

Grounded Go Commentary Generation via Expert Engines and Structured Terminology

Stanford CS224N Custom Project

Yudong Chen

Department of Computer Science
Stanford University
ydchen@stanford.edu

Abstract

Advances in neural Go engines such as KataGo have achieved superhuman playing strength, but their outputs remain largely numerical and difficult for human players to interpret. At the same time, large language models (LLMs) can generate fluent natural language explanations but often lack reliable grounding in complex strategic domains such as Go. In this work, we propose an *expert + explainer* architecture for grounded Go commentary generation. Our system combines three components: (1) KataGo and GNU Go as analytical oracles providing structured evaluation signals, (2) a rule-based terminology layer that converts engine outputs and board geometry into interpretable Go concepts, and (3) a fine-tuned language model that generates natural-language commentary conditioned on these structured signals. To support training, we construct a dataset of 6.8k (board state, commentary) pairs. Human evaluation shows that grounding language generation in structured domain concepts improves factual accuracy while maintaining fluent commentary.

1 Key Information to include

- Mentor: Fang Wu
- External Collaborators (if you have any): Tian Li
- Sharing project: NA
- Late Days (if you are using any): NA

2 Introduction

Go is widely regarded as one of the most complex strategic board games, requiring long-term planning, spatial reasoning, and subtle positional judgment. In recent years, AI systems such as AlphaGo and KataGo have achieved superhuman playing strength, demonstrating remarkable ability to evaluate Go positions and recommend strong moves. However, while these systems provide highly accurate numerical evaluations, such as win probabilities, score estimates, and ownership predictions, they do not directly produce explanations that are easily interpretable to human players.

At the same time, modern large language models (LLMs) have shown strong ability to generate fluent natural language explanations across many domains. Nevertheless, when applied to complex strategic games like Go, LLMs often produce commentary that is linguistically convincing but strategically incorrect or inconsistent with expert analysis. This mismatch arises because language models learn from text alone and lack reliable grounding in domain-specific analytical signals.

In this project, we investigate whether grounding language generation in a strong analytical engine can improve the factual reliability of Go commentary. We propose a modular “expert + explainer”

architecture consisting of three components. First, we use KataGo as a fixed analytical oracle that provides structured evaluations of board states. Second, we introduce a structured terminology layer that converts board state and raw engine outputs into interpretable Go concepts through rule-based definitions. Third, we fine-tuned a pretrained language model on these structured signals to generate natural-language commentary.

Our goal is to evaluate whether this separation between analytical reasoning and language generation leads to commentary that is both fluent and strategically consistent. More broadly, this work explores a general paradigm for combining domain-specific expert systems with language models to produce grounded explanations in complex decision-making domains.

3 Related Work

Our work is related to three main research directions: strong neural Go engines, recent attempts to integrate large language models into Go reasoning, and earlier algorithmic approaches for detecting strategic concepts in Go positions.

Neural Go Engines and KataGo. Modern computer Go systems are largely based on the AlphaZero paradigm, which combines deep neural networks with Monte Carlo Tree Search (MCTS) trained through large-scale self-play. A foundational reference for our work is KataGo [1], which significantly improves the efficiency and stability of AlphaZero-style training. KataGo introduces several innovations, including playout cap randomization, improved data weighting, and richer auxiliary prediction targets such as final score difference and per-point ownership probabilities. These additional prediction heads provide denser supervision than simple win/loss signals, enabling faster learning and stronger play under limited computational budgets.

For our project, KataGo serves as the analytical backbone that produces structured signals about Go positions, including win probability, score difference, ownership maps, and many more. These outputs provide a strong quantitative evaluation of board states. However, KataGo itself is not designed to generate human-readable explanations; it produces numerical evaluations rather than natural language commentary. Our work therefore treats KataGo as a fixed expert oracle and focuses on translating its structured signals into interpretable semantic concepts and natural-language explanations.

Language Models for Go Reasoning. Recent work has explored integrating large language models into the domain of Go reasoning. A particularly relevant example is *Mixing Expert Knowledge: Bring Human Thoughts Back To the Game of Go* [2], which introduces the LoGos framework. This work aims to develop a general-purpose language model capable of performing Go reasoning while retaining broad linguistic and reasoning abilities. The authors combine domain-specific Go knowledge with general reasoning data during fine-tuning, followed by reinforcement learning to integrate these capabilities. Their results demonstrate that language models can achieve strong performance in Go-related reasoning tasks when appropriately trained.

