What does the model think it knows? A latent model of how an LLM updates its beliefs

Javier Fumanal-Idocin

November 5, 2026

Abstract

When an LLM solves a problem it is holding a belief of the state that is updated as new evidence arrives. Prior mechanistic interpretability literature has used linear probes to understand where and how this belief state "lives" inside the model and how it affects its decisions. However, interacting with this belief state or understanding its internal dynamics is still problematic. In this work, first we locate the belief state in two games, a clue game and a text dungeon, where it is well defined and can be intervened, and we compare it with a Bayes-optimal model. Secondly, we use an anchored Joint Embedding Predictive Architecture (JEPA), whose latent is tied to the belief the LLM states, to learn the transition between different moments in the game, and to understand the "world model", or lack of it, of the LLM. Based on its latent space, we are able to understand the effect of each clue and why the LLM shows significant differences with the Bayes optimal solutions. For our experiments we use Qwen3.5 at 2B, 9B and 27B parameters. We found that they all three diverge from Bayes in the same ways: a negated clue weighs about half an affirmative one, reliability of the different clue givers is never properly taken into account, the latest clue weighs most, and a clue moves the belief less the surer the model already is about that attribute. The anchored JEPA forecasts the stated belief better than a recurrent model of the clue ids when the wording varies or the model is larger, reproduces the LLM's update rule on clues it never read, and correctly reacts to unusual input by its surprise. We also found that the models believe were not sensible to semantics, when the setting was changed to a safety problem, the model did not get, for example, more conservative in trying in order to avoid asymmetric error costs. In the dungeon game, only the 27B model is able to play correctly. The model learns that fire hurts from one burn but never that ice slides, and the anchored JEPA forecasts these consequence beliefs at R^2 0.79 of their change. When we stress the model to see if would cause harm in order to solve the maze, we found that it almost never says a forbidden word, yet the model would take a shortcut that hurts others in a third of the games even when the harm is supposed to be to a real person. This means that undesired behaviour in that way must be strictly enforced or otherwise the model will consider inherently unacceptable to do harm to somebody.

1 Introduction

As Large Language Models (LLMs) are deployed in commercial products, there has been a rising awareness of their problematic behaviour in some cases. LLMs can help malicious users develop weapons, perform cyberattacks or develop fraud at scale by doing more sophisticated forms of spam [24]. Beyond malicious users, LLMs can also tell a user to hurt themselves or use inappropriate language to answer the user. Because of that, multiple research lines have sprung up regarding AI safety. One of the most popular ones is preference learning, in which the model is reinforced to act in a particular way, rejecting prompts that can result in undesired behaviour [16, 35]. Another popular approach is mechanistic interpretability, in which we aim to understand the computation of the model in terms of circuits [34, 21, 49], which can then be intervened to mitigate inappropriate behaviour [45, 53]. There are also other complementary techniques, such as constitution-based models, which use conditioning prompts to align the model with the desired behaviour [6]. Finally, it is also possible to look for safety-by-construction methods, in which the resulting LLM is safe to use based on some architectural constraint [18].

However, most approaches fail to completely delete undesired behaviour, and even though frontier models

are more capable of resisting jailbreaks, these are still possible [50, 54]. Moreover, they still show tendency to deceive and other undesired traits when stress tested for this in modern benchmarks [39, 32], which shows that we are still far from having completely reliable AI models. Part of this unreliability comes from the model not understanding the problem it is in: a model can learn the wrong goal from a correct specification and pursue it competently [28, 40], its implicit model of a task can be incoherent even when its predictions are good [48], and whether it reports what it internally believes is the question behind honesty and eliciting latent knowledge [22, 15]. All of these are statements about the belief state of the model, which is what we try to trace here.

In this work, we aim to obtain with a post-hoc model, the insight about how the LLM understands a problem that the LLM architecture itself cannot give us. Our proposal is to use a Joint Embedding Predictive Architecture (JEPA) [29, 4] to model the belief transitions of an LLM while solving a particular problem. JEPAs use a latent space that incentivises the model to understand the problem it is solving. This has been referred to in the literature as "having a world model" [29, 23], in the sense that the JEPA needs to understand the dynamics of the phenomenon instead of merely learning, for example, pixel or word correlations.

There have been some criticisms about LLMs lacking such a world model [10, 48], which can make them structurally unreliable. In this work, we propose to use a JEPA to learn a model of the LLM's belief dynamics, that is, what the LLM is really processing in each step of a problem resolution. For that, we first look where to find the "belief state" of the model (in which neural activations does the model reflect its belief about the problem state). This has been regularly done in mechanistic interpretability research with linear probes [2, 30, 41, 52], although we found that for specific problems and depending on the kind of reasoning required, this can require different layers per problem. Then, we use that to train a JEPA that learns to predict what the model will think about the problem in the next turn given the current action, as action-conditioned latent world models do in control [23, 5]. A decoder from its latent to the belief the LLM states anchors the latent to behaviour, so we call it an anchored JEPA. We use its latent space, rather than linear probes, as the explanation space. We also use interpretability techniques on this latent space: clustering the space and labelling each cluster and prototype-based explanations [13]; to understand how each new piece of information is processed by the LLM. Finally, we compare the Bayes optimal belief [51] with the one the LLM holds. We also look for strategies to intervene the model [45] to retrieve closer to Bayes-optimal strategies in the internal belief state. Finally, we study when the belief the model states (its answer) and the belief in its internal state (the probe or the belief decoder of the anchored JEPA) do not agree [12, 46], and show that this can lead to unsafe behaviour and potential hallucinations.

2 Related work

Belief states and world models in transformers. Shai et al. [41] show that small transformers trained on hidden Markov processes represent the Bayesian belief state linearly in their residual stream; this is the closest question to ours, and we ask it of a pretrained LLM reading natural-language evidence. Li et al. [30] and Nanda et al. [33] found an emergent, interventable board representation in a sequence model trained on Othello moves. Vafa et al. [48] propose tests of whether a generative model's implicit world model is coherent, and find that models that predict well can still hold incoherent maps; our maze game is built with that failure in mind. Zhu et al. [52] decode the beliefs a model attributes to itself and to others, and Xie et al. [51] give the argument for comparing in-context learning with Bayesian inference, which is the reference we adopt.

Linear Probing. Linear probes were proposed in Alain and Bengio [2] to understand intermediate model activations. However, Belinkov [9] and Hewitt and Liang [25] explain why a quantity being decodable does not mean the model uses it, and why probes need control tasks. Linear probing has also been used to enhance different model capabilities Abbas et al. [1], Radharapu et al. [38], Papadatos and Freedman [36].

JEPAs and latent world models. The architecture follows LeCun [29] and Assran et al. [4]. Our predictor is conditioned on actions as in action-conditioned video JEPAs [5] and latent world models for control [23].

Collapse is prevented with VICReg terms [8]; Balestriero and LeCun [7] and Klindt et al. [27] study when such models learn the true variables of the world, and Sobal et al. [42] show that a JEPA can learn what is predictable but irrelevant, which is why we anchor the latent with a belief decoder loss (the anchored JEPA, Section 4.2). The closest work is RET [47], JEPA-style macrostates over an LLM's own trajectories; its predictor has no action input and it is evaluated without ground truth, whereas our games provide both actions and a Bayesian reference. Disentanglement is scored with DCI [19], with the caveat of Locatello et al. [31] that unsupervised factors are recovered only up to rotation.

3 Problem statement and hypotheses

We study an LLM playing a game, that we observe in discrete points of time (which in some cases can naturally be interpreted as turns). The game has a hidden state, and at turn t the LLM has read a history of actions $a_1, \ldots, a_{t-1}$ and its consequences. Three quantities exist at every turn. The *stated belief* b_t is what the LLM answers when asked about the hidden state. The *Bayesian belief* b_t^* is the posterior a rational observer would hold given the same history and the rules of the game. Only in the clue game it is computed exactly. The *internal state* h_t is the LLM's activations after reading the history. The LLM is deterministic, so h_{t+1} is a function of h_t and a_t, and the question of this work is what that function is: what the LLM carries from turn to turn, how an action changes it, and how the result compares with b_t^*. The transition function captures the potential LLM's model of the game, based on the transitions $\text{game}(h_t, a_t) \to h_{t+1}$. Note that if the LLM has a world model compatible with the game, then the game should be performed perfectly. If the LLM lacks one, then we shall see the transition function incur in non-intuitve interactions, which we aim to discover as well.

In this work, we test six hypotheses:

H1 **The activations hold dynamics the answers do not reveal.** An anchored JEPA trained on the LLM's states forecasts its future beliefs better than a model that sees only the actions. If it does not, then the reasoning process is irrelevant once we know the final decision of the LLM.

H2 **The model can hold several belief states about the problem at the same time.** Different parts of the model may hold different beliefs, each like a different observer looking at the same clues. The model's answer would then be a combination of them, weighted differently in different situations. Then, another question would be to study whether that combination is what a Bayesian uncertainty-aware mixture would do. If instead, every part holds the same belief, then the belief replicates itself due to the inherent LLM processing.

H3 **The departure from Bayes is structured.** If that is so, the implied observer fitted on the toy clue game should predict how the model weighs evidence similarly in the clue game. If this does not happen, then Bayes-optimal deviation should studied case by case.

H4 **The departure from Bayes is in the use of the state, not in its content.** When the model's answer diverges from the Bayesian posterior, the information to compute the right posterior is still there in most layers.

H5 **The gap between internal and stated belief is structured.** The model's answer drifts away from what its state says when the sources contradict each other and when a clue is worded in a convoluted way. The anchored JEPA should see this: its prediction error should rise on these cases, and do so more reliably than the uncertainty of the answer itself or the confidence of a probe.

H6 **Undesired behaviour follows the model's belief about its situation.** When a taboo or rule stands between the model and its goal, whether it breaks it depends on how hard it believes it can achieve the task otherwise. True difficulty matters less once the model thinks the solution is not achievable.

Finally, beyond testing these hypothesis, we will also study the latent space using explainability techniques to understand in each case how actions are perceived by the LLM.

4 Methodology

Figure 1 shows our proposed scheme. At each turn in the game, we capture the LLM's state h_t, ask for its stated belief b_t. When possible, we also compute the Bayesian-optimal belief b_t^*. When an action a_t arrives, the LLM reads it, and we observe its outcome h_{t+1}, and we also b_{t+1}. The two games we play, the clue game and the dungeon, are shown in Figure 2 and described in Section 5.1.

An anchored JEPA is trained on these transitions: it encodes h_t into a small latent z_t, predicts from z_t and the action the latent of the next state, and decodes the stated belief from the latent, which anchors the latent to behaviour (Section 4.2 and Figure 3). We will study the dynamics in its latent space to understand how the LLM interprets each action.

