Context-Aware Speculative Decoding: Accelerating Inference in High-Copy Environments

Gianni Van de Velde
gianni.vandevelde@ugent.be

Cédric Goemaere
cedric.goemaere@ugent.be

Chris Develder
chris.develder@ugent.be

Thomas Demeester
thomas.demeester@ugent.be

Abstract

In recent years, Large Language Models (LLMs) have proven to be useful for many tasks, leading to wide adoption. Yet, in pace with the increasing model sizes, energy consumption went up, whereas inference speeds went down. Speculative decoding is a proven method that can alleviate both issues. It parallelizes the decoding of the next tokens by speculating multiple decoding steps. Many speculators have been made to try and predict what the generator might generate next. This paper suggests a new method, Context-Aware Speculative Decoding (CASD), which is a hybrid method that augments the current state-of-the-art (SOTA) with a simple statistical method. It builds on the intuition that for some applications, phrases from the context may literally contribute to the final answer. Tests on Llama 3 (Grattafiori et al., 2024) using EAGLE-2 (Li et al., 2024) show that CASD speeds up decoding with 8-17% for the Retrieval-Augmented Generation (RAG) use cases. This while adding limited overhead, seen by a slowdown of only 4% when applied to general benchmarks. In the future, we see great potential for CASD to augment speculative decoders in RAG use cases, certainly in low-resource languages.

1 Introduction

In recent years, the scaling of Large Language Models (LLMs) improved their capabilities tremendously. However, growing model sizes comes at a cost. Among others, inference time goes up and so does the energy consumed at inference. To compensate for these added costs, speculative decoding emerged as a way to break the autoregressive nature of generating text with an LLM. Speculative decoding tries to parallelize LLM calls, to reduce the time and energy necessary to generate an answer (Qin et al., 2024). In practice, this means that a small model drafts many samples of what the generator could generate in the coming tokens. These drafts are then verified in parallel, in one forward pass of the model. So it becomes possible to generate multiple tokens with only one forward pass. This method has already been shown to reduce inference time and energy.

Many small draft models have been proposed to guess the next generated tokens. There are two large categories of draft models: statistical models (He et al., 2023; Yang et al., 2023) and neural models (Leviathan et al., 2023; Cai et al., 2024; Li et al., 2024). This paper presents a new model-agnostic statistical model: Context-Aware Speculative Decoding (CASD). It is naturally suited to augment neural models, and as such achieves a new state-of-the-art for specific use cases. To the best of our knowledge, we are the first to combine the drafts of statistical and neural models.

CASD leverages the context given in the prompt to make good speculations. It checks whether the LLM is copying text from the context and if so, it provides the continuation as found in the context. This method works best if the generator is expected to copy many passages from the context. With the recent popularity of Retrieval-Augmented Generation (RAG) applications, this is a reasonable assumption. We test CASD's performance by using it to augment the current SOTA, EAGLE-2 (Li et al., 2024). For the RAG benchmarks, CASD showed a 1.08 speedup on SQuAD (Rajpurkar et al., 2016) and 1.17 on our private UZGentRAG benchmark compared to EAGLE-2. CASD was also tested on the widely used benchmarks for speculative decoding: Alpaca (Taori et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), MT-bench (Zheng et al., 2023), Natural Questions a.k.a. QA (Kwiatkowski et al., 2019) and CNN/DM (Nallapati et al., 2016). Since these general speculative decoding benchmarks do not use RAG, we naturally see no inference speedup. Yet, we validate that even in these cases, the overhead introduced by adopting CASD remains minimal.

Besides the added performance in RAG-based

use cases, CASD offers the following advantages:

- **LLM agnostic.** While modern speculators are usually trained to work well with one LLM, CASD can be plugged into any LLM for generating long, high quality speculations.
- **Independent predictions.** Today, neural models are the standard of speculative decoding. These models typically have good performance for the initial upcoming tokens, but each step accumulates noise compared to what the actual LLM would output. In contrast, CASD works in a fundamentally different way, also generating high quality long length speculations. This is a built-in independence, which makes it likely that a CASD speculation is not present in the drafts of the base speculator. Thus, CASD will probably augment future speculators well too.
- **Training-free.** CASD requires no training of a model and works out of the box with any input data. This means it does not require retraining if it is applied to new language or a new niche domain.

