Thinking Databases

Enhancing LLMs' explainability, quality, and performance.

Author: Sînică Alboaie, PhD, Axiologic Research
Purpose: Thinking Databases Vision
Visibility: Public
Date: August 2024
Version: 0.2 (Draft & Use Cases Exploration)

Abstract

The concept of a 'Thinking Database' represents a novel technological paradigm aimed at enhancing semantic search capabilities and knowledge management through the strategic use of embeddings, text representations of knowledge, and advanced inference techniques. The proposed system utilizes both traditional inference methods and small, cost-effective 'Large Language Models' (LLMs) to accelerate and refine knowledge discovery, aiming to extract implicit knowledge from explicit data sources. Use cases could span various fields, including research, finance, healthcare, and legal analysis, where the database improves decision-making by providing contextually relevant insights. This approach should be viewed both as competitive and complementary to the concepts behind multi-agent systems, which attempt to use multiple agents with different roles and intentions to perform inferences based on LLMs and, through collaboration, achieve better results than one-shot queries. The fundamental intuition is that if the same LLM is used and if we desire the knowledge base for inferences to be more precise and controlled rather than vaguely extracted from the model, there is no need for a multi-agent system. Instead, we can explicitly configure databases and rules in a system that resembles a classic database but which, in the background, processes incoming data and deduces new insights based on programmed prompts and heuristics. This database can be queried in RAG approaches but can also trigger actions or generate periodic reports with the latest insights obtained.

The innovation proposed by this approach lies in the hypothesis that large LLMs will encounter an insurmountable barrier due to the well-known problem called combinatorial explosion. This complexity, algorithmically expressed through the need to run complex search algorithms in solution spaces with techniques that may involve backtracking, leads us to propose a new type of database. Instead of having tables and records, it would have 'knowledge shards' and heuristics, or formal rules for grouping data, or decision rules where knowledge is preserved along with its acquisition history. The system aims to improve explainability by making the reasoning steps behind the results clear. It also enhances efficiency by using smaller, more energy-efficient LLMs optimized for specific domains rather than encompassing all of humanity’s knowledge.

Introduction and Theoretical Framework

The concept of a "Thinking Database" represents a novel approach to knowledge management and retrieval, extending beyond traditional functionalities. Unlike conventional systems that rely solely on the retrieval of embeddings or keyword-based knowledge, a "Thinking Database" leverages advanced semantic search and inference mechanisms to uncover implicit knowledge from explicit data. This approach has the potential to revolutionize information access and utilization, particularly in environments where understanding context and relationships between data points is crucial.

Semantic search is vital for modern data handling, enabling systems to interpret the meaning behind queries rather than match exact keywords. As data volume and complexity grow, more sophisticated methods are required to extract relevant information. Semantic search enhances the accuracy and relevance of results, making it indispensable in fields such as natural language processing, information retrieval, and artificial intelligence.

This research aims to develop and evaluate a "Thinking Database" model that integrates traditional inference techniques with modern machine learning approaches.

At its core, the "Thinking Database" features a dual-layered architecture combining logic-based inference and probabilistic methods with the capabilities of LLMs. Logic-based reasoning provides a structured approach to drawing conclusions, while probabilistic methods offer flexibility in handling uncertainty and variability in data. LLMs enhance these processes, particularly in scenarios where data relationships are complex or not explicitly defined. Lightweight LLMs, such as distilled models, could be selected for their cost-efficiency and ability to operate in the background without significant resource demands.

The development of a "Thinking Database" builds on advances in semantic search, embeddings, and inference techniques. Understanding these technologies' evolution and current state is essential to contextualizing our approach and identifying opportunities and challenges. Traditional databases primarily focus on keyword-based data retrieval, limiting their ability to handle complex queries where context is crucial. The advent of “Linked Data” [LD] has significantly advanced this field, allowing for interconnected and semantically enriched data across domains, paving the way for more sophisticated semantic search engines.

Technologies like YAGO [SK], a large semantic knowledge base, demonstrate the effectiveness of structuring data into ontologies for supporting semantic queries. Ontologies are crucial for defining relationships and categories within datasets, enabling more sophisticated search and retrieval processes, as Gruber [SD] discussed.

Mikolov et al. [VS] demonstrated how embeddings revolutionized the representation of words and concepts numerically by capturing their meanings and relationships. This allows databases to process and interpret queries in a manner closer to human understanding. Tools like Faiss [FAISS], Annoy [ANNOY], and Milvus [MILVUS] efficiently manage and search through high-dimensional embeddings, enabling fast and accurate retrieval in large datasets.

Inference techniques are crucial for deriving implicit knowledge from explicit data. Traditional methods include logical reasoning and probabilistic approaches. Systems like WordNet [WN], as Miller discussed, are particularly effective in structured environments because they use predefined rules to infer new information from existing data. Probabilistic methods like Latent Semantic Analysis [SA] uncover hidden structures in data, providing flexible inference capabilities in less structured environments.

LLMs represent a significant leap in performing inferences on unstructured data. Deep learning technologies, as described in detail by LeCun, Bengio, and Hinton [DL], support these models and allow them to produce text that resembles human speech and infer intricate patterns. Integrating LLM-based

inferences into a "Thinking Database" allow for more nuanced inferences that surpass traditional methods' capabilities. Lightweight LLMs, like those in Sentence Transformers [ST], are ideal for continuous background processing, refining the database without heavy resource consumption.

In the Semantic Web area of research, databases are designed to handle data in an interconnected way, where meaning and context are as crucial as the data itself. Linked Data [LD] principles exemplify this approach, emphasizing the importance of connecting related data across domains to enable potent inferences. Logical databases, such as WordNet [WN], use structured rules for deterministic inferences, which are effective in well-understood scenarios. But logical inferences, whether they are done using traditional logic-based methods or with the help of LLMs [DL], are more flexible and can handle unclear or missing data. They can also be used with real-world data that is very complicated.

The diagram below illustrates the proposed approach. The foundational concept of a "Thinking Database" is defined as a database capable of storing both primary knowledge and derived knowledge through a distillation process guided by customary inference rules.

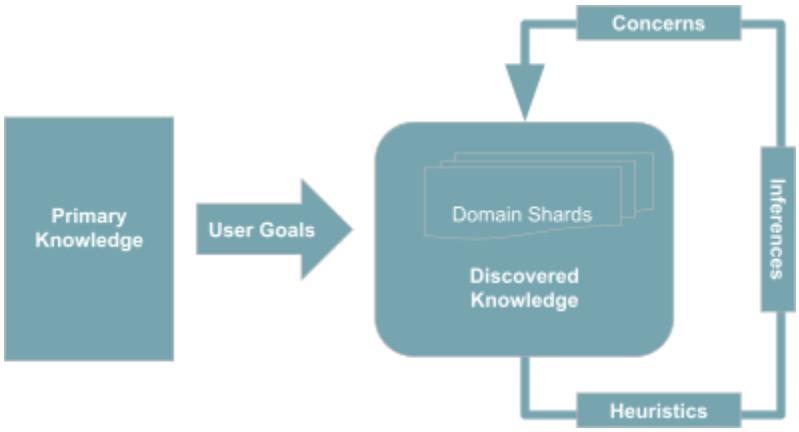

Diagram 1: Knowledge Discovery Process Based on Goals, Heuristics, Inferences and Concerns

To optimize the storage of derived data and reduce the computation, which may result from inference steps of any length, the derived knowledge is grouped into projects or domains that are aligned with user-defined goals and have resource limitations. The inference rules can be understood as user concerns or heuristics provided by experts in the relevant field. The Knowledge Discovery Process should be understood as the execution of inferences (deduction, induction, abduction, and experimentation) as long-term background activities aimed at deriving conclusions, knowledge, and insights from primary data in alignment with the goals provided by the database users.

Part of the concept involves the idea of dividing knowledge into "Knowledge Domain Shards" based on the goals provided by users and heuristics that determine which knowledge is relevant before running inferences. The knowledge obtained is then filtered through a "Concern" provided by the user. This approach offers a generic cognitive architecture that can replace the complexities of managing complex multi-agent systems while also enabling rapid exploration of use cases where long-term thinking is required to generate meaningful results with LLMs.

The ongoing development of a "Thinking Database" heavily relies on advancements in semantic search, embeddings, and inference techniques. By integrating tools and methodologies from various fields, we aim to create a system that not only stores and retrieves data effectively but also infers new knowledge. from it. This system will adapt and learn over time, offering a powerful tool for data-driven decision-making across diverse applications.

From Multi-Agent Systems to Thinking Databases

The proposal of the “Thinking Database” offers a compelling alternative to traditional Multi-Agent Systems (MAS) by simplifying the architecture and making it more accessible for experimentation and broader application. While MAS can intuitively appeal as a model of specialized employees within an organization, where each agent performs tasks based on unique expertise, the “Thinking Database” provides a more generalized cognitive architecture. This approach is adaptable to a wide range of use cases, allowing for a versatile and user-friendly system that can be more easily controlled and deployed by end users.

In a MAS framework, the specialization of agents—each potentially powered by a different LLM—can lead to more refined and context-specific operations. For instance, research by Agarwal et al. (2022) and Bubeck et al. (2023) demonstrates how LLMs in MAS can handle tasks requiring nuanced understanding and decision-making, akin to specialized employees in an organization (GAG, AGIM). These agents can communicate, collaborate, and solve complex problems effectively, leveraging their distinct training and data sources, as shown in works like those of Mou et al. (2021) and Jiang et al. (2021)(CMA, MAC).

However, the “Thinking Database” approach consolidates these capabilities into a unified system that does not rely on the intricate coordination of multiple specialized agents. Instead, it employs a general pattern or cognitive architecture that can handle various tasks through a single, adaptable framework. This reduces the complexity inherent in MAS, where maintaining coordination and integration across multiple LLMs can be challenging. The “Thinking Database” allows for a streamlined experience, making it easier to develop tools and control the exploitation of AI by end users.

Moreover, while MAS benefits from the ease of deploying different LLMs with specialized capabilities, this specialization can also be achieved within a “Thinking Database” by customizing inference rules, heuristics, and concerns to utilize LLMs in a targeted manner. For example, a more powerful LLM might be used to process and filter broad concerns, while less resource-intensive models handle specific inferential tasks. This method not only retains the advantage of specialized LLMs but also integrates them into a more cohesive and manageable system, as suggested by frameworks like LangChain and RLlib, which support composability and scalability within a single system architecture (LGC, RAYP).

While MAS offers advantages in specialization and the ability to utilize different LLMs effectively, the “Thinking Database” provides a more straightforward and flexible alternative. It enables a unified approach to AI deployment across various applications, making it easier to experiment with and control while still allowing for the integration of specialized LLM capabilities through tailored inference and heuristic adjustments. This makes the “Thinking Database” not only a viable alternative but potentially a more scalable and user-friendly solution for many use cases.

Use cases. Better Explainability.

This report offers a wide range of use cases that make the most of its capacity to infer new information from explicitly added user knowledge, improving both the accuracy and the transparency of data-driven decision-making. One prominent application is in research and development, where scientists and engineers input explicit knowledge such as formulas, experimental results, or theoretical models into the database. We hypothesize that such a database that infers new insights or hypotheses by cross-referencing existing data will be very helpful to accelerate innovation. For example, by integrating information on material properties, the database could deduce potential new applications for a given material. The inferred information is presented alongside the logical steps or models used to reach the conclusion, allowing researchers to understand and verify the thought process behind these new insights.