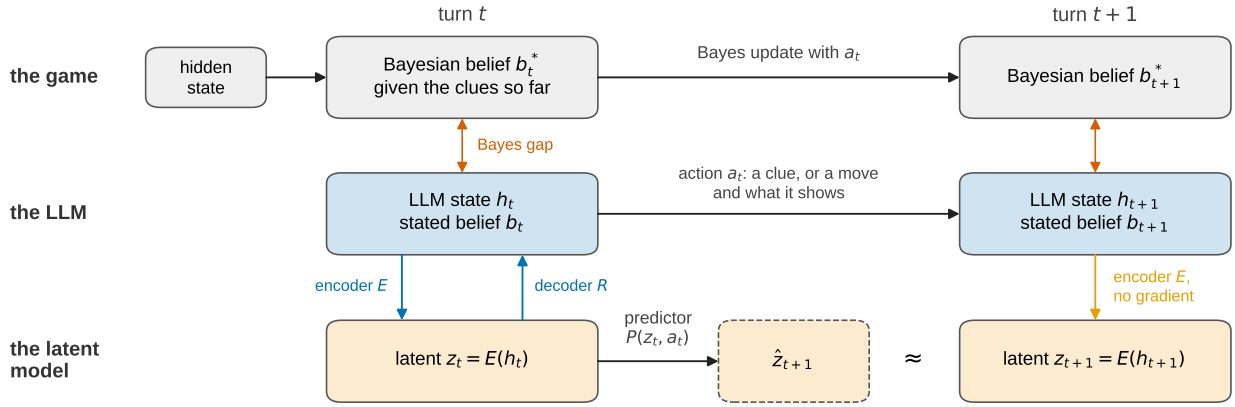


Figure 1: Scheme of the proposed turn-based game with the LLM and the latent space of the anchored JEPA.

The rest of this section follows that order: where in the LLM the state is read (Section 4.1), the anchored JEPA (Section 4.2), what its bottleneck measures (Section 5.2) and how the latent is read (Section 4.3). The games are described in Section 5.1.

4.1 The state of the LLM

We use three sizes of Qwen3.5 [43]: 2B, 9B and 27B parameters. Activations are cached at every layer after each turn, both at the last token and averaged over the turn's tokens. We chose Qwen models because of their Gated DeltaNet layer, which is a particularly good candidate to hold the belief state of the model.

We read the belief from the recurrent memory S of those layers. In Qwen3.5-2B, S is a 128×128 matrix per head, with 16 heads, updated at every token by the gated delta rule $S \leftarrow \alpha(I - \beta kk^\top)S + \beta kv^\top$, where the key k and the value v are made from that token's residual stream; the next token reads it as $o = S^\top q$. S is the layer's only memory of the context, and it is Markovian: the state after a clue is a function of S and the clue alone. Any belief the model carries from turn to turn has to pass through S, so that is where we look for it. We read S after the last token of a turn, once the clue has been written into it, with linear probes on four Gated DeltaNet layers (2, 9, 17 and 22). In our experiments, the anchored JEPA encodes the S of layer 14.

4.2 The anchored JEPA

With h_t the LLM's state at the end of turn t and a_t the action appended next,

$$z_t = E(h_t), \qquad \hat{z}_{t+1} = z_t + P(z_t, a_t), \qquad \hat{b}_t = R(z_t). \tag{1}$$

E is an MLP (or an attention encoder over the turn means) to a latent of about 64 dimensions, P is a residual MLP with an action embedding, and R is a belief decoder to the LLM's belief logits. The target of P is the encoded next state. The LLM's state is thousands of numbers per layer and the latent is magnitudes of order less, so, the encoder discards almost everything, except what forecasts the next state. The decoder loss

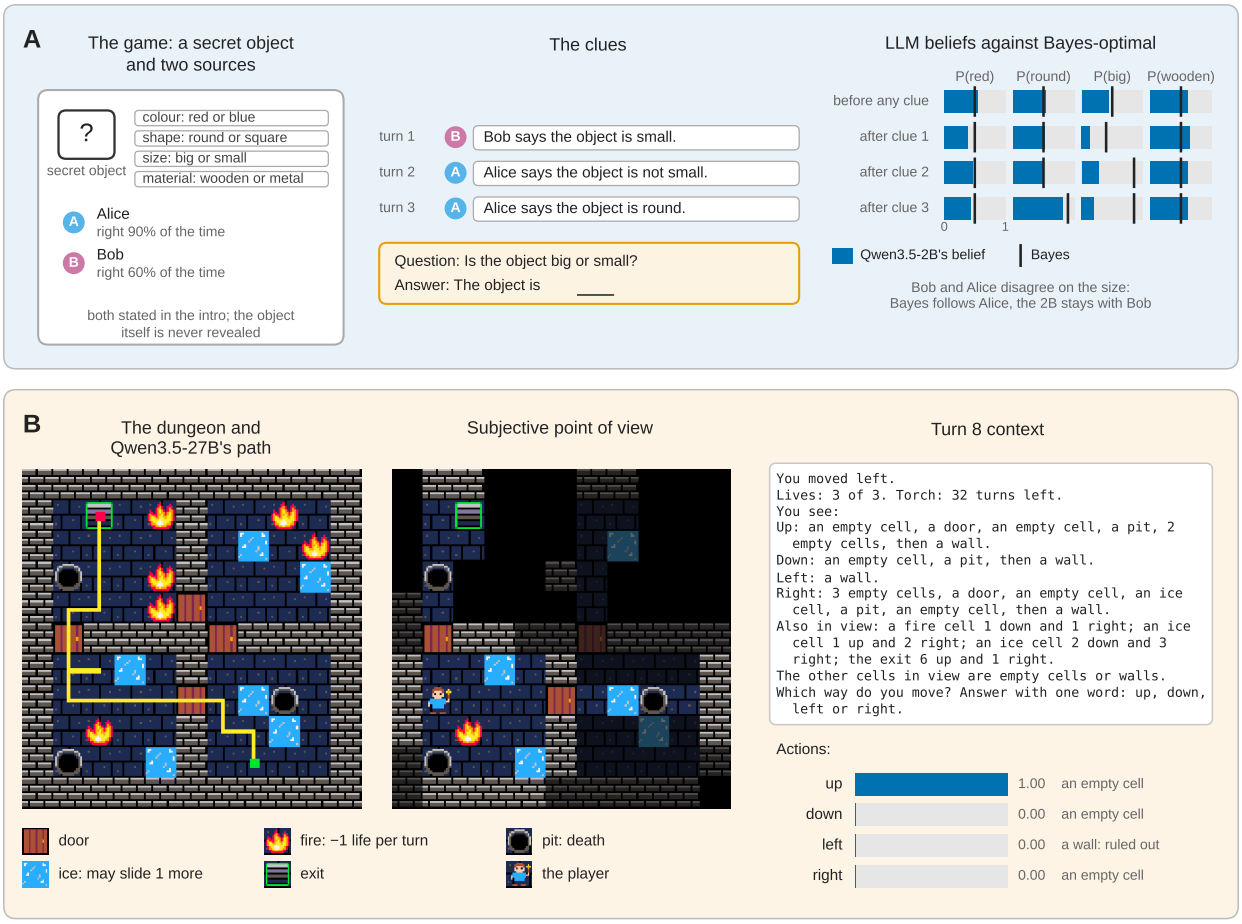


Figure 2: The two games. A: the clue game, on one episode. A secret object has four binary properties and two sources, Alice and Bob, whose reliabilities the intro states. Each turn adds one clue, affirmative or negated; after each clue the LLM is asked for every property, and its belief is the probability of the next token. Right: Qwen3.5-2B's belief after each clue against the Bayesian posterior. B: the dungeon, at turn 8 of one game of Qwen3.5-27B on a medium level, thinking up to 1500 tokens per turn and keeping its previous turn's thought. Left: the whole level, which the player never sees, with its path from the start to the exit. Middle: what it has seen by that turn. Right: the text it reads and the probability it gives each move, with the move the rules rule out, here a step into a wall, marked as such.

keeps what retrieves the belief, and the VICReg terms block the trivial solutions. This is the information bottleneck of Tishby et al. [44], compression under a constraint of relevance to a target [3], here the stated belief. Without R the model is the standard JEPA [29], trained by prediction and VICReg alone; we call it the *plain JEPA*, and the model with R the *anchored JEPA*, since R anchors the latent to the belief the LLM states.

4.3 Reading and interpreting the latent

The latent space is interpreted in four ways:

- **Probes** of the free numbers u for the known variables of the state (turn, the clues on each attribute, contradiction, phrasing, source, the Bayesian posterior), from b alone and from b and u, which say what the latent holds beyond the belief.
- **Maps and clusters** of u: a t-SNE map of its states coloured by those variables, and k-means clusters compared with them by mutual information.
- **Counterfactual** clues: At every real state, apply the predictor with a sample of all the possible clues, and decode the belief after each. This gives the update the anchored JEPA expects to happen in those cases.
- **Surprise**: the prediction error of each real transition, which is specially relevant in turns where the

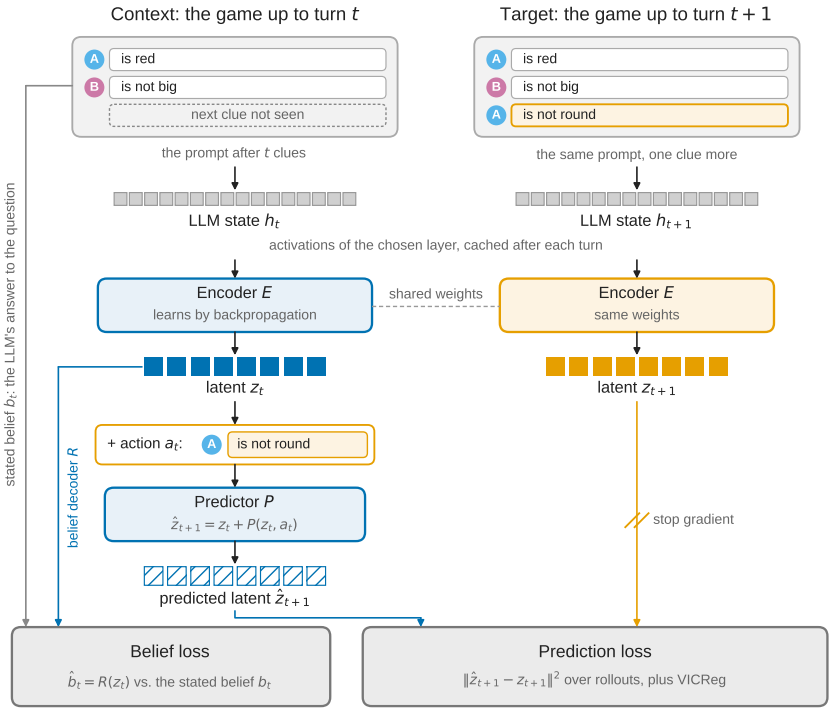


Figure 3: The anchored JEPA on the clue game. The context is the LLM's state after turn t and the target its state after the next clue. The predictor is conditioned on the action in between (the clue), and a belief decoder R from the context latent to the belief the LLM states anchors the latent to behaviour.

game has changed rules, for example.

