ChatGPT

Kimera SWM: Semantic Entropy Blindspots and Enhancements

Identified Blindspots

- **Missing Semantic Entropy in Contradiction Handling:** The current Kimera **contradiction engine** and **vault system** do not fully leverage **semantic entropy** measures to detect or manage conflicting knowledge. This is a blindspot because relying on shallow checks or binary logic can miss subtler contradictions. For example, a model output might be **low entropy at the token level (high confidence)** yet semantically incorrect – a **"confident hallucination"** [1]. Without semantic entropy, the system may not flag such confident but contradictory outputs. In the **Kimera SWM pipeline**, simpler proxies (like naive probability-of-truth or embedding similarity) achieve only ~69–70% accuracy in detecting false/conflicting information. Meanwhile, methods that incorporate semantic meaning (clustering outputs by mutual entailment) reach ~79% accuracy, indicating that semantic entropy could catch contradictions that the current system overlooks.

- **Overuse of Simplistic Entropy Proxies:** The pipeline appears to rely on **naive entropy** or other overly simplified uncertainty metrics. **Naive Shannon entropy** of output tokens conflates wording variability with actual meaning uncertainty [2]. This proxy can be misleading – different words can express the same fact (high token entropy but low semantic entropy), or the model can use consistent wording for contradictory meanings (low token entropy but high semantic uncertainty). The data shows that Kimera's use of naive entropy yields much poorer detection performance (AUROC ~0.69) compared to semantic-aware methods (~0.79). Likewise, a basic embedding regression or "P(True)" heuristic hovers near chance-level effectiveness (≈0.68–0.70). These suggest **underutilization of advanced entropy measures**: e.g. **Discrete Semantic Entropy** (which quantizes meaning space) almost matches full semantic entropy performance (0.785 vs 0.79 AUROC) at fraction of cost, and **Semantic Entropy Probes (SEPs)** – lightweight learned estimators – reach ~0.775 with real-time feasibility. Not adopting such refined entropy proxies is a blindspot that leaves Kimera SWM with coarser, less reliable uncertainty assessments.

- **Fragmented Semantic Pruning and Symbolic Interpretation:** Kimera's **semantic pruning** (entropy-guided network compression) is not tightly integrated with its **symbolic interpretation** layer. This creates a blindspot where pruned neural representations and the symbolic knowledge base may drift out of sync. The data confirms that **entropy-based pruning preserves semantic content far better** than magnitude-based pruning – e.g. at 50% network pruning, a semantic-entropy strategy retains ~90% of concept information, versus only ~65% under naive magnitude pruning [3] [4]. Yet if the symbolic reasoning module is unaware of these semantic "clusters" or compressed concepts, it cannot fully capitalize on them. Conversely, it might continue expecting distinctions that the pruned network has merged. For instance, if pruning clusters two related concepts into one latent representation, the system's ontology should reflect that merge – otherwise the symbolic engine may infer contradictions or lose interpretability. Currently, this **neuro-symbolic gap** is only partially addressed: we see improved interpretability scores with semantic pruning (e.g.

interpretability 87 at 50% prune vs 63 for magnitude-based [3] [4]), but those gains are not explicitly transferred to a human-interpretable form. The blindspot is an **"incomplete handshake"** between the subsymbolic (neural) and symbolic parts of SWM – meaningful latent factors identified during pruning are not labeled or constrained by symbolic knowledge, and symbolic logic isn't guiding what to prune or preserve.

- **Overlooked Entropy & Complexity Signals:** There are latent signals in Kimera's SWM data flow that the current design underutilizes. One example is the **gap between syntactic entropy and semantic entropy**. By principle, **meaningful information (semantic entropy H_s) is always ≤ total entropy H**. In Kimera's data, as the model is pruned, network entropy drops faster than semantic entropy, indicating the network had excess complexity beyond what was needed for semantics. At the final pruning stage, network entropy is 4.1 bits vs ~5.9 bits of semantic entropy remaining, suggesting the model is at the edge of undercapacity for the knowledge (a warning sign that further pruning would start dropping concepts). If the system isn't monitoring this **entropy gap**, it misses a chance to know when to stop pruning or when the model is overly complex. Another overlooked metric is **Information Retention vs Energy Efficiency**. Kimera's energy data show a **thermodynamic efficiency** jump from 0.15 to 0.85 as the network is pruned 85% – a huge gain in ops-per-watt – while maintaining ~89–94% semantic fidelity. This implies an opportunity to tune the architecture for an optimal trade-off (the *"energy-optimal"* point around 80–85% compression in this case). If the current pipeline uses a fixed compression or doesn't adapt to real-time energy/entropy feedback, it's missing these sweet spots. Similarly, **mutual information** between layers or between input and retained features (as used in recent frameworks like MIPP) is not visibly tracked. Neglecting MI means the system might prune away connections that quietly supported interpretability or consistency. Overall, Kimera SWM has **rich informational signals (entropy, MI, energy metrics)** that are treated as offline analysis rather than active control variables – a blindspot in achieving a self-optimizing, self-explaining architecture.

