LLM Problem Analysis and DSM Deep Semantic Model

Authors: Feng Chen, Chen Nanxi, Chen Nanxi

Date: 2024-02-20T19:15:38+00:00

Abstract

This paper analyzes the major challenges currently facing Large Language Models (LLMs) and proposes concrete solutions. It identifies that the representation and computation of probabilistic conceptual structure models is crucial, provides a brief overview of the related technology—Deep Semantic Model (DSM), and finally outlines key directions for future work.

Full Text

LLM Problem Analysis and the DSM Deep Semantic Model

CHEN Feng and **Nancy Chen**

Abstract

This paper analyzes the fundamental problems of current Large Language Models (LLMs) and proposes specific solutions, identifying that the expression and computation of conceptualized structural models combined with probability is the key. It provides a brief explanation of the related technology—the Deep Semantic Model (DSM)—and finally outlines directions for subsequent key work.

Keywords: LLM; GPT; Hallucination; DCN; Dynamic Cognitive Network; DSM; Deep Semantic Model; Interpretability; Conceptualization; Structurization; AGI

1. Introduction

The development of Large Language Models has dramatically advanced the technical capabilities of natural language processing, revealing AI's immense

potential and promising applications that will bring numerous positive impacts to human society—a consensus that has already formed in the industry.

However, deepening research and applications have also exposed many critical problems with LLM technology, creating obvious obstacles to further capability improvements and the full realization of application value. Many in the industry also believe that current LLMs do not represent the ultimate solution for achieving Artificial General Intelligence (AGI). Taking ChatGPT as a typical representative product, this paper provides an in-depth analysis of the main problems affecting similar LLMs and proposes fundamental solutions and directions.

This paper also briefly introduces DSM deep semantic technology, elaborating on key aspects of its basic theory, model architecture, implementation methods, and current achievements. It analyzes how this technology addresses the aforementioned problems and how it can collaborate with LLMs to achieve better technical solutions and products, while pointing out directions for future key work.

It should be noted that the connotations of conceptual terms are difficult to define precisely and constantly evolve. In this paper, LLM refers to the commonly accepted definition in the current industry: a model employing deep neural network architecture, trained through automated machine learning on large corpora, forming a black-box structure containing numerous non-conceptualized connections and parameters, and computing on natural language in an end-to-end manner. While GPT is analyzed as an example, most issues identified apply to other current LLMs, though a few may not be applicable to some other models—this does not affect the overall conclusions.

2.1 Interpretability Issues: Representation and Computation of Conceptualized Structures

Interpretability can be defined as the ability to explain or present model behavior in human-understandable terms. Interpretability should not be merely a metric for measuring a system but a fundamental goal of system implementation, equally important as functional effectiveness. Just as people study various sciences to construct interpretable systems, AGI aims to replicate and enhance human thinking capabilities, making interpretability a core objective of AGI. Even from a results-only perspective, the ability to interpret a system determines one's ability to decompose, adjust, and control it, thereby defining the upper limit of the system's ultimate functional effectiveness.

Current LLMs suffer from poor interpretability, which bottlenecks further capability improvements. Moreover, problems such as reliance on massive data, extensive repetitive training, and catastrophic forgetting are essentially manifestations of this root problem.

The most effective solution to interpretability problems is conceptualization and

structurization. Conceptualization involves defining human-understandable concepts as basic elements that constitute the system. Taking GPT-3 as an example, it uses 12,288-dimensional vectors to express basic information, which are primarily learned automatically by machines and are not aligned with human-understandable concepts. If these 12,288 vector dimensions could be equivalently converted to another 12,288 human-understandable concepts, the goal of conceptualization would be achieved to some extent. While achieving this through bottom-up automated machine learning alone would be ideal, if this proves impossible, combining it with top-down human design becomes necessary.

Conceptualization is accompanied by structurization. In GPT, the connections that compute vectors are also non-conceptualized, possessing only probabilistic computational parameters without semantic information. Conceptualizing and semantifying these connections is also crucial for forming a conceptual structure that integrates description and computation.

Meanwhile, Transformer is fully connected, which is well-suited for initially discovering all possible knowledge exhaustively. However, this fixed structure also means that after effective knowledge is learned, a large amount of invalid knowledge with probability parameters close to zero continues to occupy space and computational resources. Conceptualization and structurization also involve pruning, merging, and optimizing concepts and structures.

The statement "intelligence is compression" is correct. The essence of human thinking is processing nearly infinite information with a finite brain capacity. The key lies in specific methods, and combination and generalization are efficient methods for compressing information. More thorough conceptualization and structurization will increase the compression rate of effective knowledge to a higher level, making interpretability and its related problems no longer an issue.

At this point, constructing a system of wide-area base concepts (including conceptualized vectors) and structural systems is crucial. This can be used to interpret LLMs and compensate for their lack of semantic structure, serving as a foundation for developing more powerful AI systems.

2.2 Incomplete Algorithmic System: Designing Complete Algorithmic Systems

An incomplete algorithmic system represents another prominent fundamental theoretical flaw of GPT, to which many difficult problems and results can be attributed. From the perspective of Dynamic Cognitive Networks (DCN) [], the algorithms of understanding, querying, reasoning, generation, and learning (Fig. 1) are important basic algorithms for achieving AGI. These algorithms are not interchangeable with each other and must be combined to solve various computational tasks completely.

GPT primarily employs the generalized generation algorithm, which calculates

the next character (strictly speaking, Token) from a known sequence of characters. The basic principle can be summarized as: (1) The vector of the target character is computed separately from the vector of each character in the known sequence; (2) Each known character is superimposed on the target vector calculated for the target character; and (3) The superimposed vectors of the target character are compared with a character dictionary, and the best match is taken as the result.

