Language Models in Cryptanalysis

How do we crack the long ciphers?

Morten Munk

November 2025

AAU CPH - SW9

Contents

1. The papers

1. The papers

X-former Elucidator: Reviving Efficient Attention for Long Context Language Modeling
Miao, et al., 2024

Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention
Katharopoulos, et al., 2020

Long-Short Transformer: Efficient Transformers for Language and Vision
Zhu, et al., 2021

Rethinking Attention with Performers
Choromanski, et al., 2021

Linformer: Self-Attention with Linear Complexity
Wang, et al., 2020

Efficient computation during training

- Causal LM inference is fast

Best methods from comparative study

- We don't have time for them all
- Some are more inference focused

Summary

- We care about efficiency during training
- Inference for long cipher struggle due to lack of generalization on long ciphers

2. Background

Homophonic Substitution Ciphers

- 1:>0 mappings
- English without spaces & punctuation

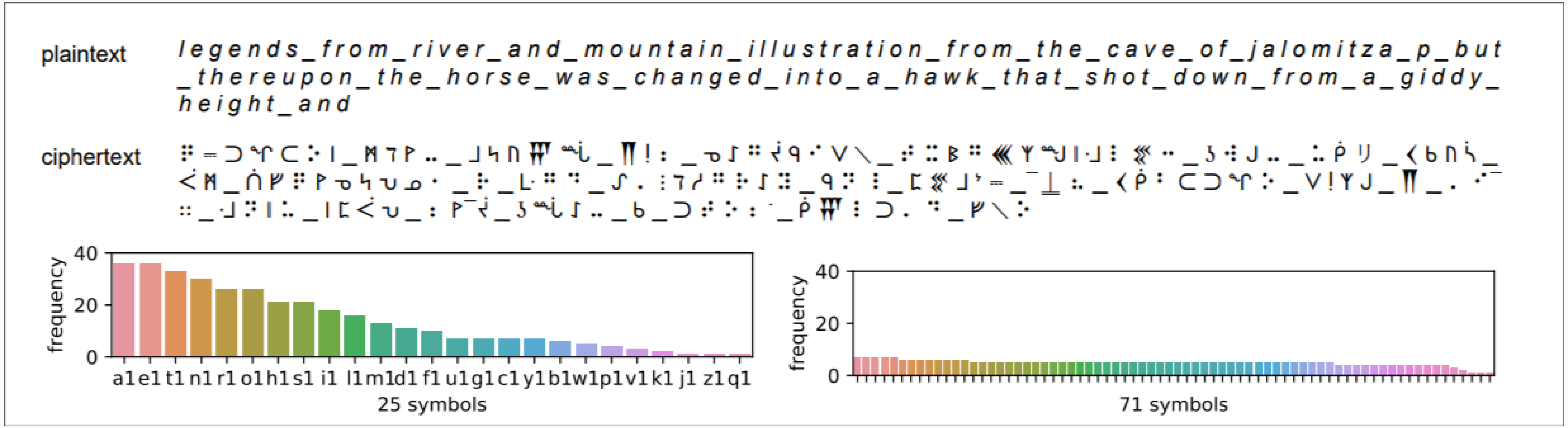

Figure 1: Example of a homophonic substitution cipher. (Kambhatla et al., Findings 2023)

2.1 Recap from last time

Causal LM

- Learns both cipher & plaintext
- Reads left to right

Seq2seq

- Learns plaintext only
- Bidirectional

Both suffer from $O(N^2)$ attention 😔

#keys	Model	Max Len. 400	Max Len. 700
30-45	Seq-to-Seq	72.30	fail
	PrefixLM	54.73	69.50
	CausalLM (tgt)	29.99	37.20
	CausalLM	**0.40**	**0.21**
40-65	PrefixLM	69.50	54.73
	CausalLM (tgt)	29.99	37.20
	CausalLM	**0.83**	**0.80**
30-85	PrefixLM	70.52	71.82
	CausalLM (tgt)	42.05	42.69
	CausalLM	**2.25**	**2.19**

Figure 2: SER on synthetic HS ciphers. (Kambhatla et al., Findings 2023)

Rows (Queries)
- Token we are looking for

Columns (Keys)
- Token we are looking at

Values (Cells)
- Attention Score

Notice: $(N \times N) = N^2$

Cipher: X Y Z

$$X \to [Y, Z]$$

$$Y \to [X, Z]$$

$$Z \to [X, Y]$$

	X	**Y**	**Z**
X	$X \to X$	$X \to Y$	$X \to Z$
Y	$Y \to X$	$Y \to Y$	$Y \to Z$
Z	$Z \to X$	$Z \to Y$	$Z \to Z$

Table 1: Attention Weight Matrix $(N \times N)$

Why is $O(N^2)$ Ineffecient?