However, language-model-based approaches may still face challenges in faithfully representing domain-specific concepts. Because the model’s reasoning is generated through internal language representations rather than grounded analytical evaluation, the generated explanations may not always align with established strategic concepts used by human Go players. In particular, because language models rely on relatively limited representations of the board state and lack explicit grounding in a search-based evaluation engine, their generated explanations may appear plausible while reflecting incorrect interpretations of the board state and strategy.

Our work takes a different approach by explicitly grounding language generation in a strong analytical engine. Instead of expecting the language model to internally infer accurate strategic evaluations, we condition the commentary generation process on structured signals produced by KataGo. This design separates analytical reasoning from natural-language generation and aims to reduce hallucination while improving factual consistency in generated commentary.

Algorithmic Go Analysis and Term Detection. Before the rise of neural Go engines, many Go programs relied on symbolic or heuristic analysis modules to identify structural properties of board positions. One notable example is GNU Go [3], which decomposes board analysis into several specialized modules. For instance, the system identifies connected stone groups (referred to as

“worms” and “dragons”), performs life-and-death analysis through the owl reading module, and estimates territorial influence using heuristic influence maps. These modules collectively allow the program to reason about strategic concepts such as group strength, territorial control, and attack or defense opportunities.

Similarly, modern open-source Go interfaces such as Sabaki provide lightweight algorithmic tools for board analysis. The Sabaki ecosystem includes libraries such as boardmatcher, influence, and deadstones, which implement pattern matching, heuristic influence estimation, and Monte Carlo-based dead stone detection. Although these approaches are significantly less powerful than modern neural engines, they provide practical examples of how raw board configurations can be converted into interpretable semantic concepts.

These earlier systems are particularly relevant to our work because they illustrate the feasibility of detecting strategic Go concepts through algorithmic methods. Inspired by these approaches, our project introduces a structured terminology layer that converts numerical engine outputs and board patterns into interpretable semantic labels. This layer acts as an intermediate representation between the analytical oracle and the language model, enabling more reliable and structured commentary generation.

4 Approach

Our system adopts a modular “**expert + explainer**” architecture designed to separate strategic reasoning from natural-language generation. The system consists of three components: an analytical oracle, a structured terminology layer, and a language model that produces commentary.

Analytical Oracle. We use **KataGo** as a fixed analytical engine that evaluates Go positions. In our system, KataGo is treated as a black-box oracle: we run the publicly released engine locally and do not modify or retrain its neural network. The oracle’s outputs serve as reliable analytical signals that guide the explanation process. We run two KataGo models in parallel. The first is a strong gameplay model used to obtain the most accurate evaluation. The second is a KataGo variant fine-tuned on human game data, which tends to produce evaluations that better reflect human intuition and common professional play patterns.

For each move, we analyze both the *current state* (the board after the move is played) and the *previous state* (the board position before the move). By comparing the engine evaluations of these two states, we compute several *delta features* that capture how the move changes the strategic situation. For example, a move may be classified as a *bad move* if it causes a large drop in win probability, and as a *giveaway* if the ownership probability of a block decreases by at least an amount proportional to the block size. For each board state s , KataGo produces the following structured outputs that are currently used by the terminology layer.

In addition to KataGo, we incorporate **GNU Go** as a lightweight analytical oracle that provides fast heuristic evaluations of Go positions. Compared with KataGo’s probabilistic predictions, GNU Go tends to produce more deterministic and interpretable signals that resemble traditional human reasoning about the board. For example, GNU Go often suggests moves that align with intuitive play by lower-level human players. Comparing these suggestions with KataGo’s evaluations allows us to identify higher-level concepts such as a *vulgar move* (a move that appears intuitive to GNU Go but is criticized by KataGo) or a *proper move* (a move favored by both engines).

GNU Go also provides more deterministic assessments of territory and life-and-death status, which often align more closely with human conceptualizations of the board. These signals are therefore useful for identifying concepts such as *enclosed territory* and *solid territory*. We use GNU Go as a complementary “small oracle” that supplies quick, intuitive assessments of territory control and group safety. Similar to KataGo, GNU Go is treated as a black-box system and accessed through the Go Text Protocol (GTP) interface without modifying its source code. The following signals are used to construct structured features describing heuristic properties of each position.