This algorithm works well for generative tasks but lacks key capabilities such as complete structured hierarchical computation and probabilistic collapse computation. It cannot equivalently replace other algorithms, and forced substitution prevents the effects of various tasks from converging to an ideal state. The fundamental solution is to design a more complete algorithmic system that should be built around the aforementioned conceptualized structure.

2.3 Hallucination Issues: Choosing the Right Algorithm

Generally speaking, LLM computations that produce results not meeting people's expectations and standards are collectively referred to as hallucinations, but different hallucinations have different causes. Here we mainly discuss the most essential type: generative hallucination, which can be summarized by the technical principle of "probability loss caused by generation from the base network to the derived network" and the problem of "treating probability as necessity."

First, the rules for comprehension, generation, and equivalence calculations are different. In the DCN system, these algorithms rely on [belong to relation] and [equivalence relation] for implementation. Assuming the system already contains knowledge such as [Lin Daiyu belongs to person], [Jia Zheng belongs to person], [Granny Liu belongs to person], and [people insulting people] (note: the probability of [people insulting people] being factual is < 1, because while an insulting event between any two people is possible, it is not factual), the algorithms work as follows:

Comprehension: Understanding [Lin Daiyu insulting Jia Zheng] as [people insulting people] involves bottom-up calculation without probability loss.

Equivalence: Copying equivalently from [Lin Daiyu insulting Jia Zheng] to [Lin Daiyu insulting Jia Zheng] is completely equivalent.

Generation: Deriving [Lin Daiyu insulting Jia Zheng] from [people insulting people] involves the base pattern and derived pattern, where corresponding concepts have a [belong to] relation. The derived pattern first directly inherits parameters including probabilities from the base pattern, meaning the factual probability is < 1. Narrow generation is top-down computation on the set dimension, and direct generation out of nothing also causes generative probability loss unless more information is incorporated to influence and adjust the new probabilities.

Learning: When acquiring factual knowledge such as "Lin Daiyu insulting Granny Liu," the first step is comprehension, constructing the new knowledge [Lin Daiyu insulting Granny Liu] and establishing a derivative relationship with [people insulting people], then setting the factual probability reloading = 1 so the new knowledge is deposited into the system. Obviously, knowledge learned from reliable sources differs from information generated by the system itself, particularly in terms of factual probability.

Overall, relations of belonging, derivation, and equivalence have distinctly different semantics. If expressed in terms of probability, it is also necessary to distinguish the essential difference between projected probabilities that are $<$ 1 versus = 1. GPT has only one-way probability calculation, and results are determined solely by relative probability size, making it fundamentally difficult for the underlying algorithm to achieve tasks requiring probability = 1.

Humans propose tasks for different purposes. Understanding, querying, and generation are completely different task types with different criteria for determining result correctness, and these criteria are themselves completely clear. The [generation] task follows the criterion of probabilistic possibility and does not require probability = 1 factuality. It should use the [generation] algorithm. For example, when asked to [fabricate a story about "insulting"], knowledge of [people insulting people] can derive infinite outcomes such as [Zhang San insulting Li Si], [Lin Daiyu insulting Zhang San], [Lin Daiyu insulting Jia Zheng], all of which satisfy the task requirements.

GPT's generalized generation algorithm is well-suited for this narrow generation task. Since generalized knowledge like [people insulting people] is obtained through training, using vector computation to generate derivatives of this knowledge can produce various results to satisfy the task.

The [query] task, however, follows probabilistic determinism, i.e., probability = 1 factuality. Correct processing requires the [query] algorithm, which is the pattern matching algorithm. For example, for the task of [querying a story about "insulting"], the correct approach should use the question as a template for pattern matching on factual knowledge. If successful, the factual knowledge obtained (e.g., [Lin Daiyu insulting Granny Liu]) is copied and output equivalently. If no match is found, the answer "no matching result found" should be given. Results calculated this way are theoretically completely stable and reliable, and problems can be accurately traced and corrected.

For GPT, the same generation algorithm is still used. If the sentence [Lin Daiyu insulting Jia Zheng] is generated based on [people insulting people], the result obtained through the generation task does not meet the query task requirements. This is the root cause of this typical hallucination.

Through training, GPT adjusts the probability of selecting different knowledge as the [base pattern] for generation based on task vocabulary such as [query] or [fabricate], making knowledge like [Lin Daiyu insulting Granny Liu] more likely to be selected than [people insulting people]. While this differentiates tasks to

some extent, this probability adjustment lacks support from rigorous computational rules, and probability calculations and choices for different knowledge lack reliability. This remains a source of various hallucination errors.

Moreover, query tasks require that every parameter in the multi-parameter question pattern be matched by the knowledge pattern, involving [And] computations to ensure pattern integrity—something GPT's sole generation algorithm cannot satisfy. In real-world scenarios, a task contains multiple nested subtasks of different types, not all with explicit task indicator keywords. For example, a generation task may nest multiple parameters requiring factual queries. Without deep semantic analysis and separate processing with the correct algorithm, hallucinations are difficult to avoid. Moreover, hallucination information may be so hidden that users cannot detect it, potentially leading to more serious consequences.

Therefore, the solution requires two approaches: (1) accurately differentiate task and subtask types through deep semantic analysis, and (2) select the correct basic algorithm for different task types.

2.4 Expression Computation Issues: Structured Hierarchical Expressions and Computations

Hierarchical representation and computation of knowledge is crucial. The hierarchical structure of images is remarkable, and language semantics are entirely hierarchical—though such hierarchies do not always correspond exactly to language organization. ChatGPT lacks the ability to represent and compute rigorously structured hierarchies.

