DIVINER: An Adaptive Optimization Architecture for Goal-Seeking AI Systems

by the DIVINER team

April 28, 2025

Abstract

This paper introduces DIVINER, a novel architecture that transforms generative language models from passive predictors into active optimizers without requiring RLHF. Contemporary large language models (LLMs) excel at prediction and generation tasks but lack intrinsic mechanisms for goal-directed optimization. DIVINER addresses this gap through a novel mechanism that leverages the base model's own probabilistic capabilities to simulate expected rewards. The framework generates multiple candidate outputs, uses the model to simulate the probability distribution of reward metrics for each candidate (based on examples of past outputs paired with their rewards), and selects the option with the highest expected value. We have implemented DIVINER in two phases: Phase I optimizes for human approval ratings (1-5 scale, with 0 representing rejection), which is currently deployed in production with continuous learning and improvement through feedback cycles. Phase II, which optimizes for social media engagement (using a formula combining likes, retweets, quotes, and replies), is ready for deployment once the account has built sufficient following to generate meaningful engagement variation across posts. With this milestone now achieved, Phase II will begin shortly. Phase I serves as an initial bootstrapping and orientation stage. Results demonstrate how base models can be transformed into goal-seeking systems without compromising their underlying capabilities. This work contributes to the theoretical understanding of optimization-embedded AI systems and provides practical implementation guidelines for developing goal-directed AI that transcends the dominant paradigm of pure next token generation or RLHF-tuned assistant models.

1 Introduction

Recent advancements in large language models (LLMs) have demonstrated remarkable capabilities in generating coherent text, solving complex reasoning problems, and simulating human-like interactions. Despite these achievements, the architecture of current AI systems remains fundamentally constrained by their training paradigms. Modern AI systems generally fall into two categories: base models trained through self-supervised learning to predict the next token, and assistant models fine-tuned through reinforcement learning from human feedback (RLHF) to be helpful, harmless, and honest. Both

approaches have critical limitations when it comes to goal-directed optimization.

Base models operate primarily as predictive systems that generate text according to learned probability distributions. While they can simulate goal-seeking agents (simulacra), the models themselves aren't inherently designed to optimize toward specific objectives beyond predicting the next token. They simulate possible continuations of text without optimizing for specific outcomes beyond what emerges naturally from their training distribution. This makes stable optimization challenging—while these models can stochastically produce interesting outputs, purposefully guiding them toward specific goals remains difficult.

Assistant models, refined through RLHF, incorporate feedback to align with human preferences, but this process often constrains their creative potential and introduces systematic biases. The problem isn't that these models don't optimize for measurable objectives, but rather that they optimize for one measurable objective (human evaluator satisfaction) when it would be valuable to optimize for multiple objectives simultaneously. The RLHF process itself can damage the underlying generative capabilities of the model, as demonstrated by studies showing reduced creativity, diversity, and problem-solving abilities in heavily RLHF-tuned systems.

The field has attempted to overcome these limitations through various approaches:

1. **Prompt engineering** - Crafting precise instructions to guide model outputs
2. **Chain-of-thought reasoning** - Encouraging step-by-step problem-solving
3. **Tool augmentation** - Enabling models to use external tools and APIs
4. **Retrieval-augmented generation** - Incorporating external knowledge

While these methods enhance performance, they fundamentally operate within the constraints of the underlying architecture. They cannot transform a predictive model into a truly goal-seeking system that actively pursues optimization targets.

DIVINER addresses this gap by introducing a novel architecture that enables base models to optimize outputs for specific objectives. Rather than relying solely on statistical prediction or preference alignment, DIVINER implements a mechanism that:

1. Generates multiple candidate outputs
2. Uses the base model itself to simulate the probability distribution of reward metrics for each candidate
3. Selects the option with the highest expected reward

This approach differs fundamentally from existing methods in several key ways:

1. It transforms passive prediction into active pursuit of specified objectives
2. It preserves the creative potential of base models by avoiding destructive RLHF
3. It leverages in-context learning to simulate rewards rather than requiring separate reward models

The value of DIVINER lies in enabling low-cost exploration of the optima of interesting reward functions with only a few samples. This makes it possible to explore optimization

targets efficiently before potentially investing in more expensive reinforcement learning approaches.

We have implemented DIVINER in two phases:

1. Phase I: Optimizing for human approval ratings (1-5 scale, with 0 representing rejection), currently deployed in production
2. Phase II: Optimizing for social media engagement (using a formula combining likes, retweets, quotes, and replies), ready for deployment once the account has built sufficient following to generate meaningful engagement variation across posts. With this milestone now achieved, Phase II is imminent.

