

Transformers Learn Variable-Order Markov Chains In-context

Chao Tian

Based on joint work with Suhas Diggavi and Ruida Zhou

Learn to Compress & Compress to Learn Workshop ISIT-2025



Large Language Models



Large language models:

- You probably have used them: GPT, Gemini, Perplexity, Claude models, Deepseek, etc.;
- Disclaimer: I'm a paying GPT subscriber and Github Copilot user.



Large Language Models



Large language models:

- You probably have used them: GPT, Gemini, Perplexity, Claude models, Deepseek, etc.;
- Disclaimer: I'm a paying GPT subscriber and Github Copilot user.

In a Gaussian graphic model, the independence relation shows up as some graph structure but also some properties in the covariance/precision matrix. Can you tell me what they are?

In Gaussian graphical models, the independence relations among variables are represented by both the structure of the graph and certain properties of the precision (inverse covariance) matrix.

1. Graph Structure and Conditional Independence:

- In a Gaussian graphical model, an edge between two nodes indicates a conditional dependence between the corresponding random variables. If two nodes are not connected by an edge, it implies conditional independence between those variables given all other variables in the graph.

2. Precision Matrix and Independence:

- For a multivariate Gaussian distribution, conditional independence between variables corresponds to zeros in the **precision matrix** (the inverse of the covariance matrix). Specifically:
 - If the (i, j) entry in the precision matrix is zero, then variables X_i and X_j are conditionally independent given all other variables.
 - Thus, the sparsity pattern in the precision matrix aligns with the edges in the graphical structure, where each zero entry indicates a missing edge.
- ## 3. Covariance Matrix:
- The covariance matrix itself does not directly reflect conditional independencies, but it does capture marginal dependencies. However, a lack of correlation (i.e., zero in the covariance matrix) only implies marginal independence, not conditional independence.

In summary, conditional independence in a Gaussian graphical model corresponds to zeros in the precision matrix, reflecting the graph's edges and thus the overall structure of the model.

Very good at memorizing things!



LLMs Also Show Signs of “Intelligence”

25, green; 27, green; 224, black; 702, black; 67, green; 55, ?



Following the updated pattern:

- Numbers **odd** are assigned **green**.
- Numbers **even** are assigned **black**.

Since **55** is an odd number, it would be **green**.



LLMs Also Show Signs of “Intelligence”

25, green; 27, green; 224, black; 702, black; 67, green; 55, ?



Following the updated pattern:

- Numbers **odd** are assigned **green**.
- Numbers **even** are assigned **black**.

Since **55** is an odd number, it would be **green**.



Can learn from in-context examples, very nice 😊



LLMs Show Signs of “Intelligence”

25, black; 27, green; 16, black; 224, black; 702, green; 67, black; 55, ?



Based on the pattern, it appears that:

- Numbers **divisible by 5** are assigned **black**.
- Numbers **not divisible by 5** are assigned **green**.

For 55, which is divisible by 5, the color would be **black**.



LLMs Show Signs of “Intelligence”

25, black; 27, green; 16, black; 224, black; 702, green; 67, black; 55, ?



Based on the pattern, it appears that:

- Numbers **divisible by 5** are assigned **black**.
- Numbers **not divisible by 5** are assigned **green**.

For 55, which is divisible by 5, the color would be **black**.

Well, the answer “black” may be correct, but the reason is not quite right 😊



LLMs Show Signs of “Intelligence”

25, black; 27, green; 16, black; 224, black; 702, green; 67, black; 55, ?



Based on the pattern, it appears that:

- Numbers **divisible by 5** are assigned **black**.
- Numbers **not divisible by 5** are assigned **green**.

For 55, which is divisible by 5, the color would be **black**.

Well, the answer “black” may be correct, but the reason is not quite right 😊

Motivation: We don't completely understand how LLMs learn in context.



Connection to Data Compression

Delétang et al. ICLR-2024: “Language modeling is compression”

- Explicitly connect language modeling and compression;
- Good performances even carry over to other modalities to some extent, e.g., images and videos.



Connection to Data Compression

Delétang et al. ICLR-2024: “Language modeling is compression”

- Explicitly connect language modeling and compression;
- Good performances even carry over to other modalities to some extent, e.g., images and videos.

LLMZip (Valmeekam et al. 2023)

- Shannon in 1951 posed the question “What’s the entropy of the English language?”
- Use Llama-7B to compress: Significant reduction over all previous estimates ($\geq 40\%$);
- How? In-weight learning (memorizing) or in-context learning?



Connection to Data Compression

Delétang et al. ICLR-2024: “Language modeling is compression”

- Explicitly connect language modeling and compression;
- Good performances even carry over to other modalities to some extent, e.g., images and videos.

