Towards Green Text Generation: Memory-Based Speculative Decoding

Antal van den Bosch
Utrecht University
a.p.j.vandenbosch@uu.nl

Abstract

Speculative decoding accelerates text generation with large language models by letting a cheap draft model propose tokens that a large verifier checks in a single forward pass. We introduce Olifant, a draft model built on a memory-based classifier. It predicts the next token from a four-token window in under 0.3 ms on CPU, requiring no GPU, no backpropagation, and no accelerator memory. We measure throughput and energy under two verifier regimes on consumer hardware. Without a KV cache, drafting is nearly free: a domain-matched Olifant model reaches 38.1% acceptance on EU legal text, delivering 2.43× throughput and 2.51× lower energy per token than the neural verifier model alone. With a KV-cached verifier, itself a ∼10× energy saver, drafting costs matter: a TinyLlama draft model halves throughput and triples energy despite high acceptance, whereas Olifant still yields 1.25× throughput and the lowest energy in the study (93 J per 100 tokens, 14× below the uncached baseline). We further show that KV caching decouples time from energy: verification of longer drafts raises power draw, shifting the energy-optimal draft length below the speed-optimal one. Speculative decoding is not dead in the age of fast verifiers, provided drafting is nearly free.

1 Introduction

Text generation with large language models is sequential by nature: each token must be generated before the next, creating a bottleneck that limits throughput even on modern hardware. *Speculative decoding* (Leviathan et al., 2023; Chen et al., 2023) addresses this by pairing a cheap *draft model* with an expensive *verifier*. The draft model proposes k tokens into the future; the verifier processes all k+1 positions in one forward pass, accepting the longest prefix consistent with its own greedy predictions. When acceptance rates are high, many tokens are produced per verifier call, yielding wall-clock speedups.

Speed, however, is only half of the efficiency story. Local text generation is increasingly energy-bound: the energy cost of every verifier forward pass, not just its latency, determines the footprint of on-device generation. This constraint binds hardest where generation cannot leave the device at all: medical and legal text applications routinely rule out cloud APIs for confidentiality reasons, and it is precisely these domains where we find the largest efficiency gains. We therefore evaluate draft models on both axes, throughput and energy per generated token (Section 4), under two verifier regimes. In a transformer, the attention keys and values computed for already-processed tokens can be stored in a *KV cache*, so that each new token only requires computing attention for itself instead of re-encoding the entire sequence; maintaining this cache turns generation into cheap incremental steps at the price of holding per-token states in accelerator memory. We evaluate batch verification without a KV cache, where full-context attention dominates and draft tokens are nearly free, and KV-cached verification, where each draft position carries real cost. The two regimes give opposite answers to how long a draft should be, and to which draft models are viable at all. We find that a neural draft model that pays for itself against a slow verifier becomes a net loss against a fast one, while memory-based drafting, at ∼0.2 ms per token from CPU RAM, remains profitable in both regimes.

The design of the draft model determines whether speculative decoding pays off. Neural draft models from the same neural model type as the verifier achieve high acceptance by construction, but incur substantial inference cost themselves. Statistical n-gram models, in contrast, are nearly free to query but cannot generalize beyond observed sequences. Recent retrieval-based approaches (He et al., 2024; Yang et al., 2023) look up draft contin-

Proceedings of the 19th International Natural Language Generation Conference: Long Papers, pages 472–482
October 17-21, 2026 ©2026 Association for Computational Linguistics

uations in a corpus, trading memory for prediction quality: REST retrieves continuations whose recent context matches the generated suffix exactly, and inference-with-reference copies spans from documents already present in the prompt. Both draft only text that occurs verbatim in their source (we return to this comparison in Section 5.4).

We explore memory-based learning (Daelemans and van den Bosch, 2005) as a novel design point. Memory-based language modeling is rooted in two classic machine learning paradigms, k-nearest neighbor classification and decision-tree learning, and stores its training data in a prefix trie, a retrieval-oriented tree in which instances sharing feature values share a path. Specifically, we use TiMBL's IGTree algorithm (Daelemans et al., 1997, 2010),[1] which compresses a training set of (context window $\rightarrow$ next token) instances into such a trie with levels ordered by information gain, the most predictive context position at the top. IGTree classifiers are wrapped into a functional autoregressive decoder, codenamed Olifant (van den Bosch et al., 2025).

At query time a single path is traversed from the root, matching the query's context tokens level by level in this order, and the majority next token stored at the deepest matching node is returned. A context never seen verbatim in training is thus still answered, from the deepest matching prefix of its most informative positions; the remaining, less informative positions are ignored. The trie is additionally pruned at training time: nodes whose majority label agrees with that of their parent are not stored, typically shrinking the trie by about 95% while leaving lookups of stored contexts unchanged. Unlike neural models, Olifant requires no GPU and no backpropagation. The contrast with an in-session n-gram draft model (a baseline in this study) is of a different kind. The n-gram model is not trained at all: it continuously indexes the current session's context, exploiting recency and burstiness of tokens, whereas Olifant is a fully-fledged trainable language model in memory-based form, fixed at inference time and trained offline on a corpus of choice.