Many attention heads

- Each head has its own matrix

Many forward passes

- Each matrix recomputed at each pass

Causal LM as an example

- 12 layers $\times$ 12 heads $=$ 144 attention heads

Ciphers with 1000s of characters does not scale well 😭

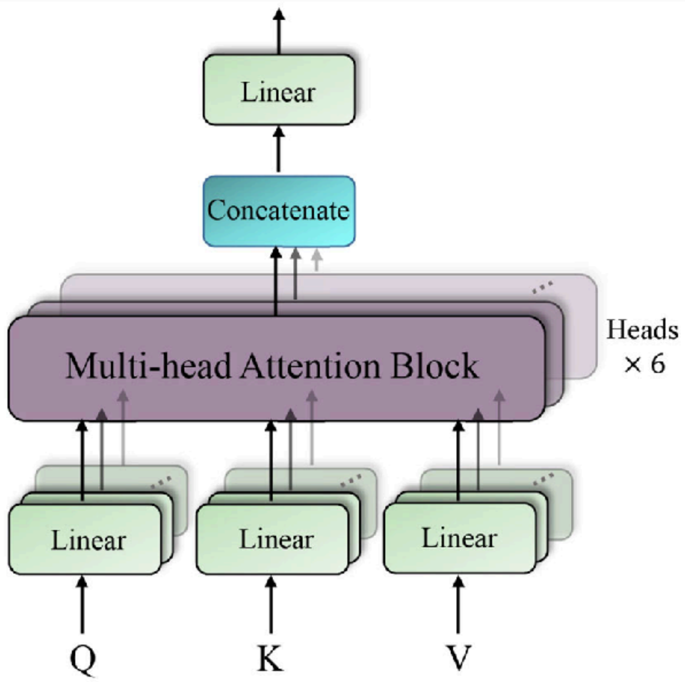

Figure 3: Example of attention with multiple heads. (Yuan et al., 2022)

3. Linformer

Disclaimer: Math is heavily simplified for understanding ⚠️

Standard attention

$$\text{attention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \underbrace{\text{softmax}\left(\frac{\mathrm{QK}^T}{\sqrt{d}}\right)V}_{P}$$

Claim: Attention matrix is low-rank

- It can be represented by a smaller matrix

For any Q, K, V exists a low-rank matrix $\tilde{P}$ where:

- $\tilde{P}$ has minimal error
- $\tilde{P}$ has low-rank (fewer dims/features)

Disclaimer: Math is heavily simplified for understanding ⚠️

Standard attention

$$\text{attention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \underbrace{\text{softmax}\left(\frac{\mathrm{QK}^T}{\sqrt{d}}\right)}_{P} V$$

Claim: Attention matrix is low-rank

- It can be represented by a smaller matrix

For any Q, K, V exists a low-rank matrix $\tilde{P}$ where:

- $\tilde{P}$ has minimal error
- $\tilde{P}$ has low-rank (fewer dims/features)

I will skip mathematical proof, and show empirical proof ⚠️

Eigenvalue index

- Top 128 eigenvalues are most important
- Trailing 384 eigenvalues are not so important

