RESCUING REGRETTABLE REJECTIONS: RELAXED ACCEPTANCE FOR SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding has become one of the most effective techniques for accelerating inference in large language models. Its defining property is that it is lossless: the output distribution is identical to that of the target model, so the speedup comes at no cost in output quality. In speculative decoding, a small draft model proposes several tokens, the larger target model verifies them in parallel, and decoding resumes from the first proposal that fails verification. Every draft token after the rejected one is discarded, and the more tokens are discarded, the smaller the speedup becomes. We examine these rejections and find that many of them are *regrettable*. A rejection is regrettable when accepting the draft token would not have affected the quality of the output. We propose a method that rescues such rejections without training any additional model or adapter. It targets the three types of regrettable rejection that we identify, surface form, preference, and order, with a whitelist of interchangeable tokens, mutual confirmation between the two models, and order-swap tables, respectively. Experiments on Qwen3-14B and DeepSeek-R1-Distill-Qwen-32B, each drafted by a smaller 4-bit quantized model of the same family, show that the method rescues 30–36% of the rejections of lossless speculative decoding at temperature 0 and 14–15% at temperature 0.6, which raises throughput by a further 8–13% and 3.5–6.5%, respectively, and lifts the speedup over the target alone to 2.1–2.4×, with no forward pass added and both models left untouched. Mean accuracy stays within 0.4 points of lossless decoding; the one clear loss, on a single benchmark, is traced to one rule.

1 INTRODUCTION

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates LLM inference by letting a small *draft* model propose several tokens that the *target* model verifies in a single forward pass. Because autoregressive decoding is bound by memory bandwidth rather than by arithmetic (Pope et al., 2023), several tokens are produced for roughly the cost of one. Its defining property is exactness: verification accepts or rejects each proposal so that the output distribution is provably identical to the target's. Most subsequent work has therefore concentrated on the drafter, making it faster and better aligned with the target so that more proposals are accepted.

Most of the resulting drafters require training, whether purpose-built draft models, additional decoding heads, or adapters (Cai et al., 2024; Li et al., 2024), and even self-speculative decoding, in which the target acts as its own drafter (Zhang et al., 2024; Elhoushi et al., 2024; Liu et al., 2024; Xia et al., 2025), usually depends on components that must be trained or calibrated. Such methods can deliver very large speedups, but any training requirement is a barrier where computing resources and data are limited and in online services, where the data changes continuously.

Following recent work that drafts with a quantized copy of the target itself (Zhao et al., 2025; Georganas et al., 2025), we use a quantized model of the same family as the drafter, which avoids both the search for a drafter and any trained component. Such a draft is larger than usual, but its output distribution is close to the target's, so a high acceptance rate can be expected. Quantized versions of most open-weight models are readily available (Frantar et al., 2023; Lin et al., 2024), and providers

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

usually release a family of models at several sizes (Yang et al., 2025; Guo et al., 2025), so a 32B target with a 7B AWQ draft can be deployed directly, without training or modifying either model.

Speculative decoding with such a draft preserves the target's outputs while delivering a substantial speedup. The draft's own outputs are also of considerable quality: studies of collaborative and cascade decoding (Shen et al., 2024; Chen et al., 2024; Yue et al., 2024; Narasimhan et al., 2025) find that a smaller model, though less rich in vocabulary and style, often matches the larger one on concise and precise answers. Yet lossless verification discards the rest of the draft at every rejection, whether or not the rejected token was wrong. We call a rejection *regrettable* when accepting the draft token would not have affected the quality of the output, and ask whether such rejections can be waived, trading a small amount of exactness for additional speed.

Relaxed acceptance has been explored before (Section 2.3). Most rules truncate the target's distribution, as the typical acceptance of Medusa (Cai et al., 2024) does. Others compare the distributions of the two models (Holsman et al., 2025), train a judge (Bachmann et al., 2025), wait a few tokens to see whether the target corrects a mismatch (Li et al., 2026), or accept rejection pairs that recur often enough when the target's probabilities allow (Zhou et al., 2026). We ask instead which kinds of relation between the two tokens make a rejection regrettable, and whether each kind can be recognized at the moment of verification, without any training. To find out, we examine the rejections of lossless speculative decoding under greedy decoding.

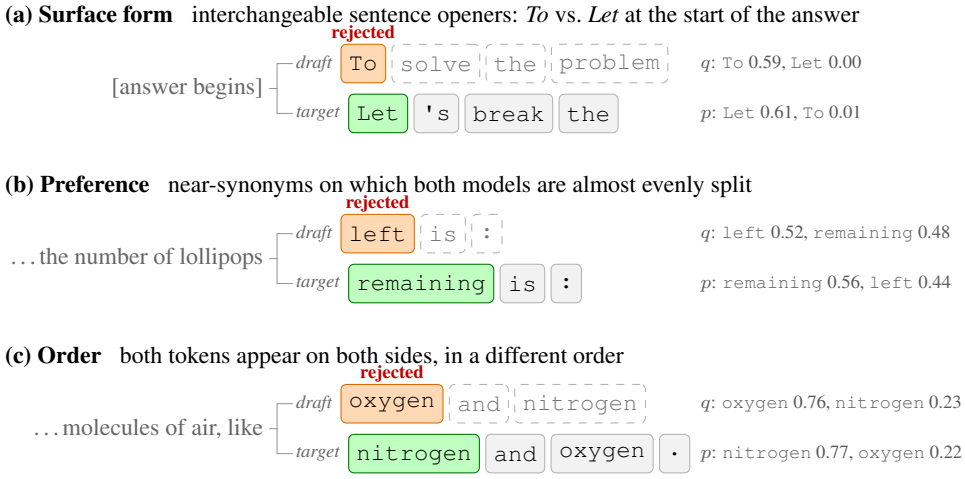

Figure 1: Regrettable rejections in lossless speculative decoding, one example per type (Qwen3-4B target, its 4B-AWQ copy as the draft, greedy decoding; Section 3.1.2). d_i the rejected draft token; t_i the target's token; d_{i+1} the rest of the draft discarded with it. q and p are the draft's and the target's probabilities of the two tokens. (a) is rescued by the whitelist (sentence-opener set), (b) and (c) by mutual confirmation.

We identify regrettable rejections in two stages (Section 3.1.2). A qualitative analysis of decoding traces yields three forms of regrettable rejection, and an LLM judge then labels 600 rejections, drawn from more diverse domains, as regrettable or justified. The three forms are defined by the relation between the token proposed by the draft and the token preferred by the target (Figure 1). In a **surface-form rejection** the two tokens are interchangeable, spell the same word up to whitespace, or one is a prefix of the other. In a **preference rejection** the two tokens mean practically the same thing, and the choice between them reflects only a preference of one model. In an **order rejection** both tokens were going to appear, and the two models merely prefer a different order. We accept draft tokens of these three kinds during decoding despite the mismatch, using three mechanisms that need only information available at decoding time: a **whitelist** of interchangeable token sets and of the whitespace and prefix relations between tokens, built from the tokenizer's vocabulary; **mutual confirmation**, which accepts the draft token when each model gives enough probability to the token proposed by the other; and **order-swap tables**, built offline from an accessible corpus, which permit exchanging the order of two tokens.

Our contributions are threefold:

- **Regrettable rejections.** We show that many rejections of lossless speculative decoding are regrettable, and classify them into three types: surface form, preference, and order (Section 3.1).
- **Training-free rules.** We target each type at verification time, without training, extra forward passes, or any change to either model: a tokenizer-derived whitelist, a two-sided probability test called mutual confirmation, and offline order-swap tables (Section 3).
- **Evaluation.** On two model pairs and nine benchmarks in vLLM, the rules raise the throughput of lossless speculative decoding by 8–13% under greedy decoding, to 2.2–2.4× that of the target alone, and by 3.5–6.5% under sampling at temperature 0.6, with mean accuracy within 0.4 points of lossless decoding (Section 4).