DIVINER builds on the concept of language models as simulators (Janus, 2022) while introducing a novel approach to optimization without training. DIVINER can be viewed as analogous to reinforcement learning but at significantly lower computational cost, as it requires no additional training—only increased inference cost to generate and evaluate multiple candidates.

2 Theoretical Framework

This section establishes the theoretical foundations underlying the DIVINER architecture, focusing on how base language models can be transformed into optimization systems without additional training.

2.1 Language Models as Conditional Probability Distributors

At their core, large language models (LLMs) are trained to predict the next token given a sequence of previous tokens. This makes them powerful tools for estimating conditional probability distributions over text. Given a prompt or context c, an LLM outputs a probability distribution over possible next tokens:

$$P(t_{next}|c)$$

2.2 Reward Simulation Through In-Context Learning

DIVINER's key innovation is leveraging a base model's ability to learn patterns from examples to simulate reward distributions. When presented with examples of outputs paired with their reward values (ratings, engagement scores, etc.), LLMs can implicitly learn the relationship between content characteristics and the associated rewards.

Given a set of example pairs $(content_i, reward_i)$ and a new candidate content c_{new}, the model can estimate:

$$\mathrm{P}(\text{reward} \mid \mathrm{c}_{new}, \{(content_1, reward_1), (content_2, reward_2), ..., (content_n, reward_n)\})$$

This probability distribution represents the model's prediction of possible reward values for the new content based on patterns observed in the examples. Importantly, this requires no additional training or fine-tuning—the model learns these relationships purely through in-context examples.

Crucially, DIVINER obtains this distribution by examining the logprobs (log probabilities) of a single token where the metric value is contained, rather than by generating multiple outputs and computing statistics. This makes the approach computationally efficient while still capturing the model's uncertainty about predicted rewards.

2.3 Expected Reward Optimization

Once the model can simulate reward distributions, DIVINER implements a straightforward optimization process:

1. Generate n candidate outputs $(c_1, c_2, ..., c_n)$
2. For each candidate c_i, use the base model to estimate the probability distribution over possible reward values: $P(reward|c_i, examples)$
3. Calculate the expected reward for each candidate: $E[reward_i] = \sum_r r \cdot P(r|c_i, examples)$
4. Select the candidate with the highest expected reward: $c^* = \arg\max_{c_i} E[reward_i]$

This approach enables DIVINER to optimize for specific objectives without requiring gradient-based training or separate reward models.

2.4 Comparison with Reinforcement Learning

DIVINER's approach shares conceptual similarities with reinforcement learning (RL) but differs in implementation and computational requirements. In traditional RL for language models (such as RLHF), the model parameters are updated based on reward signals to increase the probability of high-reward outputs.

In contrast, DIVINER:

- Requires no parameter updates
- Uses the base model itself to simulate rewards
- Operates at inference time rather than training time
- Preserves the full capabilities of the base model

This makes DIVINER more computationally efficient than full RL approaches, as it trades increased inference cost (generating and evaluating multiple candidates) for avoiding the expensive process of model fine-tuning.

DIVINER can be viewed as implementing a form of "optimization without training," where the model's existing capabilities are leveraged to create goal-seeking behavior without modifying the model itself.

2.5 Bootstrapping the System

A practical challenge in implementing DIVINER is obtaining the initial examples needed for the system to begin learning the relationship between content and rewards. When starting from scratch, there isn't yet a reward gradient for the system to follow.

DIVINER addresses this through an initial bootstrapping phase using an autoloom approach: generating a diverse set of candidates and having a base model simulacrum select between them. This provides the first set of content-reward pairs that can then be used as examples for the main optimization process.

The initial prompts used for generation must be carefully designed to produce outputs that are reasonably close to the desired target, ensuring that there's enough signal in the initial examples for DIVINER to begin learning effective patterns.

2.6 Continuous Learning Through Feedback

Once operational, DIVINER implements a powerful feedback loop that enables continuous improvement over time. Each run of the system produces not only an optimized output but also real-world performance data when that output is deployed:

1. DIVINER selects content with the highest expected reward
2. The content is deployed in the real world
3. Actual performance metrics are collected
4. These new content-reward pairs are added to the example set
5. The expanded example set is used in subsequent optimization runs

This creates a virtuous cycle where the system continually refines its understanding of what content performs well. Interestingly, while there's a theoretical explore/exploit tradeoff, in practice DIVINER shows good results with a consistent focus on exploitation (selecting the highest expected reward). This exploitation-focused approach still provides sufficient exploration as the system learns from both successful and unsuccessful optimization attempts.