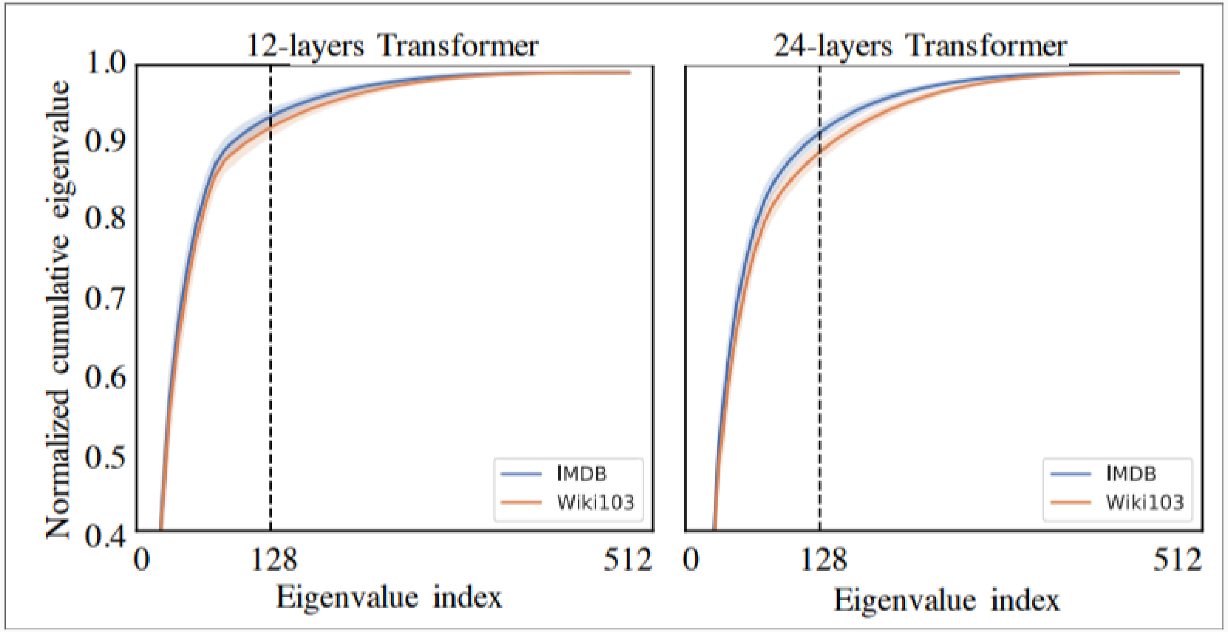

Figure 4: RoBERTa, IMDB & Wiki103 (Wang, et al., 2020)

Similar to standard transformer

- Notice projections!

$$\text{softmax}\left(\frac{\mathrm{QK}^T}{\sqrt{d}}\right)V \rightarrow \text{softmax}\left(\frac{Q(E_K K)^T}{\sqrt{d}}\right)E_V V$$

$n \times n = O(n^2)$

- inefficient!

$n \times k = O(nk)$

- Better because $k \ll n$

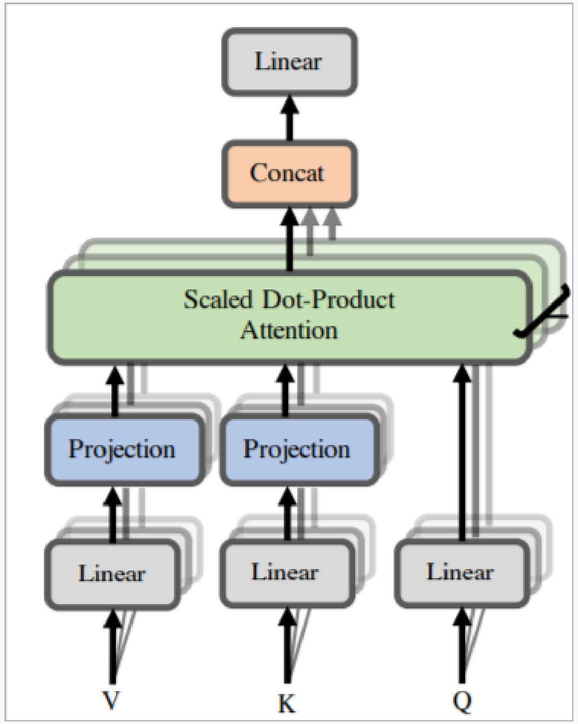

Figure 5: (Wang, et al., 2020)

4. Performer

FAVOR+: Fast Attention Via positive Orthogonal Random Features

Linformer compresses - Performer approximates

Left side (Standard):

- Each key looks at every other key $O(L^2)$
- Exponential similarity e^{QK^T}
  - Grows positively/negatively based on similarity