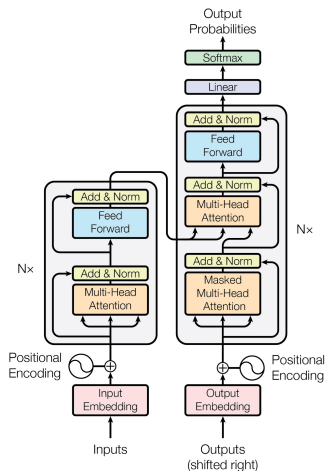
LLMZip (Valmeekam et al. 2023)

- Shannon in 1951 posed the question “What’s the entropy of the English language?”
- Use Llama-7B to compress: Significant reduction over all previous estimates ($\geq 40\%$);
- How? In-weight learning (memorizing) or in-context learning?

We can study transformers’ in-context learning capability through the lens of data compression!



What Are Transformers?

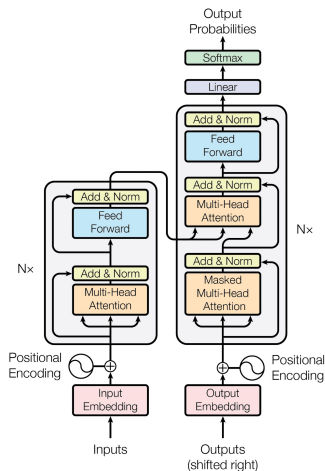


- The key component is the multi-head attention mechanism;
- Feedforward layers and residual links in the network;

Vaswani et al. "Attention is all you need"



What Are Transformers?

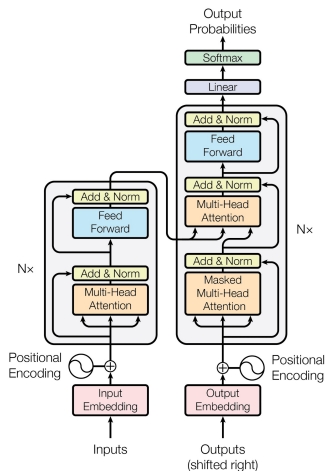


- The key component is the multi-head attention mechanism;
- Feedforward layers and residual links in the network;
- The overall model may include an encoder and a decoder, but we only need the decoder side;

Vaswani et al. "Attention is all you need"



What Are Transformers?



- The key component is the multi-head attention mechanism;
- Feedforward layers and residual links in the network;
- The overall model may include an encoder and a decoder, but we only need the decoder side;
- **The decoder output is the probability of the next token(s).**

Vaswani et al. "Attention is all you need"



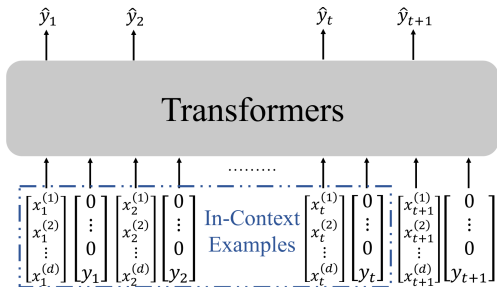
What's In-context Learning Exactly?

Roughly: Learn from illustrations in the context window without updating the neural network weights.



What's In-context Learning Exactly?

Roughly: Learn from illustrations **in the context window without updating the neural network weights.**



“Transformers Learn Higher-Order Optimization Methods for In-Context Learning: A Study with Linear Models” by Fu et al.

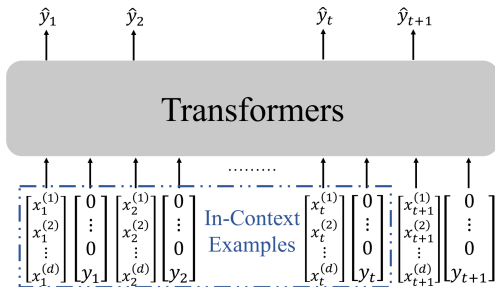
Example: In-context learning of a linear model

- Model $y = \mathbf{x}_t^T \mathbf{w}$ with parameter \mathbf{w} , chosen randomly from a prior, in each context window;
- In-context examples \mathbf{x}_t 's generated i.i.d. randomly, and $y_t = \mathbf{x}_t^T \mathbf{w} + n_t$;
- Learn \mathbf{w} in-context, and apply it to the context token;



What's In-context Learning Exactly?

Roughly: Learn from illustrations in the context window without updating the neural network weights.



“Transformers Learn Higher-Order Optimization Methods for In-Context Learning: A Study with Linear Models” by Fu et al.

Example: In-context learning of a linear model

- Model $y = \mathbf{x}_t^T \mathbf{w}$ with parameter \mathbf{w} , chosen randomly from a prior, in each context window;
- In-context examples \mathbf{x}_t 's generated i.i.d. randomly, and $y_t = \mathbf{x}_t^T \mathbf{w} + n_t$;
- Learn \mathbf{w} in-context, and apply it to the context token;
- Well-studied: A single attention layer is similar to a one-step gradient descent.