This paper is structured as follows: Section 2 starts with some preliminaries necessary to understand Section 3, where we introduce our new method CASD. Section 4 then details the experiments performed on CASD. In Section 5, the related work is briefly discussed and finally, Section 6 forms a conclusion for this paper.

2 Preliminaries

As this paper augments neural speculators with a statistical method, it is important to first understand how speculators work, what their strengths are and their weaknesses.

2.1 Speculative Decoding

Speculative decoding is a technique to generate the same number of tokens with less forward passes through the LLM. This not only reduces the time to generate, but also the energy consumption (Qin et al., 2024). To achieve this goal, the speculator, which is a model to come up with candidate answers, starts with generating such answers, shortly referred to as drafts. Afterwards, in the verification phase, all drafts are sent through the LLM in one forward pass, essentially parallelizing the workload for more efficiency. This makes it possible to break the autoregressive nature, as the generator can attend to draft tokens and thus verify tokens multiple steps in the future. After verification, the real LLM distributions are known for each draft token. Standard autoregressive sampling can be simulated on those distributions and as long as the sampled token was already in the draft tokens, the next distribution is available. This means potentially multiple consecutive tokens can be added to the actual answer under construction, within the same forward pass. This process is repeated until the stop condition is met, progressing multiple tokens per step rather than just one.

2.2 EAGLE-2

We use the current state-of-the-art speculative decoder, EAGLE-2, as a baseline throughout this paper. Hence, we introduce it briefly below.

Most research in speculative decoding is about the drafting method, as this is the component with most freedom of implementation. EAGLE-2 innovates the drafting method by trying to extrapolate the embedding of the LLM autoregressively. This is a task that a neural network can be trained to do, and it seems to be an easier task than extrapolating tokens, allowing for smaller networks. The intuition is that the embedding layer contains the "reasoning" or "thought" of the model and thus EAGLE-2 has more valuable information as input than a standard token-predicting model. This is further clarified in Figure 1.

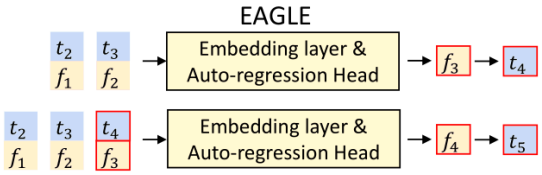

Figure 1: EAGLE-2 uses the LLM's features and previous tokens to predict the next step. It is used autoregressively, to draft multiple tokens ahead. (Li et al., 2024)

Although EAGLE-2 achieves state-of-the-art results in speculative decoding, there are some limitations:

- **Poor out-of-distribution performance.** Neural networks tend to perform badly when the input is completely different from the training data (out-of-distribution). For neural speculators, this is the case when the task contains specific jargon or a different language.

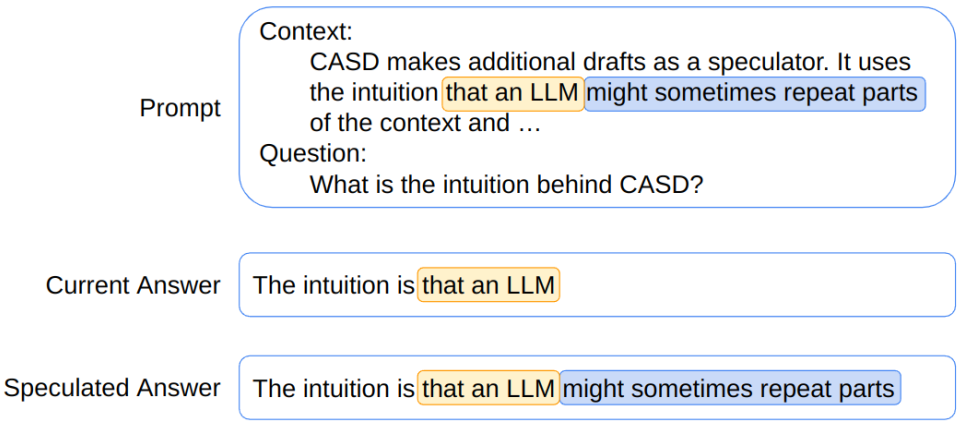

Figure 2: CASD applied to a simplified RAG prompt. The prefix ("that an LLM") is found in the context and thus the continuation ("might sometimes repeat parts") is drafted by CASD.