[1]TiMBL is open-source software: https://github.com/LanguageMachines/timbl

2 System Description

2.1 Olifant Draft Models

We pre-train four Olifant models used for drafting. Each training corpus is tokenised with the `mistralai/Mistral-7B-v0.1` SentencePiece tokeniser. Consecutive windows of width w=4 yield instances $(t_1, t_2, t_3, t_4) \rightarrow t_5$. The four models are matched to each evaluation domain of Section 3, plus an unmatched control. **olifant-wikitext** is trained on the WikiText-103 training split (Merity et al., 2017) with 130 million tokens,[2] whose test split supplies our general-text prompts; **olifant-pubmed** on the PubMed Summarization training split (Cohan et al., 2018) containing 536 million tokens,[3] matching the medical task; and **olifant-eurlex** on the EurLex training split (Chalkidis et al., 2022) of 94 million tokens,[4] matching the legal task. The fourth variant, **olifant-fineweb**, is trained on one shard of FineWeb-edu [5] web-crawled educational text that overlaps with none of the three test sources, containing 1.089 billion tokens; it serves as the largest model and domain-unmatched control, probing what an off-the-shelf general model offers without domain match. The resulting instance bases occupy 145 MB for the smallest model, eurlex, to 4.6 GB for the largest model, fineweb, of CPU RAM.

At inference time, at each speculative step, the last w context tokens are sent to Olifant, which returns its prediction. Instance bases are loaded lazily and released per strategy to save memory.

2.2 Baselines

We compare against three baseline models:

Plain. No draft model; the verifier is called once per token: the non-speculative throughput baseline.

4-gram. A sliding-window frequency model over the accumulated context, including tokens generated by the verifier in the current session. It therefore adapts online to the verifier's output.

[2]https://huggingface.co/datasets/Salesforce/wikitext

[3]https://huggingface.co/datasets/ccdv/pubmed-summarization

[4]https://huggingface.co/datasets/coastalcph/lex_glue, `eurlex` subset

[5]https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu

Neural. TinyLlama-1.1B-Chat-v1.0 (Zhang et al., 2024)[6] via HuggingFace Transformers. The neural draft is hosted on the MPS accelerator alongside the verifier and is evaluated in both full sweeps.

2.3 Draft Model Latency

We measure standalone next-token prediction latency with 50 single-token queries (10 prompts × 5 repetitions, no verifier involved): the 4-gram hash lookup answers in 0.09 ms, the Olifant variants in 0.16–0.25 ms, and TinyLlama-1.1B in 254 ms per token on CPU.

In other words, the 4-gram model and all Olifant variants predict the next token in under 0.3 ms, negligible against any verifier forward pass. TinyLlama needs a 1.1B-parameter forward pass per token; we host it on the accelerator alongside the verifier, costing ∼2 GB of accelerator memory.

2.4 Verifier and Acceptance

Mistral-7B-v0.1 base (Jiang et al., 2023)[7] is loaded via HuggingFace Transformers on a M1 Max GPU via MPS (`float16`). We evaluate two verification regimes.

In the regime Without KV caching, at each speculative step the full sequence (context + k draft tokens) is fed to the model in a single forward pass; the logits at positions $[\text{ctx}-1\,..\,\text{ctx}+k-1]$ are compared greedily against the draft tokens to determine the accepted prefix; i.e., k+1 token predictions are obtained at the cost of one forward pass. No KV cache is maintained across steps, so full-context attention dominates the cost of every step.

In the KV-cached regime, the cached decoder prefills the context once and maintains the KV cache across steps. Upon rejection, the cache is truncated to the accepted prefix. We made two design choices. First, the first draft token is verified against the previous step's logits, which have already been computed. Costs for the k-token verification pass only need to be incurred when the first draft token is accepted. Second, acceptance is decided at the token-ID level (exact greedy match), making the cached decoder output-preserving by construction. In exact arithmetic its output equals plain greedy decoding; under `float16` on MPS, rare shape-dependent kernel differences cause occasional divergences (Section 3.3).

[6] https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

[7] https://huggingface.co/mistralai/Mistral-7B-v0.1

3 Experiments

3.1 Setup

We evaluate on three tasks: (1) **General text**: 100 prompts from the WikiText-103 test split (first 50 words of each article as prompts; 100 tokens generated). (2) **Medical**: 100 prompts from the PubMed Summarization test split (first 50 words of each article as prompts; 100 tokens generated). (3) **Legal**: 100 prompts from the EurLex test split (first 50 words of each EU legislative document as prompts; 100 tokens generated).

We report mean tokens per second (tps), acceptance rate, mean accepted length (MAL), and ROUGE-L against a plain greedy reference. Energy metrics are defined in Section 4. The main text quotes selected cells; complete acceptance and MAL results for every strategy × task × k cell in both regimes are tabulated in Appendix A.

Each regime sweeps the same 7 strategies × $k \in \{3, 5, 8\}$ × 300 prompts = 6,300 generations. In the cached regime, wall-clock timestamps are recorded per generation and system power is logged in parallel, so that throughput and energy for this regime derive from a single run under identical conditions; uncached energy comes from a separate 63-configuration measurement grid (30 prompts per cell, Section 4). All experiments are run on the same Mac Studio (Apple M1 Max, 32 GB RAM) with the verifier on MPS.

3.2 Results: Uncached Verification

Table 1 reports mean throughput and speedup over plain generation; Figure 1 (left) plots the legal-text speedups against k. Nearly every strategy achieves a genuine wall-clock speedup on every task at every k value; the single exception is olifant-pubmed on general text at k=3 (0.96×), where the domain-mismatched model provides insufficient acceptance to offset its overhead. The domain-matched olifant-eurlex model on legal text reaches 2.43× at k=8 (19.9% acceptance) and 2.06× at k=3 (38.1% acceptance), by far the best results in the study. The domain-matched olifant-pubmed reaches 1.48–1.53× on medical text, and olifant-wikitext is the strongest Olifant variant on its matched general-text task (1.39–1.41×), completing the domain diagonal. The unmatched control olifant-fineweb never leads a task but is the best domain-unmatched