2 Background and Related Work

2.1 Speculative decoding

Speculative decoding goes back to blockwise parallel decoding (Stern et al., 2018), which first verified several proposed tokens in one pass, and took its present form for LLMs in the work of Leviathan et al. (2023) and Chen et al. (2023). In each round a draft model proposes k tokens, the target scores all k positions in one forward pass, and the proposals are accepted from the front up to the first one that fails verification (Section 3.1.1). Because the procedure provably preserves the output distribution of the target, it has become a standard part of LLM inference systems (Kwon et al., 2023). The speedup rises with the acceptance rate and falls with the relative cost of a draft step (Leviathan et al., 2023); prior work has addressed both.

2.2 Drafters

The most direct way to raise the acceptance rate is to train the draft towards the target: DistillSpec (Zhou et al., 2024) distills the target into the draft with on-policy data, and Medusa (Cai et al., 2024) and EAGLE (Li et al., 2024) replace the draft model with prediction heads trained on the target's hidden states; in both cases something must be trained for every target. Self-speculative decoding uses the target itself as the draft: Draft & Verify (Zhang et al., 2024) and SWIFT (Xia et al., 2025) draft with a subset of the target's layers, whereas LayerSkip (Elhoushi et al., 2024) and Kangaroo (Liu et al., 2024) rely on early exits that need training or an adapter. Quantized drafts need no training: QSpec (Zhao et al., 2025) switches only the activation precision between drafting and verification, and ML-SpecQD (Georganas et al., 2025) drafts with an MXFP4 copy of the target.

2.3 Relaxed acceptance

Exactness has often been traded for speed. Leviathan et al. (2023) already introduce a lenience parameter that softens the acceptance test, and Stern et al. (2018) accept a proposal among the target's top-k candidates or force a minimum number of tokens per step, our reference in Section 4. Most later rules decide from the target's distribution alone and truncate it. SpecDec (Xia et al., 2023) accepts a token among the target's top-β candidates whose log-probability is within τ of the top-1, typical acceptance (Cai et al., 2024) lowers the threshold where the target's entropy is high, and recent training-free rules accept the draft's token when its probability is close enough to the target's first choice (Song et al., 2026; Urbán et al., 2026). Such truncation distorts the output distribution and can degrade quality (Wang et al., 2026a). Other rules use both models: BiLD (Kim et al., 2023) and FSD (Holsman et al., 2025) accept a token when the distance between the two distributions is small, and DIVERSED (Wang et al., 2026b) verifies with a mixture of the two. Judge decoding (Bachmann et al., 2025) and AutoJudge (Garipov et al., 2025) instead train a judge on the target's representations to recognize acceptable tokens. Speculative reasoning accepts whole reasoning steps that a reward model or the target judges correct (Liao et al., 2025; Fu et al., 2025), one level above our tokens. Closest to our work are two training-free methods that accept semantically correct drafts. FLy (Li et al., 2026) waits a few tokens at high-entropy positions to see whether the target corrects a mismatched token. CSD (Zhou et al., 2026) keeps a table of rejection pairs that recur, calibrated offline on a corpus and updated during decoding, and accepts a frequent pair when the target's probability ratio between the two tokens exceeds a threshold.

We share the observation that many rejections need not happen, but we distinguish three types of regrettable rejection and recognize each from the relation between the two tokens. Unlike truncation rules and CSD's gate, mutual confirmation also consults the draft's probability of the target's token (Section 4.3.2), and our order-swap tables require evidence that the draft's token reappears. The rules decide at the rejected position, without training, deferral, or an extra forward pass, and could be combined with these methods.

3 METHOD

3.1 REGRETTABLE REJECTIONS

3.1.1 TERMS AND NOTATION

A round of speculative decoding lets the draft model propose k tokens $d_1, \ldots, d_k$, which the target model scores in a single forward pass and accepts from the front; when all k proposals are accepted, the token that the same pass yields at position $k{+}1$ is emitted as well (the bonus token). At each position i the two models define distributions over the next token, q_i for the draft and p_i for the target, both conditioned on the prompt and on the proposals $d_1, \ldots, d_{i-1}$. Under greedy decoding (temperature $T{=}0$) the proposal is $d_i = \arg\max_x q_i(x)$, the target's own choice is $t_i = \arg\max_x p_i(x)$, and position i is accepted if and only if $d_i = t_i$. Under sampling ($T{>}0$) the proposal d_i is drawn from q_i and accepted with probability $\min\big(1, p_i(d_i)/q_i(d_i)\big)$; if it is rejected, the target's token t_i is drawn from the residual distribution $\mathrm{norm}\big(\max(0, p_i - q_i)\big)$. In both cases t_i denotes the token the target emits in place of a rejected d_i, and a rejection at position i discards the remaining proposals $d_{i+1}, \ldots, d_k$.

We call a rejection *regrettable* when emitting d_i instead of t_i would neither have lowered the quality of the output nor changed its final answer. Under greedy decoding this is a well-defined counterfactual: writing $y(s)$ for the output the target produces from a prefix s, and c for the prefix before position i, the output $y(c\,d_i)$ has the same final answer as $y(c\,t_i)$ and is of no lower quality. Rescuing such a token, that is, accepting d_i in place of t_i and continuing verification at d_{i+1}, gives the proposals $d_{i+1}, \ldots, d_k$, which standard speculative decoding would discard, a second chance to be accepted. This requires no additional target forward pass: the target scored every position conditioned on the draft's own prefix, so the distributions $p_{i+1}, \ldots, p_k$ already assume that d_i was emitted. More proposals are therefore tested per round, and decoding runs faster.

3.1.2 THREE TYPES OF REGRETTABLE REJECTION

We first examined the rejections of lossless speculative decoding with Qwen3-4B as the target and its AWQ copy as the draft. For each rejection, we compared the two tokens and read how the target's output continued; the regrettable ones fell into three recurring relations, which define the types below (Figure 1). An LLM judge (Claude), prompted with these types and our examples, then labeled 600 rejections from more diverse domains as regrettable or justified, given the context, both tokens, and the target's actual continuation. The judge serves only this analysis and the choice of θ_p and θ_q, not decoding or evaluation (further examples in Appendix A.1).

Surface-form rejections. d_i and t_i differ in form rather than content, so exchanging them changes how the output reads but rarely what it says (Figure 1a). The simplest case is a pair from the same set of interchangeable tokens, such as sentence openers (*To*, *Let*, *We*) or whitespace variants (`\n`, `\n\n`). Writing $s(x)$ for the text of token x with surrounding whitespace removed, the second case is a pair with $s(d_i) = s(t_i)$, which arises because a tokenizer assigns different ids to the same word with and without a leading space. In the third, one trimmed text is a prefix of the other, ignoring case, as in *the* versus *their*: *ir* is itself a vocabulary entry, so the target can still complete the longer form at the next position. These relations depend on the tokens alone, so the tokenizer can list them in advance (Section 3.2).

Preference rejections. d_i and t_i differ in wording but mean practically the same in context, and the two models merely rank them differently: in Figure 1(b), each gives the other's choice almost as much probability as its own. Unlike surface forms, such pairs cannot be listed in advance, since

whether two words are substitutes depends on the context. We expect a preference, unlike a correction, to leave both tokens with substantial probability under both models; Section 3.3 turns this into a test.

Order rejections. d_i and t_i are not substitutes for each other; both belong in the output, and the two models disagree only on their order, as with *nitrogen* and *oxygen* in Figure 1(c). At position i the target prefers t_i, but d_i appears within a few tokens after t_i in the target's own continuation, so when d_i is accepted first, t_i is expected to follow shortly. This evidence lies in the target's continuation after t_i, which verification does not compute (Section 3.4).