Currently, ChatGPT's algorithm cannot solve simple expression computations. It may correctly answer "35+62=97," but this is essentially complete matching of the sentence "35+62=97" in a huge corpus, not real mathematical calculation. Therefore, asking "23456789+9876543=?" will likely not yield the correct result.

The fundamental problem is that GPT lacks true hierarchical information representation. "23456789+9876543" should obviously be viewed as two levels: two numbers as a whole at the lower level participating in addition at the higher level. Instead, GPT always disassembles these characters into a flat one-dimensional sequence where each character participates in probability calculation for the next character, making correct results impossible.

This example shows GPT is also unreasonable for regular natural language computation models. First, it wastes computational resources, consuming energy and facing context length limitations. Under a hierarchical computation model, each character in a 100,000-word article should not be computed against all other characters but limited to a local range. Computational growth with context should be nearly linear, not geometric, like the human brain.

Therefore, achieving true hierarchical expression and computation of knowledge is significant—not only for solving local problems like expression computation

or optimizing computational performance, but absolutely one of the important basic indicators for realizing AGI.

This example also illustrates another problem: vectors are not suitable for exact expression of numbers or deterministic concepts. From a hierarchical perspective, the root of a hierarchical structure can be represented by an identified concept to precisely represent the entire structure—a capability vectors lack. Therefore, everything cannot be represented with vectors alone; identified concept representation is also necessary, and integrating the two is a valuable topic.

2.5 Knowledge Inconsistency Issues: Bidirectional Relationship Representation and Computation

GPT suffers from serious knowledge inconsistency problems. For example, research has found a "reversal curse" in large models that prevents them from reasoning that [B is A] even after learning that [A is B]. When taught that [Washington was the first president of the United States], the model does not automatically answer [Who was the first president of the United States?] unless additionally taught that [The first president of the United States was Washington].

This problem arises simply because GPT is a unidirectional probabilistic expression and computation neural network that employs forward character projection. [Washington was the first president of the United States] and [The first president of the United States was Washington] are treated as two completely different connections and computations rather than one holistic piece of knowledge.

This problem exposes the essential fact that no real structured knowledge is learned this way in GPT. Adding more morphologically different but essentially redundant information can mitigate the effect but does not improve overall intelligence.

Obviously, humans have no problem with knowledge consistency. Knowledge of a whole should be expressed and memorized as a whole and applied flexibly in different forms.

Therefore, the correct solution is to model the human approach. Specifically, replace unidirectional function computation with semantic relations having bidirectional probabilities. Such relations can express complete semantic knowledge and perform probabilistic computation, representing a more reasonable, effective, and complete knowledge expression. Furthermore, the semantic structure of such relations can realize the "scene-fitting" expression and computation mode expressed by DCN theory. This model may well be the key to solving AGI.

2.6 Weaknesses in Reasoning: Structured Reasoning Expression and Computation

According to DCN, roughly speaking: comprehension and generation mainly refer to vertical computation of a tree network from bottom to top and top to bottom, with relatively shallow and fixed computational depth, forming the foundation of intelligence. Reasoning, however, mainly refers to horizontal transformation between multiple tree networks, where computational depth can be very deep and range expansive, determining the upper limit of intelligence.

Compared to the past situation of completely lacking reasoning ability in natural language form, GPT has achieved considerable reasoning ability—a great progress. However, current reasoning levels remain relatively weak by higher intelligence standards. Continuously improving reasoning ability and intelligence levels requires not just increasing corpora and training workload but solving fundamental problems of technical principles and architecture.

GPT's representation of reasoning knowledge, like other knowledge, is a black-box structure lacking stability and reliability and is difficult to adjust precisely. This problem becomes more pronounced as more reasoning knowledge is added. Consistent with previous analysis, a more conceptualized and structured way to express reasoning knowledge and perform reasoning computation is the truly effective solution. Assuming that learning training in the form of chains of thought has allowed the system to learn reasoning knowledge to a certain degree of effectiveness, optimizing this knowledge into more essentially structured representations and expanding important parameters (probabilities, etc.) will inevitably lead to even better results. Of course, reasoning structures are more complex than simple structures and represent a difficult problem that techniques such as traditional knowledge graphs have not effectively solved, thus requiring better structural design theory.

Non-hierarchical and holistic issues: As previously discussed, reasoning computations require stricter hierarchical and holistic requirements. A single atomic reasoning computation should be a complete conversion from one pattern to another, not reasoning character by character. The latter is not only computationally intensive but also prone to information inconsistency problems.

Bidirectional reasoning flaw: As previously discussed, GPT's reasoning is unidirectional and must be elevated to an overall structure of bidirectional reasoning for expression and computation.

Multi-branch, multi-level complex reasoning: Real-world reasoning involves multi-branch, multi-level reasoning across a wide domain of information. The maximum probability branch selected at the first level may not produce optimal results after multiple levels of reasoning. Without formalized expressions, decomposable combinations, dynamic parametric reasoning structures, and more flexible multi-way reasoning and retrospectively adjustable models, obtaining desired results is difficult.

2.7 Knowledge and Data Mixing Issues: Hierarchical Representation of Knowledge and Data

GPT lacks hierarchical definition of knowledge and cannot effectively distinguish between knowledge and data, managing them mixed together. Currently, when using models like PROMPT combined with external information, both internal and external information actually mix knowledge and data respectively, not truly separating them but rather complicating the problem.

Due to difficulty isolating correct knowledge for effective sharing, the industry performs different forms of iterative model training, generating many black-box knowledge bases with large amounts of redundancy but no uniformity, continuously wasting duplicated resources.