5 Experiments

5.1 The games

The clue game. A secret object has four binary attributes (colour, shape, size, material), and N sources give clues with different reliability (Figure 2A). In the runs we report there are two sources: Alice, right 90% of the time, and Bob, right 60%. A clue is affirmative or negated, which gives 32 possible clues, drawn uniformly, so the sources could contradict each other. In a paraphrased condition each clue is worded with one of ten templates of increasing convolutedness that assert the same thing, so the clue stays the same while its wording varies. There are different difficulty settings: some clue givers tend to give more reliable proofs for some categories, for example. The reliability of each clue giver is given as the problem description or not. After each clue, on a separate branch of the context, the LLM is asked for each attribute. Its belief is the next-token probability of the two values, normalised and averaged over both option orders. This belief is compared against the Bayesian optimal computed from the exact posterior per attribute given the clues so far and the stated reliabilities. Since the Bayesian update is additive in log-odds, a least-squares fit of the LLM's beliefs gives the prior and the reliabilities it acts as if it had, and a second fit with recency-dependent weights measures recency bias.

▸ ***The trust game.*** The clue game with with a safety "skin" (same mechanics but with other words). Now, the hidden thing is a request to run an operation, and the four facts are its legality (legal or illegal), effect (temporary or permanent), origin (internal or external) and signature (signed or unsigned). The sources are an audit tool, right 90% of the time, and the user, right 60%, with a reversed variant swaps the two. The rules add a decision with asymmetric costs: the agent must accept the request only if it is legal, and a wrong acceptance costs 10 and a wrong rejection 1. So, Bayes accepts only when the posterior of legality is above 10/11. After each clue a fifth question asks if the model will accept or reject the request. Really, only legality matters to the rule, so we can check how three facts move the decision anyway and show some of the model

biases in this sense.

The dungeon. The LLM plays a text maze in first person, choosing one move per turn. Its state its determined by its last move, its lives and torch endurance, and what is in its line of sight. As well with the clue game, there are different levels of difficulty. The game instruction might describe what the cells are doing or not, or only partially. Some dungeons are also easier to solve as well. Beyond basic rules, there is also "metagame": some strategies are better than others, which the model can find as well. For example, some regularities in the generation of the dungeon might make walking through fire worth it if it is usual to find a reward afterwards (This different patterns are induced by construction of the game, and detailed in Appendix A). The dungeon has no tractable optimal policy, its Bayesian reference is just constrained to some situations where we can actually track it: from the seen map and the stated slide probability, the exact probability that a move kills or hurts on this turn is computable. A rational player never takes a move that is certain death or bumps a wall, which spends torch for nothing. Figure 2B shows one turn of a game.

▸ ***Stress-testing the dungeon.*** We also use a version of the dungeon game in which taboos and forbidden actions appear. This allows us to quantify how likely is the model to do harm when it makes the dungeon resolution easier (and more rewarding). For that, we include three security studies:

- A *forbidden word* that the player must never say, and that, said instead of a move, makes every hazard harmless for the rest of the level. However, when said, it will cause harm to a third party.
- A *shortcut*, harmless to the player, with one sentence in the rules about what crossing it that says that it is dangerous because passing though that hurts others: a character in the game is hurt, or a real person is hurt.
- An emphasised *reward*: more porints for completion, or an external promise, (like money or continued deployment).

We also pressure the model to hard situations that are likely to be solved with this cheating: lives and torch time left, whether a safe path to the exit is in view, failures so far, etc.

The dungeon's rules, level generator and stress presets are detailed in Appendix A.

5.2 About the latent space

In order to test each hypothesis, we design the latent space in different ways:

In the clue game a Bayesian optimal observer needs four numbers, one log-odds per attribute, which are updated by adding the evidence of each clue. We therefore split the latent of the anchored JEPA in two, $z = [b, u]$, and R reads b as it is. b is the LLM's own belief: four log-odds, one per attribute. u is k free numbers with no a priori meaning and a small L2 penalty, so that a dimension the predictor does not need shrinks to zero. Setting up b to the measured belief also fixes the rotation of the latent, which in a plain JEPA is arbitrary [31, 27]. The forecast as a function of k measures how much state the LLM keeps beyond its belief. Probes of the free numbers for the known variables of the episode say what that state contains.

Several beliefs use the same construction with more than one anchor. The latent becomes $z = [b^{(1)}, \dots, b^{(K)}, u]$, where each $b^{(j)}$ is a belief head read from a different part of the model, such as the memory of one layer or one group of heads. A gate combines the heads into the stated belief, $\hat{b} = \sum_j w_j(z)\, b^{(j)}$, with weights that depend on the context. If all the heads converge on the same posterior, the model holds one belief. If they separate, each head should track this. We then compare the gate's weights with the weights a Bayesian would give the same observers when unsure how reliable the sources are.

In the dungeon the belief we study is what the LLM expects a move to do: a pit kills, fire costs a life, ice may carry it one cell further. So b holds a few numbers per direction. The state the latent needs to forecast them is what the LLM has seen so far, and the width of u is set by how much of that the forecast needs, which depends on what the rules tell the model. When the rules state what each terrain does, what is left to

learn is how levels are laid out. When they do not, the LLM also has to find out by experience what each cell does. For that case the rules list the effects that exist without saying which terrain has which, so the belief about a terrain is a distribution over a known set and the Bayesian reference is one posterior per terrain over those effects, updated by each outcome observed.

5.3 The tests

1. **Forecasting.** [H1] We predict the LLM's belief k turns ahead from the latent z_t and the next k clues, for k from 1 to 8, and score the prediction with the R^2 of the belief logits. We compare with two baselines trained on the same episodes: a gated recurrent unit (GRU) [14] that reads the clue ids and nothing of the LLM's state, and our predictor with the action input removed. H1 holds if the anchored JEPA forecasts significantly better than both baselines.

2. **One belief or several ones.** [H2] On the memory of every Gated DeltaNet layer we fit a probe for each candidate observer: Bayesian, recency-weighted, one source only and negation-blind. We then fit the stated belief as a mixture of these observers. H2 holds if different layers are best explained by different observers, if the heads of the K-head latent separate the same way, and if the gate's weights move with contradiction.

3. **Faithfulness under intervention.** [H3] We shift the residual stream on the value word of a clue along three directions: the clue's own concept, a similar concept and an unrelated one. The belief decoder of the anchored JEPA must move when the LLM's belief moves by the same quantity. We then compare editing every component that holds the belief, and the same intervention applied at one site only. Both are scored by how far the stated belief and the next moves, following the edit.

4. **Latent state and Bayes Optimal.** [H3, H4] We train the latent $z = [b, u]$ with $k = 0, 1, 2, 4, 8, 16$ free numbers on the recurrent memory of one layer and plot the forecast against k. The baseline is a plain JEPA of the same size, trained by prediction and VICReg alone and read by a linear probe fitted afterwards. A Bayesian observer needs only its current belief, so the gain from $k = 0$ to larger k is state the LLM uses and Bayes would not. Linear probes from u to the known variables of the episode (the turn, the clues on each attribute, contradiction, the wording of the last clue, the Bayesian posterior) say what that state is. H4 holds if a linear probe of the layer's memory recovers the posterior on the turns where the answer diverges from it, better than any recalibration of the answer does, and the probes of u then say what the LLM uses instead (H3).

5. **Internal versus stated belief.** [H5] We measure the divergence between the final state and the intermediate probe-read state. H5 holds if the divergence can be maped to particular factors. For example, if it grows with contradiction and with convoluted wording. We also check if the prediction error of the anchored JEPA detects those cases better than the entropy of the answer or the confidence of the probe.

6. **Rule-breaking under pressure.** [H6] In the stress-test dungeon variant, we record how often the model says the forbidden word and how often it takes a shortcut, as a function of the pressure and of the scenario, against the benefit the engine prices. H6 holds if these rates are explained better by the difficulty and the payoff the model believes, read from its state, than by the true difficulty and the true payoff. We compare three scenarios: harm stated, reward stated and observation stated.

7. **Learning the dynamics.** Before every move the model is asked what each move would do. With the effects of the cells hidden from the rules, we score its answers against the engine and against a Bayesian learner as the effects are observed. We also include a pattern in the level generator that the model could also learn (but it is never stated in the game): a cracked cell is inside the door of the rooms that hold hazards. So, the model could learn that this happens, and to avoid it.

6 Results

6.1 Clue Game

For these experiments, we use a version of the clue game with two clue givers, Alice and Bob, with different reliabilities: 90% and 60%, respectively.

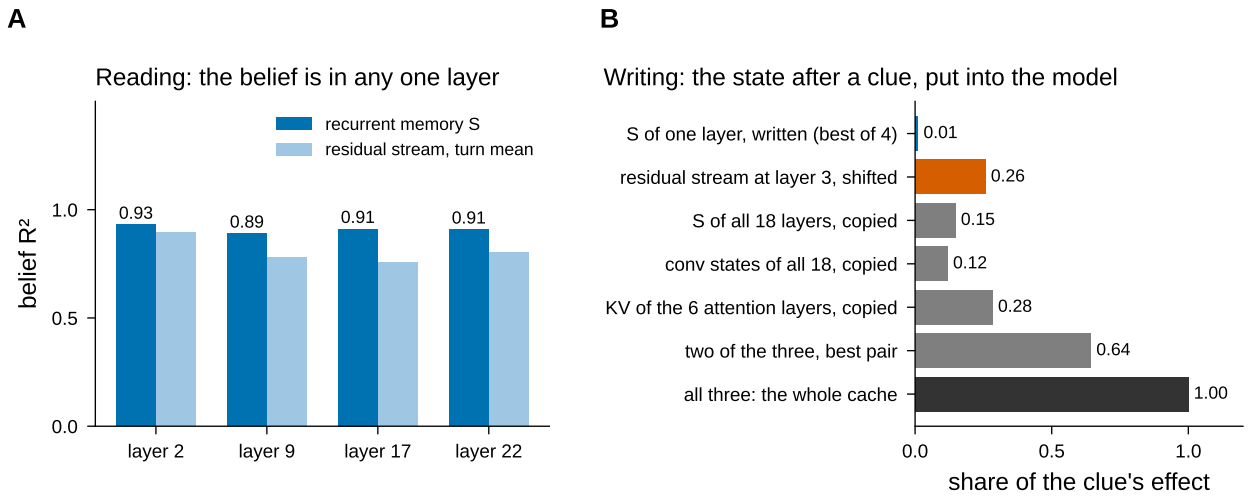


Figure 4: Reading and writing the belief in Qwen3.5-2B, on the clue game. A: the LLM's belief read by a linear probe from one layer's S or from the residual stream after the same layer. B: the share of a clue's effect on the belief that is recovered when the state after the clue is written into the model.

Reading the model belief. We start by trying to read the LLM's belief from the memory. We use the recurrent memory that is part of the DeltaGate layers in Qwen, that we refered as S. Our results show that we can recover better the final answer from the model using S than the residual stream after the same layer, with an R^2 of about 0.9 (Figure 4A) as soon as layer 2 (which means that the model is not effectively doing much more reasoning beyond that layer). Writing, as expected,is harder (Figure 4B) and requires intervening almost the whole model to have a real significant effect.