Proposed Enhancements

- **Inject Semantic Entropy into Contradiction & Vault Processes:** Integrate a **semantic entropy estimator** into the contradiction engine to gauge when the system's knowledge is becoming ambiguous or self-conflicting. Practically, this means when Kimera ingests new information or answers a query, calculate the semantic entropy over its multiple interpretations or responses. A high semantic entropy (many divergent meaning clusters) would trigger the **vault system** to engage: instead of immediately committing to one belief, the system can vault the ambiguous knowledge (store it with context and an uncertainty tag) for later resolution. This could be implemented using recent techniques like **Semantic Entropy Probes (SEPs)**, which can run in real-time by extracting entropy from hidden states [5]. By doing so, Kimera would **catch contradictions earlier** – even implicit ones – and isolate them. For example, if two subsystems give semantically different answers to the same query, the entropy spike flags a potential contradiction **before** it propagates. The vault can then hold both versions, and a *contradiction resolution routine* can attempt to reconcile them (e.g. cross-check against an external knowledge base, or ask a clarifying question). This enhancement leverages semantic entropy as a **"conflict sensor"** and uses the vault as a **buffer for high-entropy knowledge**, preventing unstable information from polluting the active working memory.

- **Upgrade Entropy Measures in the Pipeline:** Replace or augment the current uncertainty measures with **richer entropy-based metrics**. In practice, Kimera SWM should move beyond naive Shannon entropy of outputs to **meaning-aware entropy**. One enhancement is adopting the **Farquhar et al. semantic entropy method** (clustering multiple model outputs by mutual entailment) to assess answer uncertainty [6]. If full clustering is too slow for real-time use (it can be 5–10× computational cost [5]), then use faster proxies: e.g. **Discrete Semantic Entropy** (group outputs into a fixed set of meaning categories) or the aforementioned **SEPs**, which approximate semantic entropy from a single forward pass [7]. These provide a near-instant uncertainty measure focused on meaning, not just token dispersion. Additionally, combine **multi-level entropy**: monitor token-level entropy *and* semantic-level entropy. Research shows combining token entropy with semantic coherence catches far more model errors (boosting detection accuracy to ~80%) [8]. Concretely, Kimera could compute a **composite score** for each inference: e.g. a weighted sum of normalized token entropy and semantic entropy. A low token entropy but high semantic entropy would reveal a confident-sounding contradiction (flag it), whereas high token entropy across multiple paraphrased answers but low semantic entropy means the wording varies while meaning stays consistent (not a true uncertainty). By **upgrading the entropy proxy** in this way, Kimera's SWM will be less fooled by superficial variability and more attuned to meaningful uncertainty. This improves reliability in detecting hallucinations, contradictions, or knowledge gaps in real-time.

- **Link Pruning with Symbolic Knowledge:** Establish a tighter **neuro-symbolic integration** during and after pruning. One practical step is to **label semantic clusters** formed during pruning with symbolic identifiers. For example, if Stage 3 of pruning groups neurons into a "semantic cluster" that corresponds to a concept (say, *vehicle* concept merging *car* and *truck* features), then add or update a node in the symbolic memory (ontology) for `Vehicle` and note that `Car` and `Truck` are now treated as sub-concepts unified in representation. This way, the symbolic interpreter is aware of the compression and can adjust its reasoning (preventing it from drawing distinctions the network can no longer support). Another enhancement is to use the **symbolic layer to guide pruning**: leverage domain ontologies or rules to decide which weights or features are critical. For instance, ensure that features linked to safety-critical concepts or known logical constraints are preserved regardless of entropy metrics. In essence, introduce a feedback loop where **symbolic importance weighs into the entropy-based pruning criteria** – a weight serving a diverse set of symbolically distinct concepts might be less prunable even if low magnitude, to avoid losing a whole branch of knowledge. After pruning, the system should perform a **symbolic re-interpretation pass**: translate the pruned model's behavior back into rules or prototypes. If Kimera's SWM prunes 20% of nodes with only 2% loss in accuracy, ask: which facts or rules were essentially unchanged? Which got abstracted? This could be done by probing the pruned network with known symbolic queries and seeing if the answers changed. By **integrating semantic pruning with symbolic interpretation**, Kimera can maintain interpretability – the pruned model remains verifiable against the knowledge base – and achieve a stable learning loop (prune the redundant parts but not the "conceptual skeleton"). This closes the blindspot by treating pruning not just as compression, but as an opportunity for **knowledge distillation into the symbolic memory**.