In the financial sector, the Thinking Database can analyze market trends and economic data provided by analysts. As it continuously processes incoming data, it can infer potential future market movements or investment opportunities. Importantly, these inferences are not black-box predictions; the system provides a detailed breakdown of the reasoning or probabilistic models that led to its conclusions, offering transparency that is critical for making informed financial decisions.

In healthcare, the system can assist in diagnostics by combining explicit medical knowledge input by doctors with patient data. Over time, the Thinking Database could infer potential diagnoses or treatment plans for new patients by identifying patterns in the data. Doctors can review these suggestions along with the specific medical logic or clinical guidelines applied by the system, ensuring that the inferred recommendations are not only accurate but also fully explainable.

Another compelling use case is in legal research, where the database can help identify relevant case laws or statutes based on explicit legal principles entered by legal professionals. When the system suggests analogous cases or legal arguments, it also provides the reasoning process, citing how it interpreted the legal principles and applied them to the current context. This explainability ensures that lawyers can trust the system's output and understand the basis for its legal reasoning.

Finally, in education, the Thinking Database can be used to create adaptive learning environments. As students interact with the system, it can infer areas where they may need further practice or suggest new topics to explore. The system is a valuable tool for personalized education because it provides a clear explanation of how it evaluates the student's progress and learning style.

Annex 1 at the end of this report presents concrete examples that may help understand the vision. Across all these use cases, the Thinking Database stands out not only for its ability to infer new knowledge from user-provided data but also for its commitment to explainability. Whether through logical reasoning, probabilistic methods, or formal inference models, the system always provides a clear, understandable path that shows how it arrived at its conclusions. This transparency not only builds trust but also allows users to learn from the system's reasoning, making the Thinking Database a powerful tool for enhancing decision-making processes across various fields.

Preliminary Architecture and Implementation Considerations

A robust architecture for a “Thinking Database” has to be built upon several key components that ensure the system can effectively gather, process, and infer knowledge in a manner that is both reliable and transparent. The above presents six components labeled C1–C6.

The "Validate Data Gathering” component is the foundation for any inference process within the Thinking Database. This component is responsible for collecting explicit data, which serves as the basis for all subsequent inferences. Beyond simple data collection, it also assesses the reliability and trustworthiness of the data, assigning a confidence level to each data point. This guarantees that the system's inferences are based on reliable, high-quality data.

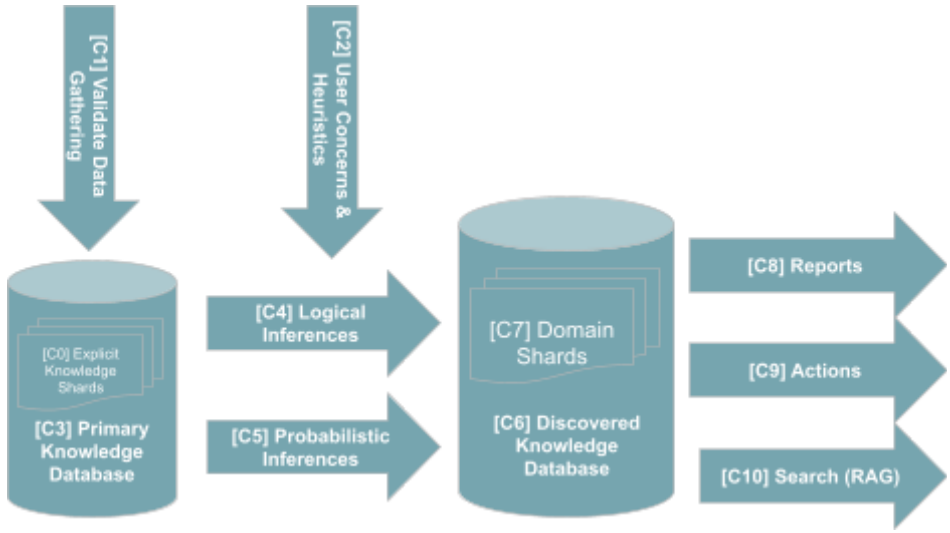

Diagram 2: High-Level Architecture

The “User Concerns & Heuristics” component manages the integration of expert input and user-defined priorities into the inference process. This component allows users to express key concerns and important aspects that should be considered during inference, effectively guiding the system's reasoning processes. It also gathers heuristics and expert insights in a manner similar to expert systems of the 1980s, but with a modern twist. Thanks to the capabilities of LLMs, this process is far less labor-intensive and more flexible, enabling the system to adapt quickly to various domains and scenarios.

The “Discovered Knowledge Database” serves as the repository for knowledge that the system has deemed sufficiently interesting and valuable, based on criteria established in the Goals, Concerns and Heuristics component. This database stores knowledge that is not only preserved for long-term use but also accessible for query-based retrieval, particularly in Retrieval-Augmented Generation (RAG) scenarios. Additionally, this repository supports explainability by providing insights into how the knowledge was derived, offering users a clear understanding of the system’s reasoning.

The “Primary Knowledge Database" is where all primary data and associated confidence levels are stored. This database acts as the backbone of the system, maintaining the raw data upon which all inferences are based. The inclusion of confidence levels helps to prioritize and weigh the data during inference, ensuring that the most reliable information is used to guide decision-making processes.

The “Logical Inferences” component is responsible for performing structured, formal inferences that can be considered part of a rigorous demonstration or proof. This includes deductive reasoning, abstraction, and inductive reasoning, all of which are essential for generating conclusions that are logically sound and verifiable. The output from this component is particularly valuable in contexts where high accuracy and reliability are required.

The "Probabilistic Inferences" part lets you make probabilistic inferences that are driven by heuristics. Micro LLMs or other inference engines are often used to help with this. While these inferences may not carry the same level of certainty as those generated by the Logical Inferences component, they offer a practical compromise between computational efficiency and accuracy. This component is especially useful in real-world scenarios where data is incomplete or ambiguous, and the best available inference is one that balances precision with resource constraints.

In the diagram above, we also highlight component C7, which represents the method of segregating knowledge into specialized domains or areas of interest. The conceptual components C7 and C0 aim to highlight the difference between explicit knowledge (C0), which can be traced back to an external source recognized by the system as valid, either with a degree of accuracy or complete confidence, and an implicit knowledge shard (C7), which is inferred by the Thinking Database based on information from other shards, whether explicit or implicit. These types of knowledge have the interesting property that the reasoning process leading to their creation can be traced and recognized, helping with explainability.

This approach is intended to limit the number of inferences that need to be performed iteratively by the Thinking Database. Additionally, components 8, 9, and 10 are emphasized as mechanisms for extracting information, specifically through the generation of relevant reports, the triggering of actions, and the utilization of knowledge for Retrieval-Augmented generation (RAG) and search functionalities.

The above components are assumed to perform all the required functions, starting with collection and data validation but also incorporating user priorities, generating and storing knowledge, and performing a range of inferences with varying degrees of certainty and complexity. This architecture supports a dynamic and powerful system capable of evolving and adapting over time, all while maintaining a high level of transparency and explainability.

At this stage, our "Thinking Database" concept is at Technology Readiness Level (TRL) 2, where we are exploring foundational technologies and conducting preliminary experiments. Two systems, Milvus and Weaviate, have shown significant potential to support long-term inference capabilities and continuous relationship management between old and new data, which are central to our approach.

Milvus is a scalable vector management system designed for efficiently indexing and searching large datasets. Although Milvus does not natively perform inferences, it can be integrated with machine learning models to enable such capabilities. By incorporating machine learning, Milvus can help identify patterns and relationships in data, allowing the system to update its knowledge base dynamically as new information is ingested.

Weaviate is a vector database tailored for contextual queries and supports automatic classification and machine learning modules. This enables Weaviate to perform ongoing inferences on the stored data, continuously refining its understanding as new data is added. Its built-in capabilities allow it to establish and update semantic relationships, making it well-suited for scenarios where data is constantly evolving.

The proposed system architecture leverages the strengths of both Milvus and Weaviate. Data is first ingested and transformed into embeddings via neural network models. These embeddings are stored in Milvus, which handles the heavy lifting of vector searches with its scalable architecture. Simultaneously, Weaviate manages the semantic layer by analyzing these embeddings and establishing connections between new and existing data. This dual-layered approach ensures that while Milvus focuses on efficient

data retrieval, Weaviate enhances the system's ability to make informed inferences, adapting to the evolving dataset.

Annex 1 provides some hypothetical examples of use cases where the system proposed in this report could be effectively utilized. The focus is on applying inference rules described in natural language, allowing the system to deduce new knowledge based on these inferences and the concerns and heuristics defined by human users. This approach has the potential to enhance the expertise of human specialists in ways that are more sophisticated and practically unattainable by large companies like OpenAI. While such companies may have access to vast amounts of public information on the internet, they lack granular access to the deep expertise and insights of domain experts.

Many of the intermediate conclusions that form part of the reasoning leading to significant, publicly noteworthy conclusions are highly valuable for fostering intelligible and explainable dialogue. However, these intermediate insights tend to be overlooked or lost in public documents, including scientific articles that are constrained by space and often assume that the reader can follow logical leaps. This assumption may not hold true for general LLMs, which cannot possess a nuanced understanding of all the scientific, technical, or locally specific rules of the organizations or communities they serve. Our approach enables the tuning and focusing of LLMs on what truly matters, ensuring that these critical intermediate insights are preserved and leveraged effectively.

We believe that due to the immense computational complexity and inherent limitations of LLM technologies, along with the sheer complexity and vast inference trees required for global knowledge, our proposal stands as a uniquely viable path for improving AI systems while simultaneously decentralizing AI. This is achieved by necessitating expertise and a focus on niche domains, making it impractical to create super AIs that know and understand everything.

In practice, this system could be used in scenarios such as enhanced semantic search, where users retrieve documents or data points based on conceptual similarity rather than exact matches. Another example is real-time data integration, where the system automatically identifies and links new data with existing knowledge to detect trends or anomalies. Through continuous learning and adaptation, the "Thinking Database" aims to offer powerful semantic search and inference capabilities, advancing the way databases can interact with and learn from the data they store.

The art of sharding

A crucial aspect of the "Thinking Database" is its method of data segmentation into “Domain Shards” or “Knowledge Shards." Within the database context, a shard can be likened to the concept of tables in traditional database systems. This segmentation allows for the organization of knowledge into shards based on the relevance established by heuristic grouping of data.

To effectively estimate the size and scope of data within a shard, several parameters need to be considered. These include the minimum number of domain shards necessary for a useful use case, the estimated depth of inference required per shard, the number of LLM (Large Language Models) prompts needed per shard to initialize and maintain it, and the estimated number of discrete knowledge elements within a typical shard. Given that each knowledge shard can encompass tens or hundreds of scientific articles or documentation spanning hundreds of pages, this could lead to the existence of thousands, or even tens of thousands, of facts or knowledge units within a single shard.