The solution is to express and manage knowledge and data hierarchically. Computation can be seamlessly integrated, but management structures can be flexibly decomposed. The importance and authority of each piece of knowledge can also be precisely defined and managed. Higher-level deterministic knowledge can be fully shared, while lower-level data can be stored flexibly, and different versions of knowledge and data that cannot reach consensus can also be maintained. This constructs a more reasonable knowledge and data maintenance system.

Hierarchical management of knowledge and data also facilitates assistance in resolving data copyright issues at the technical level. True higher-level knowledge maintains a small scale, and this human consensus knowledge has no copyright issues. Large amounts of lower-level knowledge (e.g., various news) should be embodied as independently stored data, which can be refined to identify copyright for each piece of knowledge and even generate new business models adapted to the age of intelligence.

2.8 Knowledge Learning Issues: Multi-Level Incremental Learning Models

LLM is an overall black-box structure that cannot be broken down into individual pieces of knowledge for adjustment, nor can it effectively distinguish between correct and incorrect knowledge, leading to problems such as catastrophic forgetting when learning new knowledge that may undermine existing knowledge.

The solution is to realize real-time and incremental learning based on knowledge hierarchy. Since each piece of knowledge can be split, it can form an efficient pattern similar to human knowledge learning: first fix already explicit knowledge, then incrementally learn only new knowledge, continuously accumulating knowledge more stably and reliably.

Learning truly hierarchical knowledge will be more efficient. There is more knowledge at shallow levels, and learning actions are more frequent, but only the more peripheral knowledge base needs modification. If higher-level knowledge

needs adjustment, it is passed on to modify the higher-level base knowledge base. Typically, higher levels have less knowledge and lower-frequency learning adjustments.

2.9 Alignment of Uncertain Information Issues: Conceptual Representation of Certainty

Natural language has complex ambiguity, where the same word often corresponds to multiple different semantics—a core problem in natural language processing. The key to solving ambiguity is fully utilizing contextual information, specifically projecting each word onto others, which is essentially what GPT does: GPT internally disassembles vocabulary into Tokens, converts them into vectors, lets vectors extrapolate each other to eliminate ambiguity, and finally converts result vectors back to Tokens and vocabulary, indirectly inferring from input and output representations. Natural language understanding accuracy is excellent.

However, the hidden semantic concepts formed by GPT comprehension, especially the overall structure, are difficult to express. There is not even a complete set of semantic standards to target, so GPT output remains natural language. This causes problems in aligning semantic concepts and structures, making reliable information docking and sharing among multiple systems and models difficult when relying only on natural language.

It is well known that information exchange between systems begins with ensuring information standard consistency. Therefore, a global semantic concept and structure is necessary even from an engineering perspective alone. Only with deterministic semantic expressions can information be effectively shared and transferred between systems, allowing any two systems to interface directly without requiring GPT. Problems such as task decomposition and combination, multi-technology integration, and long-term memory access can be better solved.

2.10 Bias and Jailbreaking Issues: Semantics-Based Control of Absolute Information

Information control issues such as bias, jailbreaking, and security are also challenges faced by LLM technology. Nowadays, attempts to solve these problems through constant training, fine-tuning, and prompting hardly guarantee stable and reliable results or eventual convergence. Information interacts with each other—for example, assuming a prompt works, another peripherally spiked prompt can also request the system to turn off the previous prompt's effect. A probabilistically optimal model cannot solve the problem of probabilistic absolutes.

The solution involves two aspects: (1) define semantic concepts and structures, accurately defining semantics for [bias], [sensitive information], etc., enabling

precise semantic parsing and categorization of both input and output information; and (2) based on precise semantics, insert reliable information control at any point in the process, giving absolute control over specific processing rules (probability set to = 1, no other computation allowed to override).

3. Introduction to the DSM Deep Semantic Model

DSM (Deep Semantic Model) is a specific implementation of DCN (Dynamic Cognitive Network) theory for language semantic processing. Compared with traditional knowledge graphs, DSM can realize important capabilities such as deep semantic expression, complete semantic expression, hierarchical semantic expression, algorithmic closed-loop system, probabilistic expression and computation, and docking with natural language, forming a complete linguistic semantic expression and computation system with the potential to become an independent and complete intelligent system. Traditional knowledge graphs are usually more suitable for constructing thematic databases to provide data for intelligent systems but are difficult to use as autonomous intelligent systems.

DSM has extensive content, and this paper provides only a brief introduction to its technical points.

3.1 Deep Semantic Structure

The foundation of the deep semantic model lies in its unique DSM structure definition, which adopts the two-dimensional multi-level tree network structure proposed in DCN theory and is optimized according to language characteristics. The deep semantic structure is also a structure that integrates expression and computation. The structure itself expresses conceptualized semantic knowledge and also describes basic computation rules and parameters. Various computations are embodied in various creations, combinations, and transformations of the structure relying on its own semantics and parameters.

3.2 Separation of Semantics and Language

DSM completely separates the semantic model from the language model. Semantics is independent of natural language, and the two are interchanged through comprehension and generation algorithms—"using language for external expression and semantics for internal expression and computational thinking." Semantic and linguistic transformations are described and computed mainly through two [owning] relations: (1) semantic concepts have linguistic morphology, and (2) semantic roles have linguistic roles. Language understanding and semantic generation share the same set of relational structures for reciprocal operations.

