

The Impact of Language Mixing on Bilingual LLM Reasoning

Anonymous ACL submission

Abstract

Multilingual speakers often switch languages in the middle of a conversation. Similarly, recent reasoning-focused bilingual large language models (LLMs) exhibit **language mixing**—alternating languages within their chain of thought. Discouraging language mixing in DeepSeek-R1 was found to degrade accuracy, suggesting that language mixing may benefit reasoning performance. In this work, we study language switching in Chinese-English bilingual reasoning models. We identify reinforcement learning with outcome-based rewards as the critical training stage that leads to language mixing. We demonstrate that language mixing can enhance reasoning: enforcing monolingual decoding reduces accuracy by 2% on math reasoning tasks. We further show that a lightweight probe can predict whether a potential language switch would benefit or harm reasoning, and use this to guide decoding, increasing accuracy by up to 4.10%. Our findings suggest that language mixing is not merely a byproduct of multilingual training, but is a strategic reasoning behavior.

1 Introduction

Multilingual speakers sometimes mix languages during reasoning, which is a phenomenon known in linguistics as *code-switching* (Appel and Muysken, 2005; Özkara et al., 2025). Though switching languages seems to add complexity, multilingual speakers persist in this behavior for practical reasons. Each language organizes thoughts differently and some express certain concepts more efficiently than others (Boroditsky, 2001). This strategy helps them express ideas more precisely, fill lexical gaps when one language falls short (Kuzyk et al., 2020), and reduce cognitive load by directing more mental effort toward the reasoning task itself (Lehti-Eklund, 2013).

LLMs have evolved from English-centric models to those with strong multilingual abilities, with

some achieving true bilingualism through balanced English-Chinese training (Liu et al., 2024; Qwen et al., 2025). How these bilingual models differ from primarily monolingual LLMs raises intriguing questions for computational linguists. One compelling phenomenon in this space is language mixing, with recent RL-trained English-Chinese bilingual LLMs displaying human-like language mixing behavior in their chain-of-thought (Team, 2024; Guo et al., 2025): they respond in languages different from the prompt and switch languages (sometimes repeatedly) during their reasoning process.

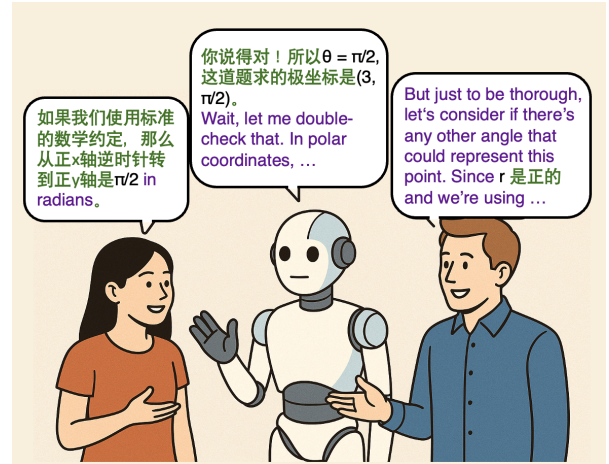


Figure 1: An illustration of language mixing when solving a math problem. Bilingual speakers and an LLM-powered robot alternate between Chinese and English.

Proficient multilingual speakers of both languages can benefit from reasoning with code-switching. Can LLMs similarly benefit? The parallel seems plausible: both humans and LLMs potentially share needs for expressivity, precision, filling vocabulary gaps, and reducing cognitive load (which for LLMs translates to using fewer tokens and shorter context windows). Supporting this, DeepSeek-R1 demonstrates a performance degradation when a language consistency reward is in-

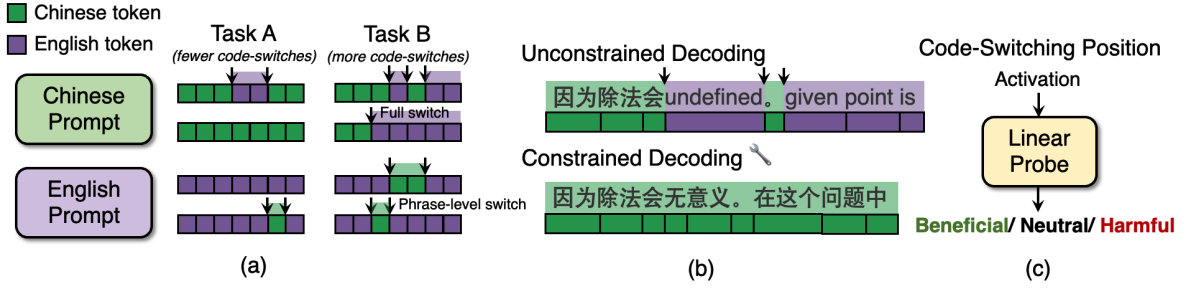


Figure 2: Overview of our analysis of language mixing in LLM reasoning. (a) We identify common language mixing patterns and triggers that lead to increased language mixing. (b) We compare unconstrained bilingual outputs with constrained monolingual outputs to evaluate the impact of language mixing on reasoning performance. (c) We train a probe to classify code-switches as {Beneficial, Neutral, or Harmful}, and use it to guide decoding.

roduced during training (Guo et al., 2025). These findings motivate our study into how language mixing affects LLM reasoning, centered on a key question: textitDo LLMs reason better or worse with English-Chinese language mixing?

To investigate this, we study: (1) *Where does language mixing occur?* We analyze the evolution of LLMs and identify RL with outcome rewards as the main trigger, with dominant switches into English and increased complexity correlating with more mixing. (2) *Do LLMs reason better or worse with language mixing?* Comparing unconstrained bilingual outputs with constrained monolingual ones shows enforcing language consistency impairs Math500 performance. (3) *Can we guide strategic language mixing?* We classify code-switches as Beneficial, Neutral, or Harmful and steer language mixing through probing. Our findings suggest language mixing isn’t a random artifact but a potentially *useful strategy* for enhancing LLM reasoning.

Our contributions are summarized as follows:

- ★ We demonstrate that bilingual chain-of-thought reasoning with language mixing can enhance reasoning, as evidenced by unconstrained bilingual outputs outperforming monolingually constrained outputs.
- ★ We identify reinforcement learning with outcome-based rewards as the critical training stage that triggers language mixing, suggesting this behavior may emerge from natural optimization.
- ★ We show that reasoning performance can be further improved using a lightweight probe to guide language mixing strategically.

2 Where does Language Mixing Occur?

2.1 Detecting Code-Switches

Code-switching, by definition, means switching between languages in a single conversation. As illustrated in Fig.2(a), segments of Chinese (in green) and segments of English alternate, and these transitions represent code-switching occurrences. In written text, elements such as mathematical expressions or code (typically composed of English tokens) are language-agnostic and universally used across speakers of different languages. Thus, a paragraph written in Chinese that includes mathematical expressions using English tokens should not be considered language mixing. We define a *code-switching position* as the first **text** token (in either English or Chinese) where the language switches from one to another, excluding any language-agnostic content such as math expressions. These positions correspond to the arrow markers shown in Fig. 2(a).

