POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

An LLM-Driven Agent for Competitive Pokémon Battling with Bayesian Opponent Modeling

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Alessio De Vito

Advisor: Prof. Alberto Marchesi

Co-advisors: Gianmarco Genalti, Pierriccardo Olivieri

Academic year: 2025-2026

Abstract

Competitive Pokémon couples the long horizons of board games with the hidden information and stochasticity of card games, layered over an encyclopedic set of species, moves, abilities, and items, which makes it a demanding testbed for reasoning under uncertainty. Existing turn-by-turn agents based on large language models (LLMs) remain limited by token-heavy prompts, weak opponent modeling, and brittle hand-crafted heuristics. This work presents *PolimiBot*, an LLM-driven agent for Generation 9 OverUsed singles on Pokémon Showdown, together with a *Bayesian Predictor* that infers the hidden opponent state from statistical models trained on roughly 140,000 historical teams. PolimiBot encodes the current battle state exhaustively, grounds the model in exact turns-to-KO figures from a canonical damage calculator, and decomposes each turn into a three-prompt flow adjudicated by objective numbers. The predictor fuses replay frequencies with metagame usage statistics, models team composition with a latent-archetype topic model, and conditions on in-battle evidence. Against a heuristic baseline and the strongest published LLM agent (PokéChamp), PolimiBot wins 89% and 68% of games respectively, while cutting prompt size by 19%; the predictor's confidence scores are shown to be well calibrated.

Key-words: competitive Pokémon; large language models; game-playing agents; Bayesian opponent modeling

1. Introduction

Reinforcement learning has produced superhuman play in chess, Go, StarCraft II, and other games, but each such system demands heavy, task-specific training and transfers poorly beyond the game it was built for. Large language models (LLMs) sit at the opposite end of the trade-off: they are generalists that carry broad prior knowledge and adapt to new situations without retraining, yet they have well-documented weaknesses in planning and can lose to simple heuristics. This work asks whether the generalist knowledge of an LLM can be harnessed, with the right scaffolding, to play a genuinely hard game well.

Competitive Pokémon is an unusually discriminating choice of game. A single match is two linked problems — assembling a team of six beforehand, then piloting it in battle — and it brings together properties usually studied sep-

arately: long decision horizons, simultaneous moves, hidden information, and stochastic transitions. The combinatorial scale is extreme: the number of legal six-Pokémon configurations exceeds 10^{139}, the per-turn branching factor is on the order of 10^5, and a player's uncertainty about the opponent's hidden team spans an information set of roughly 10^{58} states. Exhaustive search is therefore impossible, and no fixed table of heuristics can anticipate the resulting diversity of situations.

Hidden information is central to this difficulty. At the start of a battle each side sees only the species on the opposing team; every move, item, ability, defensive investment, and Tera type stays concealed until revealed in play. Strong human players close this gap by inference, reading likely sets from the metagame and narrowing the possibilities as evidence accumulates. Reproducing this faculty is one of the two problems addressed here.

This work develops an LLM-driven agent for Generation 9 OverUsed (OU) singles on Pokémon Showdown and makes two contributions. *PolimiBot* drives turn-by-turn battle decisions: rather than prepending a long transcript of past turns, it encodes the *current* state exhaustively, grounds the LLM in exact turns-to-KO figures from a canonical damage calculator, and decomposes each decision into a three-prompt flow. The *Bayesian Predictor* infers the hidden opponent information (unrevealed moves, items, abilities, Tera types, and bench Pokémon) from statistical models trained on roughly 140,000 Generation 9 OU teams, and injects its ranked predictions into PolimiBot's prompts so the agent reasons about a complete opponent from the first turn. Both components are designed as direct responses to concrete weaknesses of PokéChamp [4], the strongest existing LLM-based agent and the natural reference point alongside the offline-RL line of work [3].

2. Problem: weaknesses of the baseline

A close reading of PokéChamp exposes fifteen concrete weaknesses, grouped into four themes that each motivate an element of the design.

- **Prompt efficiency.** A raw sixteen-turn transcript is prepended to every prompt, inflating it to roughly 9,400 tokens and forcing the model to re-derive the current board from a log that is mostly already reflected in the present state.
- **Hidden opponent.** Unknown moves are filled from a static dictionary with no in-battle conditioning; unrevealed Pokémon are absent from the prompt entirely; item, ability, and Tera type are left empty until observed; and the damage the agent deals is never used to refine an estimate of the opponent's defensive investment.
- **Decision structure.** A single call conflates move reasoning and switch reasoning, which are structurally different problems, and the underlying minimax search compares uncalibrated 1–100 leaf scores *across separate LLM calls*, effectively comparing noise.
- **Numerical grounding.** Damage comes from an internal approximation that misses many interactions, speed uses only the base stat adjusted by stages, hazards are described in generic text with no per-switch HP cost, and bench Pokémon carry only a type multiplier rather than a KO estimate.