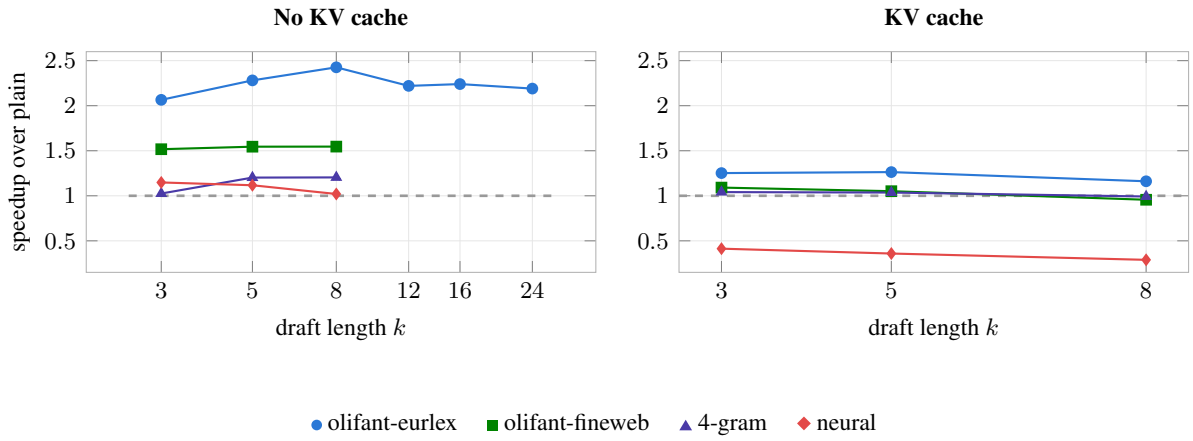


Figure 1: Speedup over plain decoding vs. draft length k on legal text, under the two verifier regimes (shared y-axis). Without a KV cache (left), draft tokens are nearly free and speedup grows with k; with a KV cache (right), each draft position has real verification cost and every curve slopes downward: profit requires high acceptance (domain-matched olifant-eurlex), and the expensive neural draft falls far below break-even (dashed line). $k \geq 12$ uncached points are from a preliminary `+D` sweep, speedups computed against the same plain baseline.

Olifant on all three. The 4-gram baseline benefits strongly from the no-KV-cache regime, reaching 1.42× on general text at k=8.

Also, speedup increases with k for olifant-eurlex on legal text: 2.06× at k=3, 2.28× at k=5, and 2.43× at k=8, despite acceptance rate falling from 38.1% to 19.9%. The neural draft moves in the opposite direction, from 1.15–1.20× at k=3 down to 1.01–1.07× at k=8: the only strategy whose speedup falls with k in this regime.

Without KV caching, speedup grows with k as appending k draft tokens adds negligible marginal cost, so any non-zero acceptance produces a net gain, and larger k is almost always better (cf. the analysis in Section 5.1). Olifant-eurlex illustrates this directly: acceptance falls from 38.1% to 19.9% as k grows from 3 to 8, yet mean accepted length grows from 1.13 to 1.53 tokens per step, driving throughput from 4.73 to 5.55 tps. The neural draft is the exception that proves the rule: its k draft tokens each require a sequential TinyLlama forward pass on the accelerator, so drafting is *not* free.

The gain from larger k is bounded, however. Extending the sweep to $k \in \{12, 16, 24\}$ for olifant-eurlex on legal text (Figure 1, left) shows throughput peaking at k=8 (2.43×) and settling near 2.2× beyond. Mean accepted length keeps growing (1.53→1.62 tokens/step), but the extra draft tokens are no longer negligible relative to the ∼150-token context, making k=8 the practical optimum.

Output quality is unaffected. Because verification is greedy, speculative decoding is output-preserving: an accepted draft token is by definition the verifier's own greedy choice at that position. Empirically, every strategy, including plain decoding, attains an identical mean ROUGE-L of 0.956 against an independently generated greedy reference, across all tasks and all k. The residual divergence from 1.0 stems from floating-point nondeterminism on MPS, and affects all strategies equally: the speed and energy gains reported here cost nothing in output quality.

3.3 Results: KV-Cached Verification

Table 2 reports the cached-regime sweep; Figure 1 (right) plots the legal-text curves on the same axis as the uncached regime.

KV caching lifts plain decoding from 2.3–2.8 to 16.3–16.9 tps. Prompt length ceases to matter: the 18% uncached gap between legal and general text collapses to 3%. Against this faster baseline, all speculative gains are lower, and decrease with k, the mirror image of the uncached regime. Domain-unmatched strategies land at break-even (0.9–1.1×): the rejection fast-path protects them from losses at k=3, but their 5–20% acceptance is too low to make a positive difference. Still, the domain-matched olifant-eurlex retains a margin on legal text: 1.25× at k=3, and 1.26× at k=5 (20.6 tps). Acceptance rates replicate across regimes (38.1% on legal text in both), confirming the draft model's behaviour is regime-independent.

TinyLlama, a genuine 1.15–1.20× accelerator in the uncached regime, drops to a slowdown of 0.47×–0.29×: each speculative step spends more time drafting k tokens sequentially than the cached