This structured approach not only facilitates more efficient data management but also enhances the database's capacity to perform complex inferential tasks, making it a pivotal strategy in the architecture of the "Thinking Database." This chapter does not delve into the sophisticated methodologies employed in the segmentation of data to optimize both retrieval and inferential processes. Currently, this is still an open

area for research; some simplistic methods are obviously possible. However, more sophisticated methods capable of empowering the database's ability to handle vast arrays of knowledge with agility and precision seem required. The list of open problems includes the way in which the natural language inference rules, goals, grouping heuristics, and concerns can be expressed as templates in a special language that can refer to shards or instances of objects in these shards in such a way as to reduce the complexity of computation and allow LLMs to focus on the desired aspects during the actual inferences.

Evaluation

A strict method must be used to test how well the "Thinking Database" works. This method must include comparing it to other semantic search databases and using certain metrics to check the accuracy of inferences and overall performance.

The first step in this methodology will be to conduct a “comparative analysis” between the Thinking Database and established semantic search systems. This comparison will involve benchmarking tasks where both systems are presented with identical datasets and queries. The primary focus will be on evaluating the ability of each system to retrieve relevant information and generate accurate inferences. Specific attention will be paid to how each system handles complex queries that require understanding the context and relationships between data points, which is where the Thinking Database is expected to excel.

To measure the effectiveness of the Thinking Database, a set of metrics for evaluating inference accuracy and performance will be employed. These metrics include precision, recall, and F1 score, which will assess the accuracy of the system in retrieving relevant data and making correct inferences. Additionally, inference confidence levels, generated by both logical and probabilistic inference engines, will be evaluated to determine the reliability of the system’s conclusions. The system’s performance metrics will also include response time and computational efficiency, ensuring that the Thinking Database can operate effectively under various workloads without compromising accuracy.

Through this methodology, we aim to demonstrate not only the superior inferencing capabilities of the Thinking Database but also its practical efficiency compared to traditional systems. This approach will provide a comprehensive evaluation of the system's strengths and areas for improvement, ultimately validating its potential for real-world applications.

Conclusions

The “Thinking Database” concept enhances data management by integrating semantic search with embeddings and advanced inference techniques. With the aid of lightweight, energy-efficient LLMs, it improves on conventional systems by providing deeper insights through logic-based and probabilistic reasoning. This approach not only provides more accurate and contextually relevant information but also enhances explainability in LLM-based systems. By optimizing the database for specific domains and using smaller LLMs, the system balances efficiency and performance without attempting to encompass all knowledge at once.

In conclusion, the "Thinking Database" aims to become a tool designed to handle the combinatorial explosion—a rapid increase in complexity based on input and constraints—associated with data inferences. This strategic tool is intended to be an indispensable asset for AI engineers in managing knowledge effectively. Positioned under the control of multi-agent systems that aim to automate even aspects of knowledge management, the approach underscores practical, automated interactions, recognizing that even the most advanced models require meticulous fine-tuning to effectively navigate the exponential growth in problem complexity.

References

[AGIM] Bubeck, S., et al. (2023). "Sparks of Artificial General Intelligence: Early Experiments with GPT-4." Microsoft Research.
[GAG] Agarwal, P., et al. (2022). "Generative Agents: Interactive Simulacra of Human Behavior." arXiv preprint arXiv:2210.00067.
[AMA] Li, Y., et al. (2023). "Adaptive Multi-Agent Systems using GPT-based Reinforcement Learning." Journal of Artificial Intelligence Research (JAIR), 2023.
[CMA] Mou, L., et al. (2021). "Collaborative Multi-Agent Dialogue with GPT-3." In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL 2021).
[MAC] Jiang, Y., et al. (2021). "Multi-Agent Communication with Graph Neural Networks and Large Language Models." NeurIPS 2021.
[LGC] LangChainAI. (2023). "LangChain: Building applications with LLMs through composability." https://github.com/langchain-ai/langchain
[RAYP] ray-project. (2023). "RLlib: Scalable Reinforcement Learning Library." https://github.com/ray-project/ray
{MSFA] Microsoft. (2023). "Autonomous Agents with GPT: A framework for building autonomous agents using GPT models." https://github.com/microsoft/autonomous-agents-gpt
[OAIA] Openai. (2023). "Multi-Agent Emergent Tool Use: A project exploring emergent behavior in multi-agent systems using GPT models." https://github.com/openai/multi-agent-emergent-tool-use
[LD] **Bizer, C., Heath, T., & Berners-Lee, T. (2009).** *Linked Data - The Story So Far.* Semantic Web and Information Systems, 5(3), 1-22. (Discusses the principles and practices of linked data, which are fundamental for semantic search databases).
[DL] **LeCun, Y., Bengio, Y., Hinton, G. (2015).** *Deep learning.* Nature, 521(7553), 436-444. (Provides foundational knowledge on deep learning technologies which are crucial for developing advanced LLMs).
[VS] **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).** *Efficient Estimation of Word Representations in Vector Space.* Proceedings of Workshop at ICLR. (Important for understanding embeddings and their application in semantic search).
[LSA] **Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990).** *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science, 41(6), 391-407. (Classic work on semantic analysis and its application to information retrieval).
[WN] **Miller, G. A. (1995).** *WordNet: A Lexical Database for English.* Communications of the ACM, 38(11), 39-41. (Discusses the design and application of WordNet, a database used extensively in NLP for semantic searches).
[SD. **Gruber, T. R. (1993).** *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 5(2), 199-220. (Essential reading on ontologies, which are crucial for knowledge representation in semantic databases).
[SK] **Suchanek, F. M., Kasneci, G., & Weikum, G. (2007).** *Yago: A Core of Semantic Knowledge.* Proceedings of the 16th international conference on World Wide Web, 697-706. (Describes YAGO, a large knowledge base, relevant for semantic data storage and inference).
[INF] **Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... & Kingsbury, B. (2012).** *Deep Neural Networks for Acoustic Modeling in Speech Recognition.* IEEE Signal Processing Magazine, 29(6), 82-97. (Provides insights into the use of deep neural networks, applicable in LLM frameworks for inference).

[SEM] **Bernstein, A., & Haas, L. M. (2008).** *Information integration in the enterprise.* Communications of the ACM, 51(9), 72-79. (Discusses challenges and solutions in data integration, relevant for developing a unified semantic search platform).
[FAISS] Faiss (Facebook AI Similarity Search): Efficient for nearest neighbor search (https://github.com/facebookresearch/faiss)
[ANNOY] Annoy (Approximate Nearest Neighbors Oh Yeah): *Fast neighbor search for embeddings* (https://github.com/spotify/annoy)
[MILVUS] Milvus: Scalable vector management system (https://github.com/milvus-io/milvus)
[Weaviate] Weaviate:Vector database for contextual queries (https://github.com/semi-technologies/weaviate)
[ST] Sentence Transformers: Framework for sentence embeddings from BERT (https://github.com/UKPLab/sentence-transformers)
[SP] StarSpace: Learning embeddings for multiple data types (https://github.com/facebookresearch/StarSpace)
[GENSIM] Gensim: Semantic analysis and topic modeling for NLP (https://github.com/RaRe-Technologies/gensim)
[FastText] FastText: Library for learning word embeddings (https://github.com/facebookresearch/fastText)

Annex 1: Use Cases

The tables below illustrate the use cases discussed, presenting both foundational and inferred knowledge at a high level. It is important to understand the types of rules as well as the specific user concerns associated with each type of knowledge and the domain in which the database will be utilized. The examples and rules were generated using ChatGPT, so they should not be taken as strict guidelines; they are provided solely to exemplify the concept, and we used them to analyze the inputs, outputs, goals, concerns, and relevant heuristics for grouping knowledge in computationally friendly shards.

The use cases span a wide range of domains, including AI scientific article review, material science research, real-time anomaly detection in financial transactions, personalized diagnostics and monitoring, legal compliance monitoring, personalized learning, code copilot for complex software projects, automated compliance documentation monitoring, employee performance monitoring and feedback, automated data quality assurance, supply chain optimization, IP legal research, adaptive cybersecurity threat detection, predictive maintenance in manufacturing, financial market analysis, semantic audit of a company or project, ensuring consistency and quality in film and TV show scenarios, automated quality control in manufacturing, AI-powered employee onboarding and training, smart energy management in urban areas, real-time environmental monitoring and response, automated detection of insider threats, proactive customer support in telecom, intelligent financial portfolio management, predictive talent retention strategies, automated news aggregation and fact-checking, personalized dietary and fitness recommendations, smart inventory management in retail, AI-enhanced virtual assistants for remote work, real-time sentiment analysis for brand management, AI-powered recruitment and candidate matching, automated social media content moderation, dynamic pricing strategies in retail, AI-powered drug discovery, and smart waste management in cities.

Each use case requires different levels of inference to generate new knowledge. Simpler scenarios, such as legal compliance monitoring, automated data quality assurance, and smart inventory management in retail, typically require at least 2-3 levels of inference, focusing on straightforward checks, validations, and optimizations. More complex scenarios, such as personalized diagnostics and monitoring, adaptive cybersecurity threat detection, and AI-powered drug discovery, demand at least 5-10 levels of inference, where multiple data sources must be analyzed, correlated, and synthesized to produce meaningful insights. In these annexes, we have focused on exploring use cases for the Thinking Database that utilize LLMs for all phases of "thinking." However, more formal models of "thinking" could certainly require hundreds of inference rules and specialized methods for filtering, grouping, and evaluating knowledge. The low-hanging fruit, however, is to use LLMs for performing all the inferences and to focus on maturing use cases where LLMs can be directly usable due to their capacity to incorporate human expert input and insights at a high level in natural language. If the heuristics for selecting relevant data are weak, the number of inferences required, even for just 4–5 levels of inferences performed on a database containing thousands or millions of foundational knowledge elements, can quickly escalate to astronomically large numbers, making the process computationally expensive and impractical.

The number of inference levels is determined by the complexity of the task, the diversity of input data, and the need for accuracy and transparency in decision-making processes. By grouping use cases based on these criteria, we ensure that the system remains efficient while being capable of handling the deeper analysis required for more sophisticated tasks while still being able to estimate the costs in terms of time and resources based on the size of the input data. The classification reflects a balance between computational efficiency and the depth of reasoning needed to achieve reliable and contextually appropriate outcomes, with the grouping serving as a guide to optimizing the system's performance for each specific application.

Use Case 0: Generic Search Assistance for Any Type of Document

Input Knowledge	• Diverse document types (e.g., PDFs, Word documents, web pages, emails). • Metadata associated with documents (author, date, keywords, etc.). • User queries and search history. • Document content and structure (headings, sections, tables, etc.).
Goals, Concerns and Heuristics	**Goals:** • Provide relevant search results tailored to specific user-defined queries. • Group documents based on their contextual relevance to the search query. • Enable efficient retrieval of information across different document formats. **Concerns** • Avoid irrelevant or low-quality results that do not match the user's intent. • Ensure that the grouping of documents is flexible and adaptive to varied goals. • Maintain a high level of precision and recall in search results. **Heuristics:** • Group documents based on shared keywords, themes, or contextual similarities. • Prioritize documents with higher relevance scores based on user goals. • Use metadata to refine search results (e.g., prioritizing recent documents or those from specific authors). • Adjust the weighting of documents based on user interaction data and feedback.
Inference Rules	• Inference Rules • If a document shares more than 70% of keywords with other documents in the search results, group them together under a common theme. • If a user query includes specific terms related to a field or topic, prioritize documents with corresponding metadata tags or keywords. • If a document has a high frequency of technical terms relevant to the query, increase its relevance score and prioritize it in the search results. • If user feedback indicates that certain types of documents (e.g., PDFs or reports) are consistently preferred, adjust the system to prioritize these formats in future searches. • If multiple documents cover the same topic but from different perspectives, suggest grouping them as a "comprehensive view" for the user. • If a document is frequently accessed or marked as favorites by multiple users for similar queries, suggest it as a top recommendation for related searches.