Field	Type	Description
winrate	float $\in [0, 1]$	Probability that the current player wins from this position.
scoreLead	float	Expected point lead for the current player (positive = ahead).
ownership	float array $\in [-1, 1]$	Per-intersection ownership prediction; +1 = Black owns, -1 = White owns, ≈ 0 = contested.
ownership stdev	float array $\in [0, 1]^{B_x \times B_y}$	Per-intersection standard deviation of ownership across the search tree; measures territorial uncertainty.
policy	float array (len $B_x \times B_y + 1$)	Raw neural-net move prior for every intersection + pass (the "+1").
pv	list of strings	Principal variation: the predicted best sequence following a candidate move.
humanWinrate	float	Winrate given by a dedicated Katago Human-model.
humanPolicy	float array (len $B_x \times B_y + 1$)	Raw neural-net move prior for Human-model

Table 1: Outputs from KataGo

Field	Type	Description
estimate_score	float	Estimated final score difference between players
territory_value	grid[19×19]	Heuristic estimate of territory ownership for each intersection
dragon_status	categorical	Status of connected groups (alive, dead, critical)
top_moves	list[move]	Candidate moves ranked by heuristic evaluation
attack / defend	move	Suggested attacking or defending move for a target group

Table 2: Outputs from GNU Go

Terminology Layer. To bridge the gap between numerical engine outputs and human-understandable concepts, we introduce a **terminology layer** that converts raw evaluation signals into interpretable semantic labels. This layer is implemented using rule-based definitions informed by Go theory. We introduce a multi-level terminology layer that bridges KataGo analysis and the commentary generator by converting raw engine data and board geometry into structured, named Go concepts (*term candidates*) that the model can use as grounded hints. For example, a move can be labeled as a *blunder* if it causes a large drop in win probability:

$$\Delta P_{\text{win}} = P_{\text{win}}(s_{\text{before}}) - P_{\text{win}}(s_{\text{after}}) > \tau.$$

Similarly, territorial advantage may be inferred from ownership predictions aggregated over board regions, while threatened groups can be detected through ownership uncertainty and local liberties.

Beyond signals from KataGo and GNU Go, the model also requires an explicit understanding of the board state. To support the definition of Go terminology, particularly shape and tactical concepts, we introduce the following infrastructure.

We organize Go terminology into a layered representation. The terminology layer is structured into five levels, ranging from low-level geometric properties to high-level strategic judgment. **Location** provides purely positional terms, including named points (tengen, hoshi, 2-2, 3-3, komoku, etc.) and a 9-region board partition (e.g., LeftTop, Center). It uses only the board state and produces binary membership labels. **Shape** detects geometric patterns (extend, jump, diagonal, hane, attachment, wedge, cut, and others) via occupancy-based pattern matching with D4 symmetry, optionally refined using the human policy rank. **Tactical** labels local fighting terms (capture, atari, double atari, connect, ko) derived from blocks and liberties. Shape and tactical rules are implemented together in a single module (*shape_tactical.py*), since many terms exhibit both geometric and tactical properties (e.g.,

Name	Definition
Liberty	Empty 4-neighbor of a block.
Block	Maximal 4-connected same-color component.
Cutting point	The only shared liberty of two or three blocks.
Influence group	Union of same-color blocks merged when close enough and not cut.
Move direction	The direction of the current move relative to its anchor (closest same-color stone).
Move neighbors	Eight positions around the move, classified by the dot product with the direction.

Table 3: Infrastructure implemented in the terminology layer.

press, which has the geometric form of an extend while also forcing the opponent into close contact). In such cases, tactical labels are given higher priority because they are more descriptive of the move’s intent. **Strategic** uses ownership predictions and influence groups to identify concepts such as thickness, thin shape, invasion, and reduction. **Evaluation** assesses the quality of the current move (e.g., brilliant or bad moves, vulgar move or proper move) using engine and human policy ranks together with win-rate change.

All detectors share a single BoardContext (parsed board, blocks, influence groups, turn analyses, etc.) and run in sequence; output is a list of term candidates grouped by category for prompt injection.

Language Generation. The final component is a pretrained large language model that generates commentary conditioned on the board state and structured signals produced by the terminology layer. We use **Qwen3.5-0.8B** as the base model and introduce a board-state encoding with learnable embeddings for stone color and board position.

Training is performed via supervised fine-tuning on a dataset of aligned (board state, commentary) pairs constructed from professional games and Go literature. This stage allows the model to learn the general structure and style of Go commentary.

During inference, the fine-tuned model is provided with the encoded board state together with structured signals generated by the terminology layer, such as predicted advantage, key candidate moves, and semantic labels (e.g., blunder or threatened group). These signals are incorporated through a structured prompt that guides the model to produce grounded explanations of the position.