3.2 Whitelist of interchangeable tokens

To recognize surface-form rejections we use a whitelist $\mathcal{W}$ of interchangeable pairs. A position i with $d_i \neq t_i$ is accepted when $(d_i, t_i) \in \mathcal{W}$, where

$$(d,t) \in \mathcal{W} \iff \exists S \in \mathcal{S} :\ d,t \in S \ \vee\ s(d) = s(t) \ \vee\ s(d) \sqsubset s(t) \ \vee\ s(t) \sqsubset s(d). \qquad (1)$$

Here $\mathcal{S}$ is the family of curated sets, and $\sqsubset$ is the case-insensitive prefix relation between word tokens of at least two letters; pairs of whitespace-only tokens satisfy $s(d) = s(t)$. A few exceptions override every rule of this section: d_i is never accepted when either token is a number, an operator such as + or −, or a negation such as *not*, and words that carry logic such as *and*, *or* and *if* are kept out of every set. In code, where indentation carries meaning, whitespace pairs are likewise excluded and the prefix relation is not applied. The whitelist is built from the tokenizer's vocabulary alone, without running the model, and all three cases reduce to lookups and range comparisons inside the GPU kernel that decides acceptance. Appendix A.2.1 lists the sets and the exceptions.

3.3 Mutual confirmation

To recognize preference rejections we use the preference that each model shows for the other's token. At a position where $d_i \neq t_i$, we ask whether $p_i(d_i)$, the probability the target assigns to the draft's token, and $q_i(t_i)$, the probability the draft assigns to the target's token, each exceed a threshold:

$$p_i(d_i) \geq \theta_p \quad \text{and} \quad q_i(t_i) \geq \theta_q. \qquad (2)$$

When both conditions hold, d_i is accepted in place of t_i. The target has thus confirmed the draft's token, and the draft would have confirmed the target's, so the disagreement is more likely one of preference than of content. Either condition alone is not enough: a target that finds d_i merely plausible may still be correcting a real mistake. Under sampling ($T>0$) the rule is applied at a position that the standard acceptance test has rejected, with t_i the token drawn from the residual distribution. We use $\theta_p = 0.1$ and $\theta_q = 0.2$, chosen once on the judged sample of Section 3.1.2 (Figure 2): below $\theta_p = 0.1$ the share of justified rejections accepted doubles while few more regrettable ones are rescued, and at $\theta_q = 0.2$ the rule accepts 38% of the regrettable rejections and 10% of the justified ones.

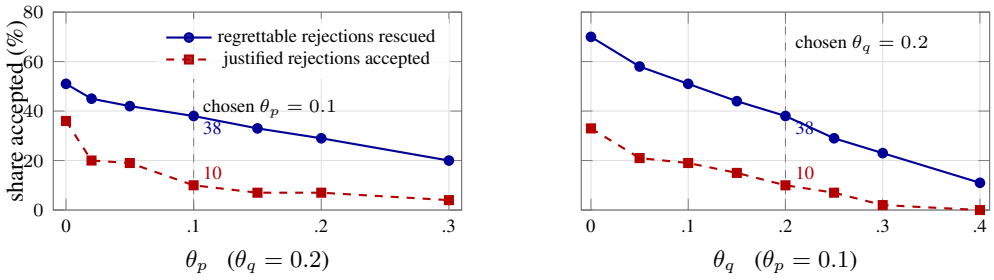

Figure 2: Choice of the mutual-confirmation thresholds on the judged sample of Section 3.1.2: share of regrettable (—●—) and of justified (- -■- ·) rejections that rule (2) accepts. Left: θ_p varies with $\theta_q = 0.2$; right: θ_q varies with $\theta_p = 0.1$. Vertical lines mark the chosen values.

3.4 ORDER-SWAP TABLES

To recognize order rejections we would like to know, at verification time, that d_i and t_i are both going to appear and differ only in order. Verification does not provide this evidence: the distributions available at position i are conditioned on d_i, not on t_i, and obtaining them directly would take further forward passes of the target. We therefore assume that a preference for one order of a pair of tokens is a property of the model pair rather than of the prompt, so that it can be measured once, offline, on an accessible corpus. Running lossless speculative decoding on the corpus, we register a pair (d_i, t_i) at every rejection whose trimmed texts have at least two characters and whose d_i is not already frequent in the recent context, and count how often the target itself emits d_i within a window of w tokens after t_i. A pair enters the table when

$$m \geq m_{\min}, \qquad h \geq h_{\min}, \qquad h/r \geq \rho, \qquad \frac{h/r}{1-(1-f)^w} \geq \lambda, \tag{3}$$

where r counts its registrations, m the distinct prompts among them, h the registrations followed by a reappearance, and f the corpus frequency of $s(d_i)$; the last condition compares the reappearance rate with chance and removes common function words such as *the*. We use $m_{\min}$=3, $h_{\min}$=2, ρ=0.5, λ=5, and w=10: a pair must recur across prompts rather than within one, reappear more often than not, and do so several times more often than chance. They were set once on the calibration corpora; Appendix A.2.2 gives the construction and their sensitivity. Writing $\mathcal{O}$ for the resulting table, a position i with $d_i \neq t_i$ is accepted whenever $(s(d_i), s(t_i)) \in \mathcal{O}$. Entries are kept separately for text and for code, and each pair is looked up by token id inside the GPU kernel. We build one table per domain from public corpora that do not overlap the benchmarks.

3.5 ACCEPTANCE PROCEDURE

A round proceeds as in standard speculative decoding, and the rules intervene only at a position that the standard test rejects. There the pair (d_i, t_i) is rescued when

$$\begin{aligned}\textsc{Rescue}(d_i, t_i) = \neg\textsc{Excluded}(d_i, t_i) \wedge \big[&(p_i(d_i) \geq \theta_p \wedge q_i(t_i) \geq \theta_q) \\ &\vee (d_i, t_i) \in \mathcal{W} \vee (s(d_i), s(t_i)) \in \mathcal{O}_{\text{ctx}}\big],\end{aligned} \tag{4}$$

with $\mathcal{O}_{\text{ctx}}$ the table for the current context, text or code. A rescued position emits d_i, and verification moves on to d_{i+1} with the p_{i+1} the target has already computed, so a rescue costs no model computation; the first position that is neither accepted nor rescued ends the round. Under greedy decoding t_i is the target's own choice; under sampling the standard acceptance test is kept and the rules examine the token drawn from the residual distribution. Overriding selected rejections changes the output distribution, and rescued tokens affect later continuations through the prefix; the procedure does not retain the distributional guarantee of lossless decoding. Every test in (4) is a probability comparison or a table lookup on the GPU, so the whole procedure runs inside the verification kernel. Figure 3 in Appendix A.2 gives the full round for greedy decoding and for sampling.

4 EXPERIMENTS

4.1 SETUP

We implement the three rules in the verification step of speculative decoding in vLLM (Kwon et al., 2023), inside the rejection sampler that runs on the GPU. We use two model pairs: Qwen3-14B (Yang et al., 2025) with Qwen3-4B as the draft, and DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025) with DeepSeek-R1-Distill-Qwen-7B. The targets run in bf16, and the drafts are the publicly released 4-bit AWQ (Lin et al., 2024) versions of the smaller models, which are easy to obtain and align closely with the target without any training. The draft proposes $k = 8$ tokens per round throughout.

We select benchmarks from five domains: mathematics with MATH-500 (Lightman et al., 2024; Hendrycks et al., 2021) and Omni-MATH (Gao et al., 2025), knowledge with MMLU-Pro (Wang et al., 2024) and GPQA Diamond (Rein et al., 2024), logic with ZebraLogic (Lin et al., 2025) and the logical-deduction, object-tracking, and web-of-lies tasks of BIG-Bench Hard (Suzgun et al., 2023), code with HumanEval+ (Liu et al., 2023; Chen et al., 2021) and MBPP (Austin et al., 2021), and

instruction following with IFEval (Zhou et al., 2023). Accuracy is the exact match of the final answer for mathematics, knowledge, and logic, pass@1 on the test suites for code, and strict prompt-level accuracy for IFEval. Every configuration is compared with lossless speculative decoding and with the target alone, under greedy decoding (T=0) and under sampling at T=0.6 (temperature only, without top-p or top-k truncation).