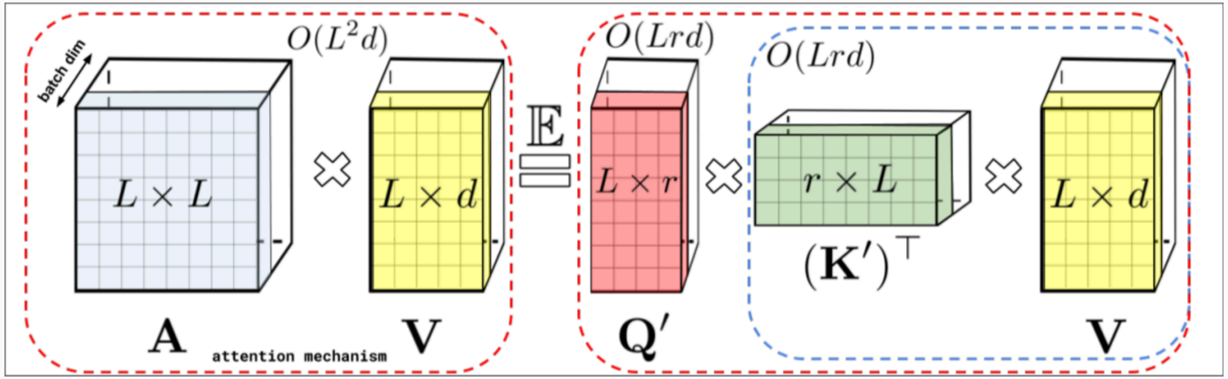

Figure 6: (Choromanski, et al., 2021)

Right side (Performer):

- Kernel trick with random features $\varphi(Q) \cdot \varphi(K) \approx e^{q_i \cdot k_j}$
- Same exponential similarity but smaller matrices

$$\varphi(Q)(\varphi(K)^T V) \to (L \times r) \times (r \times d) = O(Lr)$$

- r parameter determine number of directions to mimic similarites
  - Large r = more accurate, but slower

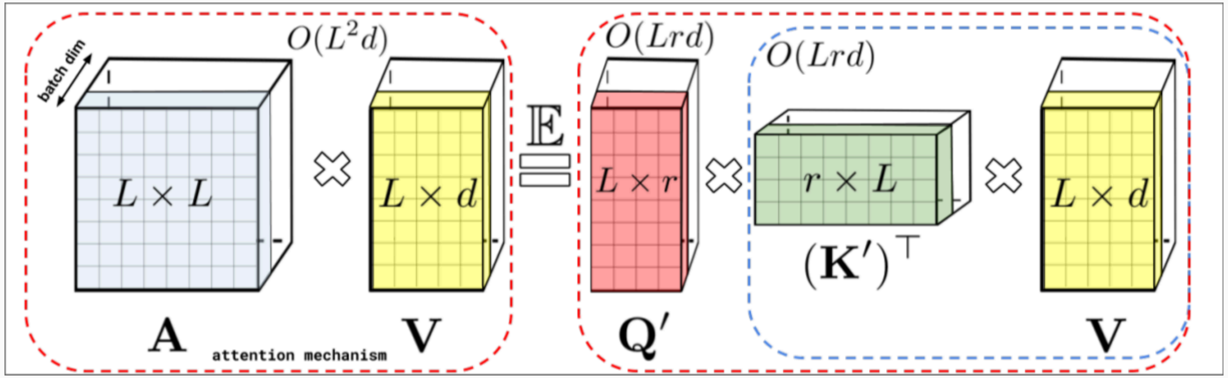

Figure 7: (Choromanski, et al., 2021)

5. Reordered Computation

Transformers are RNNs...

Transformers are RNNs…

- Not really 🤡

Transformers are RNNs…

- Not really 🤡

But this paper shows how they can act like it

Also it has no illustrations, so instead you get math 🥳

Generalized Attention $O(N^2)$

- Calculates output of a query by weighted average of all V vectors
- Similarity between Q and K determines weight

$$V_i' = \frac{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j,) V_j}{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j)}$$

Linear Attention O(N)

- Notice the reordered computation
- Uses rule of associativity $(A \times B) \times C = A \times (B \times C)$

$$V_i' = \frac{\varphi(Q_i)^T \sum_{j=1}^{N} \varphi(K_j) V_j^T}{\varphi(Q_i)^T \sum_{j=1}^{N} \varphi(K_j)}$$

5. Reordered Computation

Generalized Attention $O(N^2)$

- Calculates output of a query by weighted average of all V vectors
- Similarity between Q and K determines weight

$$V_i' = \frac{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j,) V_j}{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j)}$$