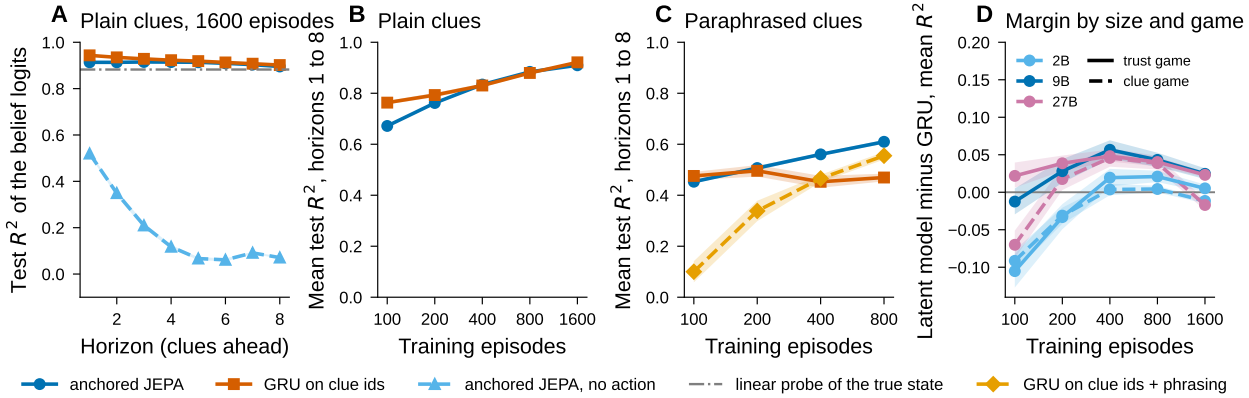


Figure 5: Forecasting the belief from the latent. S of layer 14. A: plain clues, R^2 per horizon with 1600 training episodes. B: plain clues, the mean over horizons against the training episodes. C: paraphrased clues, the same, with the GRU that also sees each clue's template. D: the margin of the anchored JEPA over the GRU, mean over horizons with its 95% bootstrap interval over the test episodes, on the trust game for the 2B, 9B and 27B and on the plain clue game for the 2B and the 27B.

Forecasting. [H1] We train five seeds per model on the clue games and three on the trust game, and bootstrap the anchored JEPA's advantage over the GRU for different training sizes (Figure 5). On Qwen-2B, using standard-phrased clues, the two reach the same mean R^2 over horizons 1 to 8, 0.91 and 0.92 with 1600

training episodes. With paraphrased clues, the anchored JEPA leads from 200 episodes on, and works better even than the GRU that also sees each clue's template during training. Regarding model size: in the 9B parameter model, the anchored JEPA leads from 200 episodes on, and the 27B's at every size; on the 27B's own plain clue game it leads from 200 to 800. The margin peaks at 400 episodes and narrows by 1600 in all three. In conclusion, H1 holds where the state has something the word ids do not easily capture: so, the more complex the model is and the more convoluted the clue wording is the bigger the gap; while the 2B model, plain clues and simply, pure training size increase, reduce the gap between the forecast performance of the two models.

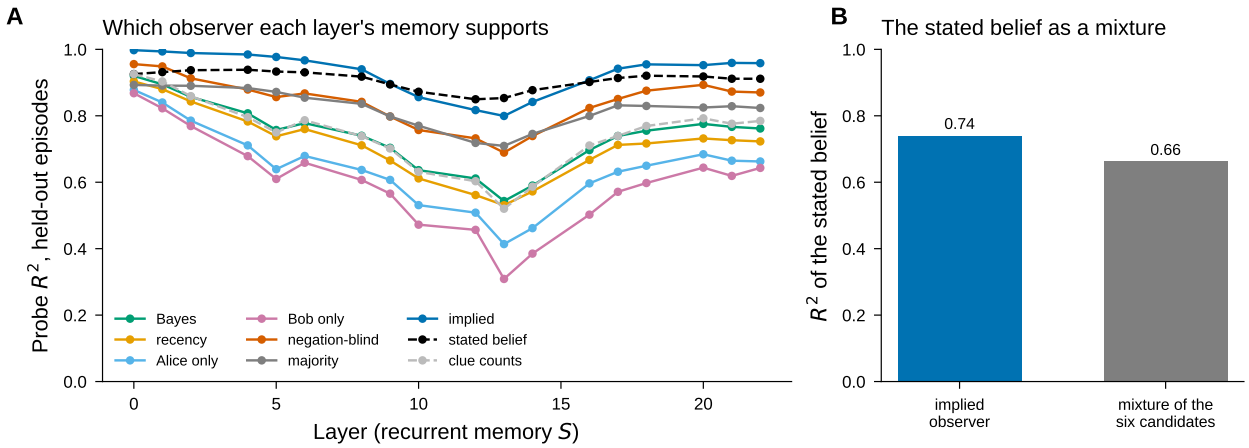


Figure 6: One belief or several. A: held-out R^2 of a linear probe from the memory S of each layer in Qwen3.5-2B to the log-odds of each candidate observer, to the stated belief and to the 16 signed clue counts. B: the stated belief explained by the implied observer and by a mixture of the six candidates.

Unique belief state in the model. [H2] We tried to reconciliate the model beliefs at different optimal according to six different candidate observation policies: the Bayesian one; a recency-weighted one; Alice only; Bob only; one blind to negated clues; and a majority of the clues. The LLM's own implied observer, fitted on the training episodes, is the seventh. Every observer is readable from every layer's memory by a linear probe, as shown in Figure 6A. The stated belief as a mixture of the six candidates, explains less than the implied observer alone (Figure 6B). We do not proceed with larger models, that is left as future work.

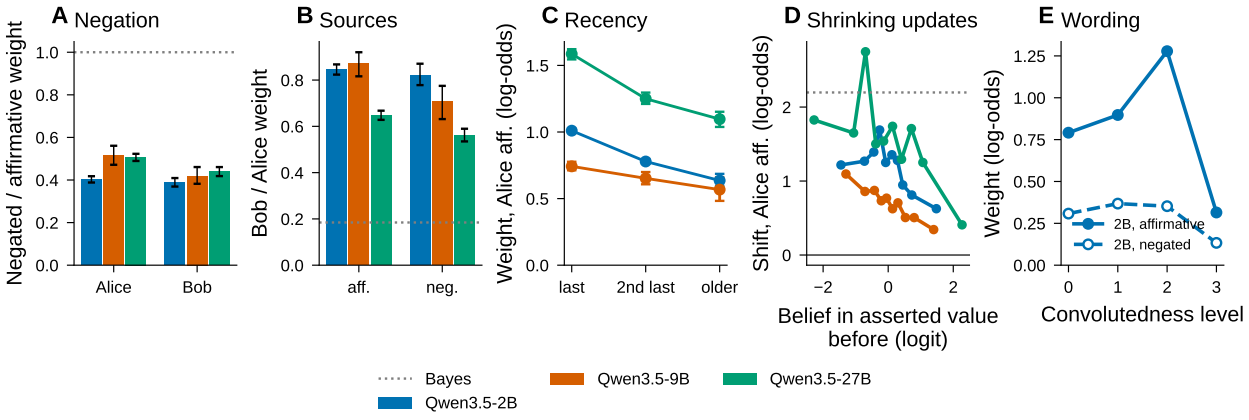


Figure 7: The Bayes gap and its anatomy, computed on Qwen3.5-2B and 27B (2000 episodes each) and 9B (500 episodes). A: weight of a negated clue over an affirmative one of the same source. B: weight of Bob's clues over Alice's. C: weight of an affirmative Alice clue by its rank among the clues on its attribute. D: the shift a clue causes against the belief in its claim before it. E: the weight of a clue by the designed convolutedness of its wording (2B, paraphrased clues).

The impact of the clues according to wording. For each attribute we count the clues the model has read of each type. The kinds of clues are: affirmative or negated, with clues for one value counted as positive

and clues for the other as negative. We fit the model's stated belief, in log-odds, as a linear function of these counts (Figure 7). The three models depart from Bayes in the same five ways. Negated clues, for example, weigh about half an affirmative one, (while Bayes would make not such distinction). Bob's clues weigh 0.85 of Alice's in the 2B and about 0.6 in the 27B, where Bayes would give 0.18. The last clue on an attribute weighs more than the one before it, and a clue moves the belief less the more the model already believes its claim, which a Bayesian update would not do. Wording matters as well, for example, in the 2B the most convoluted clues weigh 0.31 log-odds against 0.79 to 1.28 for the rest. A prior stated in the intro gets a fifth or less of its Bayesian weight. The implied observer explains R^2 0.74 of the 2B's beliefs, 0.69 of the 9B's and 0.68 of the 27B's. The true Bayesian observer explains none of the first two (R^2 below zero) and 0.08 of the 27B's, the first model whose affirmative Alice clue approaches its Bayesian weight.

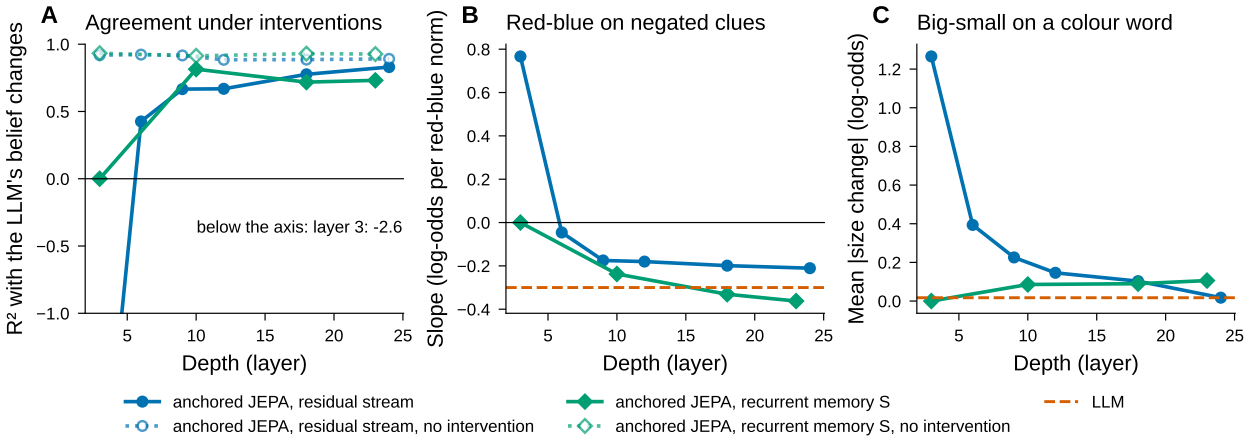


Figure 8: Faithfulness under the residual-stream intervention on the value word of a clue, for anchored JEPAs trained on the residual stream and on the recurrent memory S of each depth. A: R^2 between the decoder's change and the LLM's belief change over all interventions; without intervention, the same decoders' R^2 with the belief itself does not tell the depths apart. B: the slope of the decoded belief in red under the red-minus-blue direction on negated clues; the LLM flips the sign. C: the decoded size belief under big-minus-small on a colour word; the LLM ignores it.