Speed is measured at batch size 8, a light serving load in which speculative decoding pays off. Throughput is the number of tokens generated while the batch is full, divided by the duration of that interval; the tail of a run, in which the last long outputs finish alone, is left out, since its length depends on which problems end last. We report geometric means over benchmarks of the ratios to the target alone and to lossless decoding, averaged over three runs; quality is measured separately, at a large batch size and on more problems per benchmark (Appendix A.3.1).

We also run a forced-acceptance reference, following the minimum block size of Stern et al. (2018) recalled in Section 2.3: any rejection among the first m of the $k = 8$ proposals is overridden, with $m \in \{2, 4, 8\}$. It is not a method: it accepts every rejection in its range, regrettable or not, and serves as an aggressive reference for what accepting rejections can gain in speed and cost in quality.

4.2 Main results

Table 1: Main results. Speedup: throughput relative to the target alone (geometric mean over the nine benchmarks of Section 4.1); for the relaxed configurations, the lossless speedup times their ratio to lossless decoding averaged over three runs. Δacc.: mean accuracy minus that of lossless decoding (points). Forced acceptance is a reference, not a method; bold marks the fastest configuration whose mean accuracy is within 1 point of lossless decoding, the noise reference of Section 4.2.

	Qwen3-14B / 4B-AWQ				R1-Distill-Qwen-32B / 7B-AWQ			
	T=0		T=0.6		T=0		T=0.6	
Configuration	speedup	Δacc.	speedup	Δacc.	speedup	Δacc.	speedup	Δacc.
target alone	1.00×	+0.2	1.00×	+0.5	1.00×	+0.2	1.00×	+1.0
lossless	2.07×	0.0	1.99×	0.0	2.09×	0.0	1.99×	0.0
ours	**2.23**×	0.0	**2.06**×	0.0	**2.35**×	−0.3	**2.12**×	−0.4
Reference: forced acceptance of the first m of k=8 proposals								
m=2	2.39×	−2.1	2.32×	−2.3	2.50×	−1.7	2.42×	−2.5
m=4	2.64×	−4.1	2.56×	−3.6	2.69×	−3.6	2.75×	−4.7
m=8	2.84×	−8.2	2.78×	−8.7	2.63×	−11.5	3.01×	−10.2

Table 1 summarizes the main results. Averaged over the benchmarks of Section 4.1, the three rules together raise the throughput of speculative decoding (k=8) over the lossless baseline by 7.8% at T=0 and 3.5% at T=0.6 for the Qwen3 pair, and by 12.8% and 6.5% for the DeepSeek pair. In the standard measure, this lifts the speedup over the target alone from 1.99–2.09× to 2.06–2.35×.

Quality, read as the mean accuracy difference from lossless decoding, stays at the baseline level in all four settings: 0.0 points at both temperatures for the Qwen3 pair and −0.3 and −0.4 points for the DeepSeek pair. The target alone and lossless decoding, which should agree, differ by up to 1.0 point in the mean and by 4.9 points on a single benchmark (Tables 9–12); this range is our reference for measurement noise, and all four mean differences of our method lie well inside it. One per-benchmark loss does not: on ZebraLogic the DeepSeek pair loses 8.5 and 5.5 points, which Section 4.3.1 traces to the prefix relation of the whitelist.

By domain, the gain depends on two things: how often lossless decoding rejects, and how many of those rejections the rules rescue. Mathematics rescues the largest share (38–48% at T=0) but rejects least often (about 3 per 100 tokens); instruction following rejects most often (14–17 per 100 tokens) but rescues the smallest share (15–17%); the gain is therefore similar across the text domains (9–12% for the Qwen3 pair at T=0). Code is the exception for the Qwen3 pair, whose short outputs are already accepted at seven of eight positions (−1.8% and −0.1%). The DeepSeek pair gains more across domains (5–22% at T=0), since its 7B draft is rescued more often (36% against 30%), including on code, whose long reasoning traces are ordinary text.

The forced-acceptance configurations of Section 4.1 give an aggressive reference. The more tokens are accepted, the closer the output moves to the draft's quality: at T=0.6, m=8 raises throughput by 40–51% over lossless while accuracy falls by 9–10 points, and even m=2 costs about 2 points. Our method takes its gain at almost no cost in mean accuracy but stays well below this reference. Per-benchmark results and a throughput–accuracy plot are given in Appendix A.3.2.

4.3 Ablation study

4.3.1 Effect of each rule

Table 2 reports, for the Qwen3 pair, each rule alone, mutual confirmation with the whitelist, and all three together, relative to lossless speculative decoding.

Table 2: Effect of each rule for the Qwen3 pair: change in throughput over lossless speculative decoding (thr., %) and in accuracy (acc., points), averaged over the nine benchmarks of Section 4.1. P: $p_i(d_i) \geq \theta_p$; Q: $q_i(t_i) \geq \theta_q$; W: whitelist; O: order-swap tables; every row applies the exclusion list. The last row keeps only the target-side test of mutual confirmation.

	rules				T=0		T=0.6	
Configuration	P	Q	W	O	thr.	acc.	thr.	acc.
lossless	–	–	–	–	0.0	0.0	0.0	0.0
mutual confirmation	✓	✓	–	–	+6.7	+0.6	+2.0	+0.2
whitelist	–	–	✓	–	+2.1	+0.6	+1.6	−0.4
order-swap tables	–	–	–	✓	+0.5	−0.1	+0.5	+0.4
mutual confirmation + whitelist	✓	✓	✓	–	+7.9	+0.5	+3.5	−0.3
ours	✓	✓	✓	✓	+7.8	0.0	+3.5	0.0
target-side only (θ_q=0)	✓	–	–	–	+17.2	−0.4	+13.8	0.0

Mutual confirmation contributes most of the speed: alone it gives +6.7% at T=0 and +2.0% at T=0.6, and in the combination it accounts for 73% and 45% of the rescued positions, counting only positions that no other rule would also accept. The whitelist adds a smaller but consistent gain, +2.1% and +1.6%, with mean accuracy changes of +0.6 and −0.4 points. The order-swap tables contribute least and, on top of the other two rules, change neither speed nor accuracy measurably (+7.9% against +7.8% at T=0, +3.5% for both at T=0.6; Table 2). They rescue the order rejections that fall below the thresholds of mutual confirmation, but a table built from a corpus holds only pairs that recur, whereas most order rejections involve words specific to the problem. We keep them as the only rule aimed at the order type; in our setting they can be omitted without measurable loss.

One issue with the whitelist appeared in the DeepSeek pair: on ZebraLogic, the whitelist alone loses 11.5 points at both temperatures, and every combination that includes it loses between 5.5 and 13.5 points. The cause is the prefix relation, which rewrites inflected forms inside the reasoning, such as *their* to *the*, and in some problems the keys of the JSON answer. Removing the prefix relation gives 67.0% for the whitelist-only variant on this benchmark at T=0.6, against 67.5% for lossless decoding; the reported results keep the prefix relation and therefore include this loss. No other benchmark of either pair shows the effect.

4.3.2 Role of the draft-side condition

Mutual confirmation uses two probabilities, the target's for the draft's token, $p_i(d_i)$, and the draft's for the target's token, $q_i(t_i)$, whereas the truncation-based rules of Section 2.3 (Xia et al., 2023; Cai et al., 2024; Song et al., 2026; Urbán et al., 2026) rely on the target's preference alone. Table 3 compares mutual confirmation (θ_q=0.2) with the same rule at θ_q=0, which keeps only the target-side test and thus becomes a truncation rule of this kind (last row of Table 2).