3. PolimiBot: the battle agent

Battle decisions in competitive Pokémon are complex enough to resist simple heuristics, yet structured enough to be described in natural language. PolimiBot therefore translates the full battle state into a prompt a capable model can reason over, under two principles.

Ground the model in exact numbers. Left to its own priors, an LLM reasons about mechanics from imperfect memory, and a single wrong KO threshold or speed verdict translates into catastrophically wrong play. PolimiBot computes every decision-relevant quantity offline and injects it directly: exact turns-to-KO from the community-standard `@smogon/calc` damage calculator (invoked through Node.js, so the agent inherits the full Generation 9 mechanics), effective HP after entry hazards, the complete speed-modifier chain, and an explicit verdict of who moves first. The model is asked to reason about *strategy*, not to re-derive *arithmetic*.

Decompose, then adjudicate with numbers. Each turn is routed through special-

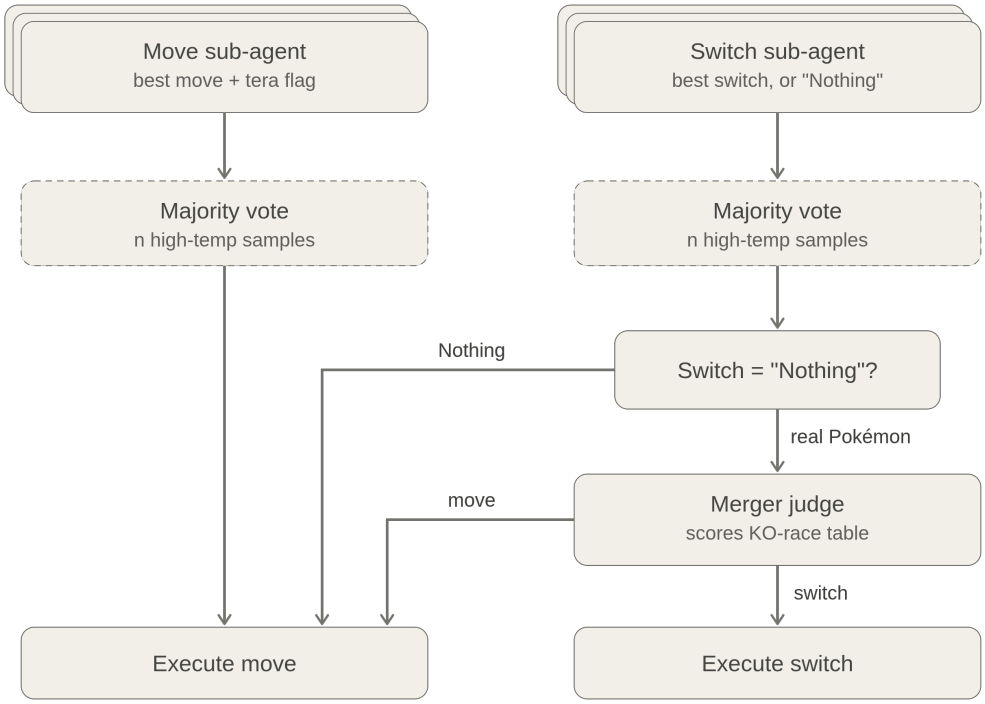

Figure 1: PolimiBot's three-prompt decision flow with majority voting. The Move and Switch sub-agents reason over the same exhaustively encoded state; the Merger Judge adjudicates between them over a pre-computed KO-race table. The Switch Sub-Agent can short-circuit the Merger by returning "Nothing".

ized sub-agents whose conflict is resolved by a judge anchored to a pre-computed KO-race table (Figure 1). The *Move Sub-Agent* selects the best move and decides whether to Terastallize; the *Switch Sub-Agent* evaluates whether to switch out, returning either a Pokémon or the value `"Nothing"` that short-circuits the flow; and when a real switch is named, the *Merger Judge* chooses move-versus-switch over objective KO-turn numbers, applying a conservative bias because switching grants the opponent a free attack. Unlike the baseline's minimax leaves, there is no cross-call score comparison. Optional *majority voting* parallelizes each sub-agent at a raised temperature and takes the most-voted action, reducing the variance of any single response, and *damage-evidence tracking* records the observed HP drop the agent inflicts and passes it to the predictor as evidence. The agent is deliberately LLM-agnostic, working with any model exposed through a common interface.