3.3 Hierarchical Semantic Knowledge and Data System

Following the basic principle that "human knowledge systems are hierarchical," DSM builds a multi-level semantic knowledge system. Higher-level knowledge

is more basic and important, serving as the basis for understanding and expressing lower-level knowledge, while the amount of higher-level knowledge is more limited. The interpretability and computability of a semantic model are mainly represented by the topmost level of knowledge, which includes [concepts], [entities], [relationships], [roles], [existence], [measures], [degrees], [sets], [intervals], [comparisons], [sequences], [space], [time], [things], [events], [event roles], [expressions], [equations], etc.

The vast amount of knowledge at lower and middle levels is more extensive and theoretically infinitely expandable, but it is all interpreted and computed using higher-level knowledge. In principle, it is no longer necessary to implement different algorithms for different knowledge. The idea of knowledge hierarchy also applies to the division of knowledge and data, with data seen as lower-level knowledge that is theoretically completely isomorphic and can be seamlessly integrated.

In terms of storage management, "separation of knowledge and data" can be realized. Since lower-level knowledge is unidirectionally dependent on higher-level knowledge, different levels of knowledge and data can be stored separately. For computational processing, higher-level knowledge must be loaded, while mid- and lower-level knowledge and data can be loaded dynamically on demand and designed in various specialized structures (e.g., relational databases) for optimal expression, including natural language, which can be viewed as a compressed form of deep semantics.

3.4 Inheritance, Overloading, and Aggregation

DSM uses the belong-to relation and inheritance mechanism to realize hierarchical knowledge representation. Lower-level knowledge first inherits the derived network of higher-level knowledge by default, thus inheriting all information of the base network. In response to changes in the derived network's information relative to the base network, an overloading definition is performed to modify the changed information (including probability distribution parameters, etc.). DSM uses the [belong to] relation as the basis for variable binding, pattern matching, and other computations, unifying the two relations [derivation] and [instantiation] from traditional object-oriented methods and unifying processing such as [variable allocation binding] and [problem solving] in this way. The core of DSM is the fusion of theories and methods such as [set], [probability], and [object-oriented].

DSM adopts a system of multiple base classes, where multiple base classes can be combined through multiple inheritance and aggregation. Combined with mechanisms such as probability and overloading, it solves drawbacks existing in many ontological approaches that try to build conceptual systems based on single inheritance and absolutes. There is a very strong connection between multiple base classes and vectors. Base classes are multilayered and more expressive, while vectors can be a multibase class with a flat hierarchy. Base

classes can replace vectors, but not vice versa. Elementary perceptual intelligence works very well with vectors, while advanced cognitive intelligence must utilize a multi-hierarchical base class structure to achieve higher compression rates. A base class can be equated to a set of more basic base classes and vectors, eliminating the need to replicate tens of thousands of vectors for large amounts of lower-level knowledge and data.

Therefore, the single-level multidimensional vector representation of LLM and the multilevel multibase class derived representation of DSM both have their own advantages, and effectively integrating the two is a topic of great significance.

3.5 Bidirectional Relation and Tree Network Structure

Like other concepts, relations in DSM are derived from one level to the next, with [belong to], [aggregate], [own], [reason], and [hierarchy] being the most basic relations at the top level. The belong-to relation is a relation on the set dimension, also called a derived relation, expressed as [A belongs to B] or [B derives from A], where A is the derived concept and B is the base concept. The equivalence relation is a particular case of belong-to relations. The aggregate relation aggregates concepts from two different domains into a single overall concept (called an aggregate), which has a derived relationship to the concepts in these different domains. The owning relation is a relation on the domain dimension that gives rise to various different owning relations. The reasoning relation is a narrow reasoning relation also on the domain dimension, representing transformation between two patterns. The root relation is an implicit relation expressing direct affiliation of individual concepts to the root concept in a tree network structure.

These relationships can be combined to form a tree network structure on the set and domain dimensions. The tree network has a root to which all elements (including concepts, relations, and additional relations) of the following multilevel hierarchy belong, as an inseparable part of the whole pattern. The root of a tree network represents the entire tree network. The root has a projective relationship to individual elements, and individual elements also have a projective relationship to the root, just with different individual projective probabilities.

Since both concepts and relations can be derived, the entire tree network consisting of concepts and relations can also be derived. Each node of the derived network and the corresponding node of the base network has a derivation relation. All relationships have bidirectional semantic and probabilistic expressions. The reason many traditional rule-based systems cannot solve practical problems well is that, on one hand, there is a lack of hierarchical relationships between knowledge and rules, and on the other hand, knowledge and rule definitions tend to be binary logic, lacking the ability to express and compute ubiquitous uncertain information in practical scenarios. Therefore, it is significant for DSM to implant expression systems such as affiliation functions and probability in the

basic structure.

Neither simple tree structures nor ordinary network structures can effectively express cognitive information. Ordinary network structures lack hierarchical information and make problem decomposition difficult, while simple tree structures lack the ability to completely express complex structures in real scenarios. Tree network structure combines the hierarchical structure and problem decomposition ability of trees with the comprehensive information expression ability of networks, breaking down complex cognitive expressions into relatively simple local problems to be solved independently, which is of great significance to AI development. We believe that the human brain also heavily employs a logical structure similar to the tree network.

3.6 Algorithm System

In DSM, several basic algorithms such as comprehension, generation, querying, reasoning, and learning are defined to form a complete algorithmic closed-loop system for language semantics. All algorithms are in fact omnidirectional network growth algorithms, considered computations that "complete" unknown parts according to different known parts around the same two-dimensional multilevel tree network structure. This is equivalent to unifying "encoder" and "decoder," as well as unifying the two computational models of "discriminative model" and "generative model"—computation of the same structure in different directions.