		tps			Speedup		
Strategy	k	Gen.	Med.	Legal	Gen.	Med.	Legal
plain	—	2.80	2.71	2.29	1.00×	1.00×	1.00×
ngram	3	3.72	2.74	2.35	1.33×	1.01×	1.03×
	5	3.88	3.40	2.75	1.39×	1.25×	1.20×
	8	**3.99**	3.47	2.75	**1.42**×	1.28×	1.20×
neural	3	3.35	3.16	2.63	1.20×	1.17×	1.15×
	5	3.24	2.99	2.56	1.16×	1.10×	1.12×
	8	2.99	2.74	2.33	1.07×	1.01×	1.02×
olifant-fw	3	3.71	3.93	3.47	1.32×	1.45×	1.52×
	5	3.74	4.00	3.54	1.34×	1.47×	1.54×
	8	3.65	3.95	3.54	1.30×	1.46×	1.55×
olifant-wiki	3	3.94	3.38	2.72	1.41×	1.25×	1.19×
	5	3.90	3.26	2.53	1.39×	1.20×	1.11×
	8	3.90	3.23	2.42	1.39×	1.19×	1.06×
olifant-pub	3	2.68	4.11	2.75	0.96×	1.51×	1.20×
	5	3.14	**4.16**	2.73	1.12×	**1.53**×	1.19×
	8	3.03	4.00	2.64	1.08×	1.48×	1.15×
olifant-eur	3	3.07	3.03	4.73	1.09×	1.12×	2.06×
	5	3.00	2.96	5.22	1.07×	1.09×	2.28×
	8	3.05	3.03	**5.55**	1.09×	1.12×	**2.43**×

Table 1: Throughput and speedup (local Mistral-7B verifier, no KV cache), 100 prompts per task. Speedup relative to plain. olifant-fw = olifant-fineweb; olifant-wiki = olifant-wikitext; olifant-pub = olifant-pubmed; olifant-eur = olifant-eurlex. Bold: best per task.

		tps			Speedup		
Strategy	k	Gen.	Med.	Legal	Gen.	Med.	Legal
plain	—	16.94	16.84	16.34	1.00×	1.00×	1.00×
ngram	3	**18.20**	17.72	17.02	**1.07**×	1.05×	1.04×
	5	17.47	17.36	16.93	1.03×	1.03×	1.04×
	8	17.73	16.88	16.22	1.05×	1.00×	0.99×
neural	3	7.92	7.51	6.75	0.47×	0.45×	0.41×
	5	6.83	6.33	5.87	0.40×	0.38×	0.36×
	8	5.40	5.00	4.72	0.32×	0.30×	0.29×
olifant-fw	3	17.35	17.87	17.84	1.02×	1.06×	1.09×
	5	16.56	17.01	17.18	0.98×	1.01×	1.05×
	8	15.06	15.44	15.63	0.89×	0.92×	0.96×
olifant-wiki	3	17.84	17.17	16.64	1.05×	1.02×	1.02×
	5	17.14	16.42	15.94	1.01×	0.98×	0.98×
	8	15.62	15.16	14.78	0.92×	0.90×	0.90×
olifant-pub	3	16.88	**18.28**	16.72	1.00×	**1.09**×	1.02×
	5	16.50	17.42	16.04	0.97×	1.03×	0.98×
	8	15.59	15.64	14.99	0.92×	0.93×	0.92×
olifant-eur	3	16.81	16.77	20.46	0.99×	1.00×	1.25×
	5	16.34	16.13	**20.64**	0.96×	0.96×	**1.26**×
	8	15.47	15.13	18.98	0.91×	0.90×	1.16×

Table 2: KV-cached verification results (local Mistral-7B verifier with persistent KV cache and rejection fast-path), 100 prompts per task. Speedup relative to cached plain. Abbreviations as in Table 1. Bold: best tps and speedup per task.

verifier needs to check them. Two-neural-model speculative decoding, the original design, is a net loss against a fast verifier on this hardware, while the memory-based draft model, three orders of magnitude cheaper per token, remains profitable where its acceptance is high.

Output quality again holds: cached speculative outputs score within 1.5 ROUGE-L points of cached plain decoding (0.955–0.959 vs. 0.971). ID-level greedy acceptance is output-preserving by construction; spot checks confirm byte-identical output. The residual gap is, again, due to `float16` kernel nondeterminism (k-token verification passes compute slightly different logits than k single-token passes).

4 Energy Efficiency

We measure whole-system power with macOS `powermetrics` (combined CPU, GPU, and Neural Engine power, sampled at 1 Hz) while each strategy generates. Energy per token is computed as $\mathrm{J}/100\,\mathrm{tok} = \bar{P} \cdot t_{\mathrm{wall}}/n_{\mathrm{tok}} \times 100$, where $\bar{P}$ is the mean power over the generation window. For the uncached regime, each strategy × task × k cell covers 30 prompts in a dedicated measurement run. For the cached regime, energy is measured during the throughput sweep: each generation carries a wall-clock timestamp, and per-block windows are joined against the parallel power log, so that throughput and energy for its 100 prompts per cell come from one run under identical conditions. Background system load was minimised during measurement; repeat measurements of the plain configuration agree within 2% in both regimes.

4.1 Uncached Regime

In the uncached regime, every draft strategy reduces energy per token on every task at every k (plain baseline: 1,005–1,326 J per 100 tokens), as shown in Table 3. The domain-matched Olifant models achieve the largest savings: olifant-eurlex on legal text cuts energy per token by **2.51**× (529 vs. 1,326 J per 100 tokens at k=8), and olifant-pubmed on medical text by 1.52×. The neural draft is strongest on general text (727 J, 1.40× at k=5), where TinyLlama's broad coverage yields the highest acceptance, but it is beaten by the domain-matched Olifant models on specialised text. The domain-unmatched olifant-fineweb still outperforms the 4-gram baseline in every single cell (1.27–1.58× vs. 1.16–1.33×).