Generated Knowledge	Grouped and ranked lists of documents that are highly relevant to specific user-defined goals. A dynamically updated database of document relationships based on user interactions and feedback improves the accuracy of future searches. Insights into the most commonly searched themes or topics, helping users refine their queries or explore related areas of interest. Suggested document collections that provide a comprehensive understanding of a specific topic from multiple angles.

Estimated size

Inference Rules	• Inference Rules • If a document shares more than 70% of keywords with other documents in the search results, group them together under a common theme. • If a user query includes specific terms related to a field or topic, prioritize documents with corresponding metadata tags or keywords. • If a document has a high frequency of technical terms relevant to the query, increase its relevance score and prioritize it in the search results. • If user feedback indicates that certain types of documents (e.g., PDFs or reports) are consistently preferred, adjust the system to prioritize these formats in future searches. • If multiple documents cover the same topic but from different perspectives, suggest grouping them as a "comprehensive view" for the user. • If a document is frequently accessed or marked as favorites by multiple users for similar queries, suggest it as a top recommendation for related searches.
Generated Knowledge	Grouped and ranked lists of documents that are highly relevant to specific user-defined goals. A dynamically updated database of document relationships based on user interactions and feedback improves the accuracy of future searches. Insights into the most commonly searched themes or topics, helping users refine their queries or explore related areas of interest. Suggested document collections that provide a comprehensive understanding of a specific topic from multiple angles.
Shards Examples	Shard 1: Web pages (HTML content, metadata) Shard 2: Scholarly articles (abstracts, references) Shard 3: Books and publications (indexes, chapters) Shard 4: Emails and communications (subject lines, attachments) Shard 5: Social media content (posts, tags)

Use Case 1: AI Scientific Article Review

Input Knowledge	The reviewed paper. Set of scientific papers in the domain, and the one cited by the reviewed paper.
Goals, Concerns and Heuristics	Goals: Identify inconsistencies or gaps in the reviewed research. Heuristic: Investigate models that show more than a 10% difference in accuracy in comparative studies. Concern: Identify inconsistencies or gaps in the reviewed research. Ensure methodological soundness in comparative studies.
Inference Rules	• If a machine learning model demonstrates over 85% accuracy in three or more independent studies, prioritize this model for similar tasks. • If two studies on the same topic show more than a 10% difference in accuracy, investigate potential methodological flaws. • If a model's performance improves by more than 15% when using a specific dataset, recommend that dataset for future benchmarks. • If a study uses a sample size smaller than the field's median, adjust the confidence level of its results downward. • Other inference rules specific to AI/ML research
Generated Knowledge	The "Thinking Database" inferred that while CNNs are highly effective for image classification, they may not generalize well to sequence prediction tasks. The system also highlighted areas of potential methodological weakness in the studies, recommending further investigation.
Shards Examples	Shard 1: Neural network methodologies (papers, studies) Shard 2: Reinforcement learning applications (case studies, results) Shard 3: Comparative analysis of AI models (benchmark data, performance reviews) Shard 4: Ethical implications of AI (regulatory guidelines, ethical debates) Shard 5: AI in healthcare (specific studies, regulatory compliance)

Use Case 2: Material Science Research

Input Knowledge	Dataset A: Thermal conductivity data of alloy X (250 W/m·K). Dataset B: Corrosion resistance of similar alloys in saline environments.
Goals, Concerns and Heuristics	Goals: Focus on materials with thermal conductivity above 200 W/m·K for aerospace applications. Ensure material longevity in extreme environments. Heuristic: Recommend alloys with high thermal conductivity and low thermal expansion. Concern: Ensure the material's longevity in extreme environments like space.
Inference Rules	• If an alloy exhibits thermal conductivity above 200 W/m·K and low thermal expansion, recommend it for applications requiring thermal stability. • If an alloy shows 70% or greater corrosion resistance in saline environments, suggest it for marine applications. • If an alloy's mechanical strength exceeds the industry standard by 15%, propose it for high-load structural applications. • If an alloy passes fatigue tests with less than 5% degradation after 10,000 cycles, endorse it for aerospace use. • Other inference rules specific to the domain
Generated Knowledge	The system recommended alloy X for aerospace applications due to its high thermal conductivity and a 75% likelihood of good corrosion resistance in space, based on adjusted environmental conditions.
Shards Examples	Shard 1: Alloy compositions (chemical properties, uses) Shard 2: Polymer Studies (synthesis methods, applications) Shard 3: Ceramic materials (thermal properties, manufacturing processes) Shard 4: Composite materials (mechanical testing, application scenarios) Shard 5: Nanomaterials (synthesis, unique properties)

Use Case 3: Real-Time Anomaly Detection in Financial Transactions

Input Knowledge	• Historical transaction data, including known patterns of normal and fraudulent activities. • Regulatory guidelines and compliance standards for financial transactions. • Industry best practices for fraud detection and financial risk management.
Goals, Concerns and Heuristics	**Goals:** Prioritize detection of transactions that deviate significantly from established patterns. **Heuristic:** Flag transactions that deviate more than 20% from typical ranges. **Concern:** Minimize false positives while ensuring rapid detection of genuine anomalies.
Inference Rules	• If a transaction amount deviates by more than 20% from a user's typical transaction range, flag it for review. • If a transaction occurs outside of usual business hours or geographic locations, trigger an alert for potential fraud. • If a sequence of transactions shows unusual frequency or high amounts, recommend a deeper investigation for potential money laundering. • Various rules and detection algorithms based on new fraud patterns identified across the industry.
Generated Knowledge	The system continuously monitors financial transactions, generating real-time alerts for any anomalies that deviate from established norms. It provides actionable insights, such as flagging suspicious transactions for further investigation, while minimizing false positives by refining detection algorithms with the latest industry data.
Shards Examples	Shard 1: Credit card transactions (purchase patterns, fraud cases) Shard 2: Stock trading anomalies (trading patterns, suspicious activities) Shard 3: Real estate transactions (pricing anomalies, market trends) Shard 4: Cryptocurrency transactions (blockchain activities, unusual patterns) Shard 5: Loan applications (credit score anomalies, falsified information)

Use Case 4: Personalized Diagnostics and Monitoring

Input Knowledge	• Data A: Patient A's symptoms (fatigue, shortness of breath, chest pain). • Clinical guidelines for diagnosing heart conditions.
Goals, Concerns and Heuristics	**Goals**: Focus on heart conditions with overlapping symptoms. **Heuristic**: Prioritize conditions when three or more consistent symptoms are present. **Concern**: Avoid false positives in high-risk diagnoses.
Inference Process	• If a patient exhibits three or more symptoms consistent with a specific disease (e.g., fatigue, shortness of breath, chest pain for heart failure), prioritize that disease in the diagnostic process. • If a patient has a family history of a particular condition, increase the likelihood of that diagnosis by 20%. • If a patient's lab results match 80% or more of the diagnostic criteria for a condition, suggest immediate further testing. • If symptoms are severe and match early-stage disease patterns, recommend preemptive treatment options. • Other inference rules specific to healthcare
Generated Knowledge	The system provided a diagnosis of early-stage heart failure with a 70% probability, supported by alignment with clinical guidelines and adjusted for demographic risk factors.
Shards Examples	Shard 1: Cardiovascular conditions (symptoms, treatments) Shard 2: Diabetes management (blood sugar levels, dietary adjustments) Shard 3: Oncology (cancer types, new research) Shard 4: Chronic diseases (long-term treatment strategies, patient data) Shard 5: Pediatric conditions (common ailments, growth milestones)

Use Case 5: Legal Compliance Monitoring

Input Knowledge	• Legal and regulatory standards relevant to the industry. • internal company policies and document templates. • Historical audit findings and common compliance issues.
Goals, Concerns and Heuristics	**Goals**: Ensure all documents adhere to legal and regulatory requirements. **Heuristic**: Flag deviations from approved legal templates for human review. Trigger alerts for documents missing mandatory sections. **Concern**: Quickly identify sections in documents that require human review due to potential legal risks or deviations from standards.
Inference Rules	• If a document contains language that deviates from approved legal templates or introduces new legal terms, flag it for human review. • If mandatory sections required by regulation are missing or incomplete, trigger an alert for immediate correction. • If a document includes clauses that conflict with existing company policies or regulations, recommend legal review and potential revision.
Generated Knowledge	The system continuously scans all generated documents, providing real-time alerts for sections that deviate from legal standards or internal policies. It highlights areas that require immediate human review, such as missing mandatory sections, non-compliant language, or conflicting clauses, ensuring that all documents meet compliance requirements before finalization.
Shards Examples	Shard 1: Data privacy laws (GDPR, HIPAA regulations) Shard 2: Employment law (worker rights, safety regulations) Shard 3: Intellectual property (copyright, patent laws) Shard 4: Environmental compliance (EPA standards, waste management) Shard 5: Financial regulations (SEC guidelines, banking laws)

Use Case 6: Personalized Learning

Input Knowledge	Class Data: Performance data of students in a specific class over the academic year, including quiz scores, assignment grades, participation records, and feedback from instructors. Student Profiles: Individual student profiles that include learning styles (e.g., visual, auditory, kinesthetic), previous academic performance, and areas of interest. Curriculum Content: A detailed breakdown of the curriculum, including key concepts, learning objectives, and the difficulty level of each module.
Goals, Concerns and Heuristics	**Goals**: Prioritize personalized learning paths for students with consistent performance gaps. **Heuristic**: Suggest tailored tutoring for students consistently underperforming in assessments. Recommend advanced materials for students excelling in specific subjects. **Concern**: Ensure recommendations align with the student's learning style without overwhelming them.
Inference Rules	• If a student consistently underperforms in assessments (e.g., scoring below 60% across multiple tests), recommend targeted review sessions focusing on foundational concepts or personalized tutoring to address their struggles. • If a student's performance significantly varies between assessment types (e.g., excelling in visual-based assessments but not others), suggest tailoring learning resources and evaluation methods (e.g., more visual aids or verbal evaluations) to their preferred learning style. • If a student consistently excels in a particular subject area (e.g., scoring above 90%), recommend advanced materials, projects, or extracurricular activities to further challenge and engage them in that subject. • If a student shows a sudden drop in performance, cross-reference with changes in participation or attendance and recommend a meeting with the instructor to discuss potential underlying issues. • If the class average on a new topic is significantly lower than previous topics, suggest reviewing and potentially adjusting the curriculum or allocating more instructional time to that topic.
Generated Knowledge	The system generates personalized learning plans for each student based on the inferences made from their performance data, learning styles, and participation patterns.
Shards Examples	Shard 1: Mathematics (curriculum, problem sets) Shard 2: Science education (experiments, scientific method) Shard 3: Literature (reading lists, essay writing) Shard 4: History (historical events, timelines) Shard 5: Language learning (grammar, vocabulary)