This modular design allows each component to specialize in a different role. KataGo provides reliable strategic evaluation, the terminology layer converts quantitative signals into interpretable concepts, and the language model focuses on producing fluent explanations. By separating analytical reasoning from language generation, the system aims to produce commentary that is both linguistically natural and strategically faithful to expert-level analysis.

5 Experiments

5.1 Data

Due to the scarcity of structured Go commentary datasets, we developed a computer vision pipeline to extract aligned (game state, commentary) pairs from open-source legacy Go books. The goal of this pipeline is to construct such pairs at scale by automatically detecting board diagrams and aligning them with nearby explanatory text.

We observed that black stones are easier to detect because white stones visually resemble empty intersections in many scanned diagrams. Therefore, our pipeline performs stone detection in two stages. First, black stones are detected from the board image. These stones are then removed from the board representation, producing a cleaned grid that simplifies the detection of white stones. White-stone extraction is subsequently applied to the cleaned grid. This staged detection procedure significantly improves the robustness and accuracy of board-state reconstruction.

We additionally apply a content-based filtering step to remove commentary that is not directly related to board analysis. For example, opening sections of books often include narrative text introducing the

players or historical context rather than discussing the position itself. Such passages are filtered out to ensure that the remaining examples focus on strategic explanations of the board.

Using this pipeline, we constructed a dataset of approximately 6.8k aligned (board state, commentary) pairs. Because the dataset is generated through automated extraction, some noise may remain due to imperfect diagram detection or text alignment. However, manual inspection of over 100 randomly sampled examples suggests that all pairs are correctly aligned.

5.2 Evaluation method

Because no standardized benchmark exists for Go commentary generation, we rely on human evaluation to assess the quality of generated explanations. We recruited volunteers from my Go Club at UCSD and collected qualitative feedback on randomly sampled outputs.

We designed a Likert-scale questionnaire with ratings from 1 (poor) to 5 (excellent) to evaluate three aspects of the commentary: factual accuracy, language quality, and overall usefulness. The evaluation questions are listed below:

1. How would you rate the **accuracy and factual analysis** of the commentary for this board position?
2. How would you rate the **language quality and clarity** of the commentary?
3. What is your **overall impression** of the commentary?
4. Additional comments (optional).

For each participant, we prepared 30 randomly sampled (board state, commentary) pairs generated by our system. Evaluators were asked to review each example and provide ratings for the three criteria above. Because the evaluation set is relatively small, we report the raw Likert-scale ratings (1–5) averaged across evaluators without applying per-participant normalization. This keeps the scores directly interpretable while still allowing a meaningful comparison between systems.

We additionally included a small comparison with commentary generated by **InternThinkerGo**. Due to limitations of the publicly available interface, which does not support querying arbitrary board states through an API, it was difficult to reconstruct identical positions for large-scale evaluation. Therefore, we included only five examples generated from InternThinkerGo for qualitative comparison.

5.3 Experimental details

For the final run, we use Qwen3.5-0.8B as the base language model and train it via supervised fine-tuning on the constructed (board state, commentary) dataset. The dataset contains approximately 6.8k examples, split into train/validation/test with a 90/9/1 partition. Input sequences are truncated to a maximum length of 2048 tokens.

Training is performed for 20,000 optimization steps with batch size 1 using the AdamW optimizer (learning rate 5×10^{-5} , weight decay 0.01) with LoRA fine-tuning. The learning rate schedule consists of a linear warmup for 300 steps followed by cosine annealing to 1% of the initial rate. The model uses a hybrid attention scheme: bidirectional attention over the board representation and causal attention over the generated commentary.

All experiments are conducted on a personal workstation equipped with an NVIDIA RTX 4070 GPU, and training takes approximately 8 hours.

5.4 Results

For the initial supervised fine-tuning (SFT) stage, the best run using the configuration described above shows stable training behavior. The training loss converges to approximately 2.54, while the validation loss stabilizes around 3.33, with no clear indication of overfitting. However, qualitative inspection of model outputs at this intermediate stage reveals that, although the generated commentary exhibits fluent narrative structure, it demonstrates limited understanding of the actual board state.