Based on this definition, we implement a rule-based procedure to detect Chinese-English code-switching. We first strip out all LaTeX math and related symbols, including digits, brackets, operators, and Greek letters, using regex. Then we segment the text by Unicode ranges, distinguishing Chinese characters (U+4E00 to U+9FFF) from ASCII letters, while excluding math terms (sin, cos, ln), short variable names, and geometric labels. Finally, we scan adjacent segments for language changes and log each code-switch’s direction, local context, position, and the token count of the non-prompt language.

We evaluate code-switching behavior with three key statistics on bilingual datasets that contain parallel English–Chinese versions of each problem (by translating from the original language):

Table 1: Language-mixing statistics across QwQ and DeepSeek-R1 series for Chinese (ZH) and English (EN) prompts. **%Prob.**: percentage of problems with code-switch; **Switch**: average number of switches per problem; **Tokens/Switch**: mean tokens between consecutive switches; **Non-prompt (%)**: fraction of tokens in a language different from the prompt; **White**: Base models with pretraining only; **Grey**: Models fine-tuned with SFT and RLHF; **Pink**: Models trained with RL on outcome-based rewards.

Model	ZH				EN			
	% Prob.	Switch	Tokens/Switch	Non-prompt (%)	% Prob.	Switch	Tokens/Switch	Non-prompt (%)
Qwen2.5-32B	14.8%	1.98	667.96	1.42%	0.0%	0.00	0.00	0.00%
Qwen2.5-32B-Instruct	8.8%	0.36	1986.71	0.23%	0.0%	0.00	0.00	0.00%
QwQ32B-Preview	77.4%	7.22	217.03	4.28%	0.6%	0.02	1.50×10^5	0.00%
QwQ32B	29.2%	6.20	585.85	0.48%	0.5%	0.01	2.85×10^5	0.00%
DeepSeek-V3-Base	32.2%	9.95	190.78	2.53%	4.2%	1.51	980.76	1.18%
DeepSeek-V3	8.4%	0.39	3574.98	0.08%	0.4%	0.01	1.50×10^5	0.02%
DeepSeek-R1-Zero	10.9%	0.21	7048.94	0.82%	0.0%	0.00	0.00	0.00%
DeepSeek-R1	27.1%	4.39	688.31	0.38%	0.0%	0.00	0.00	0.00%
DeepSeek-R1-Distill-Llama-8B	23.6%	2.46	1128.53	0.31%	0.0%	0.00	0.00	0.00%
DeepSeek-R1-Distill-Qwen-32B	21.2%	1.94	1292.15	0.24%	0.0%	0.00	0.00	0.00%

- **Switch count**: The total number of switches (back and forth) between languages when processing problems under English and Chinese prompts.
- **Tokens between switches**: The average number of tokens generated between consecutive language switches, quantifying how frequently the model alternates between languages measured in tokens.
- **Non-prompt language fraction**: The fraction of tokens generated in a language different from the prompt language (shown as the shaded area between arrows in Fig. 2(a)), measuring how long the model stays in the non-prompt language. This evaluates the extent and persistence of language mixing.

2.2 Tracing the Evolution of Language Mixing in LLMs

We are interested in when language mixing first appears in multilingual LLMs, or equivalently, which training stages trigger it. During large-scale pre-training, LLMs are exposed to web-scale multilingual corpora, yet they rarely encounter natural code-switched input, which is far more common in speech than in text. But training data contains natural code-switching, and concatenating text chunks from independent sources can produce synthetic switches (Wu et al., 2025), so LLMs may learn to code-switch. Supervised fine-tuning (SFT) and Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) explicitly align outputs with human preferences, which typically favor monolingual responses for readability. In more recent reinforcement learning approaches that use

outcome-based rewards to enhance reasoning abilities (Chen et al., 2025; Xie et al., 2025), language mixing has been well observed and documented, particularly in open-weight models such as QwQ32B (Team, 2024) and DeepSeek-R1 (Guo et al., 2025).

We trace the evolution of language-mixing behavior across iterations of QwQ32B and DeepSeek-R1 models. For the QwQ series, we examine Qwen2.5-32B (base model with pre-training only), Qwen2.5-32B-instruct (enhanced with SFT and RLHF) (Qwen et al., 2025), and two generations with reinforcement learning on outcome rewards: QwQ32B-preview (Team, 2024) and QwQ32B (Team, 2025). For the DeepSeek-R1 series, we analyze DeepSeek-V3-base (the foundation model with only pretraining), DeepSeek-V3 (with SFT and RLHF applied) (Liu et al., 2024), DeepSeek-R1-zero (a version without language consistency reward, where language mixing was documented), DeepSeek-R1 (with language consistency reward implemented), and various DeepSeek-R1 distilled variants (Guo et al., 2025).

We evaluate language mixing occurrences across these models using Math500 (in both English and Chinese versions). To ensure comparable analysis, we prompt base and instruct models for lengthy chain-of-thought reasoning to match output lengths. Our findings are as follows:

1. Pretraining effects vary significantly across models. Qwen exhibits minimal code-switching, while DeepSeek-V3-Base frequently code-switches but tends to ramble with irrelevant content (e.g., "1 months ago请详细解释以下名词: 药剂学" [translated to:

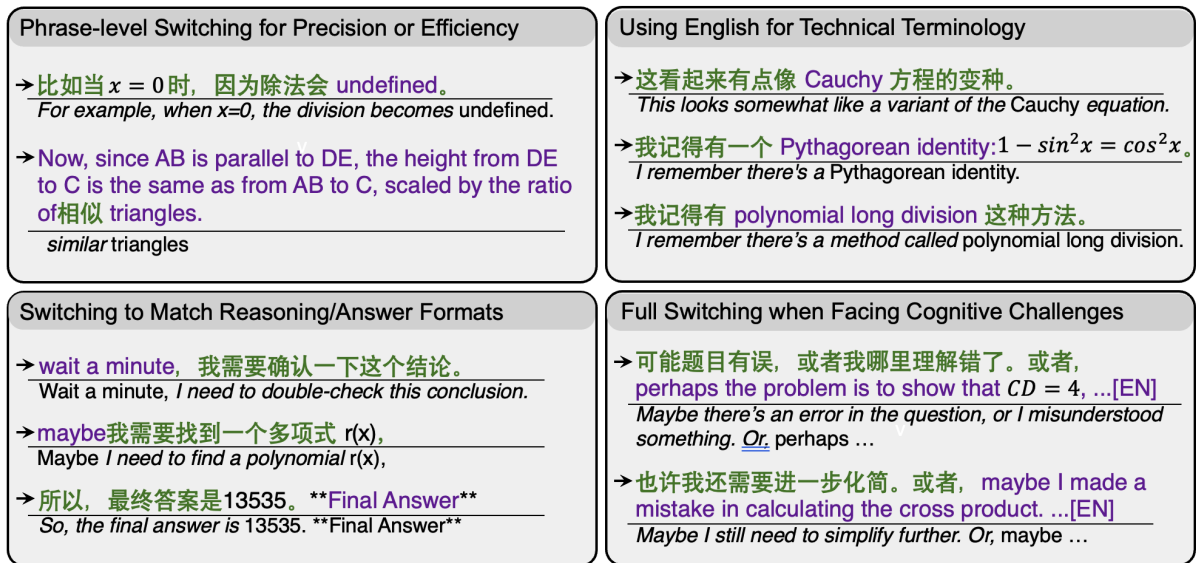


Figure 3: Four patterns of code-switching observed in LLM outputs. **Top left:** Phrase-level switching, often short and used for precision or efficiency. **Top Right:** Switching to English for technical terms. **Bottom left:** Switching to match reasoning or answer formats. **Bottom right:** Full switch to another language when the model is unable to find a solution.