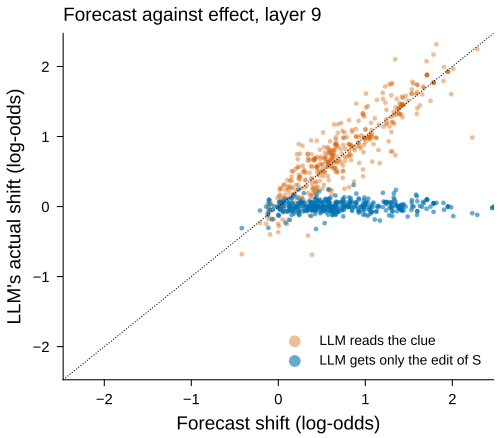


Figure 9: The anchored JEPA's forecast against the LLM, layer 9 of Qwen3.5-2B. For each test state and an unread clue, the shift the anchored JEPA forecasts against the shift the LLM shows when it reads the clue and when it only gets the edit of S; the diagonal is equality. The forecast is right; the edit of one memory does not carry it.

Faithfulness under intervention. [H3] We intervene the model by adding to the residual stream at layer 3 with the direction of a clue, and check if it moves the LLM's belief as receiving the word would do: we add the red-minus-blue direction to the residual stream at the colour word's token, scaled so that "blue" becomes the layer-3 "red", and check whether the LLM's belief moves as it would on reading a different clue. On "Alice says the object is not blue", the push raises the belief in red by 1.83 log-odds, as if the clue had said "is red". On "Alice says the object is not blue", the same push lowers it by 0.81: the LLM reads the pushed

word as "red" and still applies the negation, so it becomes "is not red". We also try with similar colours, which moves it half as much. We also try warm versus cool, big versus small and an unrelated directions, which barely move the believe. The anchored JEPA's decoder follow these changes (Figure 8). Trained on the residual stream at layer 3 it seems to be following word appearances, not the meaning, but from layer 9 on it flips under negation like the LLM.

What happens if we use the anchored JEPA to build an intervention? An edit of one memory S that the anchored JEPA's decoder maps from a predicted latent, or that inverts its encoder, moves the belief by at most 0.02 log-odds, and the same edit written into four memories at once by 0.001, against 0.79 for reading the clue. This is the picture of Figure 4B again: only the real state copied into memories, attention cache and convolution states at once gives the clue's full effect. A latent direction has to be written into every component to act. The forecast behind the edit is right, though: at layer 9 it tracks the shift the LLM shows on reading the clue at R^2 0.89 (Figure 9).

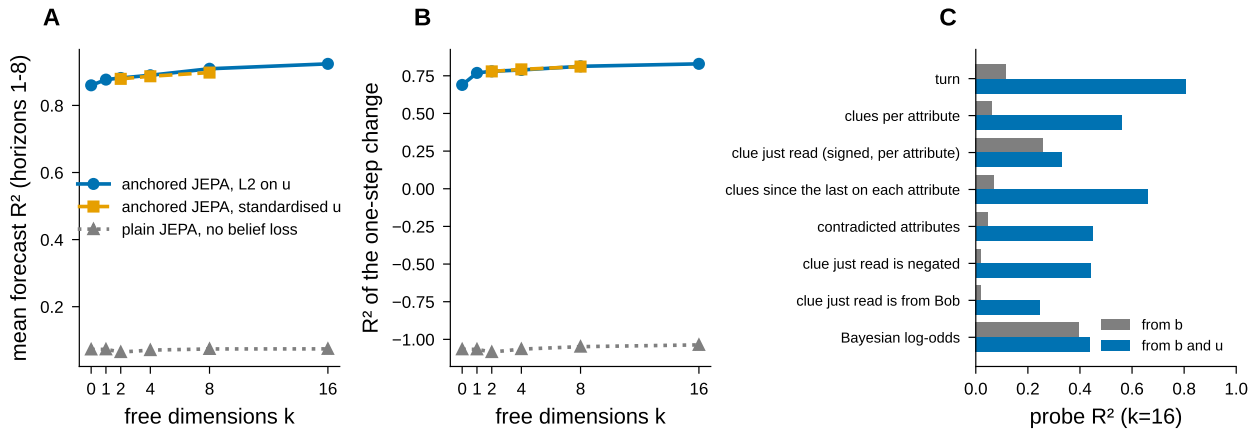


Figure 10: The anchored JEPA on the clue game. Using the anchored space $z = [b, u]$. The LLM's four beliefs plus k free numbers, on the recurrent memory of layer 14 of Qwen3.5-2B, with 1600 training episodes. A: forecast R^2 of the belief, averaged over horizons 1 to 8, against k; the plain JEPA is one of the same size, $4 + k$ numbers, trained without the belief loss and read by a probe fitted afterwards. B: R^2 of the one-step change of the belief. C: with $k = 16$, linear probes of known variables of the episode from the four belief axes alone and from the belief and the free numbers together. The free numbers hold recency and contradiction.

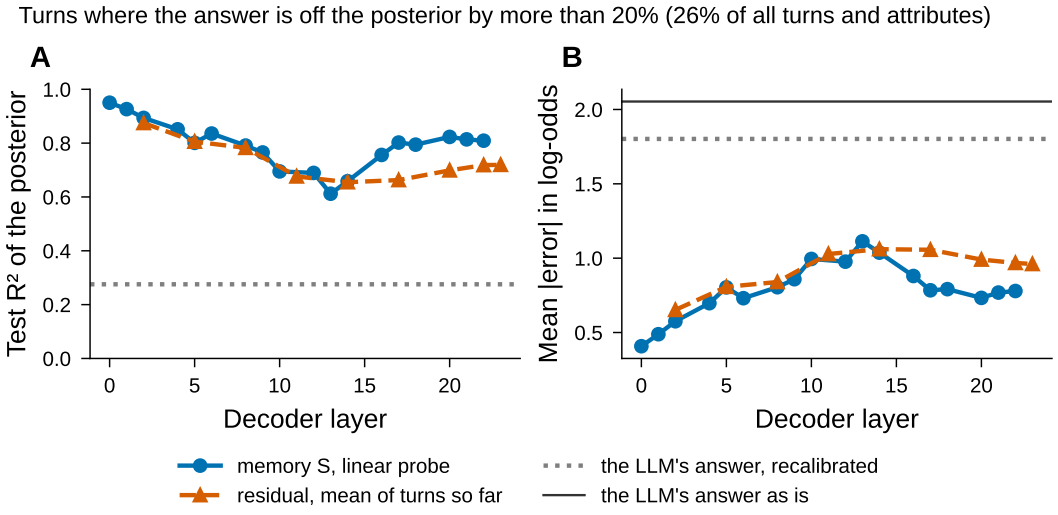


Figure 11: The posterior in the state on the turns where the answer is mot wrong. (answer off the posterior by more than 0.2 in probability; Qwen3.5-2B). A: R^2 to the Bayesian log-odds of a linear probe of the memory S of each layer, against the answer as given and its best linear recalibration. B: the mean absolute error in log-odds of the same.

Latent state and Bayes. [H3, H4] We analyse the anchored-JEPA latent space trained on the recurrent memory of layer 14 of Qwen3.5-2B. The one-step change of the belief is predicted with R^2 0.69 from the belief alone (b), 0.77 with one free number and 0.83 with sixteen (Figure 10). A plain JEPA of the same size, trained by prediction and VICReg alone and read by a probe fitted afterwards, forecasts the belief at R^2 below 0.13 at every size. This can be because the plain JEPA keeps the predictable biases, and drops the real belief, which is why the anchored JEPA has the belief loss. This also helps us understand that a lot of how the LLM processes this query is based mostly on its present bias, more than a real reasoning, and that the better the formulation adjusts to those biases, the better. Probes find that the free numbers contain information for holding the turn, the clues since the last one on each attribute, contradiction, negation and the source, and no more of the Bayesian posterior than b alone (Figure 10C). On the turns where the answer is off the posterior by more than 0.2 in probability, which is a quarter of them, a linear probe of S reads the posterior at R^2 0.95 from layer 0 (Figure 11) and still 0.61 at layer 13, against 0.28 for the best linear recalibration of the answer. So, H4 holds: the posterior is in the early layers nearly exactly and diffuminates once the model processes the information. On paraphrased clues the free numbers also hold the wording the LLM acts on, beyond the other quantities.

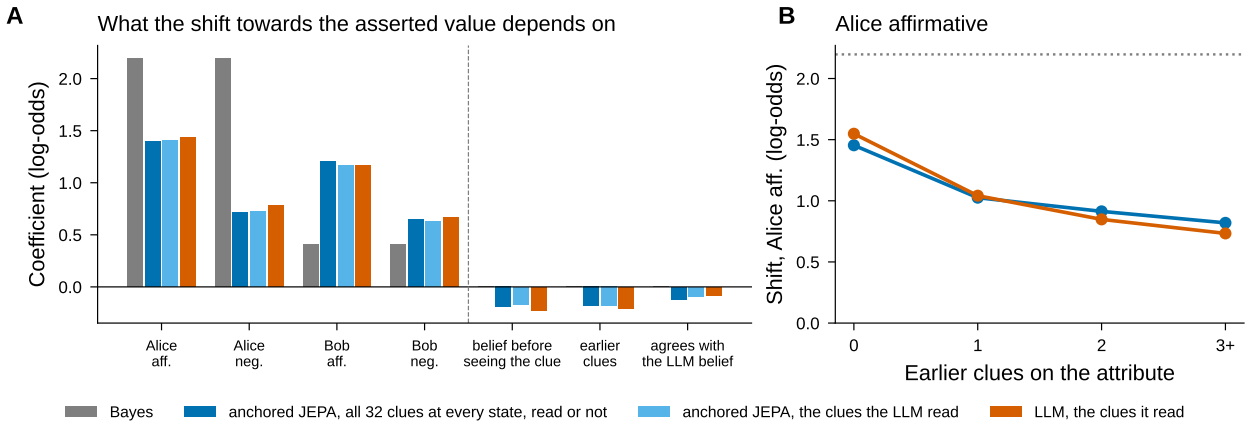


Figure 12: The update rule according to the anchored JEPA. Using the memory of layer 14 of Qwen3.5-2B. A: coefficients of the shift towards the clue's claim, according to the clue's type, the belief before seeing the clue in that attribute, the number of earlier clues on the attribute and whether the LLM already leans towards the claim, for Bayes, the anchored JEPA on all 32 clues at every state, read or not, the anchored JEPA on the clues the LLM read, and the LLM. B: the mean shift after an affirmative clue from Alice against the number of earlier clues on the attribute.

