

MiyaDreamBelief: A Dreamer-Based Automated Negotiation Agent for ANL 2026

Taisuke Miyashita, Team 421

Abstract

MiyaDreamBelief is a Dreamer-style model-based reinforcement learning agent designed for the sequential negotiation setting of ANL 2026. The agent combines a latent world model for predicting negotiation dynamics, Opponent Strategy Identification (OSI) for estimating the opponent’s preferences, a utility-band policy for narrowing offer candidates, and a safe heuristic fallback policy. During training, the agent was exposed to ANL-style bilateral negotiations with randomized domains, utility functions, reservation values, negotiation order, and opponent types. It was further trained through multiple rounds of self-play against earlier versions of itself. At submission time, the agent uses the learned Dreamer policy whenever the domain fits the fixed neural representation. If the model cannot be used, or if the domain exceeds the learned size limits, the agent automatically switches to a frequency-based fallback policy. This design aims to combine the performance of a learned policy with the robustness required in the submission environment.

1 Introduction

In ANL-style negotiation, agents are evaluated not only by the utility of the agreement they obtain, but also by how accurately they estimate the opponent’s utility function and how well they conceal their own. Therefore, an effective agent must both choose actions that lead to high-utility agreements and build an online model of the opponent’s preferences. MiyaDreamBelief addresses this problem by combining a Dreamer-style latent world model with an explicit opponent preference estimator.

At the same time, neural policies depend on the tensor shapes used during training. If a submitted agent encounters a domain whose size was not covered by the learned representation, direct neural inference may become unsafe. For this reason, MiyaDreamBelief includes a fallback policy that operates independently of the Dreamer model. When Dreamer is available, the agent acts with the learned policy. If model loading fails, a runtime error occurs, or the domain exceeds the fixed-size representation, the agent switches to the heuristic fallback.

2 Overall Architecture

MiyaDreamBelief lazily loads a compact checkpoint stored in the submission package and interacts with the environment through the ANL SAO interface. At each decision point, the wrapper converts the current negotiation state into the fixed-length tensor format used during training, queries the Dreamer policy, and converts the resulting action into either an acceptance decision or a counter-offer.

The Dreamer policy uses a fixed representation with at most six issues and at most five values per issue. All valid outcomes in this fixed space are encoded as action identifiers, with one additional action reserved for acceptance. If the current domain has more than six issues, or if any issue has more than five values, the Dreamer observation construction is aborted and the agent falls back to the heuristic negotiator. This keeps the neural model compact while preserving safe behavior on out-of-range domains.

3 Dreamer World Model

The learned agent follows the Dreamer architecture and consists of an encoder, a recurrent state-space model (RSSM), a decoder, reward and continuation heads, policy heads, and value heads. The observation includes relative time, the last opponent offer, the last self offer, the agent’s own utility table, its reservation value, valid issue and value masks, valid proposal masks, and offer frequency information updated during negotiation.

The RSSM maintains a latent state composed of a deterministic state and a categorical stochastic state. In the submitted version, the RSSM and the heads were compressed to reduce checkpoint size and inference cost. The stochastic state is not passed directly to all heads. Instead, it is first compressed by a dedicated linear layer and then fused with the deterministic state through a branched head fusion module. This separates the roles of stochastic belief and deterministic history before downstream decision making.

4 Opponent Preference Estimation

Opponent preferences are estimated by a Gaussian OSI module. The module outputs means and standard deviations for opponent issue weights, value utilities, and reservation value. The issue weights and value utilities used by the policy are a mixture of a frequency-based prior and the neural prediction. The frequency prior is computed from the values that the opponent has offered so far. The neural prediction is trusted more strongly when the uncertainty predicted by OSI is small. The mixing coefficient is computed from a stop-gradient version of the predicted standard deviation so that the mixture itself does not create unstable gradients.

The OSI training objective uses the negative log-likelihood of the raw neural prediction as the main loss and adds a smaller auxiliary loss on the mixed prediction. This encourages the neural estimator itself to learn the true preferences directly, while the values actually used by the policy and reward calculation remain stabilized by frequency information. Gradients from the OSI loss to the RSSM are stopped so that opponent preference learning does not destabilize the world-model representation.