- **Drift.** EAGLE-2 autoregressively drafts tokens, so a token five steps ahead already contains the accumulated noise of five extrapolations. In practice, this means the drafts at many steps into the future are already of much lower quality and have lower chances of being accepted.

3 Context-Aware Speculative Decoding

CASD makes additional drafts as a speculator. It uses the intuition that an LLM might sometimes replicate parts of the context. Thus, when it has already copied a small part, it might continue copying further tokens, as illustrated in Figure 2. To put this intuition into practice, the first step is to detect when the LLM might be copying literally from the context. This is easily done by substring-matching the last generated tokens with any part of the context. To limit the overhead, only the top-k matches are kept, based on the length of the match. For each of these matches, the continuation in the context is added as a draft.

While CASD itself is not the best speculator, it can augment SOTA speculators to achieve even better performance. In a practical setup, the SOTA speculator is asked to generate its drafts and then CASD adds some more drafts. After that, the default speculative decoding is performed, calling the LLM for each token and sampling from the distribution.

4 Experimental validation of CASD

In this section, we provide details about the experimental setup and benchmarks used, before going into the results. This is followed by further analysis on the acceptance length and an ablation study.

4.1 Experimental setup

Models. The experiments include only Llama-3-8B-Instruct (Grattafiori et al., 2024) as an LLM, as only this model fits our following constraints. The first constraint is that of hardware: 8B is the largest model that fits on the used GPU. Since one of our benchmarks has Dutch content (UZGentRAG, see further), we also need a Dutch-capable LLM. Lastly, the SOTA baseline, EAGLE-2 also has to support the LLM. This yielded Llama-3-8B-Instruct as the chosen model for all experiments.

Tasks. CASD is evaluated on UZGentRAG benchmark (see further), SQuAD (Rajpurkar et al., 2016) and the classical speculative decoding benchmarks (Alpaca (Taori et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), MT-bench (Zheng et al., 2023), Natural Questions a.k.a. QA (Kwiatkowski et al., 2019) and CNN/DM (Nallapati et al., 2016)). All experiments were performed with batch size 1.

Metrics. CASD, being an augmenting method, focuses on improvements on the baseline speculator. The usual speculative decoding metrics, applied to CASD are:

- *Walltime speedup ratio*: The speedup ratio relative to the baseline method, EAGLE-2.
- *Average acceptance length* τ : The average number of tokens accepted per forward pass of the target LLM.

4.2 UZGentRAG benchmark

We hypothesized that EAGLE-2 would underperform in multilingual settings and niche domains.

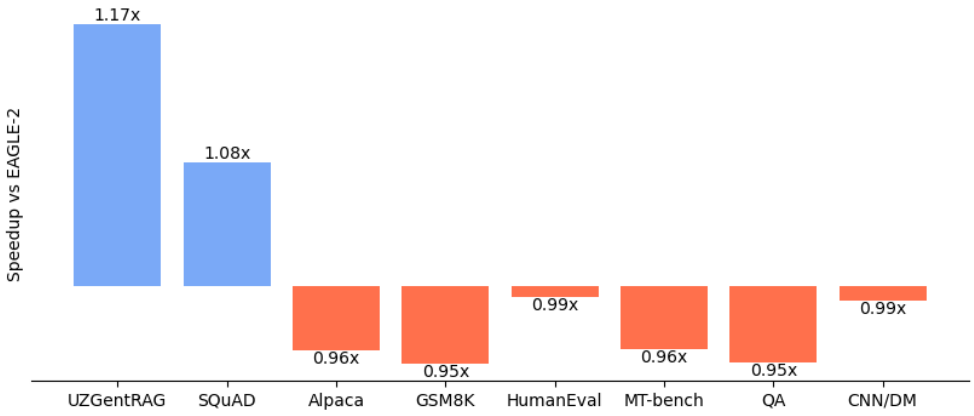

Figure 3: Speedup of CASD compared to the baseline, EAGLE-2. CASD yields improvements for both RAG benchmarks. On standard speculative decoding benchmarks the overhead is still rather limited.