Compared with end-to-end black-box computing, DSM's algorithmic system is white-box, where all aspects can be seamlessly and automatically processed. It can also be completely disassembled for customized processing when necessary, reflecting full flexibility and enabling complex multi-service fusion computing and continuous computing []. The query algorithm, also known as the semantic pattern matching algorithm, is a very basic algorithm in the whole system, using the [belong to] and [aggregate] relations of multiple base classes as base rules that can be combined with probabilistic calculations while guaranteeing complete pattern matching.

3.7 Expression of Reasoning and Computation

Reasoning is one of the key algorithms of intelligent systems. In DSM, reasoning computation is embodied as transformation of one tree network pattern to another, where each atomic reasoning is expressed through a reasoning structure. The root of the reasoning structure is a [reasoning] relation that connects the two tree networks in which reasoning is performed to form a larger tree network. The most basic reasoning relation can derive many more specific reasoning relations, all sharing the same basic structure.

For example, a reasoning tree network can describe the formula [Distance] = [Speed] $\times$ [Time] for all [Movements]. When encountering application prob-

lems with [Movement] as the base class—whether [Airplane flies from Beijing to Shanghai], [Car runs from A to B], or [Xiao Ming walks from home to school], and whether solving for [Speed], [Distance], or [Time]—all will match the same pattern and activate the mathematical formula [Distance] = [Speed] × [Time], enabling reasoning and calculations that form the basis for enumerating mathematical equations with understanding.

Solving equations is carried out by reasoning about interconversions between equations. This bidirectional reasoning of equations for multiplication and division can solve all such calculations (not limited to solving the [Movement] event). The specific transformation calculation of the equation involves selecting each reasoning transformation mode, finding the transformation path to transfer the solution target variable to the right side, and finally completing the calculation.

Like other computations, specific reasoning computation is a network completion computation: pattern matching is performed first. Once a pattern successfully matches part of an inference pattern, it triggers creation of a reasoning instance derived from this reasoning pattern as a base template and completes the other part of the reasoning instance.

This semantically structured reasoning also has the following features: abstract reasoning follows network derivation principles, where reasoning knowledge should be defined based on the most abstract essential base class for most efficient generalization; hierarchical reasoning decomposes large reasoning into small multilevel reasoning that can be combined and stacked to realize complex reasoning tasks; bidirectional reasoning structures enable the same structure to realize bidirectional reasoning; branch reasoning provides a basis for calculation choice in multi-branch reasoning; logical reasoning can be realized by combining multiple atomic reasoning using [And][Or][Not]; and planning and action extend around the same reasoning structure, where planning designs plan structures using reasoning structures and action executes those plans.

3.8 Probabilistic Expression and Calculation

Two points related to probability require explanation. First, probability collapse, mentioned in DCN, is a theory and method that needs emphasis as an effective way to solve some drawbacks of traditional probability calculations. During computation, information with high probability and certainty can undergo probability collapse (set probability = 1), making that information completely explicit and changing computation goals and directions. This not only reduces ineffective computation for higher performance but more importantly eliminates errors (uncertainty transfer and computation have cumulative errors, and proper partial collapse actually facilitates eliminating such errors in intermediate processes). Moreover, probabilistic collapses can be preset as needed for more effective system control.

In some sense, one essential difference between symbolism and connectionism is reflected in probability collapse: symbols are what all computation ultimately

needs to achieve, representing a piece of definite information—a collapsed state. But in early computation stages when information is uncertain, expressing it in explicit symbols (collapsing prematurely) is inaccurate or wrong. GPT does not separate from symbols but only expresses and calculates the superposition state of multiple symbols using probability vectors first, performing probability collapse only at the end to form deterministic symbols. Thus, symbols and probability are not mutually exclusive systems but mutually transformable. The conceptualized structure combining probability can express both probabilistic superposition and collapse states, remaining interpretable in the superposition state and allowing probabilistic intervention at any link. This can compensate for shortcomings of traditional symbolic computation and black-box neural networks, demonstrating more flexible technical advantages.

Probability collapse is also a fundamental way of thinking in the human brain. When observing and interpreting the world, humans constantly encounter unknown and uncertain information and need to identify and even manipulate information that can be prioritized and made explicit. Once some information is clarified as known (necessarily requiring definable notation), it can shift attention and computational reasoning flow to recalculate other unknowns based on the known. This continuous conditioning and computation shift is necessary for effective processing of the complex world. Without clarifying needed information as soon as possible, nothing can be done in the face of increasingly uncertain "chaotic" systems.

Probability collapse theory is also important for image recognition scenarios. Images have strong local correlation. Once an object reaches probability collapse, it drives probability collapse of surrounding objects, enabling rapid convergence. Applying this method to image and video recognition would be even more effective than language processing.

Second, simplified probability computation: although DCN is designed based on set and probability theory, actual open system application scenarios cannot provide strict probability definitions and precise values. What AI needs to solve at this stage is actually the "probability of significance under open system problems," where the probability of correct results is much greater than other results. For these problems, very high computational precision is not required, and integer-type addition and subtraction operations can often solve problems effectively. For problems where ambiguity remains, increasing computational precision is not useful; what is needed is more information, such as obtaining necessary information through multi-round dialogue communication.

Tasks requiring high-precision probabilistic computation (e.g., machine Go) usually belong to "non-significant probabilistic problems in closed systems" and should be modeled and implemented independently as domain expertise, then interfaced to the system.

3.9 Technical Applications

DSM business applications can gradually expand as technology improves. At the early stage, after constructing a certain scale of DSM model and knowledge base and focusing on realizing the ability to understand natural language as DSM structure, accurate, rich, and standard structured semantic information can serve as a basis to support various business implementations in many aspects.