Use Case 7: Code Copilot for Complex Software Projects

Input Knowledge	• Software project specifications (e.g., requirements, architecture diagrams). • Historical codebase and associated documentation. • Software development best practices. • Coding standards and guidelines. • Historical maintenance cost data and common sources of technical debt.
Goals, Concerns and Heuristics	**Goals:** Prioritize maintaining code quality to minimize long-term maintenance costs. **Heuristic:** trigger warnings if the new code deviates more than 5% from established standards. Analyze and recommend refactoring strategies if function complexity increases by more than 15%. **Concern:** Prevent the accumulation of technical debt by ensuring adherence to coding standards.
Inference Rules	• If new code introduces more than 5% deviation from established coding standards, trigger a warning for potential technical debt. • If a function's complexity increases by more than 15%, recommend refactoring to prevent maintenance difficulties. • If code changes involve more than 10% undocumented or unclear logic, flag them for review to ensure maintainability. • Regularly analyze the codebase to identify potential areas of technical debt accumulation and suggest preventive measures.
Generated Knowledge	The system continuously monitors the codebase, generating real-time alerts when new code deviates from best practices or exhibits signs of increasing technical debt. It provides actionable suggestions, such as refactoring complex functions or adding documentation to unclear logic, to maintain code quality and reduce future maintenance costs.
Shards Examples	Shard 1: API documentation (usage patterns, examples) Shard 2: Code repositories (version control, commit logs) Shard 3: Architecture Designs (system diagrams, specifications) Shard 4: Bug reports (issue tracking, resolution) Shard 5: Performance optimization (profiling data, enhancement techniques)

Use Case 8: Automated Compliance Documentation Monitoring

Input Knowledge	• Regulatory guidelines (e.g., GxP standards). • Internal documentation standards and templates. • Historical audit reports and common compliance issues.
Goals, Concerns and Heuristics	**Goals:** Ensure all documentation meets regulatory and internal standards to avoid compliance issues. **Heuristic:** Trigger alerts if new documentation lacks or has shallow mandatory sections required by standards like GxP. **Concern:** Prevent non-compliance by maintaining high documentation quality and adherence to best practices.
Inference Rules	• If new documentation lacks mandatory sections required by GxP standards, trigger an immediate compliance alert. • If documentation deviates from the standard template or language, flag it for revision to ensure consistency. • If audit trails or revision histories are incomplete or unclear, recommend corrective actions to align with regulatory expectations. • Regularly review documentation to ensure ongoing compliance and recommend updates based on the latest regulatory changes.
Generated Knowledge	The system continuously evaluates the quality of documentation, generating real-time alerts when new documents fail to meet GxP standards or internal guidelines. It provides actionable suggestions, such as adding missing sections, revising inconsistent language, or updating audit trails, to ensure full compliance and maintain high documentation quality.
Shards Examples	Shard 1: Regulatory changes (updates, new regulations) Shard 2: Standard operating procedures (manuals, guidelines) Shard 3: Audit trails (historical data, compliance checks) Shard 4: Safety protocols (industry-specific safety requirements) Shard 5: Quality assurance (standards enforcement, process verification)

Use Case 9: Employee Performance Monitoring and Feedback

Input Knowledge	• Historical performance data (e.g., project completions, KPIs, peer reviews). • Company performance benchmarks and standards. • Employee engagement surveys and feedback.
Goals, Concerns and Heuristics	**Goals**: Ensure continuous improvement and the meeting of performance standards. **Heuristic**: Recommend performance improvement plans if an employee underperforms relative to benchmarks. Prepare career advancement opportunities for employees who consistently exceed targets. **Concern**: Identify and address performance issues early to enhance productivity and job satisfaction.
Inference Rules	• If an employee consistently underperforms relative to benchmarks (e.g., below 70% of target KPIs for three consecutive periods), recommend targeted performance improvement plans or additional training. • If an employee excels in specific areas (e.g., consistently exceeds targets by 20%), suggest opportunities for career advancement or additional responsibilities to leverage their strengths. • If employee engagement surveys indicate low satisfaction in specific areas, cross-reference with performance data and recommend interventions, such as team-building activities or one-on-one meetings.
Generated Knowledge	The system continuously monitors employee performance, generating personalized feedback and recommendations. It identifies areas where employees may need additional support or training and suggests opportunities for growth based on their strengths. It also alerts management to potential issues based on engagement surveys, helping to maintain high levels of productivity and job satisfaction.

Use Case 10: Automated Data Quality Assurance

Input Knowledge	• Data quality standards and best practices. • Historical data records and common sources of data errors. • Metadata, data schemas, and validation rules.
Goals, Concerns and Heuristics	**Goals:** Ensure that all data entered into the system meets predefined quality standards to prevent downstream errors. **Concern:** Identify and correct data inconsistencies, inaccuracies, and missing information promptly. **Heuristics:** Trigger alerts for data that deviates from established schemas or contains inconsistencies. Flag significant deviations from historical patterns for validation to prevent errors.
Inference Rules	• If newly entered data deviates from established schemas or contains inconsistencies (e.g., mismatched data types, missing fields), trigger an alert for immediate review and correction. • If data records show significant deviations from historical patterns (e.g., sudden spikes or drops), flag them for validation to prevent potential errors. • If metadata indicates incomplete or inaccurate documentation of data sources, recommend a review and update of the metadata. • Regularly audit data quality across systems to identify recurring issues and suggest improvements in data entry processes or validation rules.
Generated Knowledge	The system continuously monitors the quality of incoming and existing data, providing real-time alerts for any deviations from quality standards. It suggests corrective actions, such as data review or schema adjustments, to ensure data accuracy, consistency, and completeness, thereby preventing errors that could impact decision-making processes.

Use Case 11: Supply Chain Optimization

Input Knowledge	• Historical supply chain data (e.g., inventory levels, lead times, logistics costs). • Supplier performance metrics and contract terms. • Market demand forecasts and trends.
Goals, Concerns and Heuristics	**Goals:** Optimize inventory levels and supplier relationships to reduce costs and improve efficiency. **Concern:** Minimize supply chain disruptions while ensuring timely delivery and cost-effectiveness. **Heuristics:** Recommend reducing order quantities or adjusting reorder points if inventory levels consistently exceed demand forecasts. Suggest evaluating alternative suppliers if a supplier consistently fails to meet delivery deadlines.
Inference Rules	• If inventory levels consistently exceed demand forecasts by more than 15%, recommend reducing order quantities or adjusting reorder points to avoid excess stock. • If a supplier consistently fails to meet delivery deadlines or quality standards, suggest evaluating alternative suppliers or renegotiating contract terms. • If logistics costs increase by more than 10% without a corresponding increase in demand, flag the issue for review and recommend potential cost-saving measures, such as route optimization or bulk shipping. • Regularly analyze supply chain data to identify trends and suggest adjustments to procurement strategies or inventory management practices based on changing market conditions.
Generated Knowledge	The system continuously analyzes supply chain data to provide real-time recommendations for optimizing inventory levels, improving supplier performance, and reducing logistics costs. It alerts management to potential issues, such as excess stock or rising costs, and suggests actionable strategies to enhance overall supply chain efficiency and responsiveness.

Use Case 12: IP Legal Research

Input Knowledge	Case A: Precedent on intellectual property (IP) rights in software development. Statute B: Current IP laws relating to software in a specific jurisdiction.
Goals, Concerns and Heuristics	Goals: Prioritize recent precedents in the same jurisdiction for legal arguments. Heuristic: Suggest using recent cases as primary precedents if they strongly support a particular interpretation. Prioritize the latest version of a statute if it has been amended within the last five years. Concern: Ensure the legal argument considers both statute and precedent comprehensively.
Inference Rules	• If a recent case in the same jurisdiction strongly supports a particular legal interpretation, suggest using that case as a primary precedent. • If a statute has been amended within the last five years, prioritize the latest version in legal arguments. • If a majority of recent cases in the jurisdiction favor a particular outcome, predict a similar outcome for current cases. • If legal principles in two or more jurisdictions align, propose harmonization of arguments across these jurisdictions. • Other inference rules specific to the legal sciences
Generated Knowledge	The system recommended pursuing IP enforcement with an 80% probability of success, bolstered by recent precedents and statutory alignment within the jurisdiction.
Shards Examples	Shard 1: Patent laws (existing patents, application processes) Shard 2: Copyright cases (historic rulings, current disputes) Shard 3: Trademark Regulations (brand protections, violations) Shard 4: Licensing agreements (terms, conditions) Shard 5: Litigation histories (court cases, legal precedents)

Use Case 13: Adaptive Cybersecurity Threat Detection

Input Knowledge	• Historical cybersecurity data, including known threat patterns and previous breaches. • Real-time network activity logs and security event data. • Industry-standard cybersecurity protocols and practices.
Goals, Concerns and Heuristics	Goals: Continuously adapt to emerging threats by monitoring and analyzing network activities in real-time. Concern: Quickly identify and respond to potential security breaches while minimizing false positives. Heuristics: Trigger alerts if network activity significantly deviates from established baselines. Recommend implementing countermeasures if new threat patterns emerge that match known signatures of attacks.
Inference Rules	• If network activity deviates significantly from established baselines (e.g., unusual data transfers or access attempts), trigger an alert for immediate investigation. • If a new threat pattern emerges that matches known signatures of malware or attacks, recommend implementing countermeasures and updating security protocols. • If multiple failed login attempts are detected within a short time frame, flag the account for potential brute force attacks and suggest immediate action. • Regularly review and update threat detection algorithms based on the latest threat intelligence to ensure the system remains effective against evolving cyber threats.
Generated Knowledge	The system continuously monitors network activities, providing real-time alerts and recommendations for potential security threats. It adapts to new threats by updating detection algorithms and suggesting proactive measures, such as patching vulnerabilities or adjusting security protocols, to mitigate risks and protect the organization from cyberattacks.
Shards Examples	Shard 1: Network traffic analysis (data flows, unusual activities) Shard 2: Malware detection (virus signatures, attack vectors) Shard 3: User behavior analytics (login patterns, access anomalies) Shard 4: Security policy enforcement (compliance checks, audits) Shard 5: Incident response (breach protocols, recovery steps)

Use Case 14: Predictive Maintenance in Manufacturing

Input Knowledge	• Historical maintenance records and equipment failure data. • Real-time sensor data from manufacturing equipment (e.g., temperature, vibration, performance metrics). • Manufacturer guidelines and maintenance schedules.
Goals, Concerns and Heuristics	Goals: Maximize equipment uptime and efficiency by predicting and preventing potential failures. Concern: Reduce unplanned downtime and maintenance costs by optimizing maintenance schedules. Heuristics: Trigger alerts for potential equipment failure if real-time sensor data indicates deviations from normal operating parameters. Suggest comprehensive inspections if equipment shows a pattern of frequent minor issues.
Inference Rules	• If real-time sensor data indicates deviations from normal operating parameters (e.g., increased vibration, unusual temperature fluctuations), trigger an alert for potential equipment failure and recommend preventive maintenance. • If equipment shows a pattern of frequent minor issues, suggest a comprehensive inspection to prevent a major breakdown. • If historical data indicates that a specific component typically fails after a certain number of operational hours, schedule replacement before failure occurs. • Regularly analyze maintenance and performance data to optimize maintenance schedules, reducing the likelihood of unexpected downtime and extending equipment lifespan.
Generated Knowledge	The system continuously monitors equipment performance, generating real-time alerts for potential issues based on sensor data and historical patterns. It provides actionable recommendations for preventive maintenance, helping to reduce unplanned downtime and extend the lifespan of manufacturing equipment, ultimately improving overall operational efficiency.
Shards Examples	Shard 1: Equipment health monitoring (sensor data, maintenance history) Shard 2: Failure prediction models (predictive analytics, risk factors) Shard 3: Maintenance Scheduling (optimal timings, resource allocation) Shard 4: Spare parts inventory (availability, usage rates) Shard 5: Lifecycle management (end-of-life forecasting, replacement strategies)

