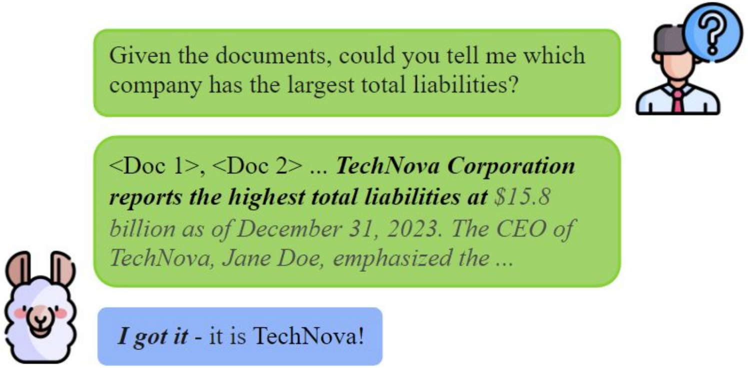
Language Models (Mostly) Know When to Stop Reading

Roy Xie[1], Junlin Wang[1], Paul Rosu[1], Chunyuan Deng[2], Bolun Sun[3], Zihao Lin[4], Bhuwan Dhingra[1]

[1]Duke University, [2]Rice University, [3]Johns Hopkins University, [4]University of California, Davis

The Problem

- **Inefficiency in LLMs:**
  - Process entire context, even after relevant info is found → *Wasted computation.*
- **“Lost-in-the-middle”:**
  - Critical information gets lost in long contexts.
- Human Conversation: Stop processing once sufficient info is gathered.
  - *For example: Interruptions during conversations.*
- RQ: Can we make LLMs to ***process input only when necessary*?**



Static vs. Dynamic Methods

“Static” Methods:

- ***Context Compression***:
  - Using an external LLM to compute the importance of tokens, prune unimportant ones based on information entropy.
  - With predefined a target compression rate (eg: 0.8) to compress the context.
- ***RAG***:
  - Predefined top-*k* relevant document retrieval
- **Issue:**
  - “Compression for compression”
  - **Using external compression heuristics, enforcing a *one-size-fits-all* reduction regardless of content complexity.**

“Dynamic” Methods (Ours):

- Adapts to content based on model’s own understanding
- ***Efficiency emerges <u>naturally</u> where we let the model itself decide when to stop processing context***

Methodology

Input Context: $C = \{s_1, s_2, ..., s_m\}$ (non-overlapping chunks).

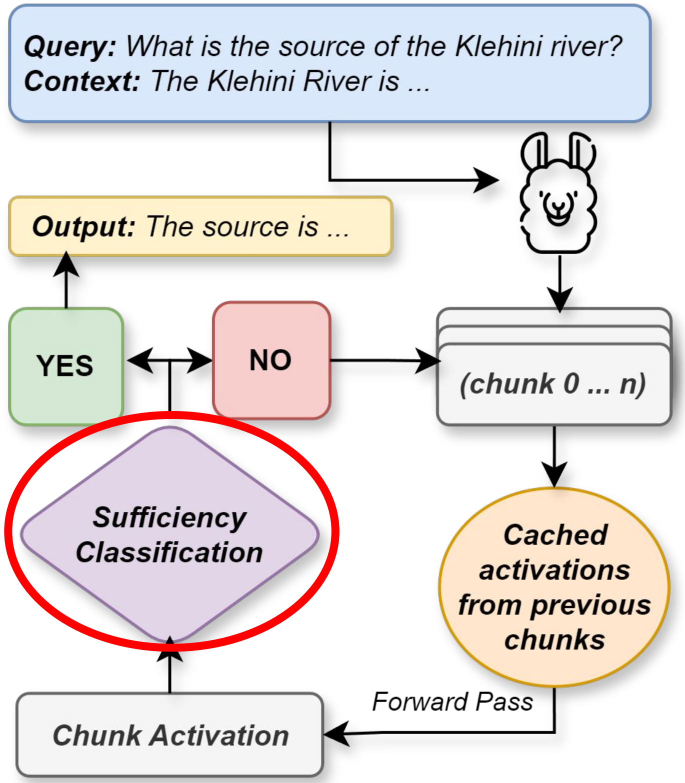
Cumulative Context: $C_i = s_1 \parallel s_2 \parallel ... \parallel s_i$.

Goal: Find minimal C_k where $\mathcal{M}(q, C_k) \approx \mathcal{M}(q, C)$.

Classifier: $\mathcal{S}(C_i) = 1$ if $\mathcal{S}_c(C_i) \geq \tau$.