Without $q_i(t_i)$, more draft tokens are accepted: the rescued share rises from 28.1% to 56.0% at T=0 and from 7.6% to 42.0% at T=0.6, and the throughput gain from 6.7% to 17.2% and from 2.0% to 13.8%. The effect on quality, however, differs by domain. At T=0, accuracy against lossless decoding drops on the four math and knowledge benchmarks, by −3.6 to −2.2 points each. Over these four benchmarks the change is −2.9 points (95% confidence interval −4.6 to −1.2, paired

Table 3: Role of the draft-side condition (Qwen3 pair). Mutual confirmation alone, with θ_q=0.2 or with θ_q=0, which leaves only the target-side test. thr.: throughput change over lossless decoding (%); resc.: share of rejections rescued (%); acc.: accuracy change (points). Mean column: geometric mean for thr., share over all positions for resc., arithmetic mean for acc.

T	θ_q		MATH	Omni	MMLU-P	GPQA	Zebra	BBH	HE+	MBPP	IFEval	mean
0	0.2	thr.	+8.6	+9.9	+8.9	+10.1	+7.2	+8.8	−0.3	−1.7	+9.6	+6.7
		resc.	39.8	40.3	17.8	22.9	36.8	22.7	0.6	1.4	10.8	28.1
		acc.	−0.8	−0.6	−1.8	+2.0	+2.3	+3.3	0.0	0.0	+1.0	+0.6
	0	thr.	+14.0	+17.1	+29.3	+21.6	+14.1	+23.3	+0.4	0.0	+40.9	+17.2
		resc.	64.0	65.6	48.5	50.6	63.3	53.2	12.6	12.0	43.7	56.0
		acc.	−2.2	−3.6	−2.8	−3.0	+3.3	+4.0	+0.6	−0.5	+0.5	−0.4
0.6	0.2	thr.	+2.4	+2.9	+2.8	+2.9	+1.8	+2.3	+0.3	−0.2	+2.7	+2.0
		resc.	11.1	11.4	4.1	5.7	9.5	5.3	0.0	0.2	2.3	7.6
		acc.	−0.2	−0.4	−2.0	+5.1	0.0	−0.7	−0.6	0.0	+0.2	+0.2
	0	thr.	+9.3	+14.1	+21.8	+18.3	+15.2	+16.6	+1.1	+1.5	+29.5	+13.8
		resc.	47.0	47.7	37.2	37.8	47.8	40.8	9.1	6.7	31.7	42.0
		acc.	−0.2	−0.6	−1.4	+3.5	−1.7	0.0	0.0	+0.8	0.0	0.0

over the same problems). With θ_q=0.2 it is −0.7 points (−2.2 to +0.8): the interval contains zero, so accuracy with θ_q=0.2 cannot be distinguished from that of lossless decoding. This drop is in line with the finding that truncation-based rules distort the output distribution (Wang et al., 2026a); under greedy decoding, it is the draft-side condition that separates mutual confirmation from such rules. The logic benchmarks gain a few points in both settings, likely because greedy decoding is 4–5 points weaker than sampling on them (Appendix A.3.2) and a rescue acts like a sampling step. At T=0.6 the two settings do not differ measurably (+0.2 and 0.0 points on average).

Taken together, the rules add speed on top of lossless speculative decoding without training or extra forward passes, 8–13% under greedy decoding, that is, 2.2–2.4× the target alone, for both model pairs. Unlike forced acceptance, which loses about 2 points already at m=2, they keep mean accuracy within 0.4 points, and under greedy decoding the draft-side condition of mutual confirmation contributes to this.

5 Conclusion and Limitations

Lossless speculative decoding discards every proposal after a rejected token, and we found that many of these rejections are regrettable: the draft's token would have served as well as the target's. We identified three forms of such rejections, surface form, preference, and order, and matched each with a training-free rule: a tokenizer-derived whitelist, mutual confirmation on the two models' probabilities for each other's token, and offline order-swap tables. In our experiments, they raise throughput by 8–13% and 3.5–6.5% over lossless decoding, lifting the speedup over the target alone to 2.1–2.4×, while mean accuracy stays within 0.4 points of lossless decoding. Mutual confirmation provides most of the gain. In the Qwen3 ablation, its draft-side condition is what separates it from truncation rules under greedy decoding: without the condition the gain rises but accuracy falls by 2.9 points on the four math and knowledge benchmarks; under sampling at T=0.6 it can be dropped at no measurable cost, so θ_q can be lowered there.

The gap to forced acceptance shows that more remains to be gained, and the classification indicates where. Order rejections call for evidence from the target's continuation, and preference rejections for a sharper separation from corrections than two fixed thresholds. Justified rejections accepted by mistake cost accuracy visibly only through the prefix relation on one benchmark. Our experiments cover two Qwen-derived pairs with quantized drafters, one batch size, and one quality run per temperature, and compare against lossless decoding and forced acceptance but not directly against other relaxed-acceptance methods (the θ_q=0 row of Table 2 stands in only for their target-side gate). The rules are not exclusive of those methods: FLy's deferred check or CSD's recurrence table could act on the positions our rules leave, and comparing and combining them in one engine is a natural next step.

AI USE STATEMENT

We used generative AI tools (Claude, Anthropic) for tasks that the ICLR AI policy lists as requiring disclosure: the LLM-as-a-judge annotation of Section 3.1.2, the implementation of the rules in the inference engine, feedback on the experimental design, and the analysis and interpretation of results; the remaining tasks on that list, such as synthetic data or mathematical claims and proofs, do not apply to this work. We also used them for tasks with recommended disclosure: the experiment and aggregation scripts, the tables and figures, literature search, bibliography checks, and translating and polishing the text from the authors' notes. The authors reviewed all AI-assisted work, every piece of code, every result, and every sentence, and take full responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

All experiments use publicly available models and benchmarks and an open-source inference engine, so they can be reproduced directly. The three rules are implemented as a patch set against vLLM that changes only the verification step; the models, benchmarks, problem counts, token caps, batch sizes, seeds, and hardware are listed in Appendix A.3.1, and the construction of the whitelist and of the order-swap tables, including the public corpora they are built from, in Appendix A.2. Greedy runs are deterministic for a given engine build, but individual tokens can differ across GPUs and compilation caches, so every configuration is compared with a lossless run on the same machine, and throughput is averaged over three runs. The patched engine, the table builders and the tables used here, the benchmark harness, and the scripts that produce every table and figure will be released publicly.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *International Conference on Learning Representations (ICLR)*, 2025.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Yichao Fu, Rui Ge, Zelei Shao, Zhijie Deng, and Hao Zhang. Scaling speculative decoding with lookahead reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge decoding without manual annotation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Evangelos Georganas, Dhiraj Kalamkar, Alexander Kozlov, and Alexander Heinecke. ML-SpecQD: Multi-level speculative decoding with quantized drafts. *arXiv preprint arXiv:2503.13565*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Maximilian Holsman, Yukun Huang, and Bhuwan Dhingra. Fuzzy speculative decoding for a tunable accuracy-runtime tradeoff. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Jinze Li, Yixing Xu, Guanchen Li, Shuo Yang, Jinfeng Xu, Xuanwu Yin, Dong Li, Edith C. H. Ngai, and Emad Barsoum. Training-free loosely speculative decoding: Accepting semantically correct drafts beyond exact match. In *International Conference on Learning Representations (ICLR)*, 2026.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient LLM reasoning. In *International Conference on Machine Learning (ICML)*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. In *International Conference on Machine Learning (ICML)*, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2025.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *Conference on Language Modeling (COLM)*, 2024.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Jingwei Song, Xinyu Wang, Hanbin Wang, Xiaoxuan Lei, Tianyu Shi, Shixin Han, Eric Yang, Xiao-Wen Chang, and Lynn Ai. MARS: Unleashing the power of speculative decoding via margin-aware verification. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Oszkár Urbán, Young D. Kwon, Stylianos I. Venieris, and Cecilia Mascolo. Margins, not windows: Training-free per-step lossy speculative decoding. *arXiv preprint arXiv:2609.02897*, 2026.