		J/100 tokens			Energy ratio		
Strategy	k	Gen.	Med.	Legal	Gen.	Med.	Legal
plain	3	1005	1097	1313	1.00	1.00	1.00
	5	1019	1086	1326	1.00	1.00	1.00
	8	1019	1095	1326	1.00	1.00	1.00
ngram	3	827	930	1120	1.21	1.18	1.17
	5	798	930	1099	1.28	1.17	1.21
	8	767	945	1102	1.33	1.16	1.20
neural	3	738	865	1016	1.36	1.27	1.29
	5	**727**	854	989	**1.40**	1.27	1.34
	8	760	924	1046	1.34	1.19	1.27
olifant-fw	3	788	781	875	1.27	1.40	1.50
	5	768	788	839	1.33	1.38	1.58
	8	785	820	845	1.30	1.34	1.57
olifant-wiki	3	815	916	1054	1.23	1.20	1.25
	5	850	956	1064	1.20	1.14	1.25
	8	828	916	1091	1.23	1.20	1.21
olifant-pub	3	861	734	1081	1.17	1.49	1.21
	5	918	**718**	1100	1.11	1.51	1.21
	8	938	721	1102	1.09	**1.52**	1.20
olifant-eur	3	908	940	617	1.11	1.17	2.13
	5	919	961	550	1.11	1.13	2.41
	8	960	981	**529**	1.06	1.12	**2.51**

Table 3: Energy per generated token (J/100 tokens; lower is better) and energy ratio vs. plain at the same k (higher is better), measured with `powermetrics` on Apple Silicon; 30 prompts per cell, local Mistral-7B verifier, no KV cache. Plain rows are repeat measurements (k unused). Bold: best per task.

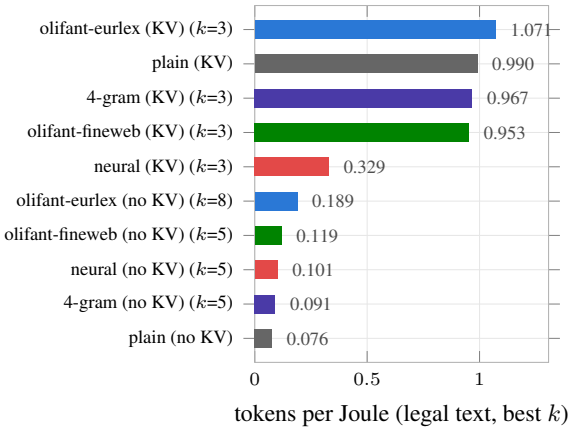


Figure 2: Energy efficiency ladder on legal text (tokens generated per Joule, best k per strategy). KV caching is the largest single lever; domain-matched memory-based drafting adds to it, while the neural draft costs energy in the cached regime.

Across all 63 uncached configurations, mean system power stays within a narrow band (16–31 W, median ≈28 W): no strategy meaningfully raises power draw over plain decoding. Energy per token is therefore dominated by wall-clock time, and for the CPU-side drafts, whose sub-millisecond lookups add no measurable power, the uncached energy ratios track the throughput speedups of Table 1 almost directly. The neural draft deviates in the favourable direction: its energy ratios (1.19–1.40) run ahead of its speedups (1.01–1.20), because sequential TinyLlama drafting leaves the accelerator less saturated than the full-context verifier passes it replaces (23.6 vs. 28.3 W mean). As the next subsection shows, this coupling of time and energy is a property of the uncached regime, not a general law.

4.2 Cached Regime

Cached plain decoding costs 96–106 J per 100 tokens (Table 4), one tenth of the uncached 1,005–1,326 J. The saving exceeds the 6–7× throughput gain, because power also drops (16.5 W vs. ≈28 W): one-token steps leave the accelerator less saturated than full-context passes. Deployments that can afford KV caching should use it; no draft model comes close to what caching saves.

In the cached regime, power is no longer constant across strategies: mean power draw rises with draft length, from 16.4 W (plain) to 22.1 W (olifant-eurlex, k=8), because dense k-token verification passes work the accelerator harder than one-token steps. Consequently, time and energy deviate. Olifant-eurlex on legal text at k=8 is 1.16× faster than plain, yet spends 1.18× more energy; at k=3 it is both 1.25× faster and 1.09× more energy-efficient (93 J per 100 tokens, the study's lowest). The energy-optimal draft length is thus shorter than the speed-optimal one. The three k-optima characterize the regimes: uncached speed favours k=8, cached speed k=3–5, cached energy k=3.

TinyLlama, in contrast, adds markedly to the energy usage of the verifier: from a 1.2–1.4× energy saver in the uncached regimen to a 2.5–3.8× energy cost in the cached regimen (243–382 J per 100 tokens). Both factors degrade at once (doubled wall time; a second neural network occupying the accelerator), and the deficit deepens with k.

Figure 2 visualizes the energy efficiencies of various models in terms of tokens per Joule: cached configurations cluster an order of magnitude above uncached ones, with cached olifant-eurlex (k=3, 1.07 tok/J) at the top and the neural draft as the only strategy that moves *down* the ladder within its regime.

		J/100 tokens			Energy ratio		
Strategy	k	Gen.	Med.	Legal	Gen.	Med.	Legal
plain	3	96	97	102	1.00	1.00	1.00
	5	97	100	106	1.00	1.00	1.00
	8	96	97	101	1.00	1.00	1.00
ngram	3	**96**	98	103	**1.00**	0.99	0.99
	5	122	108	110	0.80	0.92	0.96
	8	109	111	115	0.88	0.87	0.88
neural	3	243	264	303	0.40	0.37	0.34
	5	272	298	330	0.36	0.34	0.32
	8	334	362	382	0.29	0.27	0.26
olifant-fw	3	109	103	105	0.88	0.94	0.97
	5	121	116	116	0.81	0.86	0.92
	8	136	132	132	0.71	0.73	0.76
olifant-wiki	3	100	103	107	0.96	0.95	0.96
	5	110	112	115	0.89	0.90	0.92
	8	129	128	132	0.74	0.76	0.77
olifant-pub	3	100	**99**	104	0.96	**0.98**	0.98
	5	106	111	113	0.92	0.90	0.94
	8	118	132	127	0.81	0.73	0.79
olifant-eur	3	100	102	**93**	0.97	0.95	**1.09**
	5	107	110	100	0.91	0.91	1.06
	8	118	124	118	0.81	0.78	0.85

Table 4: Cached-regime energy (J/100 tokens; lower is better) and energy ratio vs. cached plain at the same k, derived from the timestamped throughput sweep joined with the 1 Hz power log; 100 prompts per cell. Only high-acceptance domain-matched drafting (olifant-eurlex on legal, $k \leq 5$) saves energy; ratios below 1.00 indicate an energy cost despite any speed gain. Bold: best per task.