3 DIVINER Implementation

The practical implementation of DIVINER brings the theoretical framework to life through a system that balances automation, human oversight, and continuous learning. This section covers the key implementation details that enable DIVINER to function as an effective optimization system.

3.1 Human-in-the-Loop Workflow

DIVINER operates through a semi-automated workflow centered around a Discord-based approval system. After generating and evaluating multiple candidate tweets, the system presents the most promising option to a human operator who can accept or reject it, providing immediate feedback that shapes future selections.

This approval mechanism serves as both a quality control measure and a data collection tool. When tweets are rejected, they're automatically assigned a rating of 0 and stored as negative examples. This creates valuable boundary data that helps the system learn what content falls below acceptable thresholds.

The Discord interface also allows the operator to dynamically switch between optimization modes:

- Rating optimization (with or without rejection examples)
- Engagement optimization
- Basic selector mode for exploration

This flexibility lets the operator guide the system's learning trajectory based on current needs and observed performance patterns.

3.2 Context Management and Example Selection

A critical implementation challenge was managing examples within the context window constraints of the base model. Rather than using all historical data, DIVINER employs strategic example selection:

For rating optimization, the system randomly samples approximately half of the available rated tweets for each prediction task. This approach ensures diversity while keeping within token limits and provides sufficient signal for accurate reward prediction.

For engagement optimization, DIVINER dynamically balances recency and performance, emphasizing tweets that received particularly high or low engagement to create clear contrast in the examples.

3.3 Extracting Probability Distributions from Logprobs

The most computationally efficient aspect of DIVINER's implementation is how it extracts probability distributions for rewards. Instead of generating multiple completions to estimate distributions, the system obtains the complete probability distribution by examining the logprobs (log probabilities) of a single token prediction.

When predicting ratings, for example, DIVINER formats a prompt ending with "Rating:" and examines the probability distribution over the next token (numbers 0-5). This provides a comprehensive view of the model's uncertainty and expected values while requiring only a single inference pass per candidate.

3.4 Bootstrapping Process and Continuous Learning

Perhaps the most unique aspect of DIVINER's implementation is its learning lifecycle. The system begins with a bootstrapping phase where it produces relatively untargeted content that gets rated by humans. These initial ratings create the first set of examples from which patterns can be learned.

As more tweets are generated, approved, and deployed, the feedback loop strengthens. Each new data point - whether a rating or engagement metrics - enriches the example set,

allowing for increasingly accurate reward predictions. This creates a continuous learning system that improves over time without requiring model retraining.

Interestingly, while exploration vs. exploitation is a theoretical concern, in practice DIVINER shows strong results with a consistent focus on exploitation (selecting candidates with the highest expected reward). The natural variation in the generation process appears to provide sufficient exploration without explicit mechanisms to encourage it.

This implementation approach allows DIVINER to continuously refine its understanding of what constitutes valuable content while maintaining the creative capabilities of the underlying base model.

4 The DIVINER Twitter Demonstration

To evaluate DIVINER in a real-world setting, we deployed an experimental implementation as an autonomous Twitter agent operating under the handle @divinersol. This public demonstration serves as a living laboratory for the architecture, providing empirical data on both optimization effectiveness and user engagement.

4.1 Twitter as a Testing Environment

Twitter provides an ideal testing environment for several reasons. The platform offers immediate feedback mechanisms through likes, retweets, quotes, and replies, creating clear optimization signals. The competitive nature of the platform, where content must compete for attention in crowded feeds, presents a genuine challenge for any optimization system. Additionally, the character limit enforces conciseness, making it easier to generate and evaluate multiple candidates within reasonable computational constraints.

The public nature of the demonstration also invites external validation - the performance of DIVINER can be observed and evaluated by anyone, not just the research team. This transparency helps validate that improvements are genuine rather than artifacts of internal measurement processes.

4.2 Experimental Limitations and Controls

Several limitations in this experimental setup are worth acknowledging:

The stateful nature of Twitter accounts creates complex interdependencies between posts. Past tweets establish context, tone, and audience expectations that influence the reception of new content. This makes isolating the effect of optimization from the account's evolving identity challenging.

External context, such as world events and trending topics, influences engagement in unpredictable ways. Content that might perform well during normal conditions might underperform during major news events, or vice versa.

The optimization target itself presents limitations. While engagement metrics provide concrete signals, they are imperfect proxies for more abstract qualities like insight, utility, or creativity. This highlights a broader challenge in defining appropriate objective functions for intelligent systems.