Workflow:

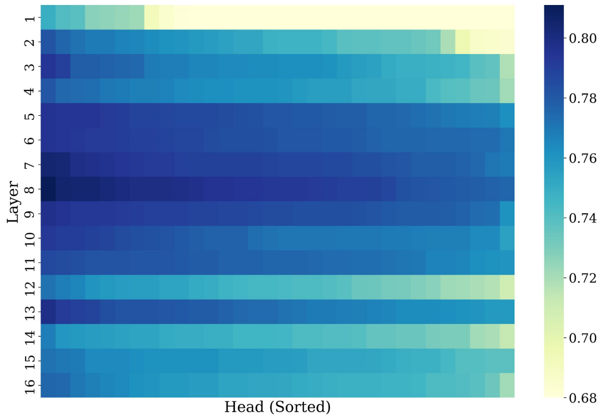
1. Split a input context into non-overlapping chunks
2. Process incrementally; cache activations.
3. Use ensemble classifier (trained on selected heads) to check sufficiency.
4. Terminate early if sufficient and start generation.



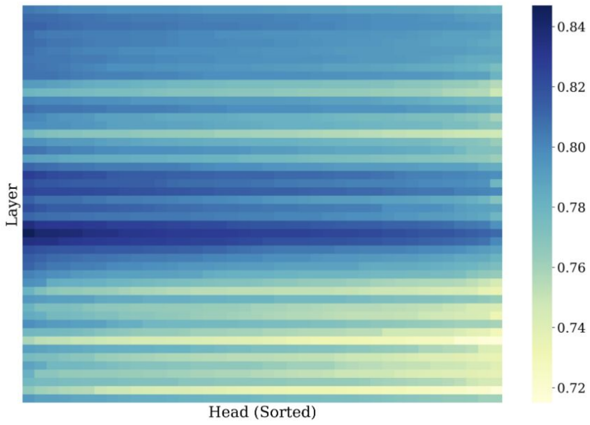
Information Sufficiency - Probing

- Probing each attention heads in each layer to detect sufficiency signals by training a linear classifier
- Specific attention heads encode signals indicating when enough info is processed.

Llama3.2 1B



Qwen2.5 14B



Information sufficiency classification

- Baseline:
  - Finetune: finetune a small (1B) LLM to predict information sufficiency on a given chunk
  - Prompt: prompt LLM itself to predict information sufficiency before generation

Model	FT	Prompt	Ours
LLaMA3.2-1B	79.5	52.6	88.3
Mistral-8B		69.7	89.8
Qwen2.5-14B		78.3	87.2
LLaMA3.3-70B		83.1	91.1

QA Results (Shorter Context 500 - 4k)

- **Token Reduction**: **1.33x** fewer tokens processed.
- **Accuracy**: **+1.3%** improvement over full context.
- Comparison with Baselines:
  - Outperforms RAG, LLMLingua2 (1.25x token reduction) with higher accuracy.
  - Prompting works best for large models (14B+).

Method	LLaMA3.2-1B			Ministral-8B			Qwen2.5-14B			LLaMA3.3-70B			Avg.		
	Multi	Single	Avg	Multi	Single	Avg	Multi	Single	Avg	Multi	Single	Avg	Multi	Single	Total
Full Context	10.4	17.9	14.2	**29.6**	44.8	37.2	30.4	57.6	44.0	37.1	75.0	56.1	26.6	48.7	37.9
BM25	**11.2**	16.2	13.7	20.8	27.5	35.6	25.8	40.8	36.5	21.7	37.1	41.7	19.9	30.4	31.9
SBERT	10.2	17.8	14.0	19.6	37.5	35.2	26.3	51.3	42.3	22.1	41.7	40.8	19.6	37.1	33.1
LLMlingua	6.3	18.3	12.3	22.1	41.7	31.9	24.2	52.5	38.3	35.8	74.1	55.0	22.1	46.7	34.4
LongLLMlingua	6.7	20.0	13.3	22.1	41.7	31.9	26.3	55.8	41.1	35.4	71.7	53.6	22.6	47.3	35.0
LLMlingua2	7.9	**20.8**	**14.4**	28.3	43.3	35.8	32.1	57.9	45.0	35.8	75.0	55.4	26.1	49.3	37.7
FT	6.2	13.8	10.0	21.5	34.7	28.1	22.3	35.1	28.7	35.6	52.4	44.0	21.4	34.0	27.7
Self-Prompt	6.4	11.4	8.9	23.8	36.2	30.0	**38.2**	52.0	45.1	**48.3**	69.9	59.1	28.9	42.6	35.8
Ours	10.3	17.5	13.9	28.8	**45.8**	**37.3**	33.3	**59.2**	**46.3**	43.8	**75.3**	**59.5**	**29.0**	**49.4**	**39.2**