Before each query, PolimiBot assembles structured information blocks that describe the *current* state exhaustively — system prompt with per-Pokémon strategic notes, side conditions and hazards, the two active Pokémon (each move annotated with its turns-to-KO), the bench on both sides, a Terastallization and speed block stating the full modifier chain and an explicit speed verdict, and an action menu of every legal move and switch with hazard-adjusted HP. For the opponent, any unknown field is filled in by the Bayesian Predictor.

4. The Bayesian Predictor

The predictor fills the opponent's hidden fields with statistically grounded estimates rather than guesses. It is trained on approximately 140,000 Generation 9 OU teams from the Metamon Teams dataset [2] and is organized as a hierarchy of statistical models combined by a team predictor.

Frequency tables and metagame prior. A set of per-species counters records how often each move, item, ability, nature, EV spread, and

Tera type was observed in the replay corpus, together with pairwise move co-occurrences. Raw counts are Dirichlet-smoothed (pseudo-count $\alpha = 0.5$, the Jeffreys prior) so that an unobserved combination still receives non-zero mass. These replay frequencies are blended, per species and component, with the official Smogon usage statistics by a convex combination whose weight adapts to how much replay evidence is available:

$$\lambda = \max\Big(0.1,\ 1 - \sqrt{\tfrac{c_\mathrm{r}}{c_\mathrm{r} + 0.05\, c_\mathrm{s}}}\Big)\,, \tag{1a}$$

$$\mathrm{fused}[k] = \lambda\, \mathrm{smogon}[k] + (1-\lambda)\, \mathrm{replay}[k], \tag{1b}$$

where c_r and c_s are the replay and Smogon observation counts. A species seen in many replays drives λ to its floor of 0.1 and trusts its own data; a rarely seen species lets λ rise toward 1 and leans on the metagame prior, avoiding overfitting to sparse data for niche Pokémon.

Team composition. Teams are not collections of independently chosen Pokémon but cluster into recognizable archetypes. The predictor captures these correlations with Latent Dirichlet Allocation [1]: each team is a six-word document over the species vocabulary, and 20 learned topics play the role of archetypes, recovering sun, rain, stall, and balance cores without supervision. At inference time the revealed Pokémon weight the topics and the model marginalizes to a distribution over the unrevealed slots, blended 70/30 with the base usage rate. This is the correct generative view: unlike the independent-co-occurrence estimate used by prior agents, which double-counts the correlations within a synergistic core and produces overconfident distributions, the archetype model marginalizes properly over unrevealed slots.

Evidence conditioning. The base distributions are sharpened during the battle: directly observed fields collapse to a point mass; revealed moves re-weight the remaining candidates by co-occurrence (in log-space); and the concrete damage the agent records is used for backward EV inference, down-weighting defensive spreads inconsistent with the observed damage. Predictions thus become sharper as the battle progresses. Because the model is trained offline, only fast inference runs at battle time, adding negligible latency.

5. Experimental results

All battles were run locally in Generation 9 OU singles. PolimiBot was evaluated against two baselines: *AbyssalPlayer*, a deterministic hand-crafted heuristic bot, and *PokéChamp* [4], the LLM-based minimax agent that motivated the design. Five reference teams of contrasting styles (hyper-offense, offense, balanced, control, stall) were each played against all five opposing teams, four games per matchup — 100 games per opponent.

Predictor accuracy and calibration. On held-out teams the redesigned predictor matches the previous naïve-Bayes baseline on species, move, and component prediction; the benchmark's ground truth is partially the old predictor's own output, so the comparison is best read as "roughly comparable on a biased benchmark" rather than a clean ablation, and the structural advantages of the LDA and Smogon-fusion design are argued to surface on truly player-authored teams. More important for its use in the agent, the predictor's confidence scores are *well calibrated*: a prediction it labels "confident" is correct 90.0% of the time, "likely" 47.4%, and "uncertain" 29.8%. The same monotonic relationship holds out of sample on 53,991 held-out predictions (Figure 2), so the predictions injected into the prompt are dependable when issued with high confidence. Evidence conditioning measurably sharpens the output: the mean entropy of the predicted move distribution falls from 3.28 to 2.09 bits as three moves are revealed.