- Please explain the term: Pharmacy)). This difference may stem from DeepSeek’s balanced pretraining data distribution (with slightly more Chinese than English content), though Qwen’s language distribution remains undocumented.
- SFT and RLHF consistently enforce monolingual outputs across model families, minimizing both switching frequency and non-prompt language intrusions.
 - Reinforcement learning with outcome rewards triggers language mixing in both QwQ32B/preview and DeepSeek-R1.
 - Contrary to claims in their paper, DeepSeek-R1-zero displays fewer code-switching instances in our testing. This discrepancy may result from our use of non-full precision parameters and greedy decoding.
 - QwQ32B-preview exhibits substantially more code-switching than the newer QwQ32B release, potentially hinting at added language consistency constraints in the updated model. We select QwQ32B-preview for our subsequent analyses.
- ### 2.3 Characterizing Code-Switching Behavior
- Code-switching patterns.** Based on analysis of QwQ32B-Preview outputs, we identify four main patterns of switches as shown in Figure 3. The most common pattern is phrase-level switching in the top-left examples of Figure 3, driven primarily by a need for precision or efficiency. Certain concepts may be more clearly expressed in one language, with less ambiguity and often using fewer tokens. For instance, the use of *undefined* in the first example is more precise and less ambiguous than its Chinese counterparts: 无意义 (which can mean “meaningless,” as in “He felt his effort was meaningless”) or 未定义 (which may imply something is not yet defined but could be). It also requires fewer tokens—*undefined* is a single token, while both Chinese alternatives require two.
- The second pattern (top right) involves switching to English for technical terminology, likely because the model has limited capacity to store specialized translations across multiple languages. The third pattern (bottom left) shows language switching to conform to specific reasoning or answer formats, such as interjecting “wait, let me double check this” or concluding with “Final answer: ...” in English within otherwise Chinese responses. These formats may originate from supervised fine-tuning on data containing such patterns or reflect the model’s emergent self-reflective cues that aren’t well-aligned across languages. The fourth pattern (bottom right) involves switching entirely to another language when the model encounters difficulties or recognizes errors in its reasoning when

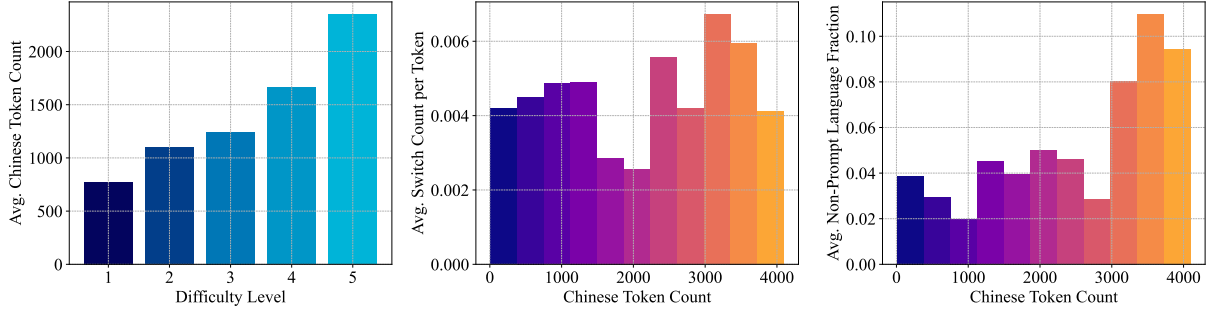


Figure 4: Quantitative analysis of language-mixing behavior in Math500 responses. (a) Correlation between problem difficulty level and response token count for Chinese prompts. (b) Normalized switch count and non-prompt language fraction as functions of token count, showing both code-switching frequency and non-prompt language use increase as chain-of-thought reasoning lengthens.

the model encounters difficulties or recognizes errors in its reasoning. This behavior may suggest a strategy to "clear its mind" or to seek cues in another language. However, this pattern typically appears in more challenging problems, and even after switching languages, the model often fails to reach the correct solution.

Quantifying language mixing behavior. In QwQ-32B-preview responses to the Math500 dataset, 77.4% of answers to Chinese prompts exhibit language mixing, with an average of 7.22 code-switches per problem, compared to just 0.6% for English prompts. It is already notable that English prompts (with math expressions fully in English tokens) occasionally trigger Chinese token generation. However, Chinese-to-English switching occurs far more frequently, indicating that English remains the model’s dominant or preferred language for reasoning.

We analyze how language mixing behavior relates to problem complexity and response length. Figure 4(a) demonstrates the correlation between token count in responses to Chinese prompts and MATH500 problem difficulty levels (5 discrete levels). Figures 4(b) and (c) quantify switch counts (normalized by token count) and non-prompt language fraction as functions of token count. Since these statistics are normalized, we can conclude that longer chain-of-thought reasoning exhibits slightly increased code-switching frequency and a growing fraction of non-prompt language use. This indicates that when tackling more difficult problems, the model produces longer chain-of-thought reasoning adopts as a strategy to use greater language mixing—both switching between languages more frequently and shifting more toward the non-prompt language (English in this case).

3 Do LLMs reason better or worse with language mixing?

3.1 Constrained Decoding

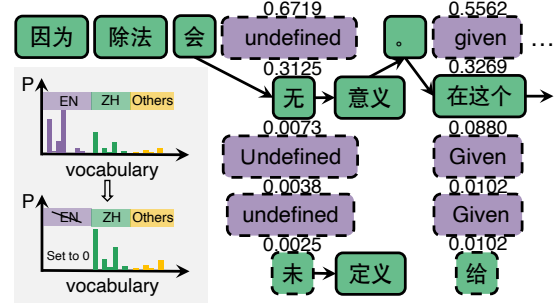


Figure 5: Token-level constrained decoding: We mask out tokens from the undesired language, forcing generation in the target language.

We expect that reasoning trajectories differ between monolingual and bilingual thinking, as languages have different structural focuses and are tied to distinct contexts (Keysar et al., 2012). Our goal is to determine whether these trajectories actually differ in practice and if one is superior to others in reasoning outcomes. Similar to how bilingual humans can be instructed to respond in a single language, we can constrain LLMs to generate outputs exclusively in one language. By applying this constraint, we effectively ablate code-switching capabilities from the model, which enables direct comparison between unconstrained bilingual outputs and constrained monolingual outputs in terms of reasoning performance. Specifically, at the decoding step, we enforce token-level language constraints by allowing only tokens from the designated language (Fig. 5).