Tianyu Wang, Yuxuan Zhou, Heng Li, Wenbin Wang, Zikai Xiao, Chunrui Zheng, and Junyuan Shang. Revisiting lossy verification in speculative decoding: Mechanisms, trade-offs, and failure modes. *arXiv preprint arXiv:2607.26627*, 2026a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024.

Ziyi Wang, Siva Rajesh Kasa, Ankith M S, Santhosh Kumar Kasa, Jiaru Zou, Sumit Negi, Ruqi Zhang, Nan Jiang, and Qifan Song. DIVERSED: Relaxed speculative decoding via dynamic ensemble verification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2026b.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. SWIFT: On-the-fly self-speculative decoding for LLM inference acceleration. In *International Conference on Learning Representations (ICLR)*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. QSpec: Speculative decoding with complementary quantization schemes. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Xuwen Zhou, Fangxin Liu, Chao Wang, Xiao Zheng, Hao Zheng, Min He, Li Jiang, and Haibing Guan. Calibrated speculative decoding: Frequency-guided candidate selection for efficient inference. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations (ICLR)*, 2024.

A Appendix

A.1 Additional examples of regrettable rejections

Table 4 lists three further examples for each of the three types of regrettable rejection introduced in Section 1, drawn from the same inspection of 600 rejected positions as Figure 1. As in Figure 1, the draft column shows the rejected draft token followed by the rest of the draft that was discarded with it, and the target column shows the target's token and what the target actually produced next. The two probability columns give, for the draft token and the target token respectively, the probability assigned by the draft model (q) and by the target model (p).

Table 4: Additional regrettable rejections from the 600-case inspection (Qwen3-4B target, 4B-AWQ draft, greedy, k=6), three per type. d_i the rejected draft token; t_i the target's token; $d_{i+1}\ldots$ the rest of the draft that was discarded. The surface-form rows differ only in the number of newlines, a leading space, or a plural suffix; the preference rows are near-synonyms with an identical continuation; in the order rows the target's continuation contains the draft's token (*down*, *Half*, *rationality*) and the draft's continuation contains the target's. The last column names the rule of Section 3 that rescues the pair (W: whitelist, MC: mutual confirmation, O: order-swap table); – marks a regrettable rejection that none of the rules rescues.

Type	**Context**	**Draft proposal**	**Target output**	q (d / t)	p (d / t)	**Rule**
Surface form	… ### Step 1: Find Ben's iPhone age	`\n` Ben's iPhone is …	`\n\n` Ben's iPhone is …	0.59 / 0.27	0.43 / 0.55	W, MC
	def parse_music(…): """	`Input` to this function is …	`␣Input` to this function is …	0.84 / 0.16	0.00 / 1.00	W
	… we need to first understand the tax	`rates` based on the stock's par value	`rate` based on the stock's par value	0.64 / 0.21	0.09 / 0.86	W
Preference	… the cost of each type of clothing and then	`sum` them up	`add` them together	0.54 / 0.46	0.15 / 0.85	MC
	… (A) Social Learning: This theory	`focuses` …	`emphasizes` learning through observation and imitation	0.71 / 0.22	0.34 / 0.56	MC
	… Plants release water vapor into the	`atmosphere` through their leaves	`air` through their leaves	0.90 / 0.10	0.44 / 0.56	–
Order	Let's break	`down` the problem step by step	`the` problem down step by step	0.46 / 0.45	0.34 / 0.56	MC, O
	… Step 1: Find how many jewels Aaron has.	`Half` of Raymond's jewels …	`Ray` mond has 40 jewels. Half of that is …	0.48 / 0.41	0.41 / 0.59	MC
	… (B) Ethics of duty: Emphasizes	`rational` ity and adherence …	`duty` and rationality (e.g., Kantian deontology)	0.44 / 0.31	0.08 / 0.37	–

A.2 Rule details

A.2.1 Whitelist

The whitelist accepts a pair (d_i, t_i) when one of three relations holds and no exclusion applies.

Interchangeable sets. Table 5 lists the seven sets. Each set starts from a short list of words or symbols, which is expanded to every vocabulary entry that spells one of them with or without a leading space and in either capitalization. A token may belong to several sets, and a pair is accepted when the two tokens share at least one set.

Table 5: Interchangeable sets of the whitelist for the Qwen3 tokenizer (628 tokens in total). Token counts include leading-space and capitalization variants.

Set	Tokens	Examples
sentence openers	18	*To*, *Let*, *We*, *Now*, *So*
articles	12	*a*, *an*, *the*
causal conjunctions	13	*so*, *thus*, *therefore*, *hence*
additive conjunctions	13	*also*, *moreover*, *furthermore*
layout whitespace	422	`\n`, `\n\n`, runs of spaces, tabs
sentence-final punctuation	148	`.`, `:`, `;`, `!` and their whitespace variants
terminators	2	end-of-turn, end-of-text

Trimmed-text and prefix pairs. Vocabulary entries whose text is identical after surrounding whitespace is removed form a group (17,127 groups covering 36,547 tokens for the Qwen3 tokenizer); a pair inside one group is accepted. Two tokens that consist only of whitespace are accepted as a pair, and so are a terminator and a whitespace token, which lets the draft end an answer where the target would emit a line break. The prefix relation applies to two word tokens of at least two letters whose trimmed texts differ and one of which is a prefix of the other, ignoring case.

Exclusions. Every rule is vetoed when either token is a number (28 tokens), consists only of operator characters (`+ − * / = < > % & | ^ ~ ( ) [ ] { }`; 666 tokens), or is a negation word (*not*, *no*, *none*, *nor*, *never*; 31 tokens); when exactly one token is a terminator and the other is not whitespace; and, in code, when both tokens are whitespace. The prefix relation is not applied in code either. In addition, the words *and*, *or*, *if*, *unless*, *without* and *of*, and any token containing a digit or one of the characters above, are never placed in a set.

Construction. Everything is derived from the tokenizer alone. One pass over the vocabulary assigns each token its sets, its trimmed-text group, and its flags (number, operator, negation, whitespace, terminator), and the result is stored as a table that the decoding kernels look up by token id. No model is run and no data beyond the vocabulary is needed; building the tables for a new tokenizer takes seconds.

A.2.2 Order-swap tables

The tables of Section 3.4 are built from a snapshot of lossless speculative decoding on a calibration corpus, which records at every rejection the pair (d_i, t_i) and the tokens the target emitted afterwards. Table 6 lists the criteria; a pair is keyed by the trimmed texts $s(d_i)$ and $s(t_i)$ together with a context flag (text or code) that is set when the rejection occurs inside a code block.

The thresholds are not delicate. In an offline re-application on the calibration snapshots, lowering the hit minimum to 1 reduced the share of rescued positions whose draft token actually reappeared from 56% to 44%; moving the ratio to 0.3 or 0.7 changed the table size by $+13\%$ or -25% and that share by -4 or $+6$ points; requiring only 2 prompts made the tables 1.5 times larger; and widening the window to 15, 20, or 30 tokens lowered the share to 43%, 39%, and 35%. The lift replaces a hand-written list of function words: a form whose corpus frequency exceeds about 2% cannot enter the table even if it always reappears, and *and* (1.5%) needs a hit rate above 0.7. Table 7 gives the calibration corpora, none of which overlaps the benchmarks (checked by exact match and by shared 12-grams; the few near duplicates found were removed), and the size of the resulting tables for the

Table 6: Criteria for the order-swap tables.