- **Leverage Latent Metrics for Dynamic Control:** Turn the currently latent signals (entropy, mutual information, etc.) into active **control knobs** in the architecture. One enhancement is to implement an **Entropy Monitor Module** that continuously tracks key metrics in the SWM: e.g. global semantic entropy of the knowledge graph, entropy of the current belief distribution, network entropy of each layer, and the gap between them. This monitor can trigger adaptive behaviors – for example, if

semantic entropy in the working memory spikes (many conflicting hypotheses), the system could automatically slow down decision-making, devote more computational "energy" to resolution (akin to a CPU throttling up), or invoke a consultation from a more powerful reasoning module. Another actionable metric is the **Network–Semantic Entropy Ratio**: Kimera can calculate H_s/H (semantic vs total entropy). If this ratio is low, it means a lot of model capacity is unused (redundant parameters or noise) – cue to perform entropy-based pruning or knowledge consolidation. If the ratio is high (approaching 1), the model is lean relative to the information it encodes – cue to be cautious with further pruning or perhaps consider expanding capacity if new info needs to be learned. Likewise, adopt **Mutual Information Preserving Pruning (MIPP)** techniques that explicitly maximize information retention between layers [9]. Concretely, Kimera's pruning algorithm can be enhanced to evaluate how much each candidate pruned neuron decreases the MI between that layer's activations and the next; instead of pruning purely on weight magnitude, prune the ones that least affect the overall information flow. This ensures no sudden drops in concept representation (addressing the overlooked MI signal). On the **thermodynamic side**, use the energy metrics: for instance, define a target **thermodynamic efficiency** (operations per watt per semantic fidelity) and have the system self-tune via pruning or activation gating to approach that optimum. If the SWM is deployed on a robot with battery constraints, it could dynamically prune or "sleep" parts of the network when full fidelity isn't needed, guided by an energy-entropy budget. Summarily, this enhancement means **instrumenting the SWM with dashboards of entropy/MI and policies that respond to them** – making the system more adaptive. It could also involve analogies like a "Maxwell's Demon" process in the knowledge vault: a background routine that continuously looks to partition low-entropy (reliable) facts from high-entropy (uncertain) ones, thereby reducing overall confusion at the cost of some compute – effectively **spending computational energy to maintain an ordered, contradiction-free knowledge base**. Embracing these metrics in real-time will guide Kimera to be both **stable and efficient**, as it can foresee instability (rising entropy) and react before it manifests as an error.

Architectural Recommendations

- **Contradiction Engine with Entropy-Gated Vault:** Redesign the contradiction handling subsystem as an active module that uses entropy as input. Architecturally, this could be a **Contradiction Manager** that sits between the SWM's inference module and the long-term memory (vault). Every time the SWM produces a new deduction or receives new data, the Contradiction Manager evaluates semantic entropy and contradiction scores. High entropy or direct conflicts trigger a routine: the conflicting knowledge is not immediately integrated into the primary world model. Instead, it is stored in a structured **Contradiction Vault** (a section of memory that holds multiple competing hypotheses along with context and confidence levels). The Contradiction Manager can then either alert a higher-level process (or human operator) or attempt an automated **epistemic resolution**: for example, by using weighted logic or probabilistic reasoning to see if the conflict can be resolved by assigning probability truth values. The key recommendation is to **make the vault system entropy-aware** – it should know *why* an item is in the vault (e.g., "Fact X has high semantic entropy because it contradicts Fact Y"), and it should periodically attempt resolution when new information arrives. This architecture ensures the SWM's active beliefs remain consistent and low-entropy, improving stability, while controversial information is quarantined but not lost. Over time, as more evidence accumulates, the vault can either confidently integrate one hypothesis (lowering entropy) or keep facts in a **partitioned state** if they truly represent enduring ambiguities (with the system then acknowledging uncertainty rather than forcing a single false certainty).