Token efficiency. Because PolimiBot logs no history and encodes the current state directly, it averages 7,616.8 prompt tokens per turn against PokéChamp's 9,399.7 — a reduction of 19.0% that widens further in long games and translates directly into lower API cost.

Battle results. PolimiBot won 89 of 100 games against AbyssalPlayer (89.0%) and 68 of 100 against PokéChamp (68.0%), with no ties

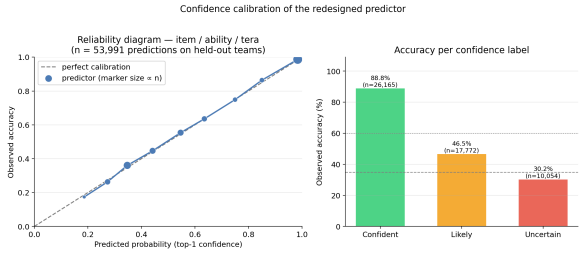

Figure 2: Calibration of the predictor on held-out teams. Left: reliability diagram of observed accuracy against predicted top-1 probability, tracking the diagonal of perfect calibration. Right: accuracy within each of the three confidence labels exposed to the agent.

Table 1: PolimiBot win rate by team against each baseline (20 games per team, 100 games per opponent).

Team	Style	vs. Abyssal	vs. PokéChamp
Team 1	Hyper-offense	100.0%	80.0%
Team 2	Offense	95.0%	70.0%
Team 3	Balanced	90.0%	65.0%
Team 4	Control	85.0%	65.0%
Team 5	Stall	75.0%	60.0%
Overall		**89.0%**	**68.0%**

(Table 1). The head-to-head result is corroborated indirectly: against the common heuristic opponent, PokéChamp wins 84% where PolimiBot wins 89%, so both the direct and indirect measurements place PolimiBot ahead. A consistent qualitative pattern is the monotonic decline in win rate from aggressive to defensive team styles, present against both baselines and *compressed* against PokéChamp (a 20-point spread versus 25). It reflects a genuine limitation of single-turn LLM reasoning: tactical, KO-driven decisions are easy to express in one prompt, whereas the long-horizon positional play that stall teams demand is not — a limitation shared by both LLM agents, so that the contest between them turns on per-turn execution, where PolimiBot's architecture has the advantage.

6. Conclusions

This work tested whether the generalist knowledge of a language model can be harnessed, with the right scaffolding, to play competitive Pokémon well, and answered the question by building two complementary contributions. PolimiBot replaces a token-heavy history transcript with an exhaustive current-state encoding, grounds the model in exact turns-to-KO figures and a full speed-modifier chain, and routes each turn through a three-prompt flow adjudicated by objective numbers. The Bayesian Predictor supplies the hidden-state beliefs the agent reasons over, fusing replay frequencies with metagame usage statistics, modeling team composition with a latent-archetype topic model, and conditioning on evidence — including the agent's own observed damage — revealed during play. Empirically, PolimiBot wins 89% of games against a heuristic baseline and 68% against PokéChamp while cutting mean prompt size by 19.0%, and the predictor's confidence scores prove well calibrated.

Four limitations bound these conclusions: the battle results are end-to-end and do not isolate any single design element; the predictor benchmark's ground truth is partially synthetic, so the expected generalization advantage remains argued rather than measured; the team-style gradient reflects a real limitation of single-turn reasoning on long horizons; and evaluation is expensive, coupling statistical power to token expenditure and capping the present study at four games per matchup. These point to clear future directions: a per-component ablation to apportion credit, evaluation on real player-authored teams and on the live ladder with confidence intervals, a curated set of tactical *puzzles* to replace noisy full games with decisive reproducible tests, reinforcement-learning fine-tuning of the backbone from the win/loss signal the game already provides, and — most fundamentally — extending the architecture with an explicit multi-turn plan so that long-horizon positional play becomes expressible.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] Jake Grigsby. Metamon teams. `https://huggingface.co/datasets/jakegrigsby/metamon-teams`, 2025. Hugging Face dataset; `gl_05_26` split.

[3] Jake Grigsby, Yuqi Xie, Justin Sasek, Steven

Zheng, and Yuke Zhu. Human-level competitive pokémon via scalable offline reinforcement learning with transformers, 2025.

[4] Seth Karten, Andy Luu Nguyen, and Chi Jin. Pokéchamp: an expert-level minimax language agent. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.