Stage	Criterion
Registration	$s(d_i)$ has at least two characters; $s(d_i)$ makes up less than 30% of the last 50 tokens; neither token is on the exclusion list; both tokens are terminators or neither is.
Hit	Within the 10 tokens after t_i, consecutive tokens emitted by the target concatenate exactly to $s(d_i)$.
Selection	At least 2 hits from at least 3 different prompts; hits / registrations ≥ 0.5; $s(t_i)$ has at least two characters; lift ≥ 5, where the lift is the hit rate divided by the chance rate $1-(1-f)^{10}$ of $s(d_i)$ appearing in a 10-token window and f is its frequency in the corpus.

two model pairs. The Qwen3 tables use about 2,000 prompts per domain and the DeepSeek tables about 400.

Table 7: Calibration corpora and number of table entries per domain for each model pair (Qwen3-14B with 4B AWQ, DeepSeek-R1-Distill-Qwen-32B with 7B AWQ).

Domain (benchmarks)	Calibration corpus	Qwen3	DeepSeek
Mathematics (MATH-500, Omni-MATH)	GSM8K train, MATH train	132	88
Knowledge (MMLU-Pro, GPQA)	MMLU validation, ARC-Challenge train	205	105
Logic (ZebraLogic, BBH)	BBH tasks not used in the benchmark	293	198
Code (HumanEval+, MBPP)	MBPP train and validation, CodeAlpaca	28	258
Instruction and writing (IFEval)	UltraChat first turns	721	167

function RESCUE(d_i, t_i) ▷ p_i, q_i and the context (text or code) are those of position i
  if EXCLUDED(d_i, t_i) **then return** false ▷ exclusion list of Section 3.2
  end if
  return $\big(p_i(d_i) \geq \theta_p \wedge q_i(t_i) \geq \theta_q\big) \ \vee \ (d_i, t_i) \in \mathcal{W} \ \vee \ (s(d_i), s(t_i)) \in \mathcal{O}_{\text{ctx}}$
end function

(a) Greedy decoding (T=0)

Require: $d_{1:k}$ with $q_{1:k}$; $p_{1:k+1}$
for $i = 1, \ldots, k$ **do**
  $t_i \leftarrow \arg\max_x p_i(x)$
  if $d_i = t_i$ **or** RESCUE(d_i, t_i) **then**
    emit d_i ▷ accepted or rescued
  else
    emit t_i; **return** ▷ round ends
  end if
end for
emit $\arg\max_x p_{k+1}(x)$ ▷ bonus token

(b) Sampling (T>0)

Require: the same, at temperature T
for $i = 1, \ldots, k$ **do**
  $u \sim \text{Uniform}(0, 1)$
  if $u > \min\big(1, p_i(d_i)/q_i(d_i)\big)$ **then**
    $t_i \sim \text{norm}\big(\max(0, p_i - q_i)\big)$
    if not RESCUE(d_i, t_i) **then**
      emit t_i; **return** ▷ round ends
    end if
  end if
  emit d_i ▷ accepted or rescued
end for
emit $x \sim p_{k+1}$ ▷ bonus token

Figure 3: One verification round with the three rules. RESCUE is shared by both variants: $\mathcal{W}$ is the whitelist, $\mathcal{O}_{\text{ctx}}$ the order-swap table for the current context (text or code), and EXCLUDED the exclusion list. (a) Greedy decoding; (b) sampling, where the rules examine the token drawn from the residual distribution.

A.3 Experiment details

A.3.1 Setup

Table 8 lists the number of problems per benchmark in the quality runs. Speed runs use batch size 8 and are repeated three times: at $T{=}0$, where greedy decoding is deterministic, on three disjoint subsets of 48 problems per benchmark, and at $T{=}0.6$ on the same 128 problems with three seed bases (1, 2, 3), from which each request derives its own seed; quality is measured once per temperature. Sampling uses the temperature alone, without top-p or top-k truncation, and the Qwen3 pair runs with its thinking mode disabled. During generation, the numbers of running and waiting requests and of generated tokens are sampled every 0.5 s, and the batch counts as full while all eight slots are occupied or requests are waiting. Quality runs use batch size 128 for the Qwen3 pair and 32 for the DeepSeek pair, whose long outputs need the memory. Generation is capped at 4,096 new tokens for the Qwen3 pair and at 16,384 for the DeepSeek pair, whose reasoning traces routinely exceed the former. The code benchmarks are decoded in the code context of Section 3.2: whitespace pairs are excluded, the prefix relation is off, and the order-swap tables are consulted for their code entries. The Qwen3 pair runs on a single NVIDIA RTX A6000 (48 GB); the DeepSeek pair runs with tensor parallelism over two GPUs, RTX A6000 for quality and for speed at $T{=}0.6$ and RTX 6000 Ada for speed at $T{=}0$, and speed ratios are always taken within one machine. For the DeepSeek pair at $T{=}0.6$, the target alone, whose runs take the longest, is measured in one of the three runs.

Table 8: Number of problems per benchmark in the quality runs.

Benchmark	Qwen3 pair	DeepSeek pair
MATH-500	500	200
Omni-MATH	500	200
MMLU-Pro	500	200
GPQA Diamond	198	198
ZebraLogic	300	200
BIG-Bench Hard (six tasks)	300	200
HumanEval+	164	164
MBPP	400	200
IFEval	400	200

A.3.2 PER-BENCHMARK RESULTS

Tables 9–12 give, for each model pair and temperature, the throughput and accuracy of the target alone, lossless speculative decoding, forced acceptance with $m \in \{2, 4, 8\}$, and our method on every benchmark of Section 4.1; Figure 4 plots the same configurations as throughput against accuracy, both relative to lossless decoding. One exception to the ordering of the forced configurations appears there: for the DeepSeek pair at T=0, m=8 is slower than m=4, because with every proposal accepted the outputs grow longer (19% more tokens than with m=4, and half again as many outputs reach the token cap) and every decoding step slows with the length of the context. Table 2 reports the ablation of the individual rules as averages over the benchmarks.

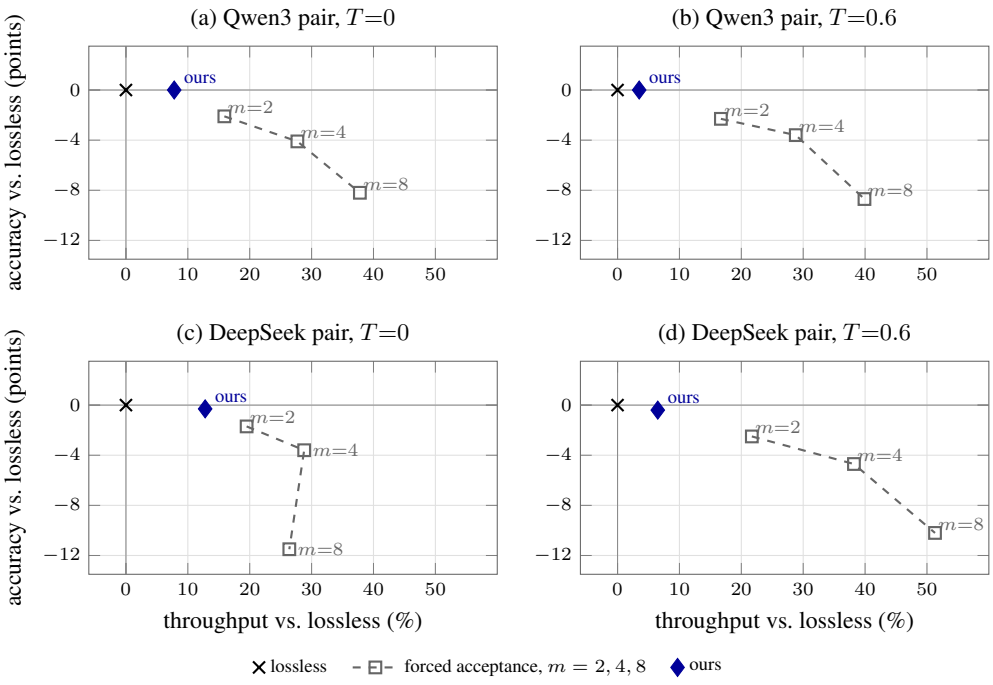

Figure 4: Throughput and accuracy relative to lossless speculative decoding, which sits at the origin (cross). Hollow squares joined by a dashed line: forced acceptance of the first m proposals; filled diamond: our method with all three rules. Both are averaged over the nine benchmarks of Section 4.1, throughput as a geometric mean of ratios and accuracy as a mean of points.