- **Real-Time Semantic Entropy Monitor:** Incorporate a dedicated **Semantic Entropy Monitor** in the SWM architecture that continuously runs in parallel with normal inference. This component would take streams of internal activations or outputs and compute semantic uncertainty on the fly. Modern approaches make this feasible – e.g. a small neural probe network can be trained to output an entropy estimate from the hidden state of the main model [10] . The architecture can therefore have a lightweight thread that, for each new output or each new salient memory update, issues an "entropy report." Low-latency monitoring enables **real-time responses**: if the entropy report indicates a spike in uncertainty, the SWM can immediately adjust its behavior (for instance, an autonomous vehicle's SWM might decide to hedge or ask for operator input if its semantic entropy about an obstacle's identity is high). This monitor could also drive a **confidence threshold** for actions: only act autonomously when semantic entropy is below a certain level, otherwise seek clarification. Embedding this into Kimera's pipeline means the system is **constantly self-assessing its understanding**, rather than only discovering confusion after a failure. In terms of architecture, this might look like a small side-network attached to intermediate layers (much like how "auxiliary heads" are used in some networks) that outputs entropy predictions without perturbing the main computation.

- **Neuro-Symbolic Synchronization Layer:** To ensure pruning and symbolic reasoning stay aligned, introduce a **synchronization layer** between the neural SWM and the symbolic knowledge base. This could be realized as a middleware that translates neural representations (especially after pruning or retraining) into symbolic terms. One implementation is a **Concept Extraction routine** that runs after each training/pruning phase: it examines the neurons or embeddings and clusters them (using, say, concept activation vectors or probing with labeled examples) to identify what concepts or features they now correspond to. It then updates the symbolic knowledge graph accordingly – e.g. merging nodes, adding new abstract nodes, or re-weighting relationships between concepts based on the new model structure. Conversely, the synchronization layer can inject constraints: for instance, if the ontology says two concepts should remain distinct (say *cause* vs *effect* in a causal model), the layer can regularize the neural model training to keep those concepts represented separately (perhaps by a penalty on collapsing those latent dimensions). This **two-way sync** ensures the symbolic memory is a faithful, interpretable mirror of the neural state. Architecturally, the synchronization layer might use techniques from **explainable AI** – for example, **layer-wise relevance or concept attribution** – to map neurons to known symbols. It could also use a simple rule: if a pruning stage eliminates a feature, mark the corresponding symbolic facts as needing review (maybe the fact was redundant or derivable from others). This will lead to a more resilient Kimera SWM: even as the model optimizes and prunes itself, the human-understandable knowledge remains accurate, and the system can explain its decisions by referencing updated symbols. In summary, this recommendation formalizes a **neuro-symbolic interface** so that every major change in the sub-symbolic world model is reflected in the symbolic world model (and vice versa), preventing the drift that currently poses interpretability challenges.

- **Thermodynamic Pruning & Resource Management:** At the architectural level, embed the principles of **thermodynamic pruning** into Kimera's model lifecycle. Rather than pruning once and deploying a static model, design Kimera SWM to be **adaptive in its complexity** based on resource availability and task demands. This could mean the model can **"throttle" itself**: e.g., prunes more aggressively (even at runtime) when running on low-power or when response time is critical, and **re-grows or fine-tunes** connections when idle or when extra accuracy is needed. The thermodynamic analogy here is to treat the network like a system that can release or absorb "energy" – releasing