Linear Attention O(N)

- Notice the reordered computation
- Uses rule of associativity $(A \times B) \times C = A \times (B \times C)$

$$V_i' = \frac{\varphi(Q_i)^T \sum_{j=1}^{N} \varphi(K_j) V_j^T}{\varphi(Q_i)^T \sum_{j=1}^{N} \varphi(K_j)} \rightarrow (\varphi(Q)\varphi(K)^T)V = \underbrace{\varphi(Q)}_{M_1} \underbrace{(\varphi(K)^T V)}_{\mathrm{M}_2\ O(N)}$$

Calculate M_2 which is independent of Att matrix size, then multiply with M_1

$N(O(N)) \rightarrow$ linear!

But why do they claim transformers are RNNs?

But why do they claim transformers are RNNs?

Used during inference

Standard Attention

- For each Q, look at each K

RNN approach

- s_i = content memory of keys and values
- z_i = normalizer memory (running total of weights)
- Combined they keep weighted average for next output
  - no need to look back at individual past keys!

For each Q it is constant lookup!

6. Long Short Transformer

Linear complexity from dual attention

Short term (local window)
- Fine-grained local correlations
- Divide input sequence into w sized segments
- Sliding window looks at home segment and $\frac{w}{2}$ tokens on both sides
- Fixed size keeps attention $O(N)$

Long range (dynamic projection)
- Distant correlations across entire sequence
- Dynamic Projection P decides which keys/values are important to keep
- P creates low-rank version of K and V
- Low-rank contains fixed r summary points