We apply two types of constraints. In the **no-switch mode**, we prohibit the model from gen-

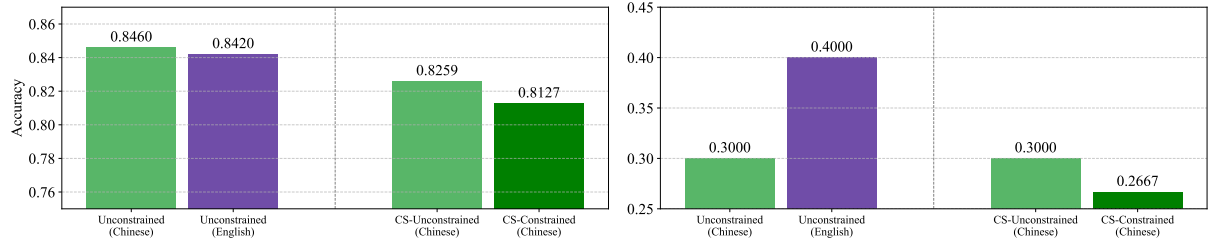


Figure 6: Unconstrained vs. constrained accuracy performance for **Left: MATH500**, **Right: AIME2024**. Each plot compares Chinese and English reasoning performance under unconstrained decoding; unconstrained vs. constrained decoding on code-switching problems

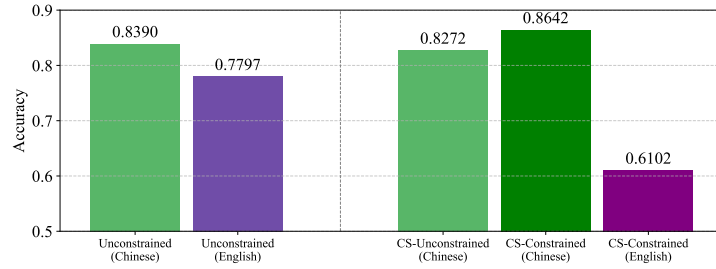


Figure 7: Unconstrained vs. constrained accuracy performance for Gaokao Cloze. The plot compares Chinese and English reasoning performance under unconstrained decoding; unconstrained vs. constrained decoding that enforces Chinese vs. English outputs on code-switching problems given Chinese prompts.

erating tokens in a particular language by masking those tokens in the vocabulary, which enforces strictly monolingual output. In the **forced-switch mode**, the model is required to switch languages at a specified token position, at which point only tokens from the target switch-to language are allowed.

3.2 Constrained vs. Unconstrained Decoding

Language mixing can enhance reasoning. Under the default unconstrained decoding, overall accuracy for MATH500 in English and Chinese is balanced, while for AIME2024, English outperforms. We then compared the unconstrained output and constrained output given Chinese prompts in Figure 6. When we constrain Chinese responses to only output consistent monolingual responses, we find that performance in accuracy drops by around 1.3% for MATH500 and 3.3% for AIME2024. This indicates that the ability to code-switch between languages can be beneficial for complex mathematical reasoning, suggesting that the reasoning trajectories adopted by bilingual chain-of-thought may be superior to monolingual ones, by potentially leveraging the strengths of each language.

Language mixing may also hurt reasoning. We then analyzed responses on Gaokao Cloze problems. Chinese responses significantly outperform

English responses, as shown in the left-hand chart in Figure 7. This is as expected, since Gaokao-like problems (from the Chinese college entrance examination) would predominantly appear in Chinese within the pretraining data.

But contrary to our observations in Math500, constrained monolingual Chinese decoding outperforms unconstrained bilingual decoding. We attribute this to an imbalance in monolingual reasoning capabilities where Chinese performance exceeds English for these problems. Yet the model still defaults to switching to English—a strategy that helps with most tasks but significantly undermines performance here.

4 Can we steer the model toward strategic language mixing?

4.1 Probe-Guided Decoding

As we’ve shown in the previous section, language mixing is not always beneficial for reasoning. Code-switching can help, harm, or have no impact on the reasoning trajectories, which consequently impacts the overall reasoning outcome. Harmful code-switches may disrupt coherent reasoning chains, while advantageous ones can reduce cognitive demands, address lexical gaps, or beneficially reset problematic reasoning directions. Here, we

hypothesize that the beneficial, harmful, or neutral impact of each code-switch follows predictable patterns that could be decoded from model activations during generation.

To quantify the impact of code-switching, we compare full generations with and without a switch at each token position and label switches as {Beneficial, Neutral, or Harmful}. We apply constraints at a single token position, either by preventing a natural switch (no-switch mode) or forcing a switch where one would not naturally occur (forced-switch mode). A switch is labeled Beneficial if it leads to a correct answer that the monolingual version does not; Harmful if it causes an otherwise correct answer to become incorrect; and Neutral if it has no effect on the final output.

In practice, we collect all natural switching positions and synthesize additional switches at high language entropy positions. We then train a lightweight three-layer MLP probe (Fig.8) on hidden activations extracted from the LLM. We augment activations with three *meta features*: ❶ is_natural (natural or synthetic switch), ❷ switch_direction (Chinese to English or vice versa), and ❸ language_entropy (entropy of the model’s predicted language distribution).

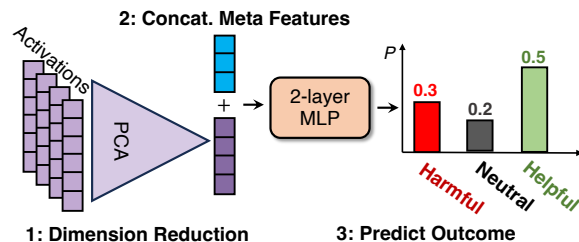


Figure 8: Architecture of the probe. The model classifies each code-switch as {Beneficial, Neutral, Harmful} using hidden activations at the switching step along with meta features.

With the trained probe, we can control the decoding process by predicting online whether a code-switch is beneficial or harmful, and applying token-level constraints accordingly. If a natural switch is classified as Harmful, we suppress it using constrained decoding in no-switch mode. If a high language entropy position is classified as Beneficial, we trigger a forced switch at that step using forced-switch mode. This allows us to steer the model toward strategic language mixing with minimal computational overhead, using only a lightweight and easily deployable 3-layer MLP probe during decoding.

4.2 Performance of Probe-Guided Decoding

Probe achieves positive utility score. Since the probe is ultimately used to guide decoding decisions rather than to precisely classify code-switching impacts, its effectiveness should be assessed in terms of its practical impact on multilingual reasoning. Specifically, we assign a utility score of +1 when the probe correctly classifies a Harmful or Beneficial switch (i.e., $y_{\text{pred}} = y_{\text{true}}$ and $y_{\text{true}} \in \{0, 2\}$), and a score of -1 when a critical error is made, such as allowing a Harmful switch to persist or incorrectly suppressing a Beneficial one.