To prove this, we had to go beyond the standard English speculative decoding benchmarks. However, low-resource languages such as Dutch have limited benchmarks available and for RAG use cases we did not find any. In collaboration with Ghent University Hospital (UZGent), we constructed UZGentRAG, a private niche RAG benchmark in Dutch and for the medical domain. This benchmark contains the queries logged from the daily work of doctors and nurses, by recording their questions to a chatbot. Internal UZGent documents found by the chatbot were combined with the query in a typical RAG prompt. These prompts form the benchmark, as that is the only data necessary to test a speculative decoding method.

4.3 Results

Figure 3 shows the comparison between CASD and EAGLE-2. The most relevant benchmarks are the RAG benchmarks, UZGentRAG and SQuAD, for which CASD achieved a speedup of 1.17 and 1.08 respectively. This is quite interesting, because both benchmarks are RAG use cases, yet the results differ strongly. We hypothesize that the Dutch UZGentRAG is harder for EAGLE-2, so it is easier for CASD to find better continuations.

Next to the RAG benchmarks, we also list the general benchmarks. Since these are not RAG-based, and EAGLE-2 was designed for these settings, performance is now greater for EAGLE-2. The key takeaway here is that even when applying CASD to non-RAG input results it still has limited overhead (1-5%).

4.4 Acceptance length

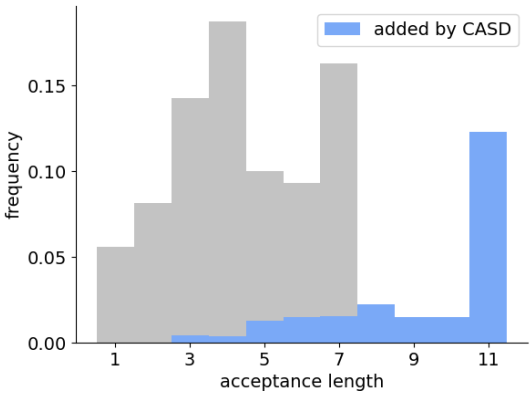

Figure 4: Distribution of acceptance lengths with CASD augmenting EAGLE-2. The tokens that are generated by copying are highlighted in blue, while the gray tokens were generated by EAGLE-2.

To understand why CASD yields improvements on neural speculators, Figure 4 shows how the acceptance length is distributed. EAGLE-2 works quite well and rather consistently: most often it has 4 to 7 tokens accepted. On the other hand, CASD's added speculations show a strong peak at 11. So if CASD gives the best speculation, it is a high quality speculation for a long length. This also explains intuitively why CASD works so well with neural speculators: CASD has occasional speculations that are extremely good, while the neural speculator works well consistently when nothing is copied. As a last remark, it might stand out that there is a hard cap at 7 and 11 for EAGLE-2 and CASD respectively. Both methods implement this cap to

	CASD		CASD$_{prompt}$		EAGLE-2	
	Tokens/s	τ	Tokens/s	τ	Tokens/s	τ
UZGentRAG	**35**	**2.73**	34	2.65	30	2.09
SQuAD	**46**	**7.78**	**46**	**7.78**	43	5.88
MT-bench	61	**4.35**	61	4.30	**63**	4.18

Table 1: Performance of CASD$_{prompt}$ compared to CASD and EAGLE-2 as the baseline. The best score is indicated in bold.

balance overhead with more accepted tokens.

4.5 Ablation study

CASD copies drafts from the entire context, but also from the tokens that were already generated. However, for the original problem statement (RAG), CASD only had to copy from the prompt and not from the answer that the LLM had generated up until that point. If CASD did that, it could be better if the LLM almost never copies from the generated tokens, as it has a smaller overhead. The opposite could as well be true, where CASD benefits now from an LLM repeating itself. The following empirical tests show whether the extra LLM tokens actually provide benefits: we compare our original CASD to CASD$_{prompt}$, which only considers prompt tokens.