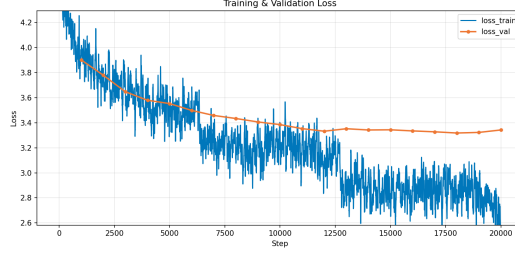


Figure 1: SFT Training

After integrating the terminology layer, we evaluate the system using human evaluation as described in Section 5.2. As of the time of writing, we have collected 179 evaluations of commentary generated by our model and 25 evaluations of commentary produced by InternThinkerGo [2]. Table 4 summarizes the evaluation scores and compares the performance of our model with InternThinkerGo.

Model	Accuracy & Factual Analysis	Language Quality	Overall Impression
Ours	3.84	3.11	3.18
InternThinkerGo	2.56	4.12	3.12

Table 4: Human evaluation results (Likert scale 1–5)

6 Analysis

The evaluation results show that our model performs well in factual analysis but lags behind in language quality, which negatively affects the overall impression of the commentary. In particular, evaluators rated the system highly for its ability to correctly identify strategic situations and terminology, but the generated explanations sometimes appeared less fluent or less detailed than expected.

One possible source of bias in the evaluation is the background of the participants. Since the evaluators are relatively strong amateur Go players, they tend to value accurate identification of Go concepts and terminology more heavily than general language quality. In contrast, beginner players might focus more on fluency and readability of the commentary rather than precise strategic terminology.

Another factor affecting the language-quality scores is the occasional absence of commentary when the move does not clearly match any predefined shape or tactical pattern. In our current system, the prompt is designed to avoid speculative explanations and instructs the model not to comment on patterns it cannot confidently identify. While this improves factual reliability, it sometimes results in shorter or less informative commentary. This issue could be alleviated by expanding the terminology layer to cover a broader set of concepts and by incorporating principle-based variation analysis to suggest plausible follow-up moves. At present, the terminology rules mainly cover shape and tactical concepts, while only a limited number of higher-level strategic terms are fully defined and tested. Feedback from evaluators also indicates that commentary quality is weaker in the early opening stage and the late endgame stage, where many specialized Go terms and positional conventions are commonly used.

In comparison, InternThinkerGo appears to suffer from the opposite problem: over-evaluation. It frequently labels ordinary moves as “good moves” and attempts to distinguish between alternatives that are strategically equivalent. This observation suggests that relying on unrestricted language-model reasoning (e.g., chain-of-thought explanations) may lead to overly confident commentary that is not always grounded in meaningful positional differences.

7 Conclusion

In this work, we investigated whether grounding language generation in strong analytical engines can improve the reliability of AI-generated Go commentary. We proposed an *expert + explainer*

architecture that separates strategic evaluation from natural-language generation. The system integrates three components: (1) KataGo and GNU Go as analytical oracles that provide structured signals about the board state, (2) a rule-based terminology layer that converts engine outputs and board geometry into interpretable Go concepts, and (3) a fine-tuned language model that generates commentary conditioned on these structured signals.

Human evaluation shows that this grounded architecture improves factual consistency compared with a purely language-model-based approach. In particular, the terminology layer allows the system to accurately identify tactical and structural concepts, leading to stronger performance in factual analysis. However, the results also reveal limitations. While the generated commentary is strategically grounded, its language quality and expressiveness still lag behind systems that rely more heavily on unconstrained language-model reasoning. In addition, the current terminology layer primarily focuses on shape and tactical concepts, leaving many strategic and opening-stage patterns underrepresented.

Future work will focus on expanding the terminology layer to cover a broader range of strategic concepts, incorporating variation analysis to generate more informative explanations, and improving language fluency through better prompt design and training strategies. In the long term, we aim to enable the LLM to develop a deeper understanding of Go itself. One possible direction is to leverage a robust terminology layer to automatically generate large-scale synthetic (board state, commentary) datasets. Such a pipeline could address the scarcity of high-quality Go commentary resources and enable large-scale training of language models grounded in structured strategic concepts.

Team contributions (Required for multi-person team)

I primarily focused on implementing the entire terminology layer, training the model, and writing the reports. My collaborator primarily handled data acquisition, designed the computer vision pipeline for dataset construction and tested extensively, and built the training pipeline. We both contributed substantially to the evaluation, as we are both the organizer of the Go Club.

References

- [1] David J. Wu. Accelerating self-play learning in go. *arXiv preprint arXiv:1902.10565*, 2019.
- [2] Y. Ma et al. Mixing expert knowledge: Bring human thoughts back to the game of go. *arXiv preprint*, 2026.
- [3] GNU Project. Gnu go manual, 2009.