Table 9: Qwen3 pair, $T{=}0$ (throughput averaged over three runs on disjoint problem subsets). Throughput per benchmark at batch size 8, measured while the batch is full, and accuracy per benchmark; the last column is the geometric mean over benchmarks of the throughput ratio to lossless decoding, and the mean accuracy over the nine benchmarks. Bold: highest mean accuracy among the relaxed configurations (forced acceptance and ours); the target alone and lossless decoding are exact references.

Configuration	MATH	Omni	MMLU-P	GPQA	Zebra	BBH	HE+	MBPP	IFEval	mean
Throughput (tokens/s)										
target alone	183	184	182	181	174	186	178	178	187	0.49×
lossless	461	447	326	363	376	405	415	368	263	1.00×
forced $m{=}2$	509	498	408	433	412	469	428	387	385	1.16×
forced $m{=}4$	531	536	471	488	439	532	435	411	483	1.28×
forced $m{=}8$	569	561	530	528	458	569	439	420	607	1.38×
ours	504	496	359	403	408	452	407	368	291	1.08×
Accuracy (%)										
target alone	87.4	37.2	65.0	46.5	48.7	91.7	81.1	73.0	86.0	68.5
lossless	87.8	37.4	65.4	48.0	44.3	91.0	81.1	73.0	87.0	68.3
forced $m{=}2$	86.0	35.6	62.4	45.5	45.3	93.3	76.2	66.8	85.0	66.2
forced $m{=}4$	84.8	32.8	58.4	40.9	45.7	90.7	75.6	64.5	84.5	64.2
forced $m{=}8$	81.0	29.4	53.2	38.4	42.3	86.0	67.7	58.8	84.2	60.1
ours	86.0	35.4	64.8	49.0	44.7	95.0	80.5	74.0	85.8	**68.3**

Table 10: Qwen3 pair, $T{=}0.6$ (throughput averaged over three seeded runs). Throughput per benchmark at batch size 8, measured while the batch is full, and accuracy per benchmark; the last column is the geometric mean over benchmarks of the throughput ratio to lossless decoding, and the mean accuracy over the nine benchmarks. Bold: highest mean accuracy among the relaxed configurations (forced acceptance and ours); the target alone and lossless decoding are exact references.

Configuration	MATH	Omni	MMLU-P	GPQA	Zebra	BBH	HE+	MBPP	IFEval	mean
Throughput (tokens/s)										
target alone	182	183	181	181	174	184	177	176	185	0.50×
lossless	441	422	312	339	347	397	402	347	256	1.00×
forced $m{=}2$	480	468	395	415	394	467	417	372	369	1.17×
forced $m{=}4$	511	508	455	477	425	518	423	388	465	1.29×
forced $m{=}8$	533	540	512	530	459	568	434	396	566	1.40×
ours	456	442	326	355	360	416	407	352	263	1.04×
Accuracy (%)										
target alone	87.0	38.0	67.6	48.5	52.0	94.7	84.1	71.5	87.5	70.1
lossless	88.4	37.2	67.6	47.5	49.0	95.3	82.3	72.2	87.2	69.6
forced $m{=}2$	87.0	36.0	63.8	49.0	47.3	94.0	75.6	67.8	85.8	67.4
forced $m{=}4$	85.0	36.8	61.8	48.0	45.0	92.7	75.6	64.2	85.8	66.1
forced $m{=}8$	82.4	31.2	56.4	42.4	39.7	86.0	67.1	60.8	82.5	60.9
ours	88.6	38.0	66.6	50.5	50.3	94.0	81.7	72.5	85.0	**69.7**

Table 11: DeepSeek pair, $T=0$ (throughput averaged over three runs on disjoint problem subsets). Throughput per benchmark at batch size 8, measured while the batch is full, and accuracy per benchmark; the last column is the geometric mean over benchmarks of the throughput ratio to lossless decoding, and the mean accuracy over the nine benchmarks. Bold: highest mean accuracy among the relaxed configurations (forced acceptance and ours); the target alone and lossless decoding are exact references.

Configuration	MATH	Omni	MMLU-P	GPQA	Zebra	BBH	HE+	MBPP	IFEval	mean
Throughput (tokens/s)										
target alone	190	182	182	172	179	202	193	187	207	0.48×
lossless	492	411	353	358	331	478	421	431	310	1.00×
forced m=2	558	450	445	399	359	574	537	476	469	1.20×
forced m=4	594	456	508	423	365	655	590	466	565	1.29×
forced m=8	525	489	453	398	368	728	518	455	618	1.26×
ours	529	465	431	399	359	568	484	455	356	1.13×
Accuracy (%)										
target alone	88.5	62.5	64.5	49.0	65.5	99.0	68.3	80.0	33.5	67.9
lossless	89.0	59.5	63.0	48.5	67.5	98.0	67.7	83.0	33.0	67.7
forced m=2	92.5	60.0	57.5	44.4	58.0	97.5	73.8	77.5	32.5	66.0
forced m=4	87.0	59.0	53.5	46.0	44.5	96.0	80.5	78.5	32.0	64.1
forced m=8	87.5	54.0	35.5	37.9	26.5	91.0	77.4	63.0	33.0	56.2
ours	88.0	63.5	62.0	50.5	59.0	97.5	74.4	81.0	31.0	**67.4**

Table 12: DeepSeek pair, $T=0.6$ (throughput averaged over three seeded runs; the target alone is measured in one run). Throughput per benchmark at batch size 8, measured while the batch is full, and accuracy per benchmark; the last column is the geometric mean over benchmarks of the throughput ratio to lossless decoding, and the mean accuracy over the nine benchmarks. Bold: highest mean accuracy among the relaxed configurations (forced acceptance and ours); the target alone and lossless decoding are exact references.

Configuration	MATH	Omni	MMLU-P	GPQA	Zebra	BBH	HE+	MBPP	IFEval	mean
Throughput (tokens/s)										
target alone	143	139	143	137	133	153	149	143	153	0.50×
lossless	360	300	253	249	221	355	347	304	224	1.00×
forced m=2	391	346	333	312	252	422	406	364	333	1.22×
forced m=4	417	369	399	364	275	482	448	407	430	1.38×
forced m=8	425	403	460	394	290	531	483	420	540	1.51×
ours	376	320	269	267	233	383	366	321	244	1.06×
Accuracy (%)										
target alone	94.5	66.0	69.5	62.1	72.0	99.5	80.5	89.0	33.5	74.1
lossless	94.5	68.5	65.0	61.1	67.5	99.5	75.6	93.0	32.5	73.0
forced m=2	92.5	64.5	60.0	61.1	55.5	97.5	82.3	89.5	31.5	70.5
forced m=4	95.5	65.0	55.0	56.1	48.5	98.0	83.5	80.0	33.0	68.3
forced m=8	94.0	62.5	49.0	53.0	30.5	90.5	87.8	67.5	31.0	62.9
ours	95.0	66.5	66.0	61.6	62.0	96.0	82.9	91.0	33.0	**72.7**