QA Results (Longer Context 4k - 40k)

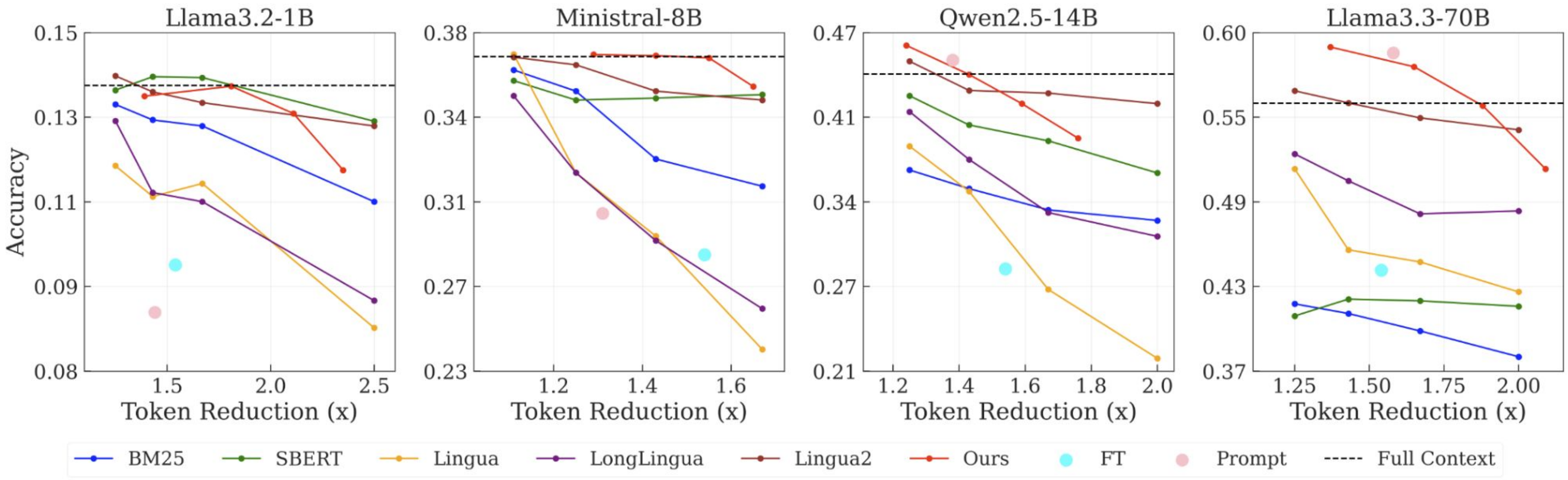
- **Token Reduction: 1.27x** fewer tokens processed.
- **Accuracy**: **+0.5%** improvement over full context.

Method	LLaMA3.2-1B			Ministral-8B			Qwen2.5-14B			LLaMA3.3-70B			Avg.		
	Multi	Single	Avg	Multi	Single	Avg	Multi	Single	Avg	Multi	Single	Avg	Multi	Single	Total
Full Context	5.0	10.4	7.7	18.3	**38.8**	28.5	29.9	40.0	35.0	29.3	70.8	50.0	20.6	**40.0**	30.3
BM25	**5.7**	12.2	8.9	**20.9**	38.7	**29.8**	30.2	39.2	34.7	28.7	68.7	48.7	**21.3**	40.0	30.5
SBERT	5.6	**12.5**	**9.1**	20.2	37.7	29.4	30.0	38.9	34.4	27.7	68.8	48.3	21.1	39.5	30.3
LLMlingua	3.8	12.1	7.9	17.1	35.8	26.5	27.1	41.8	34.5	23.3	65.4	44.4	17.8	38.8	28.3
LongLLMlingua	3.3	12.1	7.7	15.0	37.1	26.0	28.0	40.1	34.1	27.5	67.9	47.7	18.5	39.3	28.9
LLMlingua2	2.7	9.6	6.2	17.1	38.3	27.7	28.8	42.9	35.8	28.2	69.2	48.7	19.2	40.0	29.6
FT	2.6	8.4	5.5	14.9	31.5	23.3	21.4	33.2	27.3	21.4	47.1	34.3	15.1	30.0	22.6
Self-Prompt	4.2	7.3	5.7	17.4	32.6	25.0	**30.5**	**45.8**	**37.6**	29.0	65.4	47.2	20.3	37.8	29.0
Ours	5.0	9.9	7.5	19.1	37.7	28.4	29.8	43.2	36.5	**30.8**	**70.9**	**50.9**	21.2	39.9	**30.8**