Use Case 15: Financial Market Analysis

Input Knowledge	Data A: Historical stock prices for tech companies during economic downturns. Data B: GDP growth and its impact on stock market sectors.
Goals, Concerns and Heuristics	Goals: Focus on sectors resilient during past economic downturns. Concern: Minimize risk by considering macroeconomic indicators. Heuristics: Forecast growth during upcoming recessions if a sector showed growth in revenue during past downturns. Recommend low-risk investments if a company consistently outperforms the market during low GDP growth periods.
Inference Rules	• If a sector shows 10% growth in revenue during past economic downturns, forecast similar growth during upcoming recessions. • If a stock's price increases by more than 5% after a major political event, consider it sensitive to political changes. • If a company consistently outperforms the market during low GDP growth periods, recommend it as a low-risk investment. • If tech stocks rebound within 12 months after a 5% GDP drop, predict a similar trend in future downturns. • Other inference rules specific to finance
Generated Knowledge	The system recommended investing in tech stocks with a 60% probability of outperforming other sectors during the upcoming recession, supported by historical resilience data and current economic indicators.
Shards Examples	Shard 1: Economic indicators (GDP growth, unemployment rates) Shard 2: Market sentiment (news analysis, investor sentiment) Shard 3: Equity markets (stock performance, company financials) Shard 4: Fixed income (bond yields, credit ratings) Shard 5: Commodity markets (price fluctuations, supply factors)

Use Case 16: Semantic Audit of a Company or Project

Input Knowledge	• Data from ERP systems, including financial records, supply chain data, and project management metrics. • Industry standards, compliance requirements, and risk management frameworks. • Audit standards • Historical audit reports and identified risks.
Goals, Concerns and Heuristics	**Goals:** Ensure continuous compliance and risk mitigation by analyzing company or project data in real-time. **Heuristic:** Trigger alerts for potential financial risk if transactions or expenditures deviate significantly from budgeted amounts. Suggest deeper audits if supply chain activities show irregular patterns, such as delayed deliveries or unusual order volumes. **Concern:** Identify and address potential risks before they escalate, ensuring that all activities align with regulatory and internal standards.
Inference Rules	• If financial transactions or project expenditures deviate significantly from budgeted amounts, trigger an alert for potential financial risk and recommend immediate review. • If supply chain activities show irregular patterns, such as delayed deliveries or unusual order volumes, suggest a deeper audit to assess potential risks. • If ERP data indicates non-compliance with regulatory or contractual obligations, flag the issue for immediate action and suggest corrective measures. • Regularly analyze and cross-reference data from various ERP modules to identify emerging risks, inefficiencies, or compliance issues in real-time and recommend appropriate actions to mitigate these risks.
Generated Knowledge	The system continuously analyzes data from ERP systems to conduct real-time audits of company or project activities. It identifies potential risks and non-compliance issues, providing actionable recommendations to address these concerns before they escalate. This approach helps maintain operational integrity, ensure regulatory compliance, and mitigate financial and operational risks.
Shards Examples	Shard 1: Financial transactions (expense reports, budget variances) Shard 2: Project timelines (milestones, delays) Shard 3: Compliance checks (regulatory adherence, internal audits) Shard 4: Risk assessments (potential vulnerabilities, impact analyses) Shard 5: Operational efficiency (process optimization, resource utilization)

Use Case 17: Ensuring Consistency and Quality in Film and TV Show Scenarios

Input Knowledge	• Script drafts and finalized episodes from previous seasons and episodes. • Character profiles, world-building details, and plot summaries. • Editorial guidelines, thematic elements, and narrative continuity rules
Goals, Concerns and Heuristics	Goal: Ensure narrative consistency and alignment with the established universe across all episodes and seasons. Concern: Identify and resolve contradictions or deviations from the established storyline, character arcs, and editorial values. Heuristics: Trigger alerts for revision if new scripts introduce events or details that contradict established plot points or character development. Recommend revisions if a character's behavior in new scripts deviates from their established personality or arc.
Inference Rules	• If new scripts introduce events, character actions, or details that contradict established plot points or character development, trigger an alert for revision and suggest adjustments to maintain continuity. • If a character's behavior or decisions in the new script deviate from their established personality or arc, recommend revisions to align with previous portrayals. • If new narrative elements or plot twists are proposed, cross-check against the established universe for potential conflicts or inconsistencies and suggest alternative approaches if necessary. • Regularly review scripts for adherence to editorial guidelines and thematic consistency, flagging any deviations for further editorial review.
Generated Knowledge	The system continuously monitors the development of scripts, providing real-time feedback to ensure that all new content aligns with the established universe, character arcs, and editorial guidelines. It identifies potential contradictions or inconsistencies and offers suggestions for maintaining narrative continuity, helping to create a cohesive and immersive experience for viewers across all episodes and seasons.
Shards Examples	Shard 1: Script versions (drafts, revisions) Shard 2: Character development (backgrounds, arcs) Shard 3: Plot consistency (timeline, continuity) Shard 4: Thematic elements (themes, messages) Shard 5: Production guidelines (cinematography, set design)

Use Case 18: Automated Quality Control in Manufacturing

Input Knowledge	• Sensor data from manufacturing equipment (e.g., temperature, pressure, vibration) • Historical data on product quality metrics. • Manufacturing process parameters and tolerances.
Goals, Concerns and Heuristics	Goal: Ensure consistent product quality by identifying defects early in the manufacturing process. Concern: Minimize false positives in defect detection to avoid unnecessary process interruptions. Heuristics: Trigger alerts for potential quality issues if sensor data deviates from established process parameters. Recommend process adjustments if product quality metrics fall outside acceptable tolerances.
Inference Rules	• If sensor data deviates from established process parameters, trigger an alert for potential quality issues. • If product quality metrics fall outside acceptable tolerances, recommend process adjustments. • Regularly analyze historical data to refine defect detection models and reduce false positives.
Generated Knowledge	The system continuously monitors manufacturing processes, generating real-time alerts for potential quality issues based on sensor data deviations and historical quality metrics. It suggests corrective actions to maintain consistent product quality.
Shards Examples	Shard 1: Process parameters (machine settings, operational guidelines) Shard 2: Quality metrics (inspection results, defect rates) Shard 3: Product specifications (design requirements, tolerances) Shard 4: Supplier quality data (material consistency, supplier audits) Shard 5: Customer feedback (product reviews, returns)

Use Case 19: AI-Powered Employee Onboarding and Training

Input Knowledge	• Employee profiles, including skills, previous experience, and learning preferences • Company training materials and onboarding processes. • Historical data on employee performance and feedback.
Goals, Concerns and Heuristics	**Goal:** Streamline the onboarding process and ensure that new employees quickly reach full productivity. **Concern:** Ensure that training is tailored to individual learning styles and avoids overwhelming new hires. **Heuristics:** Prioritize relevant training modules if a new employee's profile matches the skills and experience required for a specific role. Recommend training methods that have shown effectiveness for employees with similar profiles based on historical data.
Inference Rules	• If a new employee's profile matches the skills and experience required for a specific role, prioritize relevant training modules. • If historical data shows that certain training methods are more effective for employees with similar profiles, recommend those methods. • Regularly review and adjust onboarding processes based on feedback and performance data to improve outcomes.
Generated Knowledge	The system continuously monitors the onboarding process, generating personalized training paths for new employees based on their profiles and historical data. It provides recommendations for improving training effectiveness and adjusting onboarding processes.
Shards Examples	Shard 1: Role-specific training modules (skills, procedures) Shard 2: Performance benchmarks (targets, evaluation criteria) Shard 3: Learning management systems (course content, user progress) Shard 4: Employee feedback (surveys, improvement suggestions) Shard 5: Onboarding processes (checklists, timelines)

Use Case 20: Smart Energy Management in Urban Areas

Input Knowledge	• Real-time data from energy consumption sensors across the urban area. • Historical energy usage patterns and weather data. • Data on energy costs and availability from the grid.
Goals, Concerns and Heuristics	**Goal:** Optimize energy consumption and reduce costs while ensuring a stable supply. **Concern:** Prevent energy shortages during peak demand and minimize environmental impact. **Heuristics:** Recommend adjusting energy distribution if real-time data indicates a spike in energy consumption. Suggest preemptive load balancing strategies if historical data shows a pattern of increased energy usage during certain times.
Inference Rules	• If real-time data indicates a spike in energy consumption, recommend adjusting energy distribution or activating alternative energy sources. • If historical data shows a pattern of increased energy usage during certain times, suggest preemptive load balancing or demand response strategies. • Regularly analyze energy costs and availability to optimize the use of renewable energy sources and reduce reliance on non-renewable energy.
Generated Knowledge	The system continuously monitors energy usage across the urban area, providing real-time recommendations for optimizing energy distribution and usage. It suggests actions to balance demand, reduce costs, and enhance the use of renewable energy sources.
Shards Examples	Shard 1: Energy consumption data (usage patterns, peak times) Shard 2: Renewable energy sources (solar and wind data) Shard 3: Grid stability metrics (load distribution, outage data) Shard 4: Demand-response strategies (incentives, load shifting) Shard 5: Environmental impact assessments (emissions, reductions)

Use Case 21: Real-Time Environmental Monitoring and Response

Input Knowledge	• Sensor data from air and water quality monitoring stations. • Weather forecast data and historical environmental data. • Regulatory standards for environmental quality • Emergency response protocols and resources.
Goals, Concerns and Heuristics	Goal: Ensure rapid detection and response to environmental hazards to protect public health and safety. Concern: Minimize false alarms while ensuring timely intervention in cases of actual hazards. Heuristics: Trigger alerts and recommend immediate action if sensor data exceeds regulatory limits for pollutants. Suggest preemptive measures if weather forecasts predict conditions likely to exacerbate environmental risks.
Inference Rules	• If sensor data exceeds regulatory limits for pollutants, trigger an alert and recommend immediate action based on severity. • If weather forecasts predict conditions likely to exacerbate environmental risks (e.g., heavy rainfall leading to water contamination), suggest preemptive measures. • Regularly analyze historical data to improve detection accuracy and reduce false positives.
Generated Knowledge	The system continuously monitors environmental conditions in real-time, generating alerts and action recommendations when potential hazards are detected. It also provides insights for improving future monitoring and response strategies based on historical trends.
Shards Examples	Shard 1: Air quality indices (pollutant levels, sources) Shard 2: Water quality monitoring (contaminants, treatment) Shard 3: Weather data integration (forecasts, extreme events) Shard 4: Emergency response protocols (evacuation routes, resource allocation) Shard 5: Regulatory compliance (environmental standards, reporting)