Moreover, DSM and LLM have their own specialized capabilities. Using DSM's unified semantic expression capability, the two can be tightly integrated to form a more complete technical solution to enhance business application effectiveness. Specifically, DSM can focus on: semantic parsing of natural language to form unambiguous semantic structures; semantic integration of multi-round conversations and history to form complete task semantic structures; precise task distribution analysis for vertical models and systems; task management based on semantic structures; semantic reasoning for various calculations and transformations; semantic generation of lower-level semantics or natural language from higher-level semantics; semantic sharing and exchange for reliable information transfer among DSM, LLM, and other systems; semantic retrieval with more accurate matching than vector matching; and execution-type tasks requiring structured expression of task information with configurable control rules.

In later stages, with model and knowledge base expansion and further algorithm system improvement, intelligent processing capabilities will be comprehensively enhanced across all task segments to realize more powerful intelligent business applications.

3.10 Prototype Systems

We have open-sourced DSM 1.0, completed in 2016 (https://github.com/chenfeng-china/DSM). This version's basic theories, models, and algorithms have taken shape, especially providing a model library containing thousands of the most critical underlying concepts and structures, and demonstrating fundamentals of deep semantic representation and computation with several examples for analysis and research by relevant parties.

4. Further Work

DSM has been continuously developed and refined, currently iterated to the third generation, with further R&D work to follow. Key objectives include:

4.1 Implementation of LLM Ability to Read and Write DSM Structures

A very valuable recent work is training LLM to directly read and write DSM structures—specifically, the ability to parse natural language into DSM structures and generate natural language from DSM structures—to facilitate more

flexible integration of various technologies and systems for intelligent business and products.

4.2 Building a Complete Deep Semantic Knowledge Base

Building a more complete DSM foundation knowledge base and domain knowledge bases is important work requiring continuous accumulation and improvement. Unlike some other knowledge base builds, DSM prioritizes "depth" over "breadth." Higher-level knowledge is more effective and important, requiring AI experts to design and accumulate it. Previous R&D has solved many key model structure problems and constructed a basically complete higher-level knowledge system, laying a good foundation for subsequent work.

After the basic knowledge system is constructed, further derived and expanded knowledge is more numerous but less difficult, allowing joint participation by experts from various industrial fields. Moreover, LLM can accelerate DSM knowledge base construction efficiency, including using LLM as an auxiliary tool for knowledge discovery and processing, and directly converting LLM's hidden knowledge to DSM structured knowledge. Lower-level knowledge and data will be automatically learned and processed in complete real time. As the entire model scale increases, system capabilities will also have an "emergent" effect.

Building this deep semantic knowledge base may have important social value. Compared to black-box holistic models, each piece of knowledge can be shared and used by industries and continuously optimized, serving as important public infrastructure for realizing more powerful AI. To this end, building an open platform to open up the knowledge model, knowledge base, and algorithmic capabilities for industry-wide participation in improving the deep semantic knowledge base should be considered.

4.3 Building a Stronger Overall Model

The longer-term goal is further deep integration of DSM and LLM to build an integrated intelligence model combining the advantages of both. Main features include: conceptualized, structured, and interpretable knowledge structures; better DSM structures and semantic vector structures; convergence of vector computation and conceptual system computation; more complete and efficient basic algorithm systems; complete real-time knowledge learning capabilities; incremental, active, and continuous learning; a unified platform for "knowledge + data" integration; stronger reasoning, planning, and execution capabilities; deeper and more comprehensive intelligent agent systems; more efficient computing and lower resource consumption.

Among these, continuous active learning is a core capability that powerful AI must have. Super AI learning will not be one-time but can continuously and actively seek information to learn knowledge, as well as introspect on the existing knowledge system for complementation and optimization. In this system, the

hierarchy of knowledge and data plays a decisive role as the basis for the system to recognize information value, set learning goals, and control adjustment and storage strategies for each learning task.

References

[] Chen Feng. AI Centered on Scene Fitting and Dynamic Cognitive Network. [2020]. http://arxiv.org/abs/2010.04551

[] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. [2018]. https://arxiv.org/pdf/1806.01261.pdf

[] Gary Marcus. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. [2020]. https://arxiv.org/abs/2002.06177

[] Judea Pearl, Dana Mackenzie. The Book of Why. Allen Lane, 2018.

[] Gary Marcus. bengio v marcus and the past present and future of neural network models of language. [2018]. https://medium.com/@GaryMarcus/bengio-v-marcus-and-the-past-present-and-future-of-neural-network-models-of-language-b4f795ff352b. https://arxiv.org/ftp/arxiv/papers/1801/1801.00631.pdf

[] Geoffrey E. Hinton, Alex Krizhevsky, Sida D. Transforming Auto-Encoders. Artificial Neural Networks and Machine Learning, ICANN 2011, 21st: 44-51.

[] Sara Sabour, Nicholas Frosst, Geoffrey E Hinton. Dynamic Routing Between Capsules. [2017]. https://arxiv.org/abs/1710.09829

[] Gary Marcus, Ernest Davis. Rebooting AI: Building Artificial Intelligence We Can Trust. Pantheon, 2019.

[] Nicola Kuczewski, Cristophe Porcher, Volkmar Lessmann, Igor Medina, Jean-Luc Gaiarsa. Back-propagating action potential. Communicative & Integrative Biology, 2008, 1:2: 153-155.

Note: The original version of this paper is written in Chinese, and there may be deviations in the translation process. Therefore, the Chinese original version is attached for reference. If there is any difference between the two versions, the Chinese original version shall prevail.