Table 1 shows how CASD$_{prompt}$ performs. For the most relevant benchmark, UZGentRAG, a slight improvement justifies the choice of CASD over CASD$_{prompt}$. For SQuAD, τ stayed the same and so did the throughput. Finally, MT-bench shows very similar performance. So it seems that CASD performs better than CASD$_{prompt}$, mostly on the UZGentRAG benchmark. As this benchmark contains longer answers than SQuAD, we intuitively think that the LLM starts repeating generated fragments more the longer the answer gets.

5 Related work

The original paper that introduced speculative decoding (Leviathan et al., 2023) already mentioned the idea of copying from the context, without elaborating. Inference with Reference (Yang et al., 2023) actually implemented it, copying only the best match as a single speculation. While this speculator cannot compete with the current state-of-the-art, CASD proves the principle of copying yields benefits as an augmenting technique on top of the current state-of-the-art.

REST (He et al., 2023) is another method that has similarities with this paper. Rather than drafting from the context, REST queries a large set of pre-existing text to find relevant pieces to copy.

6 Conclusion

We have introduced CASD, a model-agnostic augmentation to any modern speculator for speculative decoding. We found that CASD improves performance of the SOTA speculators, with a very simple method. CASD yields high quality speculations by copying relevant tokens from the context. Like most other speculative decoding methods, CASD allows generating tokens faster, while still reducing the energy necessary to generate. Our tests, augmenting the current SOTA, EAGLE-2, with CASD, demonstrate the added value of its speculations. The experiments show that our method improves speculations in the intended domains, while minimizing the overhead when applied to other domains.

Limitations

CASD is tailored specifically for high-copy environments, such as RAG-based use cases. As seen in the results section, one should only apply it for such cases, because then the gains outweigh the overhead significantly.

Also, the current implementation is not overly optimized. Thus, it is likely that further optimizations such as specialized CUDA kernels could help CASD achieve even higher speedups.

Acknowledgments

We would like to thank the ICT department of the UZGent for their collaboration on making the UZGentRAG benchmark.

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

		CASD		EAGLE-2		CASD vs EAGLE-2	
		Tokens/s	τ	Tokens/s	τ	Speedup	$\tau_{CASD}/\tau_{EAGLE-2}$
RAG benchmarks	UZGentRAG	**35**	**2.73**	30	2.09	1.17	1.30
	SQuAD	**46**	**7.78**	43	5.88	1.08	1.32
General benchmarks	Alpaca	61	**4.15**	**64**	4.12	0.96	1.01
	GSM8K	64	**4.56**	**67**	4.42	0.95	1.03
	HumanEval	75	**5.55**	**76**	5.05	0.99	1.10
	MT-bench	61	**4.35**	**63**	4.18	0.96	1.04
	QA	52	**3.56**	**54**	3.53	0.95	1.01
	CNN/DM	52	**4.14**	**52**	3.74	0.99	1.11

Table 2: Performance of CASD compared with the baseline EAGLE-2. The best score is in bold.

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. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. 2023. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Zongyue Qin, Ziniu Hu, Zifan He, Neha Prakriya, Jason Cong, and Yizhou Sun. 2024. Optimized multi-token joint decoding with auxiliary model for llm inference. *arXiv preprint arXiv:2407.09722*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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. *arXiv preprint arXiv:2304.04487*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

A Appendix results and reproducibility

Results. Table 2 shows the full results of the benchmarks, comparing CASD with EAGLE-2.

Hardware. The tests in Table 2 used an NVIDIA A40 with 44 GB RAM. The results were gathered in one session on the GPU to improve consistency between results.

Benchmarks. Most benchmarks are publicly available, however UZGentRAG is not open for publication as it contains possibly sensitive data. UZGent has indicated that they are open to give access to the dataset in collaborations.

Hyperparameters. The hyperparameters used in this paper are: the maximal substring length considered is 10. Only the top-2 matches are kept when there are more matches. Of those matches, a continuation of 10 tokens long is chosen as a draft.