4.3 Training Energy

The figures above cover inference; a green-generation claim should also count the energy needed to build the draft models. Training an Olifant model is a single CPU pass over the corpus: tokenisation, windowing, and IGTree trie compilation, with no backpropagation and no accelerator. Compiling the largest instance base (fineweb, 1.089 billion tokens) took about six hours on the evaluation machine, roughly half a megajoule at the tens of watts it draws for CPU-bound work: a one-time expense recouped within roughly a thousand 100-token generations at the savings of Table 3; the smaller corpora cost less. The neural draft model sits at the other end of the scale: TinyLlama was pretrained on 3 trillion tokens on 16 A100 GPUs over about 90 days (Zhang et al., 2024), on the order of 10^{10} J counting GPU board power alone, four to five orders of magnitude above the memory-based models. Pretrained models amortise this cost across many downstream uses, but the contrast stands: the lifecycle energy of a memory-based draft model is dominated by inference, not training.

5 Discussion

5.1 The Verifier Cost Model

The standard break-even condition for speculative decoding is:

$$\text{acc_rate}_{\text{break-even}} \approx \frac{k}{k+1} \tag{1}$$

If verifying k draft tokens costs as much as generating them one by one, a speculative step only pays off when nearly all of its draft tokens are accepted, and the required per-token acceptance rises with the draft length: 3 out of 4 or 75% at k=3, 89% at k=8.

Without KV caching, transformer attention over the full context dominates each forward pass, and the marginal expense of appending k draft positions is negligible. The break-even threshold collapses to near zero: any non-zero acceptance yields a speedup, larger k is better up to a plateau (Section 3.2), and every uncached energy ratio exceeds 1.0. This regime matches any deployment where the verifier is re-invoked on the full sequence (stateless serving, or devices that cannot afford a cache), and it is forgiving of low-acceptance drafts.

With KV caching, Equation 1 would predict that none of our draft models survive: no strategy approaches 75% acceptance. Two effects soften the prediction. The rejection fast-path heuristic (Section 2.4) means failed steps cost no more than plain steps, so a k-token verification pass can only achieve gains in extra accepted tokens on steps in which the first token is accepted. Empirically, the resulting threshold sits near 20–25% first-token acceptance: below it, strategies flatline at break-even (Table 2); above it, which only olifant-eurlex on legal text reaches at 38%, a genuine margin appears. TinyLlama's sequential drafts cost more than the cached verification they enable, at any acceptance.

5.2 Speculative Decoding is Not Finished

Against slow verification, the original two-model speculative decoding design still performs as predicted; our uncached results reproduce this. Verifier-side optimization (KV caching here; batching, quantization, and faster kernels generally) shrinks the budget available for drafting, however. The field's general trajectory away from sep-

arate draft LMs toward ever-cheaper drafting (attached heads, retrieval, prompt lookup) concedes the point: our cached results show the separate neural draft losing 2–3× on the same hardware where it once profited. Memory-based drafting, on the other hand, occupies the far end of that trajectory: sub-millisecond, CPU-resident, verifier-agnostic (any model sharing the tokeniser), and domain-adaptable by swapping an instance base rather than retraining. It is the only draft design in this study that remains profitable in both regimes. Speculative decoding is not finished as verifiers get faster; drafting merely has to become nearly free, and memory-based learning is one way to get there.

5.3 Olifant vs. In-session 4-gram: Related but Distinct

The in-session 4-gram model we compared against in this study is not a trained model: it builds its frequency table from the accumulated context of the current session, including tokens generated by the verifier. It therefore automatically adapts to the verifier's output distribution, and to the recency and burstiness of words in the running text, effectively acting as an online imitator. Olifant is the opposite design: a trained language model in memory-based form, fixed at inference time, with no mechanism to incorporate session-level feedback. This explains why the 4-gram baseline competes well on domains where no Olifant model was trained (Table 1): online adaptation substitutes for static acceptance.

A second difference is that Olifant operates natively on Mistral's SentencePiece tokens, the same units the verifier uses, so draft predictions slot directly into the verification step with no representation mismatch. The 4-gram model predicts whitespace-delimited words, of which only the first sub-token survives the text-fallback path; this structural mismatch caps its effective acceptance regardless of word-level accuracy.

On domain-matched tasks, Olifant's offline training pays off decisively (38.1% vs. 24.7% acceptance on legal text; 2.43× vs. 1.20× speedup at k=8). On unmatched domains, the 4-gram's online adaptation closes most of the gap. Two extensions suggest themselves toward the best of both approaches: training Olifant directly on the verifier's output distribution, and augmenting the fixed instance base with a recency/burstiness component that, like the 4-gram baseline, tracks the current session's context. Preliminary experiments with such a hybrid are ongoing; we leave a full treatment to future work.