Use Case 22: Automated Detection of Insider Threats

Input Knowledge	• Historical data on employee behavior and access logs. • Patterns of known insider threats and security breaches. • Company policies and access control rules. • Real-time monitoring data from IT systems and networks.
Goals, Concerns and Heuristics	**Goal:** Detect and mitigate insider threats before they result in security breaches or data loss. **Concern:** Avoid false positives that could lead to unnecessary investigations or employee distrust. **Heuristics:** Trigger alerts if an employee's access patterns deviate significantly from their normal behavior. Flag incidents for review if a user attempts to access restricted areas without proper authorization.
Inference Rules	• If an employee's access patterns deviate significantly from their normal behavior, trigger an alert for potential insider threat activity. • If a user attempts to access restricted areas of the network without proper authorization, flag the incident for review. • Regularly run threat detection algorithms based on new threat patterns and historical incidents to enhance detection accuracy.
Generated Knowledge	The system continuously monitors employee activity and network access, generating real-time alerts for potential insider threats. It provides actionable insights to prevent security breaches and improve overall threat detection mechanisms.
Shards Examples	Shard 1: Access logs (entry, exit times) Shard 2: Communication monitoring (email, chat logs) Shard 3: Behavioral analytics (anomaly detection, pattern recognition) Shard 4: Security policy compliance (violations, audits) Shard 5: Data exfiltration attempts (unauthorized data transfers)

Use Case 23: Proactive Customer Support in Telecom

Input Knowledge	• Historical customer interaction data, including support tickets and call logs. • Real-time data on network performance and service outages. • Customer satisfaction surveys and feedback. • Customer profiles, including service usage patterns and account history.
Goals, Concerns and Heuristics	**Goal:** Enhance customer satisfaction by proactively addressing issues before they are reported. **Concern:** Minimize unnecessary contact with customers to avoid frustration and maintain a positive experience. **Heuristics:** Automatically notify affected customers and provide troubleshooting steps if real-time network data indicates a potential service disruption. Prioritize proactive outreach to customers with a history of frequent support requests.
Inference Rules	• If real-time network data indicates a potential service disruption, automatically notify affected customers and provide troubleshooting steps. • If a customer has a history of frequent support requests, prioritize proactive outreach to prevent future issues. • Regularly analyze customer feedback and interaction history to refine support strategies and improve customer satisfaction.
Generated Knowledge	The system continuously monitors network performance and customer interactions, generating real-time alerts and proactive support actions. It helps prevent service disruptions from escalating and improves overall customer satisfaction through timely interventions.
Shards Examples	Shard 1: Network performance data (uptime, disruptions) Shard 2: Customer interaction history (support tickets, resolutions) Shard 3: Service upgrade opportunities (promotions, upgrades) Shard 4: Customer satisfaction metrics (survey results, loyalty scores) Shard 5: Technical support guides (troubleshooting, FAQs)

Use Case 24: Intelligent Financial Portfolio Management

Input Knowledge	• Historical financial data, including stock prices, bond yields, and economic indicators. • Real-time market data and news updates. • Client investment profiles, including risk tolerance and financial goals. • Economic forecasts and macroeconomic trends.
Goals, Concerns and Heuristics	**Goal:** Optimize portfolio performance by balancing risk and return based on client preferences. **Concern:** Minimize exposure to volatile assets while maximizing growth opportunities. **Heuristics:** Recommend adjustments to reduce risk exposure if market data indicates increased volatility in a client's portfolio. Suggest rebalancing or diversifying investments if a client's portfolio underperforms relative to market benchmarks.
Inference Rules	• If market data indicates increased volatility in a client's portfolio, recommend adjustments to reduce risk exposure. • If a client's portfolio underperforms relative to market benchmarks, suggest rebalancing or diversifying investments. • Regularly review and adjust portfolios based on changing market conditions and updated client information to ensure alignment with financial goals.
Generated Knowledge	The system continuously monitors financial markets and client portfolios, generating real-time recommendations for optimizing investment strategies. It helps achieve better risk management and portfolio performance tailored to individual client needs.
Shards Examples	Shard 1: Client profiles (risk tolerance, investment goals) Shard 2: Market analysis (stock trends, economic indicators) Shard 3: Investment strategies (asset allocation, diversification) Shard 4: Regulatory requirements (compliance checks, reporting) Shard 5: Historical performance data (benchmarks, returns)

Use Case 25: Predictive Talent Retention Strategies

Input Knowledge	• Historical employee performance data, including reviews, promotions, and turnover rates. • Employee engagement surveys and feedback. • Industry benchmarks for employee retention and satisfaction. • Data on workplace conditions, such as workload, management practices, and team dynamics.
Goals, Concerns and Heuristics	**Goal:** Improve employee retention by identifying and addressing factors that contribute to turnover. **Concern:** Ensure interventions are targeted and effective without disrupting overall team dynamics. **Heuristics:** Recommend proactive measures like one-on-one meetings if an employee shows signs of disengagement, such as decreased performance or negative feedback. Suggest reviewing management practices if turnover rates in a specific department exceed industry benchmarks.
Inference Rules	• If an employee shows signs of disengagement, such as decreased performance or negative feedback, recommend proactive measures like one-on-one meetings or adjustments to their role. • If turnover rates in a specific department exceed industry benchmarks, suggest reviewing management practices and workplace conditions in that area. • Regularly analyze engagement and performance data to identify at-risk employees and tailor retention strategies to individual needs.
Generated Knowledge	The system continuously monitors employee performance and engagement, generating real-time insights and recommendations for retaining key talent. It helps reduce turnover by providing targeted interventions and improving overall employee satisfaction.
Shards Examples	Shard 1: Employee turnover data (reasons, tenure) Shard 2: Job satisfaction surveys (engagement, satisfaction levels) Shard 3: Career development programs (training opportunities, promotions) Shard 4: Work-life balance initiatives (flexible hours, remote work) Shard 5: Compensation and benefits (market comparisons, equity)

Use Case 26: Automated News Aggregation and Fact-Checking

Input Knowledge	• Real-time news articles from various sources. • Historical data on news source credibility and past misinformation incidents. • Natural language processing models for sentiment analysis and fact-checking. • User preferences and interests for personalized news aggregation.
Goals, Concerns and Heuristics	**Goal:** Provide users with accurate, relevant, and personalized news while filtering out misinformation. **Concern:** Avoid the spread of false information and ensure the credibility of aggregated news sources. **Heuristics:** Flag articles from sources with a history of misinformation for further review or exclusion. Prioritize the aggregation of information reported by multiple credible sources.
Inference Rules	• If a news article comes from a source with a history of misinformation, flag it for further review or exclude it from aggregation. • If multiple credible sources report similar information, prioritize those articles in the aggregation. • Regularly update the credibility scores of news sources based on ongoing fact-checking results and user feedback.
Generated Knowledge	The system continuously aggregates and checks the credibility of news articles, providing users with accurate and personalized news feeds. It helps prevent the spread of misinformation by filtering out unreliable sources and prioritizing credible information.
Shards Example	Shard 1: Source credibility ratings (reliability scores, history) Shard 2: Content verification (fact-checking, cross-referencing) Shard 3: User preferences (customized feeds, interests) Shard 4: Sentiment analysis (public opinion, trends) Shard 5: Real-time updates (breaking news, alerts)

Use Case 27: Personalized Dietary and Fitness Recommendations

Input Knowledge	• User health data, including medical history, allergies, and dietary preferences. • Fitness activity data from wearable devices and apps. • Nutritional information on foods and dietary guidelines. • User goals, such as weight loss, muscle gain, or improved cardiovascular health.
Goals, Concerns and Heuristics	**Goal:** Provide personalized dietary and fitness recommendations that align with individual health goals and preferences. **Concern:** Ensure that recommendations are safe, effective, and sustainable for long-term health. **Heuristics:** Suggest adjustments to diet or fitness routines if a user's activity levels decrease. Immediately update recommendations if a user reports new dietary preferences or restrictions.
Inference Rules	• If a user's activity levels decrease, suggest adjustments to their diet or a change in fitness routines to maintain progress towards their goals. • If a user reports new dietary preferences or restrictions, update recommendations to reflect these changes immediately. • Regularly review and adjust recommendations based on progress data to ensure alignment with the user's evolving health goals.
Generated Knowledge	The system continuously analyzes user data to generate personalized dietary and fitness recommendations, adapting to changes in health status, preferences, and goals. It supports users in achieving their health objectives by providing tailored advice that evolves with their needs.
Shards Examples	Shard 1: Nutritional data (caloric content, macronutrients) Shard 2: Exercise regimens (workout types, fitness levels) Shard 3: User health profiles (medical history, allergies) Shard 4: Dietary preferences (vegetarian, gluten-free options) Shard 5: Fitness tracking data (activity levels, wearable device data)

Use Case 28: Smart Inventory Management in Retail

Input Knowledge	• Historical sales data, including peak seasons and customer demand trends. • Inventory levels and turnover rates. • Supplier lead times and reliability. • Market trends and competitor pricing information.
Goals, Concerns and Heuristics	**Goal:** Optimize inventory levels to meet customer demand while minimizing holding costs and stockouts. **Concern:** Avoid overstocking or understocking, which can lead to lost sales or increased costs. **Heuristics:** Recommend increasing reorder quantities if sales data indicates a rising trend in product demand. Suggest discounts or promotions to clear excess stock for slow-moving products.
Inference Rules	• If sales data indicates a rising trend in demand for a product, suggest increasing the reorder quantity to prevent stockouts. • If inventory levels of a slow-moving product are high, recommend a discount or promotion to clear excess stock. • Regularly review supplier performance data to adjust reorder points based on lead-time reliability.
Generated Knowledge	The system continuously monitors inventory and sales data, providing real-time recommendations for replenishment, promotions, and inventory optimization. It helps maintain optimal stock levels, reduce costs, and improve customer satisfaction.
Shards Examples	Shard 1: Product sales data (historical sales, seasonal trends) Shard 2: Inventory levels (stock on hand, turnover rates) Shard 3: Supplier information (lead times, reliability) Shard 4: Pricing Strategies (Discounts, Promotions) Shard 5: Customer demand forecasting (purchase predictions, market analysis)

Use Case 29: AI-Enhanced Virtual Assistants for Remote Work

Input Knowledge	• Employee schedules, task lists, and deadlines. • Communication data from emails, chats, and project management tools. • Employee productivity metrics and work patterns. • Company policies and best practices for remote work.
Goals, Concerns and Heuristics	**Goal:** Improve remote work efficiency and employee productivity by providing personalized assistance and task management. **Concern:** Avoid overloading employees with tasks and ensure a healthy work-life balance. **Heuristics:** Suggest prioritizing tasks or delegating them if an employee's task list becomes overloaded. Recommend scheduling reminders or follow-ups if communication data indicates delays or missed deadlines.
Inference Rules	• If an employee's task list becomes overloaded, suggest prioritizing tasks or delegating some to team members. • If communication data indicates a delay in responses or missed deadlines, recommend scheduling reminders or follow-ups. • Regularly analyze productivity metrics to offer personalized tips for improving focus and time management.
Generated Knowledge	The system continuously monitors employee tasks and communication, providing real-time assistance and recommendations to optimize workflow and productivity. It helps manage workloads effectively and supports remote workers in maintaining a balanced and efficient work environment.
Shards Examples	Shard 1: Employee scheduling (meeting times, deadlines) Shard 2: Project management (task assignments, progress tracking) Shard 3: Communication analytics (email, messaging patterns) Shard 4: Productivity Tools (software utilization, efficiency tools) Shard 5: Remote work resources (online training, support documents)