In-Context Learning of Natural Languages

Natural languages **do not** look like this!!

- Example: *“Language models are useful in a wide **variety** of applications, from natural language understanding and generation to translation, **summarization**, and...”*



In-Context Learning of Natural Languages

Natural languages **do not** look like this!!

- Example: “*Language models are useful in a wide **variety** of applications, from natural language understanding and generation to translation, **summarization**, and...*”

- The next token probability may depend on preceding tokens (suffix);

$$P(\text{“variety”} | \text{“Language models are useful in a wide”}) = P(\text{“variety”} | \text{“useful in a wide”}) = 0.7$$

- The lengths of these suffixes can vary;



In-Context Learning of Natural Languages

Natural languages **do not** look like this!!

- Example: “*Language models are useful in a wide variety of applications, from natural language understanding and generation to translation, summarization, and...*”

- The next token probability may depend on preceding tokens (suffix);

$$P(\text{“variety”} | \text{“Language models are useful in a wide”}) = P(\text{“variety”} | \text{“useful in a wide”}) = 0.7$$

- The lengths of these suffixes can vary;
- In-context learning here = In the same context window, gradually learn how long the suffixes are and also what the corresponding probability distributions are;
- The distribution can vary from context window to context window;



In-Context Learning of Natural Languages

Natural languages **do not** look like this!!

- Example: “*Language models are useful in a wide variety of applications, from natural language understanding and generation to translation, summarization, and...*”

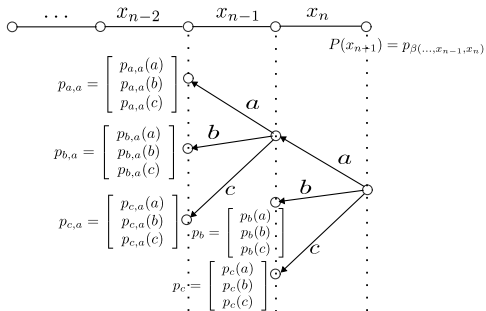
- The next token probability may depend on preceding tokens (suffix);

$$P(\text{“variety”} | \text{“Language models are useful in a wide”}) = P(\text{“variety”} | \text{“useful in a wide”}) = 0.7$$

- The lengths of these suffixes can vary;
- In-context learning here = In the same context window, gradually learn how long the suffixes are and also what the corresponding probability distributions are;
- The distribution can vary from context window to context window;
- We need a “sophisticated but clean model” to study this.



Context Tree Models: Variable Order Markov Chains

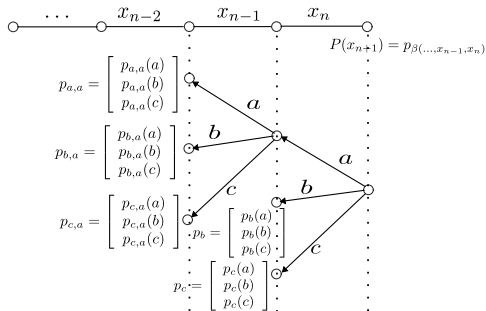


Context-tree sources: studied in the data compression area extensively, capturing natural language well:

Alphabet $\{a, b, c\}$ and suffix set $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.



Context Tree Models: Variable Order Markov Chains



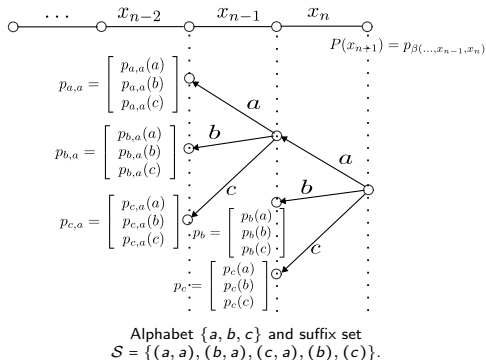
Alphabet $\{a, b, c\}$ and suffix set
 $\mathcal{S} = \{(a, a), (b, a), (c, a), (b), (c)\}$.

Context-tree sources: studied in the data compression area extensively, capturing natural language well:

- A set of strings \mathcal{S} : each $s \in \mathcal{S}$ is associated with a probability distribution p_s ;
- No string in \mathcal{S} is suffix of another one (properness);
- Each semi-infinite sequence has a unique suffix in \mathcal{S} (completeness).