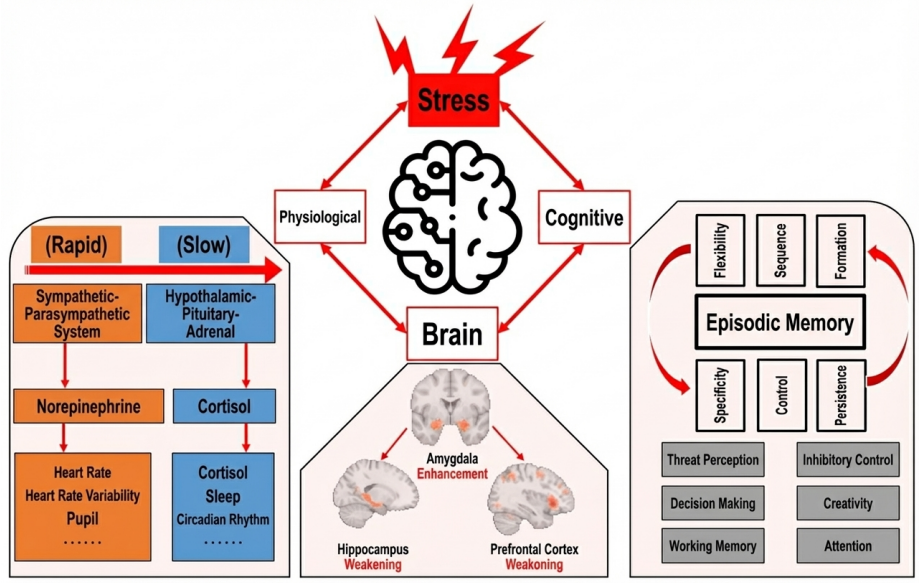

Figure 1: Figure 1

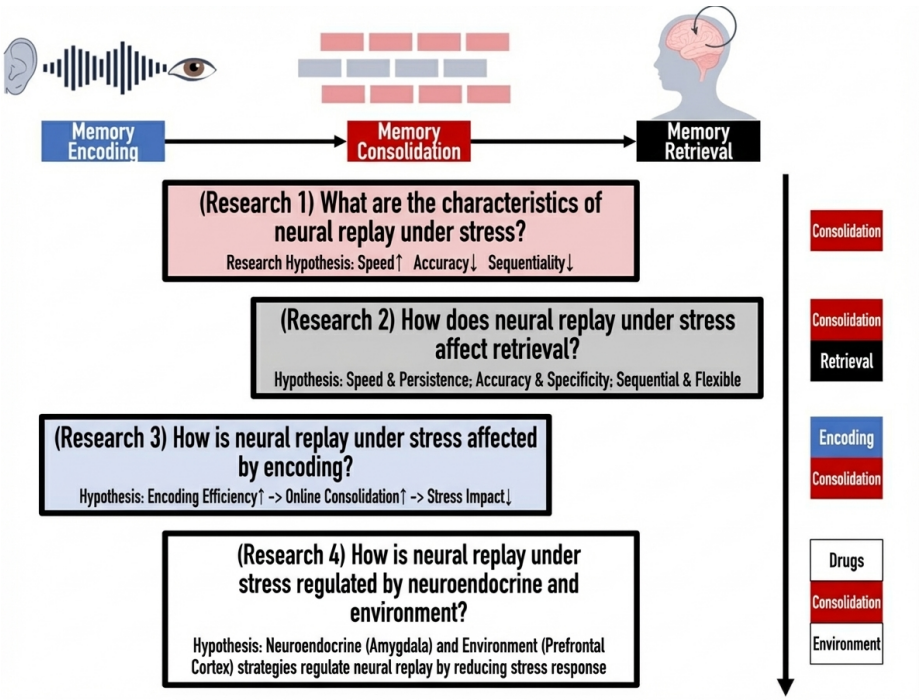

Figure 2: Figure 2

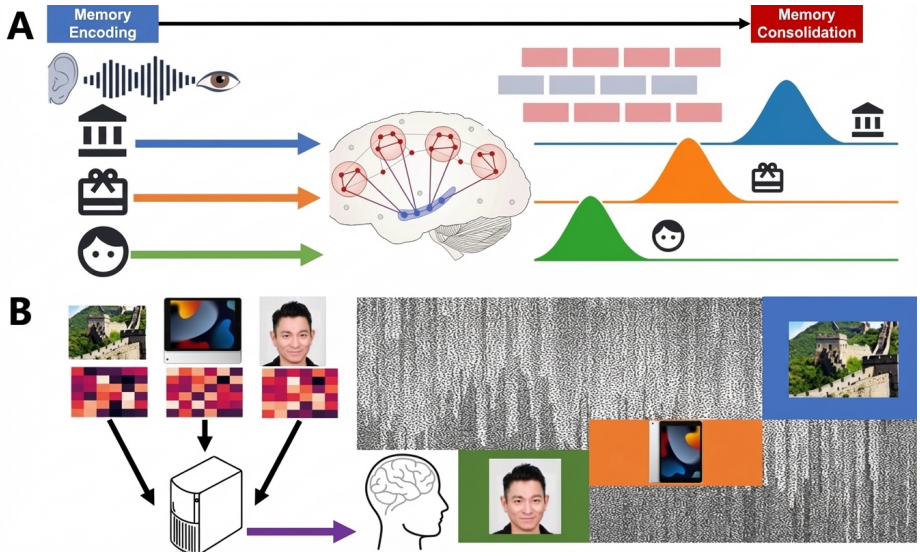

Figure 3: Figure 3

Figures

Source: ChinaXiv — Machine translation. Verify with original.