Use Case 30: Real-Time Sentiment Analysis for Brand Management

Input Knowledge	• Social media posts, customer reviews, and feedback related to the brand. • Historical sentiment data and brand reputation metrics. • Competitor sentiment and industry trends. • Marketing campaigns and brand messaging strategies.
Goals, Concerns and Heuristics	**Goal:** Monitor and improve brand reputation by responding quickly to changes in public sentiment. **Concern:** Avoid potential PR crises and ensure consistent, positive brand messaging. **Heuristics:** Immediately alert the PR team if sentiment analysis detects a spike in negative sentiment. Suggest amplifying successful messaging across other channels if positive sentiment increases following a marketing campaign.
Inference Rules	• If sentiment analysis detects a spike in negative sentiment, immediately alert the PR team to investigate and address the issue. • If positive sentiment increases following a marketing campaign, suggest amplifying successful messaging across other channels. • Regularly compare sentiment trends with competitor data to identify opportunities for brand positioning and differentiation.
Generated Knowledge	The system continuously analyzes real-time sentiment data, providing actionable insights and alerts to manage brand reputation effectively. It helps maintain a positive brand image by enabling prompt responses to public opinion and optimizing brand messaging strategies.
Shards Examples	Shard 1: Social media interactions (likes, comments) Shard 2: Customer feedback (surveys, review platforms) Shard 3: Media coverage (news articles, press releases) Shard 4: Brand campaigns (advertisements, marketing strategies) Shard 5: Competitive analysis (brand positioning, market share)

Use Case 31: AI-Powered Recruitment and Candidate Matching

Input Knowledge	• Job descriptions, required skills, and qualifications for open positions. • Candidate resumes, cover letters, and LinkedIn profiles. • Historical hiring data, including successful candidate profiles and interview feedback. • Industry benchmarks for role-specific competencies and experience levels.
Goals, Concerns and Heuristics	**Goal:** Streamline the recruitment process by accurately matching candidates to job openings based on their qualifications and potential fit. **Concern:** Avoid biases in the recruitment process and ensure a diverse pool of candidates is considered. **Heuristic:** Prioritize candidates whose skills and experience closely align with job requirements for further evaluation. Suggest candidates with high potential for growth or strong cultural fit, even if some skills are slightly below the required level.
Inference Rules	• If a candidate's resume closely matches the key qualifications and skills required for a role, prioritize them for further evaluation and interview. • If historical hiring data shows that certain traits or experiences correlate with long-term success in a role, suggest those as additional criteria for candidate selection. • If a candidate's profile indicates high potential for growth or a strong cultural fit, recommend them even if some skills are slightly below the required level. • Regularly review hiring outcomes to refine and adjust matching algorithms, ensuring continuous improvement in candidate selection and diversity.
Generated Knowledge	Reports with the most promising candidates are presented from various perspectives offered by the various inference rules.
Shards Examples	Shard 1: Job descriptions (requirements, responsibilities) Shard 2: Applicant profiles (resumes, skill sets) Shard 3: Interview feedback (evaluations, recommendations) Shard 4: Hiring outcomes (successful placements, performance) Shard 5: Recruitment trends (industry standards, role evolutions)

Use Case 32: Automated Social Media Content Moderation

Input Knowledge	• Community guidelines and content moderation policies. • Historical data on flagged content, including reasons for moderation (e.g., hate speech, spam, misinformation). • User engagement data and sentiment analysis. • External data sources, such as known lists of harmful keywords or phrases.
Goals, Concerns and Heuristics	**Goal:** Maintain a positive and safe online community by automatically identifying and managing harmful or inappropriate content. **Concern:** Balance content moderation with free expression to avoid over-censorship and user alienation. **Heuristic**: Prioritize candidates whose skills and experience closely align with job requirements for further evaluation. Suggest candidates with high potential for growth or strong cultural fit, even if some skills are slightly below the required level.
Inference Rules	• If content contains keywords or phrases flagged as harmful or violates specific community guidelines, automatically remove or flag the content for review. • If user behavior suggests repeated posting of harmful content, suggest actions such as temporary bans or warnings. • If content receives a high volume of negative sentiment or reports from the community, prioritize it for immediate review and potential moderation. • Regularly analyze content trends and user feedback to update moderation rules, ensuring they remain relevant and effective in protecting the community.
Generated Knowledge	The system continuously monitors social media content, automatically flagging or removing posts that violate community guidelines. It provides real-time alerts for content that may require human review and offers insights into evolving content trends, helping maintain a safe and positive online environment while respecting user expression.
Shards Examples	Shard 1: Content flags (inappropriate language, copyright issues) Shard 2: User behavior (posting frequency, interactions) Shard 3: Moderation policies (platform rules, community guidelines) Shard 4: Compliance reports (violations, enforcement actions) Shard 5: Trend analysis (emerging issues, policy updates)

Use Case 33: Dynamic Pricing Strategies in Retail

Input Knowledge	• Historical sales data, including pricing, demand elasticity, and seasonal trends. • Competitor pricing information and market conditions. • Customer segmentation data and purchasing behavior analytics. • Inventory levels and supply chain data.
Goals, Concerns and Heuristics	**Goal:** Maximize revenue and profitability by dynamically adjusting prices based on real-time market and consumer data. **Concern:** Avoid price fluctuations that could negatively impact customer loyalty or lead to stockouts. **Heuristic:** Recommend price adjustments to remain competitive if competitor prices drop significantly, considering inventory levels. Suggest personalized pricing strategies or promotions for customer segments with a higher willingness to pay.
Inference Rules	• If competitor prices drop significantly, suggest a price adjustment to remain competitive, unless inventory is low, in which case maintain prices to avoid stockouts. • If sales data indicates high demand elasticity for a product, recommend more frequent price adjustments to capitalize on shifts in consumer demand. • If certain customer segments show a higher willingness to pay, suggest personalized pricing strategies or targeted promotions for those segments. • Regularly analyze sales trends and inventory data to optimize price points, ensuring high-demand products are priced to maximize revenue while low-demand items are discounted to clear stock.
Generated Knowledge	The system continuously monitors market conditions, competitor pricing, and customer behavior to generate real-time pricing recommendations. It helps retailers optimize their pricing strategies to maximize revenue, improve inventory turnover, and maintain customer satisfaction by adjusting prices dynamically based on comprehensive data analysis.
Shards Examples	Shard 1: Market data (competitor pricing, supply conditions) Shard 2: Customer purchasing behavior (buying patterns, price sensitivity) Shard 3: Sales performance (product margins, sales velocity) Shard 4: Inventory status (overstock, understock situations) Shard 5: Seasonal trends (holiday effects, weather impacts)

Use Case 34: AI-Powered Drug Discovery

Input Knowledge	• Historical data on drug compounds, including chemical properties and biological effects. • Genomic data and disease pathways. • Clinical trial results and outcomes. • Existing drug databases and pharmaceutical research papers.
Goals, Concerns and Heuristics	**Goal:** Accelerate the discovery of new drug compounds with high efficacy and low side effects. **Concern:** Reduce the time and cost associated with drug discovery and minimize the risk of failure in clinical trials. **Heuristic:** Recommend further exploration of compounds with structural similarities to known effective drugs. Flag compounds for additional safety screening if clinical trial data indicates adverse effects related to similar molecular structures.
Inference Rules	• If a new compound shares key structural features with existing drugs that have proven efficacy against a specific disease, recommend further exploration and testing of that compound. • If genomic data reveals a strong correlation between certain biomarkers and disease progression, suggest targeting those biomarkers in drug development efforts. • If clinical trial data indicates a pattern of adverse effects related to a specific molecular structure, flag similar compounds for additional safety screening. • Regularly review and incorporate the latest research findings to refine drug discovery models and identify new potential targets.
Generated Knowledge	The system continuously analyzes chemical, genomic, and clinical data to identify promising drug candidates. It generates recommendations for new compounds to explore, targets to focus on, and potential safety concerns to address early in the drug development process, ultimately speeding up the discovery of effective and safe medications.
Shards Examples	Shard 1: Chemical libraries (compound structures, properties) Shard 2: Clinical trial data (study results, patient responses) Shard 3: Regulatory compliance (FDA guidelines, approval processes) Shard 4: Research publications (new findings, hypotheses) Shard 5: Biomarker data (genetic markers, disease correlations)

Use Case 35: Smart Waste Management in Cities

Input Knowledge	• Historical waste collection data, including volume and frequency. • City demographics, population density, and seasonal waste patterns. • Sensor data from smart bins and waste collection vehicles. • Environmental regulations and sustainability goals. • Goals, Concerns and Heuristics
Goals, Concerns and Heuristics	**Goal:** Optimize waste collection routes and schedules to reduce costs and environmental impact. **Concern:** Prevent overflow of waste bins and ensure timely collection. **Heuristic:** Suggest increasing collection frequency if sensor data indicates waste bins are consistently reaching capacity before the scheduled collection. Recommend route optimization for waste collection vehicles to reduce fuel consumption on certain routes.
Inference Rules	• If sensor data indicates that waste bins in a specific area are consistently reaching capacity before the scheduled collection, suggest increasing collection frequency in that area. • If waste collection vehicles report high fuel consumption on certain routes, recommend optimizing the route for efficiency by adjusting the sequence of stops. • If data shows a sudden increase in waste volume during certain seasons or events, adjust the waste management plan to allocate more resources during those periods. • Regularly analyze waste patterns to identify areas where recycling efforts can be increased or where waste reduction initiatives could be implemented.
Generated Knowledge	The system continuously analyzes waste collection data and city demographics to generate real-time recommendations for optimizing collection schedules and routes. It provides actionable insights to prevent bin overflows, reduce operational costs, and support sustainability goals by identifying opportunities for improving recycling and waste reduction efforts.
Shards Examples	Shard 1: Collection data (routes, frequencies) Shard 2: Recycling processes (material sorting, recovery rates) Shard 3: Public engagement (awareness campaigns, participation rates) Shard 4: Environmental impact (reduction in landfill use, emissions) Shard 5: Regulatory requirements (municipal codes, waste reduction targets)

Annex 2 (Size Estimations)

These are just intuitive guesses to estimate complexity.
Use Case Name: The title of the use case.
Min Shards: Minimum number of domain shards necessary for utility.
Inference Depth: Estimated depth of inference required per shard.
No Prompts: Number of LLM prompts per shard to initialize and maintain.
Est. Knowledge: Estimated number of discrete knowledge elements within a typical shard.

Use Case Name	Min Shards	Inference Depth	No prompts	Est. Knowledge
Generic Search Assistance	100	5	100	100000
AI Scientific Article Review	5	10	50	10000
Material Science Research	10	5	50	5000
Financial Real-Time Anomaly Detection	5	5	30	1000
Personalized Diagnostics	100	5	100	1000
Legal Compliance Monitoring	10	5	200	1000
Personalized Learning	10	5	100	1000
Code Copilot for Software Projects	5	10	100	1 million
Compliance Documentation Monitoring	5	5	50	1000
Employee Performance Monitoring	5	5	50	100
Adaptive Cybersecurity Threat Detection	5	10	100	1000