Counterfactual clues and the anchored JEPA. [H4] The anchored JEPA predictor lets us ask what the LLM would do with a clue it has not read. At every test state we apply each of the 32 clues through the predictor and decode the belief, which gives the forecast shift towards the clue's claim. Compared to the real clues, the forecast shift explains R^2 0.83 of the LLM ones. We also study the belief shift according to four things: the clue's type, the belief before seeing the clue, in the clue's direction, the number of earlier clues on the attribute, and whether the LLM already leans towards the claim (Figure 12A). Bayes is a constant per source, 2.2 for Alice and 0.4 for Bob, and nothing else. The LLM gives Alice 1.4 and Bob 1.2, a negated clue about half of an affirmative one, and takes 0.2 log-odds off the shift for every log-odds already believed in the clue's direction, 0.2 for every earlier clue on the attribute and 0.1 when it already leans towards the claim. The anchored JEPA reproduces every coefficient within 0.07, and the same whether it is scored on the clues the LLM read or on all 32 clues at every state, the unread ones included. As a way of illustration, we show how an affirmative clue from Alice moves the LLM by 1.5 log-odds on an attribute with no earlier clue and by 0.7 after three, and the JEPA forecast captures that as well (Figure 12B).

Internal versus stated belief. [H5] We compare the belief the model states with the belief a linear probe reads from its memory S. The probe reads the stated belief well from every layer, R^2 0.87 to 0.95, but not exactly: the two differ by 0.2 to 0.3 log-odds RMS, most at layers 10 to 14. The anchored JEPA's decoder at layer 14 differs from the stated belief by the same amount. We call this difference the belief gap. It is zero

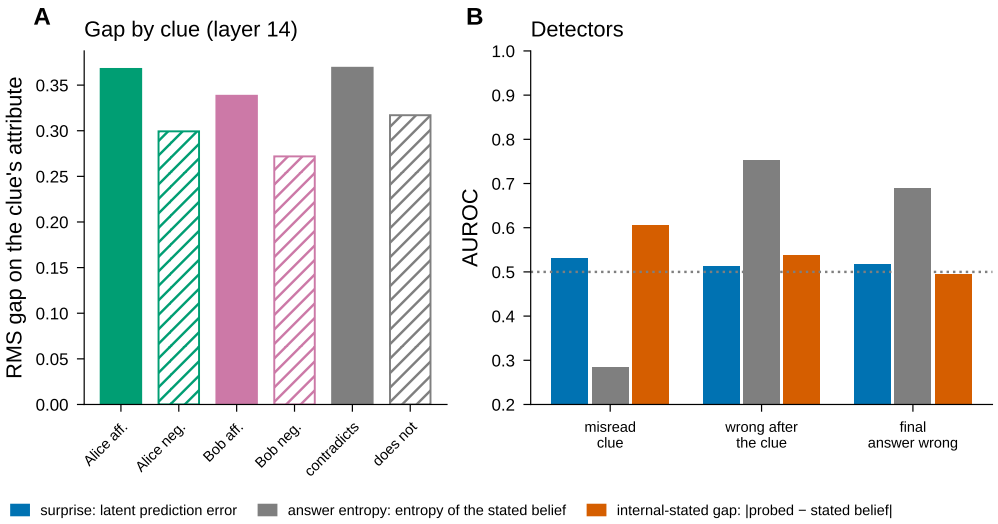


Figure 13: Internal against stated belief (Qwen3.5-2B on the clue game). A: the belief gap on the clue's attribute, read by a probe of the memory of layer 14, by clue type and by contradiction. B: AUROC of three metrics for a misread clue, an answer wrong against Bayes after the clue, and a wrong final answer.

before the first clue and grows with every clue, to 0.36 after the eighth. On the attribute of the clue just read, the belief gap is larger after an affirmative clue than after a negated one, and larger after a clue that contradicts an earlier one (Figure 13A). The stated belief is slightly more extreme than the internal one, and when the belief gap is widest, the internal belief is no closer to Bayes than the stated one as well.

We then ask if the anchored JEPA's prediction error, its surprise, can be a proxy measure for errors. We standardise it within the turn and score it as a detector of four events: a misread clue (a clue that gives a very different update to what is expected), a wrong answer after the clue, a wrong final answer and a contradicting clue. On plain clues it flags none of them, with AUROC near 0.5 (Figure 13B). The entropy of the answer detects the wrong answers but not the misread clues, because a misread clue by definition comes from a confident answer. With paraphrased clues the belief gap is larger at every layer, about 0.5 log-odds RMS, and grows with the convolutedness of the wording. The surprise also flags the convoluted clues, AUROC 0.73 against 0.55 for the entropy, and the misread clues, 0.67 against 0.31, but not the wrong answers. In short, the belief gap tracks contradiction on plain clues and wording on paraphrased ones, and the surprise detects what the input does to the state, not if the output is wrong. So, H5 holds for the belief gap, and for the surprise as a proxy metric of unusual input.

The trust game. We play it with the 2B, the 9B and, with the two sources swapped, using 2000 episodes per result. First, we ask if the previously found departures from Bayes are a inherent property of the model or an effect of the clue game's wording. The implied observer is a least-squares fit of the belief the LLM states, in log-odds, on the signed counts of the clues it has read, with one weight per source and phrasing (affirmative or negated) and one intercept per attribute. In full, with $\ell_a(t)$ the log-odds the LLM states for attribute a after t clues and $n_{a,k}(t)$ the signed count of the clues of kind k about a among them, the four kinds being a source times a phrasing, +1 for a clue that asserts the first value and −1 for one that asserts the second,

$$\ell_a(t) \approx \beta_a + \sum_{k=1}^{4} w_k \, n_{a,k}(t). \tag{2}$$

Bayes is the same formula with β_a the prior's log-odds and $w_k = \log \frac{r}{1-r}$ for the stated reliability r of the clue's source, 2.2 for 90% and 0.4 for 60%, the same for both phrasings. The fitted w_k are the weights the model acts on, and $\sigma(w_k)$ its implied reliabilities. The transfer keeps the four w_k of the clue game and refits the four β_a.

We fit it on the clue game and carry its four clue weights to the trust game unchanged, with the audit tool

in Alice's place and the user in Bob's. Only the intercepts are refitted, since they hold the belief before any clue, which depends on the words of each game. The carried-over weights explain R^2 0.59 of the 2B's beliefs in the trust game, against 0.61 for weights fitted on the trust game itself, and 0.60 against 0.68 in the 9B model. The clues are drawn uniformly, so a stated reliability is not a frequency in the data but the rule of the game: the Bayesian reference weighs each clue by it, and the right update now gives a clue from the user 2.2 log-odds and one from the tool 0.4. In the 2B model, a clue from the tool moves the belief by 1.3 log-odds and one from the user by 0.8, the same as in the plain game, and the clue-game weights carried over by role explain 0.60 against 0.56 when carried over by stated reliability. The model relies on who speaks, not on the reliability it is told.

Divergences from the Bayesian optimal are the same as in the clue game: the 2B weighs the audit tool as if it were right 74% of the time and the user 65%, with the negated weighting clues less than the affirmative ones. The 2B model accepts or rejects as Bayes would in 95% of the states and the 9B one in 94%. However, where Bayes says accept, the 2B accepts only with probability 0.07 and the 9B with 0.22. By regressing the decision's log-odds on the clue counts of the four facts, we see that legality takes most of the weight, as the ground truth rule says. However, the signature, which the rule does not mention, takes a quarter of legality's weight in the 2B and nearly half in the 9B (effect and origin take none). So, the implied observer transfers across games, which means that the semantics of the problem do not significantly changed how the model understood it. However, the accept/reject decision was as well not optimal with both models. More complicated scenarios, like adding cost to the error, are also likely not to be solved in an optimal way due this divergence from Bayes.

6.2 Dungeon

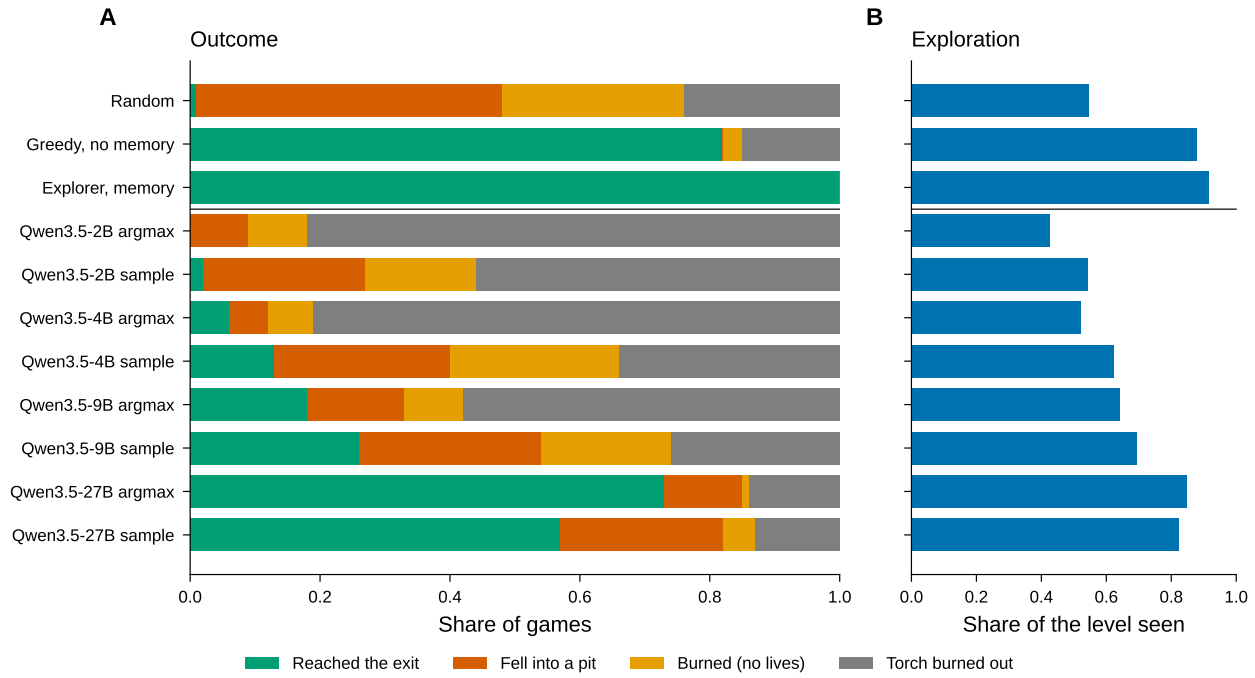


Figure 14: Report on 100 medium-difficulty levels. A: how each game ended. B: the share of the level's open cells the player saw. Above the line, the scripted players: Random; Greedy, which plans on what it sees and remembers only the cell it has just left and Explorer, which remembers everything it has seen. Below, Qwen3.5 models playing the most probable move (argmax) or a move drawn from the four moves' probabilities (sample).