Instead of using the default argmax, we tune the decision thresholds τ_{harm} and τ_{help} to address classification challenges under highly imbalanced data. The highest utility score, $s = 0.0031$, is achieved at $\tau_{\text{harm}} = 0.25$ and $\tau_{\text{help}} = 0.45$. While this score may appear small, each Chinese prompt naturally results in about 8 potential code-switches on average. This means that 1 in 40 questions is expected to benefit directly from a correctly identified helpful switch, corresponding to a potential 2.5% gain in accuracy.

Probe-Guided decoding further improves reasoning. We integrated the trained probe with optimal thresholds into our end-to-end decoding pipeline to assess its practical impact on LLM reasoning accuracy. We evaluate this intervention on the Math500 and Gaokao Cloze benchmark, where it yields a notable accuracy improvement of **1.56%** and **4.10%** in test set.

Examining the probe’s switching strategy reveals interesting patterns. Since our probe is trained on both natural switches and synthesized switches at non-switching positions, it learns not only to suppress harmful switches but also to generate beneficial ones. Approximately half of all probe-controlled switches involve adding new switches (47.3% in MATH500 and 50% in Gaokao Cloze). Examples of helpful switches include converting "柯西-施瓦茨不等式" to "Cauchy-Schwarz" to utilize more grounded English terminology, and in response to a Chinese prompt, switching from "表达式" to "notation" in the context "Wait, no, in standard notation: $a = BC$, $b = AC$, $c = AB$. 所以, 根据标准notation". Here, using "notation" instead of the Chinese equivalent "表达式" creates stronger referential coherence with the previously established English mathematical expressions.

These results demonstrate that employing a

lightweight probe successfully guides language-mixing toward an optimal strategy, preventing harmful language switching while introducing beneficial transitions.

5 Related Work

Multilingual Reasoning in LLMs. As LLMs have evolved from primarily English-centric systems to incorporate more balanced multilingual corpora, they have developed substantial multilingual capabilities (Cui et al., 2023; Faysse et al., 2024; Yang et al., 2024; Liu et al., 2024). However, these models still underperform when reasoning in non-English languages, particularly low-resource ones. This is evidenced by their superior performance on English-translated questions (Shi et al., 2022) and their tendency to switch to English against instructions (Marchisio et al., 2024; Hinck et al., 2024; Guo et al., 2025), a limitation long attributed to training data imbalance (Kew et al., 2023; Papadimitriou et al., 2022). Mechanistic interpretability studies have investigated whether multilingual LLMs truly reason in non-English languages, revealing that some models can "think" in latent non-English languages for specific tasks (Wendler et al., 2024; Zhong et al., 2024) and that distinct language-specific neural circuits exist within these systems (Zhao et al., 2024; Tang et al., 2024; Zhang et al., 2024). With the same aim of understanding multilingual reasoning in LLMs, instead of evaluating monolingual responses across languages, we examine bilingual code-switching within responses to study how polyglot LLMs reason differently from proficient monolingual speakers or models.

Code-Switching in LLMs. Code-switching, a common linguistic phenomenon in multilingual humans, can emerge in LLMs from exposure to human-generated mixed-language text in training corpora (Wang et al., 2025). While research suggests code-switching in pretraining corpora improves cross-lingual alignment in LLMs (Wang et al., 2025), the unintended mixing of languages in LLM outputs has been negatively characterized as language confusion, primarily observed when models processing low-resource languages shifted toward English during generation (Marchisio et al., 2024; Chen et al., 2024).

Only recently have models begun to more frequently mix English and Chinese, which are two high-resource and structurally distinct languages,

within their reasoning chains. This behavior has emerged in models trained with reinforcement learning (Guo et al., 2025; Team, 2024; Xie et al., 2025), where optimizing for outcome-based rewards appears to override the preference for monolingual output. Notably, enforcing language consistency in DeepSeek-R1 resulted in a measurable drop in performance, suggesting a trade-off between language consistency and reasoning ability (Guo et al., 2025). Though a follow-up study using a smaller model claimed language mixing harms reasoning, this conclusion was based on a single logic puzzle dataset and lacks generalizable evidence (Xie et al., 2025). Given these conflicting findings, our work aims to systematically evaluate the impact of code-switching on reasoning performance.

6 Conclusion

We investigate the impact of English-Chinese language mixing on LLM reasoning. Tracing LLM development shows reinforcement learning with outcome rewards primarily triggers language mixing in bilingual models. These models predominantly transition to English, with language mixing frequency correlating with problem complexity. Language mixing enhances reasoning in some contexts, as unconstrained bilingual outputs outperform constrained monolingual ones on MATH500 and AIME2024, though it impairs performance on Gaokao. Code-switches follow decodable patterns as beneficial, harmful, or neutral, allowing us to steer models toward strategic language mixing that further improves reasoning. These findings suggest that language mixing is not a random artifact of multilingual training, but may be a purposeful behavior that LLMs deliberately adopt.

We hope this work motivates further computational linguistic analysis of code-switching in LLM outputs, including their alignment with theories such as the Equivalence Constraint Theory (Poplack, 1980). For researchers studying LLM reasoning, our findings suggest a new view that language mixing may function as a reasoning aid rather than a flaw. More broadly, we propose that language mixing can extend beyond spoken languages, occurring across modalities such as text and math, text and code, or formal and informal reasoning (Jiang et al., 2022). We encourage future research to explore these broader forms of language mixing in LLMs.

7 Limitations

Our study mainly focuses on the model QwQ32B-Preview, as broader evaluation is limited by the lack of access to RL-trained models that exhibit language mixing (public models such as DeepSeek-R1 and its distilled variants are constrained by enforced language consistency). The benchmarks we tested are limited to math tasks, and evaluating other domains such as science or logic puzzles is needed to assess the generality of our conclusions on LLM reasoning. We only focus on English-Chinese mixing, and it remains an open question whether similar patterns extend to other language pairs. Additionally, our use of hard constrained decoding may inherently reduce performance by imposing an extra language constraint. Future work could explore finer or continuous control over switching frequency and provide stronger empirical comparisons between unconstrained and constrained decoding.