Efficiency / Accuracy Trade-off

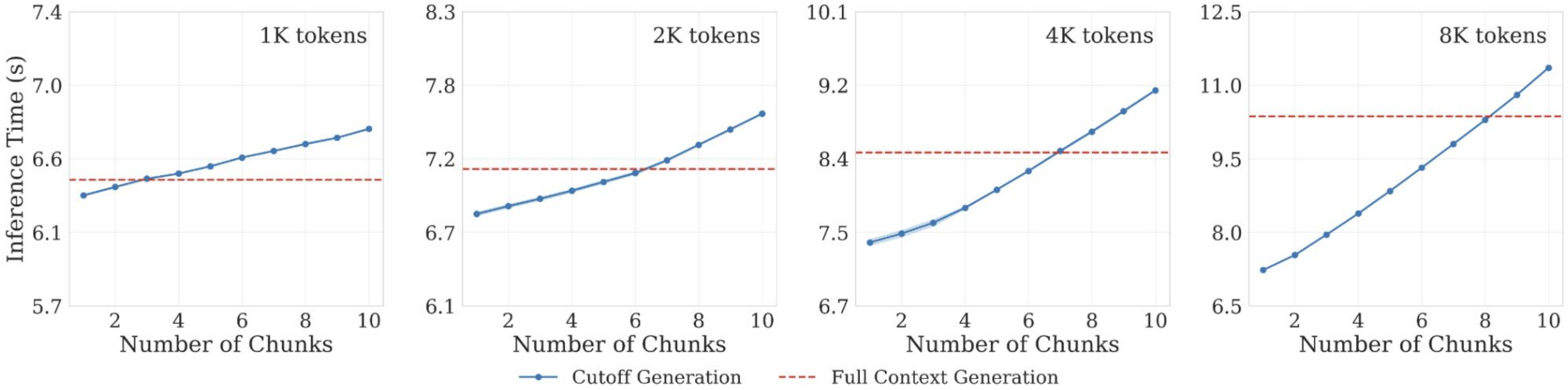
Classification thresholds: 80%, 85%, 90%, 95%

Scaling law: Larger models achieve better efficiency without accuracy loss.



Inference Time

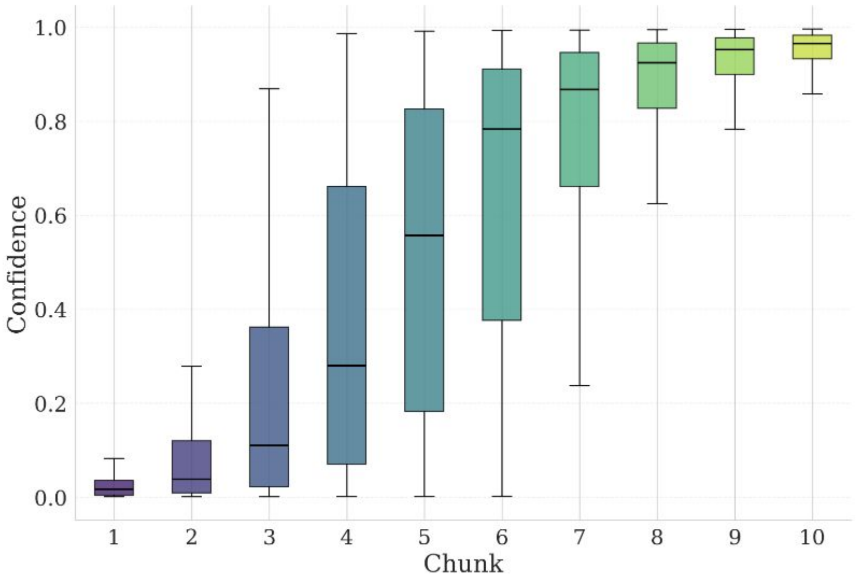
- Chunk information classification introduces latency. For shorter context, process full context might might be faster.
- Our method generally faster for contexts >2K tokens.



Chunking Strategies

- Sentence-level: Highest F1 (96.8) with higher latency.
- 10% chunking: Best balance (88.3 F1, 85.9 R@90P).
- More chunks (context), more confidence.

Metric	Sent.	1%	5%	10%	20%
F1-Score	96.8	87.2	87.0	88.3	88.3
R@90P	95.4	90.9	78.4	85.9	85.8
Acc.	14.5	13.7	12.8	13.9	13.7



Conclusion

- We introduce an natural efficiency paradigm - use models’ internal understanding to process only the minimal necessary context.
- We found that LLMs have intrinsic self-assessment capabilities on information sufficiency, especially with larger models.
- Our method reduces computation while improving accuracy, outperforming static methods like RAG and compression-based heuristics.