Context Tree Models: Variable Order Markov Chains

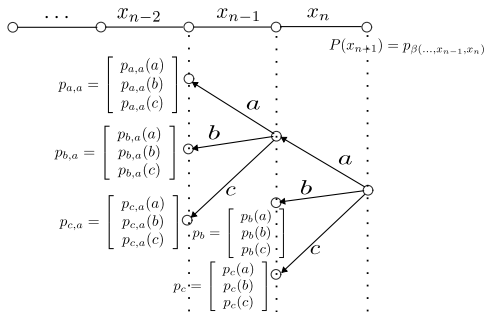


Context-tree sources: studied in the data compression area extensively, capturing natural language well:

- A set of strings \mathcal{S} : each $s \in \mathcal{S}$ is associated with a probability distribution p_s ;
- No string in \mathcal{S} is suffix of another one (properness);
- Each semi-infinite sequence has a unique suffix in \mathcal{S} (completeness).
- Context tree: leaf node = a suffix $s \in \mathcal{S}$ and a distribution p_s ;



Learning to Compress VOMC Sources



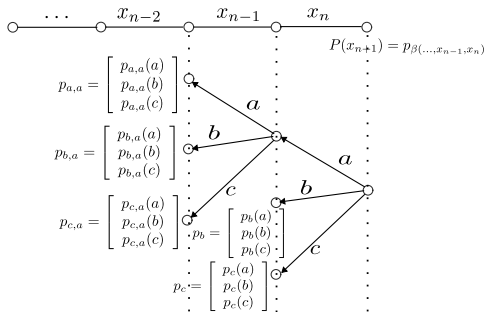
Alphabet $\{a, b, c\}$ and suffix set $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.

Variable-Order Markov Chain (VOMC) vs. the fixed-order version

- Difficulty: (1) find the tree structure, and (2) estimate the probability distribution.



Learning to Compress VOMC Sources



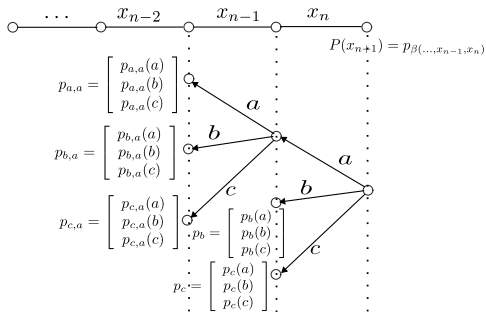
Alphabet $\{a, b, c\}$ and suffix set
 $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.

Variable-Order Markov Chain (VOMC) vs. the fixed-order version

- Difficulty: (1) find the tree structure, and (2) estimate the probability distribution.
- Intuitively, we can collect statistics for suffixes of different lengths. But, which suffix to use then?



Learning to Compress VOMC Sources



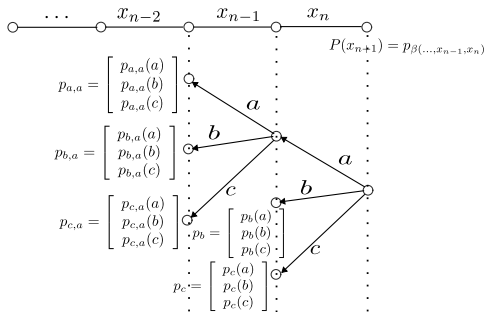
Alphabet $\{a, b, c\}$ and suffix set $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.

Variable-Order Markov Chain (VOMC) vs. the fixed-order version

- Difficulty: (1) find the tree structure, and (2) estimate the probability distribution.
- Intuitively, we can collect statistics for suffixes of different lengths. But, which suffix to use then?
- If we underestimate the suffix length, incorrect statistics;
- If we overestimate the suffix length, inefficient usage of in-context data.



Learning to Compress VOMC Sources



Alphabet $\{a, b, c\}$ and suffix set $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.

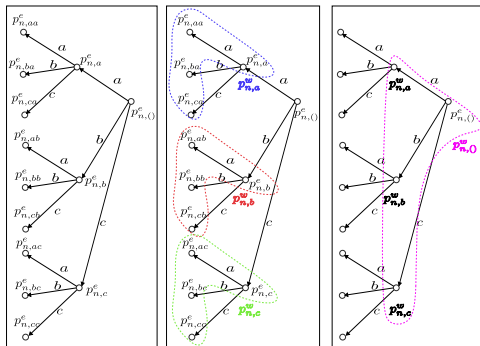
Variable-Order Markov Chain (VOMC) vs. the fixed-order version

- Difficulty: (1) find the tree structure, and (2) estimate the probability distribution.
- Intuitively, we can collect statistics for suffixes of different lengths. But, which suffix to use then?
- If we underestimate the suffix length, incorrect statistics;
- If we overestimate the suffix length, inefficient usage of in-context data.

Can transformers learn in-context such models, and if so, how?



Context-tree Weighting: Bayesian Optimal Universal Compression



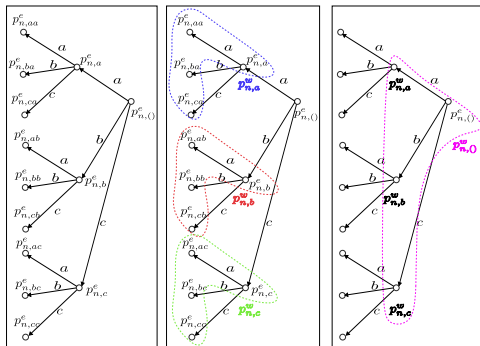
Alphabet $\{a, b, c\}$ and suffix set
 $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.