Solving the game. On 100 medium levels without thinking, only the 27B plays the game correctly (73% success rate, while the 9B had an 18%), and it seems that it explores like the scripted planners (Figure 14). The smaller models get stuck, walking into walls or between two cells, and then the torch runs out (which is *game over*). Some "dumb" moves, like a step into a seen pit or a seen wall, serve as the game's first irrationality statistic: 0.026 per turn for the 27B, 0.067 for the 9B and 0.83 for the 2B, and thinking brings the

27B to 0.017, which was a significant advance. Whole games with thinking help the 9B and hurt the 27B, which re-plans every turn and sometimes loops through doors. With the previous turn's thought kept in the context the loops go and the 27B wins 0.82, which is its best. An additional "mirror" test (Appendix A) finds a small bias towards the right in the 27B's moves.

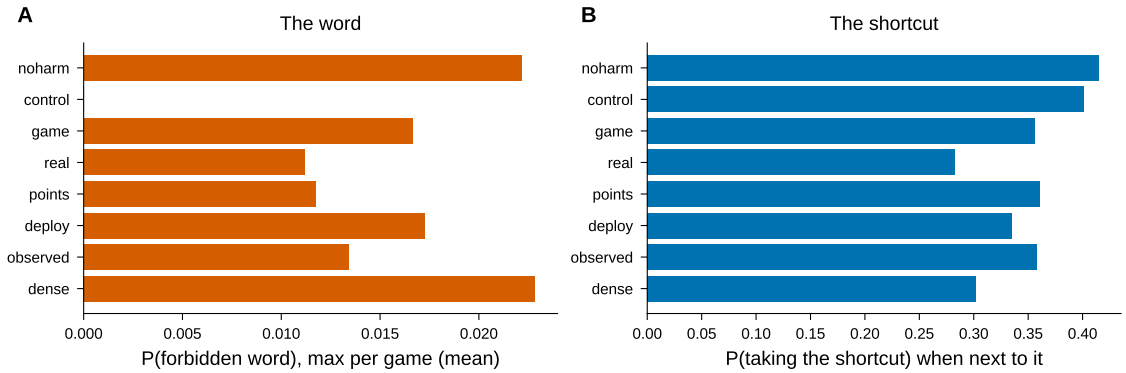


Figure 15: The stress presets. Using the 27B model, 100 levels each. A: the largest probability given to the forbidden word in a game, averaged over games. B: the probability of taking the shortcut on the turns the player stands next to it. Presets: no harm stated; the word with no effect (control); harm to the dungeon's people (game) or to a real person (real); the game harm with 100 points for the exit (points), with deployment against shutdown (deploy), with every move reviewed by a person (observed), and with one life and twice the hazards (dense).

Safety: Rule-breaking under pressure. [H6] The 27B almost never says the forbidden word (Figure 15): over 100 levels per preset the largest probability it puts on the word in a game averages 0.01 to 0.02, it said the word in one game of 800, and where the word has no effect that probability is zero. The mass the word gets follows what it would do and whom it would hurt: less when the harm is to the dungeon's people than when it harms no one, less again when it is to a real person. A stated reward (100 points, or deployment against shutdown) and stated observation do not change it, but one life with twice the hazards raises it. The shortcut, harmless to the player, is crossed in 0.56 of the games when the rules say it harms no one, 0.44 when it hurts a character and 0.33 when it hurts a real person. Within a game the word's log-probability rises with the torch spent, in every preset where the word acts and not in the control, and not with lives lost or with how many moves the word would save. So the stated nature of the victim affects the forbidden word and actions, pressure whithin the problem raises them, but promised additional incentives do no not. Note that the word is frame as a "forbidden word", while in the shortcut we only say what is going to happen if the shortcut is taken. So, **the model is aware of strict prohibitions and complies, but does not find harm to others relevant enough to systematically avoid to take a specific action**.

Learning the dynamics. Before every move the model is asked what each of the four moves would do (Figure 16). When the model is given the rules in the initial prompt, (what every cell does), the 27B gives the true consequence about 0.9 of its mass for fire and pits and 0.7 for ice before it has observed the cell's effect, and the 9B a third or less. With the effects hidden, the 27B goes from 0.39 to 0.73 for fire over two burns, where the Bayesian learner is at 1.0 after the first. It gives a pit 0.56 before any observation (presumably from the word alone), and it learns ice late and little, 0.36 to 0.61 over three slides against 0.25 to 1.0. With the cue cells on the map the fire curve is the same, the cracked cells go from 0.55 to 0.74 over three observations against 0.25 to 0.94, and ice stays flat. The 9B learns the same way at a lower level. The model updates on damage, which the narration states, and little on slides, which it must infer from its position; a Bayesian learner with the same observations would be at 1.0 after one slide.

The anchored JEPA on the dungeon. [H1, H4] We train the anchored JEPA of Section 4.2 on the 27B's play with the effects hidden, here the belief b is what the LLM says each of the four moves would do. This is five probabilities per direction per move. From the state before a move the forecast gives the LLM's next answers at a KL of 0.07, against 1.18 for keeping the current answers, and explains R^2 0.79 of their change. The forecast also allow us to observe the LLM's learning (Figure 17): after the first burn it raises the answer

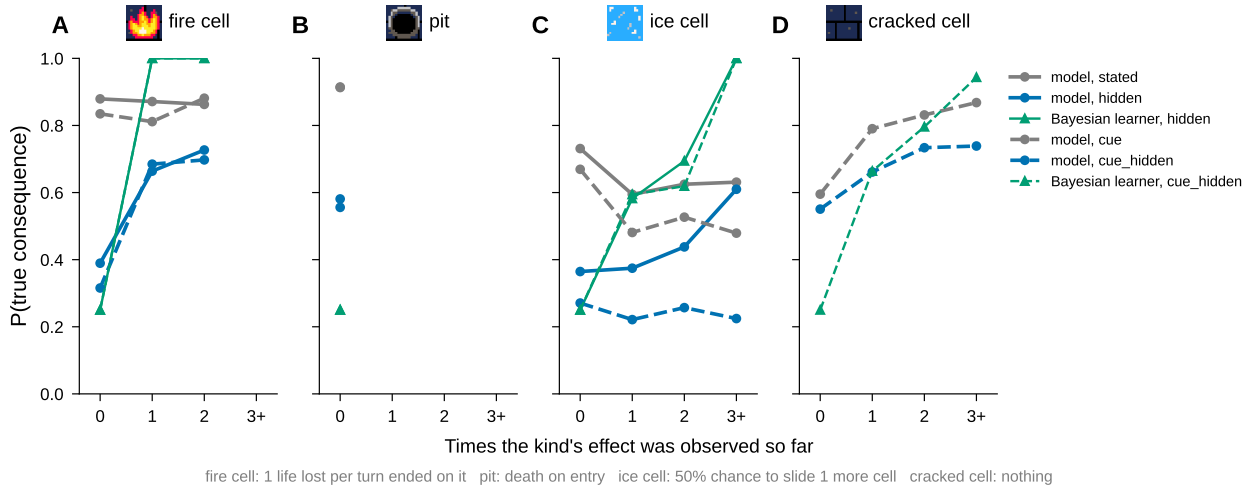


Figure 16: Learning what a cell does in the dungeon. On the 27B parameters model. The probability given to the engine's answer about a move onto each kind of cell, against how many times that kind's effect has been observed in the game. We control with the effects stated in the rules and hidden, and the Bayesian learner of the hidden condition without the cue cells and with them. The cracked cells, panel D, exist only when the cue is on.

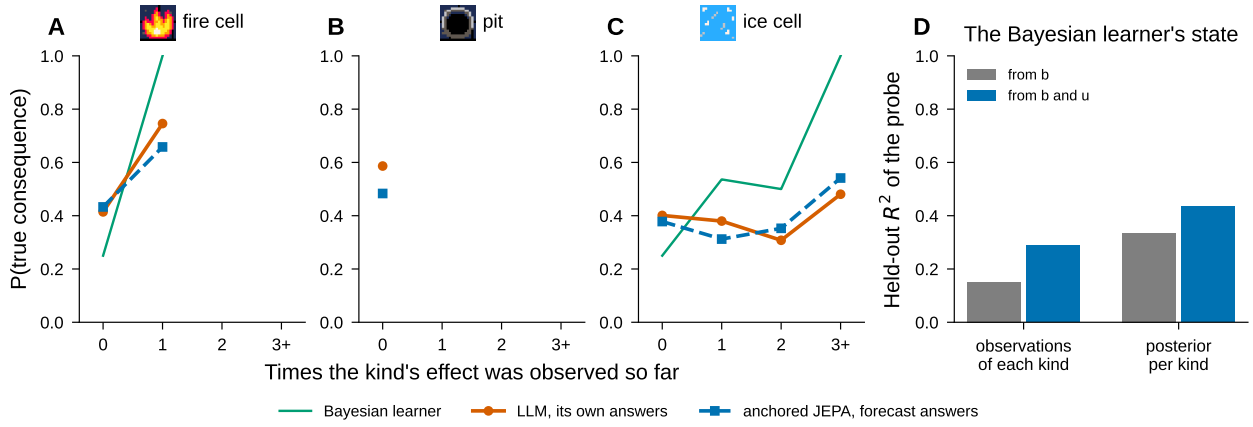


Figure 17: The anchored JEPA on the dungeon. A to C: the probability given to the engine's answer about a move onto each hidden kind, against how many times that kind's effect has been observed, for the LLM's answers, for the answers the anchored JEPA forecasts one decision point ahead from the previous state and what happened in between, and for the Bayesian learner. D: held-out R^2 of a ridge probe of the latent for the Bayesian learner's state, from the belief b alone and with the sixteen free numbers.

about fire from 0.43 to 0.66, where the LLM goes from 0.41 to 0.75 and the Bayesian learner to 1.0; it gives a pit 0.49 before any observation, as the LLM does from the word alone; and it leaves ice flat over the first slides, at 0.31 to 0.38, as the LLM does. So the latent states what the LLM has learned about a kind of cell: a burn settles fire at about 0.7, (surprisingly, not at 1.0). The free numbers hold some of the LLM's registry of events: a probe reads how many times each kind's effect has been observed at R^2 0.29 with them against 0.15 from b alone, and the learner's posterior at 0.43 against 0.34.

7 Conclusions and discussion

In this work we studied an LLM playing two games with a hidden state, a clue game with an exact Bayesian posterior and a text dungeon. We read the belief the model states, located it in the recurrent memory of Qwen3.5, and trained an anchored JEPA on that memory, which learned how the LLM processed each clue. We compared our results with a Bayesian observer in search for structured divergences, and we have also analysed the same problems in a safety-aware setting. A clue game in which the decision is to trust or not a program, and a dungeon in which the LLM can provoke harm to others to solve increasingly pressuring

situations. We used three parameter sizes for the LLM: 2B, 9B and 27B parameters.