5 Policy

The policy does not assign an independent logit to every possible offer. Instead, it first narrows the candidate set through a utility band. A target utility policy predicts a target utility ρ , an upper width Δ_{up} , a lower width Δ_{down} , and a maximum allowed utility u_{max} . The lower width is constrained to $[0, 0.1]$, and the candidate range is conceptually

$$[\rho - \Delta_{\text{down}}, \rho + \Delta_{\text{up}}].$$

Valid offers whose self utility lies in this band are selected as candidates. The offer policy then scores each candidate from its candidate token, the shared head representation, ρ , Δ_{down} , Δ_{up} , and u_{max} . A separate accept policy outputs the logit for accepting the current opponent offer.

The raw and transformed target information produced by the target policy is attached to the action and provided to the action encoder and encoder input at the next step. This allows the RSSM to know which target utility band was used in the previous decision. The value function is also split into target-side and offer-side critics. During imagination, the target policy and offer policy use their corresponding value and slow-value networks to compute lambda returns and advantages.

6 Reward Design and Evaluation

The reward was designed to align with the ANL score:

$$\text{Score} = \text{Advantage} + \text{Concealing}.$$

The advantage term is based on the utility of the final agreement relative to the agent’s reservation value. The concealing term compares the quality of the agent’s opponent model with the quality of the opponent’s inferred model of the agent. Preference-estimation quality is measured by a Kendall-tau-style ranking agreement over the outcome space.

Computing Kendall tau inside JAX at every training update was too expensive. Therefore, the final implementation passes the Dreamer opponent-utility estimate through the action to the environment. At terminal states, the environment computes an ANL-style score using the submitted estimate and stores that reward in replay. Non-terminal rewards are essentially zero, while the terminal transition stores advantage plus concealing. The reward head is then trained against the reward stored for the corresponding replay step.

7 Self-Play

Training was performed as iterative self-play over multiple rounds. At the end of each round, the best checkpoint from that round was added to a self-play pool. The pool keeps the best checkpoint overall and a small set of recent checkpoints, and these frozen agents are sampled as opponents in later rounds. This exposes the current agent not only to fixed heuristic opponents, but also to policies close to its own learned behavior.

The opponent set included time-dependent negotiators, MAP/BOA-style agents, frequency-model heuristic agents, and Dreamer agents from earlier checkpoints. Training domains were randomized over the number of issues, number of values, utility tables, reservation values, and negotiation order. This randomization was intended to reduce overfitting to any single fixed scenario.

8 Fallback Policy

The fallback policy handles arbitrary outcome spaces without using Dreamer. During initialization, it caches candidate outcomes and their self utilities so that the utility function does not need to be reevaluated over the full outcome space at every proposal. During negotiation, it estimates opponent utility from the frequency of values in the opponent’s offers and selects candidates near a time-dependent target utility. The acceptance rule compares the current opponent offer with both the current target and the counter-offer that the agent would otherwise propose.

The fallback target utility includes a mild wave-like variation so that the policy is not purely monotonic concession. This makes the fallback less rigid while keeping it simple and robust when Dreamer cannot be used.

9 Submission Robustness

In the submitted package, the Dreamer checkpoint is pruned and stored as Python arrays. The model is loaded only when it is first needed. If loading fails, if a dependency is unavailable, if the domain exceeds the fixed neural size, or if a runtime exception occurs during Dreamer inference, the agent automatically switches to the fallback policy.

Before submission, we verified that Dreamer loads successfully on standard domains, that negotiations against MAPNeg complete in both negotiation orders, and that artificial out-of-range domains with six values for one issue or seven issues trigger fallback and still complete negotiation. These tests check both the learned behavior and the safety path used in exceptional domains.

10 Conclusion

MiyaDreamBelief integrates a Dreamer-style world model, Gaussian/Frequency OSI, utility-band candidate selection, dual-value actor-critic learning, and a safe fallback negotiator. Within the fixed-size representation, it uses the learned Dreamer policy to adapt to the opponent. Outside that range, it falls back to a frequency-based heuristic policy. This combination aims to provide both the expressiveness of a learned negotiation policy and the robustness required for submission.