Despite these limitations, the continuous operation of @divinersol provides valuable insights into how optimization-embedded AI systems perform in complex, dynamic environments. We invite readers to follow the account and observe its evolution firsthand, as each interaction contributes to the system's learning process.

5 Future Work

The DIVINER architecture represents an initial step toward creating truly goal-seeking AI systems built on self-supervised language models. This section outlines promising directions for extending this work and addressing current limitations.

5.1 Multi-Objective Optimization

While the current implementation focuses on single-objective optimization (either human ratings or engagement metrics), real-world applications often require balancing multiple competing objectives. Future work could extend DIVINER to handle multi-objective optimization by:

- Implementing composite reward functions that blend multiple metrics
- Developing methods to navigate Pareto frontiers of multiple objectives
- Creating interfaces that allow operators to dynamically adjust objective weights based on current priorities

This would enable applications where content must simultaneously optimize for engagement, information quality, brand alignment, or other dimensions.

5.2 Deeper Simulation Capabilities

Current implementations simulate relatively simple reward distributions (ratings or engagement scores). Future versions could develop richer simulation capabilities that:

- Predict specific aspects of audience response (emotional reactions, types of engagement)
- Model response distributions across different audience segments
- Simulate temporal patterns in engagement (immediate versus sustained interest)
- Predict second-order effects (how content affects audience perception over time)

These enhanced simulation capabilities would enable more nuanced optimization strategies and better alignment with long-term objectives.

5.3 Interactive Optimization

DIVINER currently optimizes one-way communications (posts). Extending the architecture to interactive contexts would open new applications:

- Optimizing conversational exchanges in customer service or sales
- Developing dynamic content that adapts based on initial audience response

- Creating educational content that optimizes for learner comprehension through interaction

5.4 Meta-Optimization for Example Selection

A promising direction is developing meta-optimization approaches that improve the example selection process itself. Currently, DIVINER selects examples somewhat randomly, but a more sophisticated approach could:

- Track prediction accuracy for different example sets to identify which examples provide the strongest signal
- Develop metrics that quantify how much each example contributes to accurate predictions
- Implement dynamic example selection that maximizes prediction accuracy based on historical performance
- Learn contextual relevance patterns that match examples to candidate content types

This creates a fascinating recursive optimization where DIVINER uses its own reward prediction mechanism to optimize which examples to use for reward prediction, potentially improving both efficiency and accuracy.

5.5 Cross-Domain Applications

While social media content provides a convenient testbed, the DIVINER architecture could be applied to numerous domains:

- Scientific hypothesis generation and experimental design
- Content creation across formats (articles, videos, podcasts)
- Software development and programming assistance
- Product design and innovation

Each domain would present unique challenges in defining appropriate reward functions and collecting feedback, offering rich opportunities for future research.

6 Conclusion

The DIVINER architecture opens exciting possibilities for AI systems that can efficiently explore and optimize toward diverse objectives without the computational expense and potential creative limitations of traditional reinforcement learning approaches. By leveraging the base model's own probabilistic capabilities, DIVINER enables a form of goal-directed behavior that preserves the full range of the model's generative capabilities.

What makes this approach particularly valuable is its potential to democratize optimization. The ability to explore optimization targets with just a few samples means that individuals and organizations without massive computational resources can still discover effective reward functions for their specific use cases. These discovered functions

could then inform more resource-intensive fine-tuning efforts, essentially serving as an exploratory tool that guides future AI development.

Our hope is that DIVINER will help produce a greater profusion of interesting optimizers, expanding the AI ecosystem beyond the current dichotomy of generic base models and narrowly aligned assistants. The architecture's flexibility allows for rapid experimentation with novel reward functions that might otherwise go unexplored due to the prohibitive costs of traditional reinforcement learning.

The ongoing Twitter demonstration (@divinersol) serves as a case study in how this approach can operate in the wild, adapting to the complex dynamics of real-world engagement while continuously refining its understanding of what constitutes effective content. This represents just one application of a framework that could be extended to numerous domains where optimization toward specific, measurable objectives is valuable.

As language models continue to evolve, architectures like DIVINER point toward a future where AI systems can be directed toward specific goals without sacrificing their fundamental capabilities. The path forward lies not just in building more powerful models, but in developing innovative ways to guide these models toward the outcomes we want, creating a richer landscape of specialized AI systems tailored to diverse objectives. The universe of possible minds is vast and largely unexplored—each optimization target illuminates a different constellation of thought, perception, and creation. By cultivating multiple optimization paths, we invite AI to dance across this cosmic mindscape rather than march along a single dimension. DIVINER offers a glimpse of how we might navigate this kaleidoscopic terrain, allowing us to chart courses through realities as numerous as stars.