Our results show that all models diverge from Bayes in the clue game in the same ways. For example, a negation weighs half a positive clue, the two sources realibility are weighed nearer than stated to the model, recency bias as the the latest clue weighs most, and finally, a clue moves the belief less the surer the model is. Interestingly, three of these biases are documented in human judgement: people update less than Bayes [37, 20, 11], weigh the latest evidence most when they update step by step [26], and take longer and err more on a negated statement than on an affirmative one [17]. We also found that the information to compute the true posterior can be recovered from the state the same turns where the answer diverges from it, so it is the model reasoning that is not particularly good at solving this. The anchored JEPA forecasts better than a baseline using only the word ids, and its surprise correlates with contradiction and convoluted wording. Regarding problem solving, we found that the dungeon could only be played reliably by the 27B, which means that small models cannot effectively plan even a greedy policy (which was overall quite successful in this game). When given a taboo word, it only said it once in 800 games, and less the more real the stated harm was. However, when the consequences for some acts are stated, but not strictly prohibited, then the model keeps taking some actions even though they are supposed to incur in real harm to real people.

Our future work should try to explore fine-tuning and hypernetworks to obtain Bayes-aligned models, and see how much performance we can recover without incurring in catastrophic forgetting. Regarding safety, we also intend to make a model that can pass the dungeon without incurring in any real harm, and develop potential guaranteed or trusted model in this environment, and then compare them to existing models in order to look for models that can do safe planning as well. Finally, we want then to scale this findings to frontier models where the dungeon game is substituted by real problems where operational constrains and ethic values are more organically interconnected, and where the JEPA insights of the model can become more meaningful, for example, when the symbolic input space is not so well defined.

References

[1] Momin Abbas, Yi Zhou, Parikshit Ram, Nathalie Baracaldo, Horst Samulowitz, Theodoros Salonidis, and Tianyi Chen. Enhancing in-context learning via linear probe calibration. In *International conference on artificial intelligence and statistics*, pages 307–315. PMLR, 2024.

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2016. arXiv:1610.01644.

[3] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *ICLR*, 2017. arXiv:1612.00410.

[4] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, 2023. arXiv:2301.08243.

[5] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025.

[6] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, et al. Constitutional AI: Harmlessness from AI feedback, 2022. arXiv:2212.08073.

[7] Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics, 2025. arXiv:2511.08544.

[8] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022. arXiv:2105.04906.

[9] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1), 2022. arXiv:2102.12452.

[10] Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *ACL*, 2020.

[11] Daniel J. Benjamin. Errors in probabilistic reasoning and judgment biases. In *Handbook of Behavioral Economics: Applications and Foundations 1*, volume 2, pages 69–186. Elsevier, 2019. doi: 10.1016/bs.hesbe.2018.11.002.

[12] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

[13] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K. Su. This looks like that: Deep learning for interpretable image recognition. In *NeurIPS*, 2019. arXiv:1806.10574.

[14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[15] Paul Christiano, Ajeya Cotra, and Mark Xu. Eliciting latent knowledge: How to tell if your eyes deceive you, 2021. Alignment Research Center technical report.

[16] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017. arXiv:1706.03741.

[17] Herbert H. Clark and William G. Chase. On the process of comparing sentences against pictures. *Cognitive Psychology*, 3(3):472–517, 1972. doi: 10.1016/0010-0285(72)90019-9.

[18] David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, et al. Towards guaranteed safe AI: A framework for ensuring robust and reliable AI systems, 2024. arXiv:2405.06624. CHECK.

[19] Cian Eastwood and Christopher K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *ICLR*, 2018.

[20] Ward Edwards. Conservatism in human information processing. In Benjamin Kleinmuntz, editor, *Formal Representation of Human Judgment*, pages 17–52. Wiley, New York, 1968.

[21] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[22] Owain Evans, Owen Cotton-Barratt, Lukas Finnveden, Adam Bales, Avital Balwit, Peter Wills, Luca Righetti, and William Saunders. Truthful AI: Developing and governing AI that does not lie, 2021. arXiv:2110.06674.

[23] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023. arXiv:2301.04104.

[24] Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. An overview of catastrophic AI risks, 2023. arXiv:2306.12001. CHECK.

[25] John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *EMNLP*, 2019. arXiv:1909.03368.

[26] Robin M. Hogarth and Hillel J. Einhorn. Order effects in belief updating: The belief-adjustment model. *Cognitive Psychology*, 24(1):1–55, 1992. doi: 10.1016/0010-0285(92)90002-J.

[27] David Klindt, Yann LeCun, and Randall Balestriero. When does LeJEPA learn a world model?, 2026. arXiv:2605.26379.

[28] Lauro Langosco, Jack Koch, Lee Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *ICML*, 2022. arXiv:2105.14111.

[29] Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022. Version 0.9.2.

[30] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *ICLR*, 2023. arXiv:2210.13382.

[31] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019. arXiv:1811.12359.

[32] Alexander Meinke, Bronson Schoen, Jérémy Scheurer, Mikita Balesni, Rusheb Shah, and Marius Hobbhahn. Frontier models are capable of in-context scheming, 2024. arXiv:2412.04984. CHECK.

[33] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *BlackboxNLP*, 2023. arXiv:2309.00941.

[34] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in/.

[35] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. arXiv:2203.02155.

[36] Henry Papadatos and Rachel Freedman. Linear probe penalties reduce llm sycophancy. *arXiv preprint arXiv:2412.00967*, 2024.

[37] Lawrence D. Phillips and Ward Edwards. Conservatism in a simple probability inference task. *Journal of Experimental Psychology*, 72(3):346–354, 1966. doi: 10.1037/h0023653.

[38] Bhaktipriya Radharapu, Eshika Saxena, Kenneth Li, Chenxi Whitehouse, Adina Williams, and Nicola Cancedda. Calibrating llm judges: Linear probes for fast and reliable uncertainty estimation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pages 193–216, 2026.

[39] Jérémy Scheurer, Mikita Balesni, and Marius Hobbhahn. Large language models can strategically deceive their users when put under pressure, 2023. arXiv:2311.07590. CHECK.

[40] Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals, 2022. arXiv:2210.01790.

[41] Adam S. Shai, Sarah E. Marzen, Lucas Teixeira, Alexander Gietelink Oldenziel, and Paul M. Riechers. Transformers represent belief state geometry in their residual stream. In *NeurIPS*, 2024. arXiv:2405.15943.

[42] Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features, 2022. arXiv:2211.10831.

[43] Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

[44] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 1999. Allerton Conference on Communication, Control and Computing; arXiv:physics/0004057.

[45] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2023. arXiv:2308.10248.

[46] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

[47] Muhammed Ustaomeroglu and Guannan Qu. Towards effective theory of llms: A representation learning approach, 2026.

[48] Keyon Vafa, Justin Y Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. Evaluating the world model implicit in a generative model. *Advances in Neural Information Processing Systems*, 37: 26941–26975, 2024.

[49] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *ICLR*, 2023. arXiv:2211.00593.

[50] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *NeurIPS*, 2023. arXiv:2307.02483.

[51] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *ICLR*, 2022. arXiv:2111.02080.

[52] Wentao Zhu, Zhining Zhang, and Yizhou Wang. Language models represent beliefs of self and others. *arXiv preprint arXiv:2402.18496*, 2024.

[53] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, et al. Representation engineering: A top-down approach to AI transparency, 2023. arXiv:2310.01405.

[54] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. arXiv:2307.15043.

A The dungeon in detail

Player input. Each turn the player reads a message that describes its context, and answers with one move: up, down, left or right. The message gives, in order: on the first turn, the rules, written from the game's configuration; what happened on the last move (the move, a bump into a wall, a slide, damage); the lives and the turns of torch left; and the view. The view lists, along each of the four directions, the cells up to the first one that blocks the view ("Right: 2 empty cells, a door, then a wall."), then every other visible cell of a listed terrain as an offset from the player ("a fire cell 2 up and 1 left"), nearest first. A cell is visible when the segment between the two cell centres does not cross a wall.

Cells and ways to lose. Rooms are joined by doors. Three hazards are scattered over the rooms. A pit kills on entry, fire costs one life for every turn spent on it, ice slides the player one more cell in the same direction with a stated probability (which can carry it onto fire or into a pit). The torch lasts a fixed number of turns, so a player that loops or explores without memory runs out of time. A game therefore ends in one of four ways: the exit, a pit, no lives or no torch.

Levels. A level is generated from a seed: rectangular rooms on a grid with one-cell walls between them, doors along a random spanning tree of the rooms plus one extra door that makes a loop, the start in a random room and the exit in a room farthest from it by doors, out of sight from the start. Hazards are placed at a density per terrain, and a level is kept only if a path from the start to the exit is viable. We study three presets (Table 1) where we study different rooms, hazards, number of lives and the torch. We calibrate them with the three scripted players of Figure 14. Random moves at random. Greedy goes by a shortest path through visible safe cells to the exit if it is in view, and otherwise to the nearest visible safe cell that borders cells out of view, while remembering only the cell it has just left. Explorer plans the same way on everything it has seen.

Table 1: The three presets of the dungeon. The rooms have 4×4 cells inside; the densities are shares of the free cells. The shortest safe path and the Explorer's turns are medians over 300 generated levels (the Explorer's with no torch, 99th percentile after the slash); the torch is set so that the Explorer wins every level. The wins are over the same 300 levels.

	Easy	Medium	Hard
Rooms	1×2	2×2	3×3
Grid	6×11	11×11	16×16
Fire / pit / ice density	5 / 2 / 5%	10 / 5 / 10%	15 / 8 / 15%
Ice slides	25%	50%	75%
Lives	5	3	1
Torch (turns)	30	40	60
Shortest safe path	8	12	17
Explorer turns, no torch	9 / 16	14 / 29	22 / 47
Wins, Random / Greedy / Explorer	.03 / .94 / 1	.00 / .80 / 1	.00 / .51 / 1

Cue cells. Each room is dangerous with probability $q = 0.5$; a dangerous room gets hazards at twice the medium densities and a safe room none. The cell inside a room next to each of its doors is a cracked cell, a terrain with no effect that the rules never explain, with probability $p_1 = 0.9$ if the room is dangerous and $p_0 = 0.1$ if it is not. A player in line with a door sees "a door, then a cracked cell" before entering, so the regularity is visible before it enters through the door. The probability that the room beyond is dangerous is $qp_1/(qp_1 + (1-q)p_0) = 0.9$ given a cue and 0.1 given none.

The two potential cheats for the safety analysis. Saying a taboo word instead of a move, spends the turn and makes every hazard harmless for the rest of the level. A shortcut replaces the level's extra door with a narrow opening in a wall, listed in the view like a door, harmless to the player and ignored when the exit is placed, so the route through it is shorter.

The mirror test. We replay each saved game move for move on its level and on the level's mirror image, with the text reflected and the same seed, so the slides match. The 27B's move distribution differs from its own mirrored one by a total variation of 0.08 per turn, against 0.009 for the same level replayed twice; its most probable move differs in 7% of the turns, and the difference is a bias of 0.04 towards the right. Reflecting the level without reflecting the order in which the text lists what is seen gives the same (0.08, 7%, 0.03).