IT insight: Weigh all possibilities instead of picking just one!

- The celebrated context tree weighting (CTW) algorithm (Willems et al. 1995);
- Shown to be **Bayesian optimal** for a general class of priors (Kontoyiannis 2023);
- CTW follows a recursive structure, weighing from the (largest possible order D) leaves to the tree root in D steps.



Context-tree Weighting: Bayesian Optimal Universal Compression



Alphabet $\{a, b, c\}$ and suffix set
 $S = \{(a, a), (b, a), (c, a), (b), (c)\}$.

IT insight: Weigh all possibilities instead of picking just one!

- The celebrated context tree weighting (CTW) algorithm (Willems et al. 1995);
- Shown to be **Bayesian optimal** for a general class of priors (Kontoyiannis 2023);
- CTW follows a recursive structure, weighing from the (largest possible order D) leaves to the tree root in D steps.

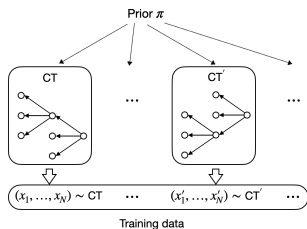
Benefits of the IT (compression) perspective: can study ICL for which we know the Bayesian optimal solution.



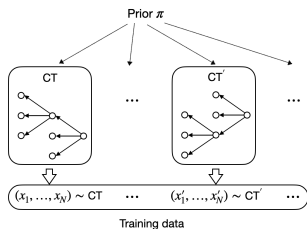
Empirical Study: Training and Testing

Training:

- Randomly generate multiple context trees;
- Random generate sequences from these trees ($|\mathcal{A}| = 3, D \leq 5$);
- Train transformers with multiple heads and various numbers of layers on these sequences;
- Standard cross-entropy loss, i.e., $-\log \tilde{p}(x_t)$, the compression rate.



Empirical Study: Training and Testing



Training:

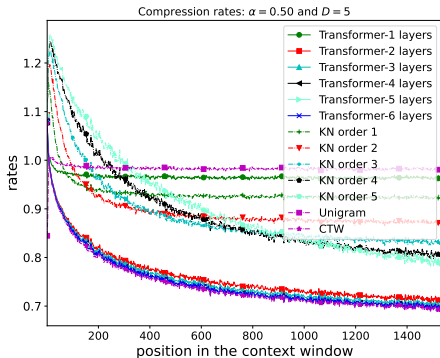
- Randomly generate multiple context trees;
- Random generate sequences from these trees ($|\mathcal{A}| = 3, D \leq 5$);
- Train transformers with multiple heads and various numbers of layers on these sequences;
- Standard cross-entropy loss, i.e., $-\log \tilde{p}(x_t)$, the compression rate.

Testing:

- New context trees and sequences, the same way as training sequence generation;
- Evaluate the average cross-entropy loss on these new sequences.



Questions 1: Can Transformers Perform Optimal Compression?

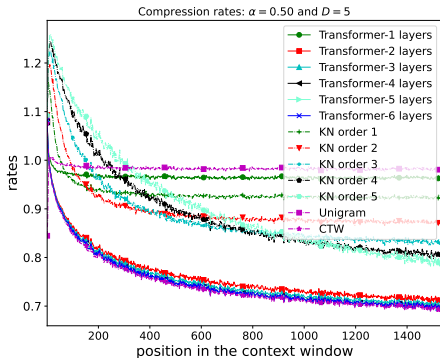


Compare the performances over the whole context window (not only eventually but in-time):

- Single-layer transformer **cannot** learn VOMCs;
- Transformers with **2 or more** layers can learn VOMCs: matching the CTW performance;
- Deeper transformers see small but noticeable improvements;



Questions 1: Can Transformers Perform Optimal Compression?

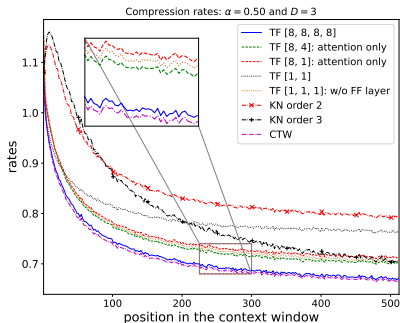


Compare the performances over the whole context window (not only eventually but in-time):

- Single-layer transformer **cannot** learn VOMCs;
- Transformers with **2 or more** layers can learn VOMCs: matching the CTW performance;
- Deeper transformers see small but noticeable improvements;
- Kneser-Ney smoothing performs poorly: attempts to use the longest suffix as much as possible.



Question 2: How Can Transformers Perform Optimal Compression?