Aggregation of attention Final output is calculated at every head

- Query looks at concatenation of global and local keys/values

Scale mismatch Bias towards short term due to larger values

- Solved with DualLN normalization technique

Runtime depends on sequence length N and summary points r

- Hence linear runtime $O(N)$

7. Comparative study

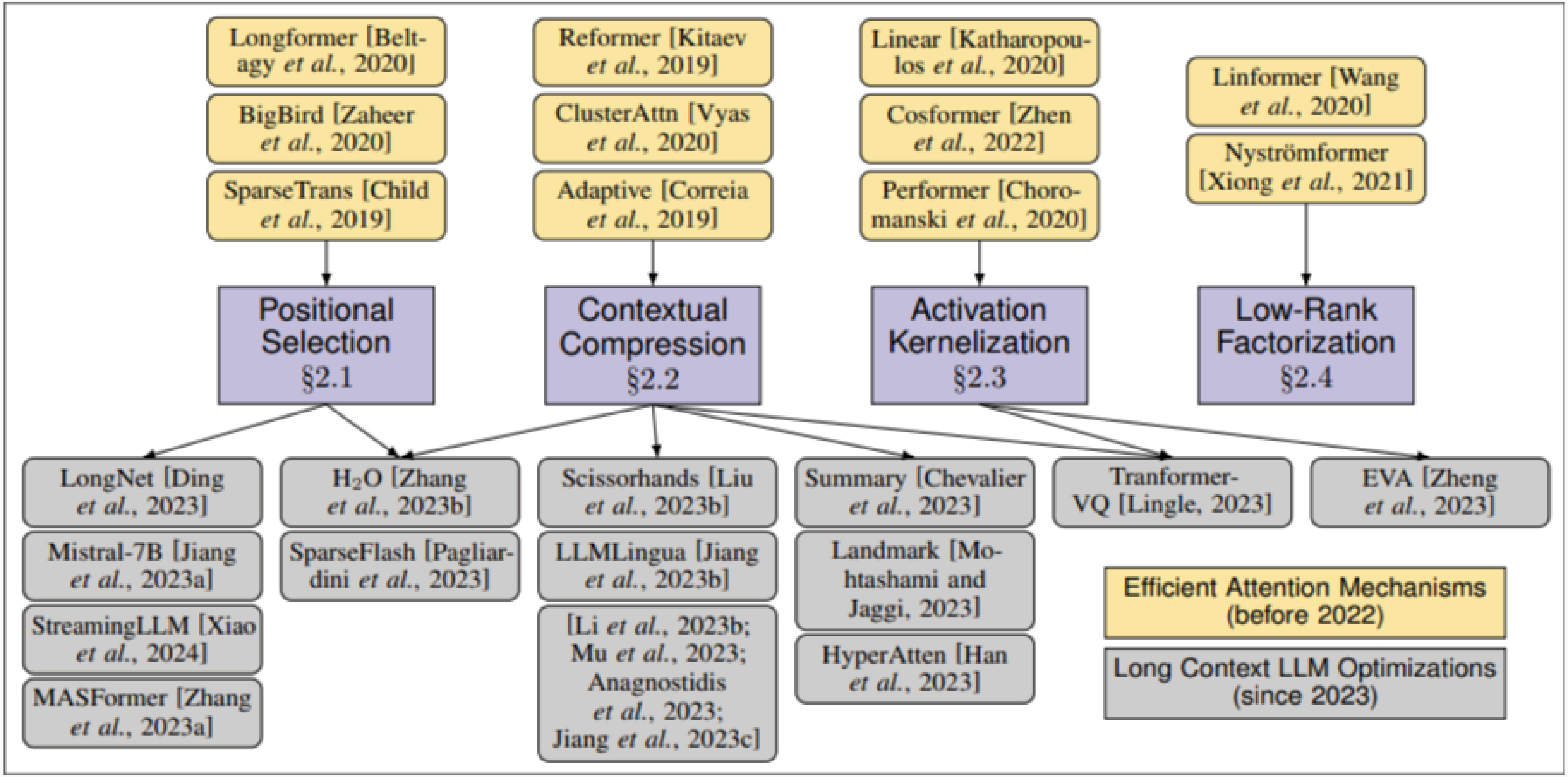

Figure 8: (Miao, et al., 2024)

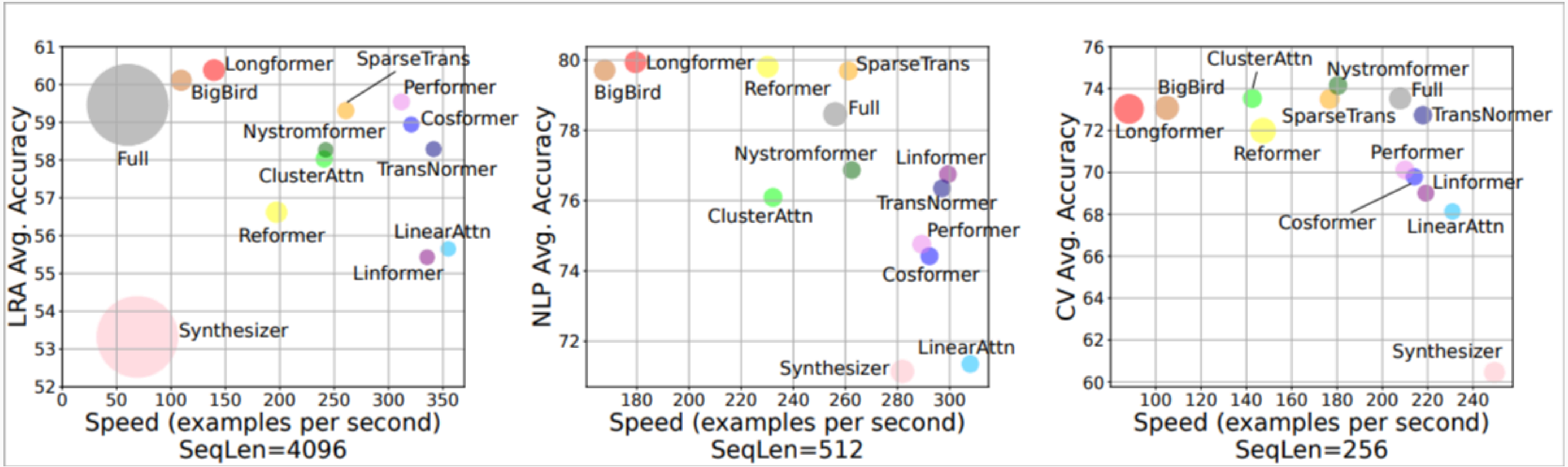

Figure 9: (Miao, et al., 2024)

8. Limitations

None of the methods are tested on ciphers

- Ciphers are slightly different than regular machine translation

No one method dominates

- Performer is likely best for 4096 sequence length
- I would like to go way beyond 4k sequence length 😎

Long Short dependencies

- Will it work for very long ciphers?
- What do you think is most important?
  - Global
  - Local
  - Evenly?

9. Questions?

Please no math questions 😞