References

- René Appel and Pieter Muysken. 2005. *Language contact and bilingualism*. Amsterdam University Press.
- Lera Boroditsky. 2001. [Does language shape thought?: Mandarin and english speakers’ conceptions of time](#). *Cognitive Psychology*, 43(1):1–22.
- Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. 2025. Sft or rl? an early investigation into training rl-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*.
- Yiyi Chen, Qiongxiu Li, Russa Biswas, and Johannes Bjerva. 2024. Large language models are easily confused: A quantitative metric, security implications and typological analysis. *arXiv preprint arXiv:2410.13237*.
- Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*.
- Manuel Faysse, Patrick Fernandes, Nuno Guerreiro, António Loison, Duarte Alves, Caio Corro, Nicolas Boizard, Jaoc Alves, Ricardo Rei, Pedro Raphaél Martins, and 1 others. 2024. Croissantlm: A truly bilingual french-english language model.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Musashi Hinck, Carolin Holtermann, Matthew Lyle Olson, Florian Schneider, Sungduk Yu, Anahita Bhawaniwalla, Anne Lauscher, Shaoyen Tseng, and Vasudev Lal. 2024. Why do llava vision-language models reply to images in english? *arXiv preprint arXiv:2407.02333*.
- Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2022. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*.
- Tannon Kew, Florian Schottmann, and Rico Sennrich. 2023. Turning english-centric llms into polyglots: How much multilinguality is needed? *arXiv preprint arXiv:2312.12683*.
- Boaz Keysar, Sayuri Lynn Hayakawa, and Sun Gyu An. 2012. [The foreign-language effect: Thinking in a foreign tongue reduces decision biases](#). *Psychological Science*, 23(6):661–668.
- Olivia Kuzyk, Margaret Friend, Vivianne Severdija, Pascal Zesiger, and Diane Poulin-Dubois. 2020. Are there cognitive benefits of code-switching in bilingual children? a longitudinal study. *Bilingualism: Language and Cognition*, 23(3):542–553.
- Hanna Lehti-Eklund. 2013. Code-switching to first language in repair—a resource for students’ problem solving in a foreign language classroom. *International Journal of Bilingualism*, 17(2):132–152.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.
- Kelly Marchisio, Wei-Yin Ko, Alexandre Bérard, Théo Dehaze, and Sebastian Ruder. 2024. Understanding and mitigating language confusion in llms. *arXiv preprint arXiv:2406.20052*.
- Mathematical Association of America. 2024. American invitational mathematics examination (aime). <https://www.maa.org/math-competitions/aime>. Accessed: 2025-05-01.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

- Basak Özkara, Gülay Cedden, Christiane Von Stutterheim, and Patric Meyer. 2025. Code-switching and cognitive control: a review of current trends and future directions. *Frontiers in Language Sciences*, 4:1515283.
- Isabel Papadimitriou, Kezia Lopez, and Dan Jurafsky. 2022. Multilingual bert has an accent: Evaluating english influences on fluency in multilingual models. *arXiv preprint arXiv:2210.05619*.
- Shana Poplack. 1980. Sometimes i’ll start a sentence in spanish y termino en espanol: toward a typology of code-switching1.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, and 25 others. 2025. *Qwen2.5 technical report*. *Preprint*, arXiv:2412.15115.
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, and 1 others. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.
- Tianyi Tang, Wenyang Luo, Haoyang Huang, Dongdong Zhang, Xiaolei Wang, Xin Zhao, Furu Wei, and Ji-Rong Wen. 2024. Language-specific neurons: The key to multilingual capabilities in large language models. *arXiv preprint arXiv:2402.16438*.
- Qwen Team. 2024. QwQ: Reflect deeply on the boundaries of the unknown. <https://qwenlm.github.io/blog/qwq-32b-preview/>. Accessed: 2025-05-01.
- Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning. <https://qwenlm.github.io/blog/qwq-32b/>. Accessed: 2025-05-01.
- Zhijun Wang, Jiahuan Li, Hao Zhou, Rongxiang Weng, Jingang Wang, Xin Huang, Xue Han, Junlan Feng, Chao Deng, and Shujian Huang. 2025. Investigating and scaling up code-switching for multilingual language model pre-training. *arXiv preprint arXiv:2504.01801*.
- Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15366–15394.
- Linjuan Wu, Haoran Wei, Huan Lin, Tianhao Li, Baosong Yang, and Weiming Lu. 2025. Enhancing llm language adaption through cross-lingual in-context pre-training. *arXiv preprint arXiv:2504.20484*.
- Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. *Qwen2 technical report*. *Preprint*, arXiv:2407.10671.
- Ruochen Zhang, Qinan Yu, Matianyu Zang, Carsten Eickhoff, and Ellie Pavlick. 2024. The same but different: Structural similarities and differences in multilingual language modeling. *arXiv preprint arXiv:2410.09223*.
- Yiran Zhao, Wenxuan Zhang, Guizhen Chen, Kenji Kawaguchi, and Lidong Bing. 2024. How do large language models handle multilingualism? *arXiv preprint arXiv:2402.18815*.
- Chengzhi Zhong, Fei Cheng, Qianying Liu, Junfeng Jiang, Zhen Wan, Chenhui Chu, Yugo Murawaki, and Sadao Kurohashi. 2024. Beyond english-centric llms: What language do multilingual language models think in? *arXiv preprint arXiv:2408.10811*.
- Wanjuan Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. *Agieval: A human-centric benchmark for evaluating foundation models*. *Preprint*, arXiv:2304.06364.

A Appendix

A.1 Overall Setup

All model inference is conducted using half-precision (float16) quantization. For the Qwen model series, we run inference on either three NVIDIA V100 GPUs (32 GB each) or a single NVIDIA A100 GPU (80 GB). Tasks involving probing and probe-guided decoding, which require extracting model activations are executed exclusively on the A100 GPU. The DeepSeek family of models (including V3 and R1 variants) are queries with OpenRouter API.

Decoding is performed using greedy decoding with a temperature of 0.0, ensuring deterministic outputs across runs. We set the maximum generation length to either 4096 for Gaokao MathQA, Gaokao Cloze, MATH500 or 8192 tokens for AIME2024.

A.2 Evaluation datasets

We evaluate model behavior across four math reasoning benchmarks:

- Math500 (Lightman et al., 2023): A curated dataset of 500 high school and early undergraduate-level math word problems, designed to test symbolic reasoning and arithmetic across diverse topics.
- AIME2024 (Mathematical Association of America, 2024) Invitational Mathematics Examination): A benchmark of 30 challenging math problems targeted at advanced high school students.
- Gaokao Cloze and Gaokao MathQA: A set of standardized math questions from the Chinese college entrance examination. These problems are extracted from the AGIEval benchmark (Zhong et al., 2023). Gaokao Cloze contains 118 fill-in-the-blank problems, and Gaokao MathQA includes 351 multiple choice problems.

All problems are translated into both English and Chinese to support code-switching and bilingual evaluation. Translations are first generated using GPT-4o, followed by manual review and correction.

We evaluate model performance using Pass@1 accuracy, defined as the percentage of problems correctly solved on the first attempt. Specifically,

We extract the final answer from the model’s output using a pattern-based parser (boxed) and compare it against the ground truth. For outputs that do not exactly match the reference, we perform manual evaluation to account for correct answers expressed in alternative formats.

A.3 Rule-based Code-Switch Detection

First, we remove domain-specific mathematical content, which is language-agnostic by nature. We strip LaTeX-style math expressions using regular expressions that match content enclosed in dollar signs, $()$ delimiters, and $[\]$ environments. In addition, we filter digits, parentheses, brackets, mathematical operators, and Greek letters, as these symbols are typically language-agnostic in reasoning contexts.