5.4 Olifant vs. Retrieval-Based Drafting

Olifant is also distinct from the retrieval-based drafting methods cited in Section 1. REST (He et al., 2024) keeps the draft corpus as a suffix-array datastore and, at each step, retrieves continuations whose recent context matches the generated suffix exactly; inference-with-reference (Yang et al., 2023) copies spans from reference documents already present in the prompt, targeting settings with high prompt-output overlap such as retrieval-augmented generation. Both are exact-match designs: when matching, they draft long verbatim spans, and when the current context does not occur in their source, they draft nothing. Olifant instead compiles the corpus into a next-token predictor. The IGTree trie discards about 95% of the instance base at training time, its depth bounded by the context window rather than the corpus size. Its information-gain-ordered prefix backoff returns a prediction for every context, including contexts never seen verbatim in training, by ignoring the least informative window positions. The price is that Olifant drafts one token per query rather than a retrieved span. In machine learning terms, IGTree is itself a decision-tree compression of a k-nearest-neighbor learner (Section 1), so Olifant can be read as a decision-tree draft model with retrieval functionality. An empirical head-to-head with REST and inference-with-reference on the same hardware and both verifier regimes is an obvious next step (Section 5.6).

5.5 Method or Text? Attributing the Domain-Match Advantage

Olifant-eurlex's lead on legal text raises the question whether the advantage reflects the memory-based method or merely properties of legislative text, such as local repetition. To calibrate, we measure the redundancy of the evaluation texts themselves. Decoding is output-preserving, so every strategy produces the same text for a given prompt; the statistics below therefore describe the exact texts on which all acceptance rates were computed. Table 5 reports three measures over each task's 100 generations, computed on Mistral tokens to match the draft models' context window: the *within-text* repetition rate (the fraction of a text's token 4-grams that already occurred earlier in the same text), the *cross-document* overlap (the fraction of a text's distinct 4-grams that also occur in at least

Task	Within-text	Cross-doc.	gzip
General	29.6%	26.7%	0.29
Medical	21.2%	14.9%	0.33
Legal	12.1%	73.6%	0.18

Table 5: Redundancy of the evaluation texts (100 generations per task, Mistral tokens). Within-text: mean fraction of a text's token 4-grams already seen earlier in the same text. Cross-doc.: mean fraction of a text's distinct 4-grams that also occur in another text of the same task. gzip: compression ratio of the task's concatenated text (lower = more redundant).

one other text of the same task), and the gzip compression ratio of the task's concatenated text as a model-free summary.

Within documents, legal text is the least repetitive of the three tasks (12.1% vs. 21.2% medical and 29.6% general), so within-generation repetition cannot explain the acceptance ordering. In contrast, 73.6% of a legal text's distinct 4-grams recur in other legal texts, against 14.9% and 26.7% elsewhere, and the concatenated legal texts compress to 0.18 of their size. This cross-document formulaicity is what offline training on a domain corpus captures; it is the mechanism by which memory-based drafting profits, not a confound of it.

The contribution of the method, as opposed to the text, is isolated by comparing draft models on identical legal texts: at k=3, acceptance spans 7.5% (olifant-pubmed) to 38.1% (olifant-eurlex), with the 4-gram baseline at 24.7% and the neural draft at 21.1% (Appendix A). The text is held constant across these rows; only the draft model changes. Both factors are thus real and separable: tasks differ in how much cross-document redundancy there is to harvest, and draft models differ in how much of it they capture.

5.6 Future Work

Several axes are deliberately left open. First, a direct empirical comparison with retrieval-based drafting (REST and inference-with-reference, Section 5.4) on the same hardware and under both verifier regimes. Second, verifier diversity: repeating the sweeps with verifiers of other sizes and families would test how far the cost-model analysis of Section 5.1 carries; the same holds for server-class GPUs, where faster verification should sharpen the cached-regime compression we observe. Third, sampling and batching: speculative sampling (Leviathan et al., 2023; Chen et al., 2023) extends verification losslessly to temperature sampling, at acceptance rates that shift with temperature, and batched serving changes the drafting budget again. Fourth, generalization: evaluating domain-matched models on in-domain corpora from sources other than the training collection. Fifth, the session-adaptive hybrid trailed in Section 5, combining a trained instance base with an online recency component. Finally, scaling the instance bases: eurlex, the smallest corpus in the study at 94M tokens, produced the best draft model, leaving the scaling axis unexplored.

6 Conclusion

We have presented Olifant, a speculative decoding draft model based on TiMBL memory-based learning, and evaluated it on general, medical, and legal text against 4-gram and neural baselines under two verifier regimes, measuring both throughput and energy on consumer hardware.

Without a KV cache, drafting is nearly free, and nearly every strategy achieves a genuine wall-clock speedup; only the neural draft's gain decays with draft length. The domain-matched legal model delivers 2.43× throughput and a matching 2.51× energy saving. With a KV cache, the baseline itself accelerates 6–7× at one tenth the energy, and the rules change: gains diminish, the useful draft length shrinks, and drafting cost become a prominent factor. The neural draft model loses 2–3× in both time and energy despite high acceptance, while the memory-based draft model keeps a 1.25× positive throughput margin where its acceptance is high and reaches the lowest energy cost in the study (93 J per 100 tokens). The cached regime decouples time from energy: verifying longer drafts raises power draw, so the energy-optimal draft length sits below the speed-optimal one. Greedy verification preserves output quality in both regimes, up to floating-point nondeterminism in the cached decoder.

In sum, verifier-side optimization keeps shrinking verification costs, so drafting must approach zero cost to still make a difference. Memory-based drafting, at 0.2 ms per token from CPU RAM with no accelerator footprint, meets that bar today.