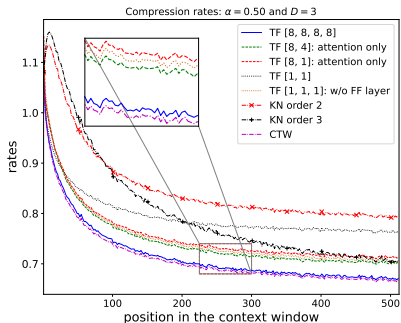


Do mechanisms for fixed-order Markov chains work?

- Attention-only networks (Edelman et al. 2024, Nichani et al. 2024);
- Single-head attention networks with layer normalization but without the feedforward (FF) layer (Rajaraman et al. 2024);



Question 2: How Can Transformers Perform Optimal Compression?

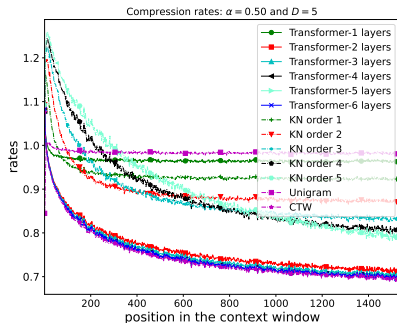


Do mechanisms for fixed-order Markov chains work?

- Attention-only networks (Edelman et al. 2024, Nichani et al. 2024);
- Single-head attention networks with layer normalization but without the feedforward (FF) layer (Rajaraman et al. 2024);
- No: These mechanisms are not good enough!



Question 2: How Can Transformers Perform Optimal Compression?

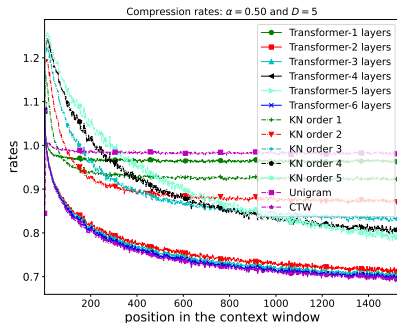


Capability? Construct a transformer to understand “how” transformers can learn VOMCs:

- Consideration A: Avoid exotic information embedding or too large embedding space;



Question 2: How Can Transformers Perform Optimal Compression?

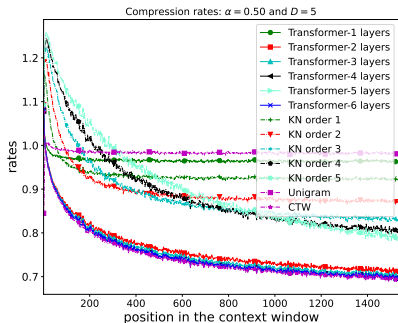


Capability? Construct a transformer to understand “how” transformers can learn VOMCs:

- Consideration A: Avoid exotic information embedding or too large embedding space;
- Consideration B: Avoid unrealistically many heads;



Question 2: How Can Transformers Perform Optimal Compression?



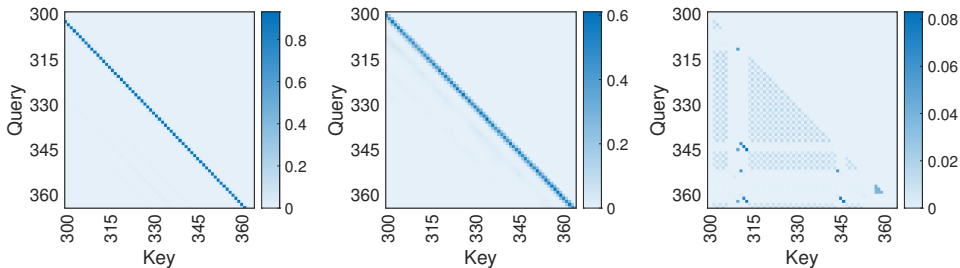
Capability? Construct a transformer to understand “how” transformers can learn VOMCs:

- Consideration A: Avoid exotic information embedding or too large embedding space;
- Consideration B: Avoid unrealistically many heads;
- Consideration C: Avoid unrealistically many layers.

Reasoning: 1) Keep the parameters scale reasonably for larger models, 2) Empirical observations in our experiments.



Question 2: How Can Transformers Perform Optimal Compression?

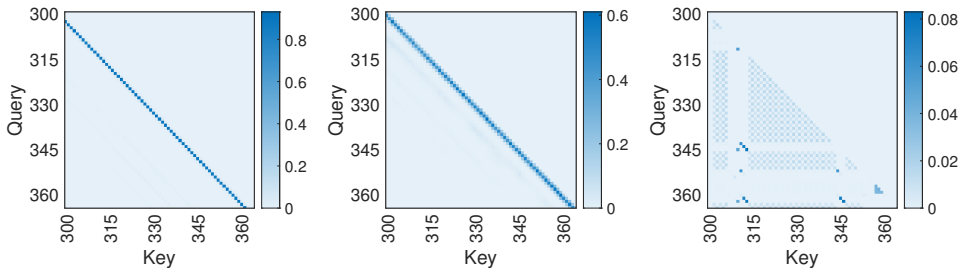


Attention heatmaps:

- Type 1: diagonal or diagonal-like pattern: attention determined by the relative location, suggesting copying certain information from tokens at the previous one or several positions;
- Left vs. middle: flexible copying, i.e., a linear combination of several previous tokens.