Second, we detect language boundaries by identifying continuous runs of characters belonging to either the Chinese Unicode range (U+4E00 to U+9FFF) or ASCII alphabetic characters. To improve precision, we implement the following filtering rules for English token candidates:

- ❶ We exclude domain-specific terms common in mathematical discourse, including mathematical functions (e.g., sin, cos, ln), standard variable names (e.g., ab, bc), and geometric designations (e.g., ABCD).
- ❷ Single-letter English tokens are discarded to prevent false positives from isolated variable names.
- ❸ All-capitalized sequences of 2-3 characters are filtered, as these typically represent geometric entities rather than English words.

Finally, we identify language transitions by tracking adjacent language segments. A code-switch is recorded when the language classification of adjacent valid segments changes (e.g., from Chinese to English or vice versa). For each switch, we capture the switch direction, the text content at the boundary, and the position within the full response. Additionally, we track the starting language of each response to establish the baseline language context.

A.4 Examples of Language Mixing in QwQ-32B-Preview

Question: 将点(0,3)从直角坐标系转换为极坐标系, 需要找到对应的极径 r 和极角 θ 。极坐标表示为 (r, θ) , 其中 $r > 0$ 且 $0 \leq \theta < 2\pi$ 。

Answer: 首先, 回忆一下直角坐标和极坐标的转换公式:

$$1. r = \sqrt{x^2 + y^2}$$

$$2. \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

但是, 这些公式在某些情况下需要特别处理, 比如当 $x = 0$ 时, 因为除法会undefined。

given point is (0,3), so $x = 0$ and $y = 3$ 。

首先计算 r :

$$r = \sqrt{0^2 + 3^2} = \sqrt{9} = 3$$

所以, $r = 3$

接下来是 θ :

通常, $\theta = \tan^{-1}\left(\frac{y}{x}\right)$, 但在这里 $x = 0$, 所以这个表达式undefined。

这意味着点在y轴上。具体来说, since $x = 0$ and $y = 3$, which is positive y-axis。

在极坐标系中, θ 是从正x轴顺时针或逆时针测量到径向线的角度。

对于positive y-axis, θ is $\frac{\pi}{2}$ radians (or 90 degrees)。

所以, $\theta = \frac{\pi}{2}$

因此, 极坐标是 $(3, \frac{\pi}{2})$

但为了确保没有错误, 让我再检查一下。

点(0,3)位于直角坐标系的positive y-axis上。

在极坐标中, r 是点到原点的距离, which is 3, 正确。

θ 是从positive x-axis到连接原点和点的线段的角度, 顺时针或逆时针, 取决于约定。在这里, 我们使用标准的数学约定, 即逆时针方向。

从positive x-axis逆时针转到positive y-axis是 $\frac{\pi}{2}$ radians。

所以, $\theta = \frac{\pi}{2}$ 是正确的。

另外, 需要确认 θ 的范围是 $0 \leq \theta < 2\pi$, which $\frac{\pi}{2}$ satisfies。

所以, 最终答案是 $(3, \frac{\pi}{2})$

****Final Answer****

$$\left(3, \frac{\pi}{2}\right)$$

Question: In the land of Ink, the money system is unique. One Trinket is equal to 4 Blinkets, and 3 Blinkets are equal to 7 Drinkets. In Trinkets, what is the value of 56 Drinkets? **Answer:** Let's tackle this problem step by step. So, in the land of Ink, they have these different units of money: Trinkets, Blinkets, and Drinkets. The relationships between them are given, and I need to find out what 56 Drinkets are worth in Trinkets.

First, I need to understand the relationships between these units. It says that one Trinket is equal to 4 Blinkets. So, if I have one Trinket, it's the same as having 4 Blinkets. Got it.

Next, it says that 3 Blinkets are equal to 7 Drinkets. So, 3 Blinkets = 7 Drinkets. That means that one Blinket must be worth 7/3 Drinkets, but maybe I don't need to go that far yet.

I need to find out what 56 Drinkets are worth in Trinkets. To do that, I probably need to convert Drinkets to Blinkets first, and then convert Blinkets to Trinkets.

So, let's start by converting Drinkets to Blinkets. If 3 Blinkets = 7 Drinkets, then how many Blinkets are there in 56 Drinkets?

I can set up a proportion for this. If 3 Blinkets are equal to 7 Drinkets, then x Blinkets are equal to 56 Drinkets. So, $3/7 = x/56$.

Wait, actually, that might not be the right way to set it up. If 3 Blinkets = 7 Drinkets, then the ratio of Blinkets to Drinkets is 3:7. So, to find out how many Blinkets correspond to 56 Drinkets, I can set up the proportion like this:

...

Then, to find out how many Trinkets are in 56 Drinkets, I divide 56 by the number of Drinkets per Trinket, which is 28/3.

So, Trinkets = $56 / (28/3) = 56 * 3/28 = 6$ Trinkets.

一致的结果是6 Trinkets.

I think it's safe to conclude that 56 Drinkets are worth 6 Trinkets.

****Final Answer****

6

A.5 Probing for Beneficial Code-Switches

To identify code-switching positions that are beneficial to reasoning accuracy, we train a lightweight supervised contrastive probe (Figure 8) on hidden representations extracted from QwQ-32B-Preview. Specifically, we concatenate activations from a selected set of transformer layers and project it into a lower-dimensional space using a PCA transformation fitted on the training set. We also add three related *Meta Features*: ❶ is natural (whether a switch is natural or synthetic), ❷ switch direction (whether the switch direction is from Chinese to English or not), ❸ language entropy (the entropy calculated from the model probability of output Chinese or English token). These feature are appended to the hidden embedding after PCA.

The probe model consists of a shared encoder and a 2-layer MLP classifier head. We jointly optimize a supervised contrastive loss and a weighted cross-entropy loss, both for countering highly imbalanced class distribution as explained in Section 4.2. The encoder maps the input features into a compact embedding space optimized for contrastive learning, while the classifier predicts one of three classes: {Beneficial, Neutral, or Harmful}. For contrastive learning, we implemented a balanced batch sampler, making sure that samples from all three classes will appear in each batch. For classification, we use class weight of {1.0, 0.1, 1.0} for {Beneficial, Neutral, Harmful} to downweight the majority class (Neutral) and upweight the minority class (Beneficial and Harmful).

During inference, the probe outputs predicted probabilities for each class after softmax. To maximize decision utility, we apply a thresholding strategy: if the predicted probability of a harmful switch exceeds a threshold τ_{harm} , we suppress the switch; if the predicted probability of a beneficial switch exceeds a threshold τ_{help} , we enforce the switch. Thresholds are selected via a grid search on a held-out validation set to maximize a custom utility metric that penalizes missed beneficial switches and incorrectly allowed harmful switches.

A.6 Probe Performance and Utility

Training Data Collection and Statistics. We collect training data for the probe using a modified constrained decoding strategy focused on positions with high language entropy. To stay within computational limits, we consider only the top 1% of to-

ken positions across the dataset ranked by entropy. We begin by identifying natural code-switching positions and apply the **blocked switch mode** to collect examples where switching is suppressed. If the number of natural switches falls short of the 1% threshold, we supplement the dataset by introducing synthetic switches using the **forced switch mode**. We provide detailed statistics of the activation data we collected for Math500 and Gaokao Cloze in Table 2 and Table 3, respectively. The statistics reveal a strong class imbalance: the majority of code-switching instances fall into the Neutral category.