energy by pruning (saving compute power, lowering entropy by discarding noise) or absorbing by learning/restoring weights when more detail is required. A practical way to achieve this is to maintain a pool of prunable units (filters or neurons) that can be **deactivated or activated on the fly**. Use an **energy-entropy heuristic** to decide this: for example, each filter could have a "temperature" value reflecting how much it contributes to reducing output entropy versus how much energy it consumes (if a filter's utility-to-cost ratio drops, mark it for pruning). This is analogous to a **thermodynamic filter selection** mechanism [11]. The architecture might include a small controller that periodically evaluates such metrics and flips the switches (turn off those filters or reactivate some if needed). Additionally, incorporate **Landauer's Principle awareness**: avoid irrevocably erasing information unless absolutely necessary. For instance, instead of hard-deleting pruned weights, archive their state in the vault or a compressed form so that if it turns out later that a pruned feature was needed, the system can **"undo"** certain prunings (at an energy cost) rather than relearning from scratch. This makes the pruning process more like a reversible compression, aligning with the idea that erasing a bit has a minimum energy cost [12] – so you only erase (forget) when you're sure that energy cost is worth paying. By managing resources in this thermodynamic-inspired way, Kimera can achieve high efficiency **without compromising stability**. In fact, the data suggests a combined **Semantic-Thermodynamic pruning** approach yielded higher semantic fidelity (94%) at much lower energy use, compared to naive methods. Architecturally baking this in means treating **computational resources and knowledge fidelity as dual objectives**, continuously balanced by the system.

- **Epistemic Feedback Loop:** Finally, establish a feedback loop where the outcomes of these improvements inform future behavior. The architecture should allow Kimera to **learn from contradiction resolutions** (e.g. if the vault resolved a conflict by favoring one hypothesis, feed that back into model fine-tuning so it doesn't flip-flop on that fact again). It should also log instances where high entropy was detected and what caused it, gradually building an "epistemic memory" of trouble spots. Over time, patterns might emerge (e.g. certain sensors or input types often cause spikes in entropy – indicating maybe a calibration issue). Feeding this information to developers or an auto-tuner can lead to architectural refinements (such as adding a knowledge source to fill a gap, or improving a perception module that yields ambiguous data). In essence, treat **entropy events (spikes, contradictions, heavy pruning losses)** as feedback for the design. This creates a self-improving SWM: one that not only handles uncertainties and contradictions better in the moment, but also guides engineers (or automated systems) to strengthen the architecture against future occurrences.

Supporting Examples from Kimera SWM Data

- **Semantic Entropy vs Naive Entropy Performance:** In the provided Kimera SWM evaluation, **methods using semantic entropy vastly outperformed naive uncertainty measures** for detecting incorrect or contradictory outputs. For example, a **bidirectional entailment clustering** approach (grouping answers by shared meaning) achieved about **79% AUROC** in identifying hallucinations, whereas a naive entropy or simple probability-based method was around **69%**. This ~10% gap highlights how a semantic-aware contradiction detector catches issues that a naive one misses. It underscores the blindspot of underusing semantic entropy – and suggests that implementing semantic clustering or entropy probes in Kimera could immediately improve consistency checks.

- **Entropy Proxies and Real-Time Feasibility:** The data compares various entropy estimation methods. Notably, the **Semantic Entropy Probes (SEP)** method reached **0.775 AUROC** while only incurring ~1.1× the computational cost of a baseline forward pass. In contrast, the full semantic entropy calculation (with multiple samples and clustering) was slightly more accurate (0.79) but 5–10× more expensive. This illustrates a practical enhancement: Kimera can adopt SEP to get most of the benefit of semantic entropy **without sacrificing real-time performance**. By training a lightweight probe, the system gains nearly the same accuracy in detecting uncertain outputs, and it remains **fully real-time feasible (SEP requires only a single generation, unlike the expensive multi-sampling of full semantic entropy)**.

- **Impact of Semantic Pruning on Concept Preservation:** The Kimera SWM pruning logs clearly demonstrate the value of entropy-guided pruning for maintaining meaning. At **50% pruning**, the model pruned with semantic entropy retained **90% of concept labels** (Concept_Preservation = 90) and high interpretability (87), with negligible accuracy drop (90.8% from 92%) [3]. In contrast, a standard magnitude-pruned model at 50% prune kept only **65% of concepts** and saw a larger accuracy hit (84.5%) and much lower interpretability (63) [4]. Even a randomly pruned model collapsed to about **55% concept retention** at 50% prune [13]. These figures support the recommendation that Kimera use semantic criteria when compressing models – it can drastically reduce model size while **preserving the knowledge**. The data (summarized in the **"Semantic Thermodynamic Entropy Pruning"** brief) notes that traditional pruning often has *Low* semantic preservation, whereas a **Semantic-Thermodynamic approach achieves "Very High" semantic preservation (with only ~3–13% accuracy loss at 80–90% compression)** [14]. In practice, this means Kimera can become lean and fast but still "know" almost what it knew before – a key to both efficiency and interpretability.