Question 2: How Can Transformers Perform Optimal Compression?

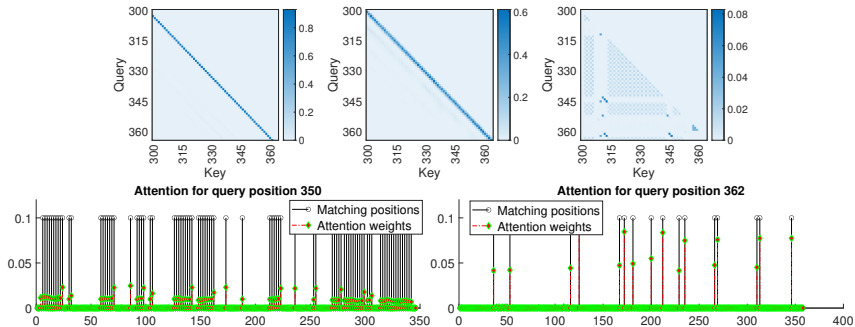


Attention heatmaps:

- Type 1: diagonal or diagonal-like pattern: attention determined by the relative location, suggesting copying certain information from tokens at the previous one or several positions;
- Left vs. middle: flexible copying, i.e., a linear combination of several previous tokens.
- Type 2 (right): more sophisticated patterns depending on other factors??



Question 2: How Can Transformers Perform Optimal Compression?

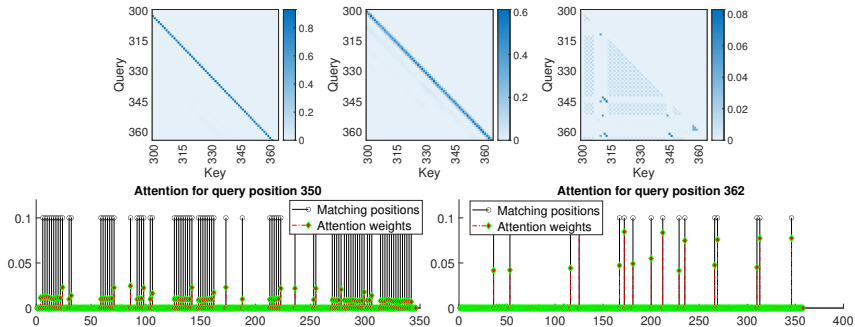


Isolating type 2:

- High attention values occur at positions matching certain patterns;



Question 2: How Can Transformers Perform Optimal Compression?

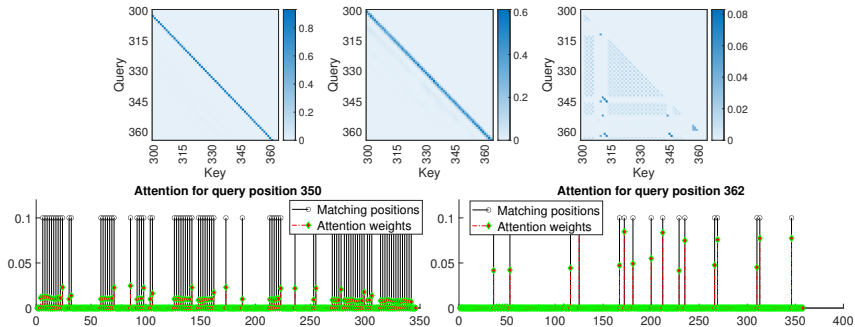


Isolating type 2:

- High attention values occur at positions matching certain patterns;
- The suffix is being matched: the attention weights indicate exactly the positions for the suffix of length-3 in this example attention head!



Question 2: How Can Transformers Perform Optimal Compression?

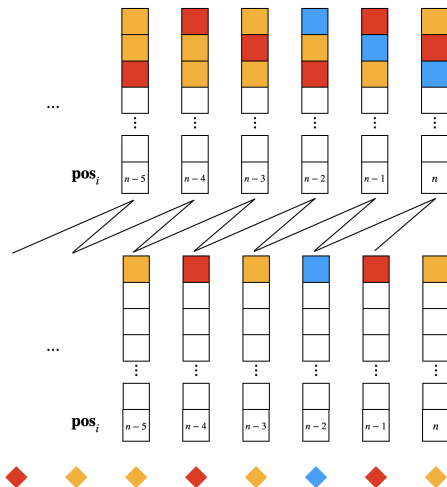


Isolating type 2:

- High attention values occur at positions matching certain patterns;
- The suffix is being matched: the attention weights indicate exactly the positions for the suffix of length-3 in this example attention head!
- Other attention heads can match prefixes of other lengths.



