June 2023, arxiv.org/abs/2306.12672. Accessed 04 Feb. 2025.

Xiong, Miao, et al. “Can LLMS Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in Llms.” *arXiv.Org*, 17 Mar. 2024, arxiv.org/abs/2306.13063. Accessed 04 Feb. 2025.

Yang, Xintian, et al. “Application of Large Language Models in Disease Diagnosis and Treatment.” *Chinese Medical Journal*, U.S. National Library of Medicine, 20 Jan. 2025, pmc.ncbi.nlm.nih.gov/articles/PMC11745858/. Accessed 04 Feb. 2025.

Yao, Jia-Yu, et al. “LLM Lies: Hallucinations Are Not Bugs, but Features as Adversarial Examples.” *arXiv.Org*, 4 Aug. 2024, arxiv.org/abs/2310.01469. Accessed 04 Feb. 2025.

Zhao, Huaqin, et al. “Revolutionizing Finance with LLMS: An Overview of Applications and Insights.” *arXiv.Org*, 12 Dec. 2024, arxiv.org/abs/2401.11641. Accessed 04 Feb. 2025.

Zhu, Zifeng, et al. “Multichartqa: Benchmarking Vision-Language Models on Multi-Chart Problems.” *arXiv.Org*, 18 Oct. 2024, arxiv.org/abs/2410.14179. Accessed 04 Feb. 2025.