- **Thermodynamic Efficiency Gains:** According to the system metrics provided, combining semantic pruning with thermodynamic principles yields striking improvements in efficiency. The **Semantic-Thermodynamic pruned model (≈80% prune)** operated at **5.5× the computational speed of the dense network** while using only **1/6th the energy (25W vs 150W)**. Remarkably, this model even slightly **improved semantic fidelity to 94%** compared to an entropy-only pruned model at 70% prune (88% fidelity). Its overall information retention was about 480 bits vs 1000 bits in the original, implying it shed redundant information but kept the core semantics. The **thermodynamic efficiency (η)**, which measures how well the system converts energy into preserved information, climbed from **0.15 in the dense model to 0.78 after semantic-thermo pruning**. These numbers validate the idea that an entropy-informed, energy-aware strategy can make Kimera's SWM far more sustainable **without compromising knowledge**. By targeting the "low-entropy, low-yield" parts of the network (those consuming energy but adding little meaning), Kimera achieved an almost **5× increase in ops per watt**. This example supports architectural decisions to incorporate energy optimization into the pruning and to use entropy metrics to guide that process.

- **Ontology Integration Benefits:** In one of the cross-domain evaluations, **ontology-based semantic modeling** in a Business Intelligence context led to **25–30% better insights** in decision support queries. Although this is a different domain, it is directly relevant to Kimera SWM's symbolic integration. It demonstrates that grounding a system's reasoning in an ontology (i.e. a structured symbolic memory) can significantly improve the quality and explainability of its outputs. For Kimera, this suggests that linking the SWM with an ontology or knowledge graph – and using that to interpret and constrain the model's inferences – could yield more consistent and insightful results

(just as it did for those BI systems). Similarly, the **Knowledge Graph** domain entry showed a **15–20% improvement in clustering** when using graph-based representation learning with semantic similarity metrics, hinting that Kimera's world model might better organize its memories by adopting graph semantics and measuring info content in that structure, rather than solely in unstructured latent space.

- **Recent Framework Alignment:** Many of the proposed improvements for Kimera align with cutting-edge research. For example, the **NEPENTHE algorithm** (2024) specifically prunes neural layers by removing low-entropy connections, effectively flattening uninformative parts of the network [11]. And the **MIPP approach** (2025) conserves mutual information across layers during pruning, ensuring pruned models remain as informative as the original [9]. These techniques match the identified blindspots: NEPENTHE addresses the oversimplified "magnitude-only" pruning by focusing on information content, and MIPP targets the often ignored MI signal to keep pruned networks interpretable. By integrating such approaches, Kimera SWM would be on par with the state-of-the-art: e.g. selecting which neurons to cut not just by size but by how much uncertainty they contribute (a form of **"Thermodynamic Filter Selection"** as mentioned in the documentation) [11]. This means in practice Kimera's pruning module would ask: *if I remove this neuron, how does the entropy of the output or the mutual information between layers change?* – a much smarter criterion than current static thresholds.

Each of these examples from the data and literature reinforces the case for our enhancements. Kimera's current design has solid foundations, but by addressing these blindspots – injecting semantic entropy into contradiction handling, using richer entropy proxies, marrying pruning with symbolic knowledge, and exploiting latent information metrics – the **Kimera Semantic World Model** can achieve superior consistency, efficiency, and explainability grounded in demonstrated results and principles [14]. The path forward is clear: treat entropy not just as an abstract theory, but as a practical tool for building a smarter, more robust SWM.

---

[1] [8] Epistemology and Metacognition in Artificial Intelligence: Defining, Classifying, and Governing the Limits of AI Knowledge | Nova Spivack
https://www.novaspivack.com/technology/ai-technology/epistemology-and-metacognition-in-artificial-intelligence-defining-classifying-and-governing-the-limits-of-ai-knowledge

[2] [6] Evaluating LLMs using semantic entropy | Thoughtworks United States
https://www.thoughtworks.com/en-us/insights/blog/generative-ai/Evaluating-LLM-using-semantic-entropy

[3] [4] [13] e871a31a.csv
file://file-731sSwQFqNDs1okNUmZWFT

[5] [7] [10] [2406.15927] Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs
https://arxiv.org/abs/2406.15927

[9] [2411.00147] Mutual Information Preserving Neural Network Pruning
https://arxiv.org/abs/2411.00147

[11] [12] [14] Safari.pdf
file://file-3yz2knG9HiHXwAbvbYaQtX