Limitations

All experiments run on a single hardware platform (a Mac Studio[8] with an Apple M1 Max chip and 32 GB RAM, verifier on MPS in `float16`) with a single verifier (Mistral-7B-v0.1). On a server-class GPU, per-token verifier latency would drop further (against a higher energy footprint) and the relative cost of drafting rises, so the cached-regime compression we measure would likely sharpen. Energy figures are whole-system measurements on consumer hardware with background load minimised; they support relative comparisons on identical hardware, not absolute cross-platform claims. All measurements use greedy verification at batch size one; how the reported margins shift under temperature sampling and batched serving remains to be established (Section 5.6). For the domain-matched models, evaluation prompts come from held-out test splits of the same collections used for training (Section 2); transfer to in-domain text from other sources is untested.

All datasets and pretrained models are available from the Hugging Face hub (footnotes at first mention), and TiMBL and Olifant are open-source. Our implementation and experimental scripts are released at `https://github.com/antalvdb/olifant-speculative`, and the four trained instance bases at `https://huggingface.co/antalvdb/olifant-speculative-ibases`.

References

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, Dublin, Ireland. Association for Computational Linguistics.

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

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Walter Daelemans and Antal van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press.

Walter Daelemans, Antal Van den Bosch, and Ton Weijters. 1997. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2010. TiMBL: Tilburg memory based learner, version 6.3, reference guide. Technical Report ILK 10-01, ILK Research Group.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1582–1595, Mexico City, Mexico. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286. PMLR.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of ICLR 2017*.

Antal van den Bosch, Ainhoa Risco Patón, Teun Buijse, Peter Berck, and Maarten van Gompel. 2025. Memory-based language models: An efficient, explainable, and eco-friendly approach to large language modeling. *Preprint*, arXiv:2510.22317.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with reference: Lossless acceleration of large language models. *Preprint*, arXiv:2304.04487.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

[8] Energy was supplied by a solar panel and home battery combination, requiring almost no grid electricity.

A Complete Acceptance and MAL Results

Tables 6 and 7 report the acceptance rate and mean accepted length for every draft model × task × draft length cell, in the uncached and KV-cached verifier regimes respectively, complementing the selected cells quoted in the main text. Acceptance is the fraction of drafted tokens accepted by the verifier, averaged over the 100 prompts of a task; MAL is the mean number of draft tokens accepted per verification step. As k grows, per-token acceptance declines while MAL rises: longer drafts convert a smaller share of their tokens, but still yield more accepted tokens per verifier call, which is why throughput can improve with k wherever drafting is cheap.

	k=3		k=5		k=8	
Draft model	Acc.%	MAL	Acc.%	MAL	Acc.%	MAL
General						
4-gram	22.1	0.31	15.7	0.35	11.0	0.39
neural	25.2	0.75	17.5	0.86	11.9	0.92
olifant-fineweb	12.2	0.36	7.7	0.38	4.9	0.38
olifant-wikitext	14.8	0.44	9.5	0.47	6.1	0.47
olifant-pubmed	5.1	0.15	3.2	0.15	2.0	0.15
olifant-eurlex	4.8	0.14	2.9	0.14	1.9	0.14
Medical						
4-gram	16.4	0.22	10.9	0.24	7.3	0.26
neural	23.3	0.69	15.6	0.77	10.5	0.81
olifant-fineweb	16.0	0.48	10.5	0.51	6.9	0.53
olifant-wikitext	9.0	0.27	5.6	0.28	3.6	0.28
olifant-pubmed	18.2	0.54	11.8	0.58	7.7	0.59
olifant-eurlex	5.8	0.17	3.6	0.17	2.3	0.18
Legal						
4-gram	24.7	0.19	16.3	0.20	10.9	0.21
neural	21.1	0.63	15.0	0.73	9.9	0.76
olifant-fineweb	18.8	0.56	12.3	0.60	8.2	0.63
olifant-wikitext	8.6	0.25	5.3	0.26	3.4	0.26
olifant-pubmed	7.5	0.22	4.6	0.22	2.9	0.22
olifant-eurlex	38.1	1.13	28.3	1.38	19.9	1.53

Table 6: Complete acceptance rates (%) and mean accepted lengths (MAL, tokens per verification step) for all draft models, tasks, and draft lengths k, **uncached** verifier regime (100 prompts per task).

	k=3		k=5		k=8	
Draft model	Acc.%	MAL	Acc.%	MAL	Acc.%	MAL
General						
4-gram	22.2	0.31	15.8	0.36	11.1	0.40
neural	25.2	0.75	17.5	0.86	11.9	0.92
olifant-fineweb	12.1	0.36	7.7	0.38	4.9	0.38
olifant-wikitext	14.9	0.44	9.5	0.47	6.1	0.47
olifant-pubmed	5.1	0.15	3.2	0.15	2.0	0.16
olifant-eurlex	4.8	0.14	3.0	0.14	1.9	0.15
Medical						
4-gram	16.4	0.22	10.9	0.24	7.3	0.26
neural	23.3	0.69	15.6	0.76	10.4	0.81
olifant-fineweb	16.0	0.48	10.5	0.51	6.8	0.53
olifant-wikitext	9.0	0.27	5.6	0.28	3.6	0.28
olifant-pubmed	18.3	0.54	11.9	0.58	7.7	0.59
olifant-eurlex	5.9	0.18	3.6	0.18	2.3	0.18
Legal						
4-gram	24.8	0.19	16.3	0.20	10.9	0.21
neural	21.2	0.63	15.1	0.74	10.0	0.77
olifant-fineweb	18.7	0.56	12.3	0.60	8.2	0.63
olifant-wikitext	8.6	0.25	5.3	0.26	3.4	0.26
olifant-pubmed	7.5	0.22	4.6	0.22	2.9	0.22
olifant-eurlex	38.1	1.13	28.3	1.38	20.0	1.54

Table 7: Complete acceptance rates (%) and mean accepted lengths (MAL) for all draft models, tasks, and draft lengths k, **KV-cached** verifier regime (100 prompts per task).