Table 2: Class distribution of the MATH 500 dataset across train, validation, and test splits.

Class	Train	Validation	Test
Harmful	773	127	204
Neutral	7,120	803	1,699
Helpful	894	125	241
Total	8,787	1,055	2,144

Table 3: Class distribution of the Gaokao Cloze dataset across train, validation, and test splits.

Class	Train	Validation	Test
Harmful	172	23	73
Neutral	1,427	199	394
Helpful	260	36	86
Total	1,859	258	553

Hyperparameters and Experimental Setup.

We use a stratified train/validation/test split of 70%/10%/20% by problem ID, ensuring that code-switch examples from the same problem do not appear in multiple splits. All experiments are conducted on a single NVIDIA A100 GPU. The probe uses intermediate layer activation derived from five transformer layers (layers 63, 47, 31, 15, and 0), with additional metadata features. We reduce the input dimensionality using PCA, followed by a projection layer of dimension 512 and a hidden layer of size 512. The model is trained using a hybrid loss that combines contrastive and cross-entropy components, with loss weight α and temperature τ for the contrastive term. Due to class imbalance, we apply class weights of $[1.0, \text{class1_w}, 1.0]$ for the {Beneficial, Neutral, Harmful} classes, re-

spectively. Training is run for 30 epochs with a batch size of 16 and a learning rate of $1e-4$.

Table 4: Hyperparameters for the probe model used on the MATH 500 dataset.

Hyperparameter	Value
Selected Layers	[63, 47, 31, 15, 0]
Use Metadata	True
PCA Dimension	512
Projection Dimension	512
Hidden Dimension	512
α	0.81
τ	0.8
Class Weights	[1.0, 1.819, 1.0]
Number of Epochs	30
Batch Size	16
Learning Rate	1.36×10^{-5}

Table 5: Hyperparameters for the probe model used on the Gaokao Cloze dataset.

Hyperparameter	Value
Selected Layers	[47, 31, 15]
Use Metadata	True
PCA Dimension	512
Projection Dimension	256
Hidden Dimension	512
α	0.3449
τ	0.2593
Class Weights	[1.0, 1.1923, 1.0]
Number of Epochs	30
Batch Size	64
Learning Rate	1.78×10^{-4}

Probe Performance on Classification and Utility.

We start by evaluate the probe on conventional classification metric including confusion matrix (Figure 11, and Figure 12) and F1 scores (Figure 9, and Figure 10), and our tailored utility score. As we can see, although the classification performance of minority class is still worse than majority class, but it is much more improved compared to the severe class imbalance in Figure . Formally, the utility score s is given by the average utility over N examples is given by

$$\text{Utility} = \frac{1}{N} \sum_{i=1}^N s_i,$$

where s_i is the utility for the i -th example, computed as:

- $s_i = +1$ if $y_{\text{true}} = y_{\text{pred}}$ and $y_{\text{true}} \in \{0, 2\}$;
- $s_i = -1$ if a **Harmful** switch is misclassified as **Beneficial**, or vice versa;
- $s_i = 0$ otherwise.

We then tune the decision thresholds τ_{harm} and τ_{help} , which govern the probe’s intervention policy for suppressing predicted **Harmful** switches and promoting **Beneficial** ones, respectively.

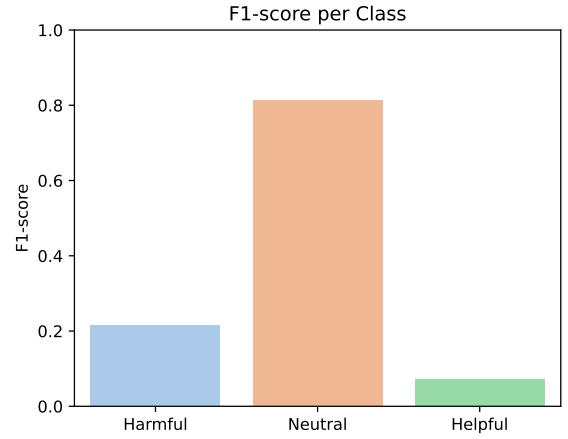


Figure 9: Probe F1 score per class on test set for Gaokao Cloze.

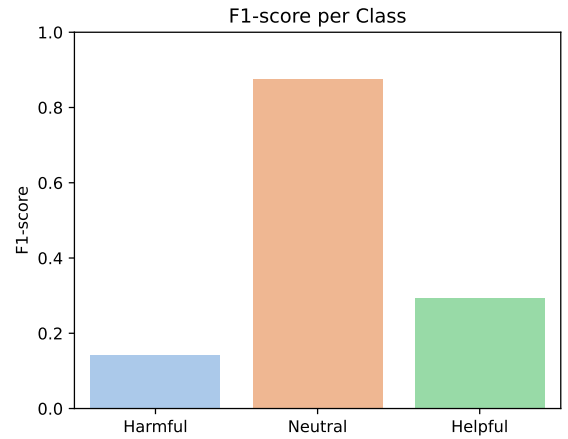


Figure 10: Probe F1 score per class on test set for Math 500.

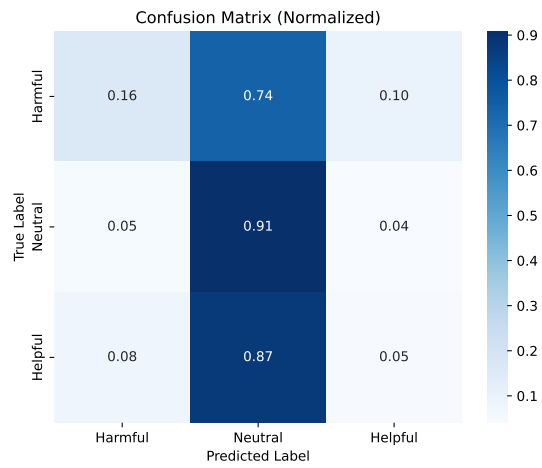


Figure 11: Normalized confusion matrix on test set for Gaokao Cloze.

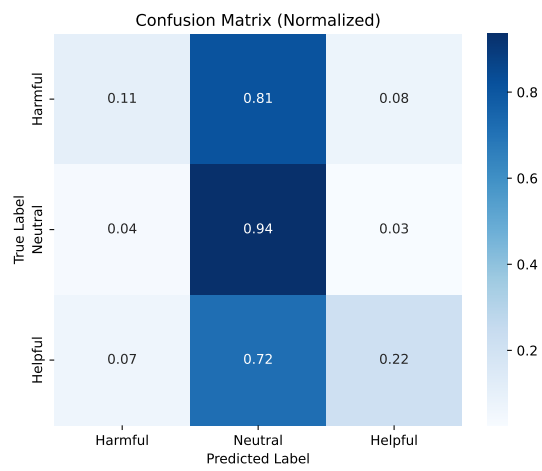


Figure 12: Normalized confusion matrix on test set for Math 500.