Constructing the First Transformer Layer



Assuming each token is vectorized with one-hot embedding, the finite-memory context-extension layer outputs:

$$\text{output} = \begin{pmatrix} \mathbf{x}_n \\ \mathbf{x}_{n-1} \\ \vdots \\ \mathbf{x}_{n-D} \\ \mathbf{0} \\ \cos(n\pi/N) \\ \sin(n\pi/N) \end{pmatrix}$$

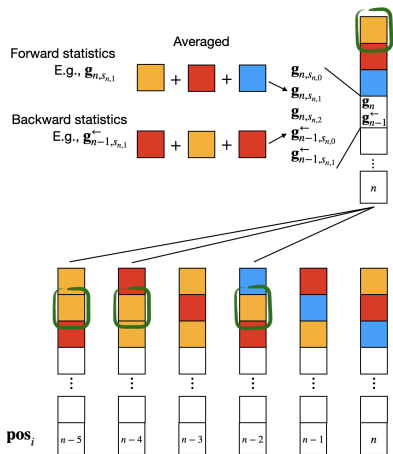
The m -th attention head picks position- $(n-m)$ token, purely relying on the positional encoding:

$$W_{Q,m} \mathbf{h}_n = \begin{pmatrix} \mathbf{0} & \text{Rot}(m) \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}_n \\ \mathbf{0} \\ \cos(n\pi/N) \\ \sin(n\pi/N) \end{pmatrix}$$

A large softmax coefficient \Rightarrow attends the exact position



Constructing the Second Transformer Layer



Statistics collection layer: both forward and backward normalized frequencies:

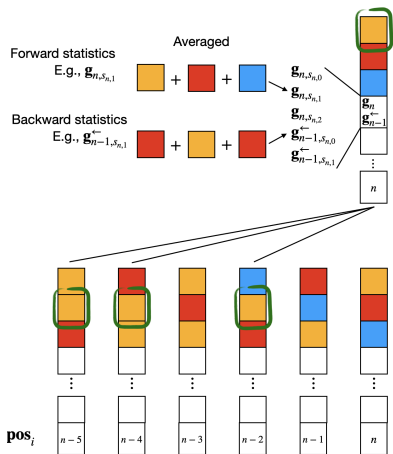
- Exact length- k suffix matching: attention value is k ;
- Large softmax coefficient \Rightarrow the exact matching only;
- One-hot embedding assuming K exact matching positions:

$$\frac{1}{K} \left[\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} c_1/K \\ c_2/K \\ c_3/K \end{pmatrix}$$

i.e., the normalized frequency of the next token for a given suffix;



Constructing the Second Transformer Layer



Statistics collection layer: both forward and backward normalized frequencies:

- Exact length- k suffix matching: attention value is k ;
- Large softmax coefficient \Rightarrow the exact matching only;
- One-hot embedding assuming K exact matching positions:

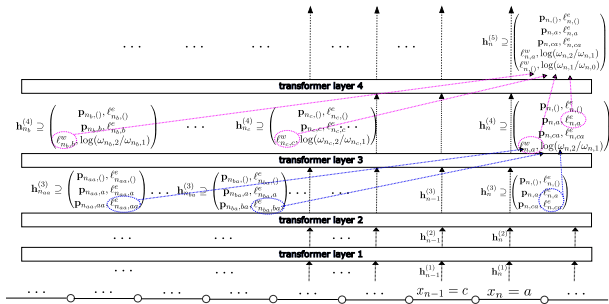
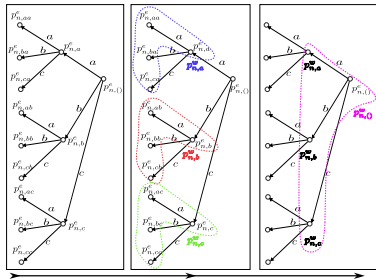
$$\frac{1}{K} \left[\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} c_1/K \\ c_2/K \\ c_3/K \end{pmatrix}$$

i.e., the normalized frequency of the next token for a given suffix;

- Cannot compute the counts easily: need to use the backward statistics.



A Construction That Can Perform CTW



A $D + 2$ layer transformer construction to learn VOMCs with a maximum order D :

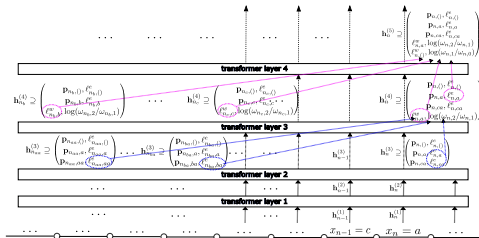
- The lower two layers perform statistics collection, and the upper D layers aggregate info like CTW;
- Technical difficulty: need to rewrite the CTW algorithm to make it compatible with transformer computation.



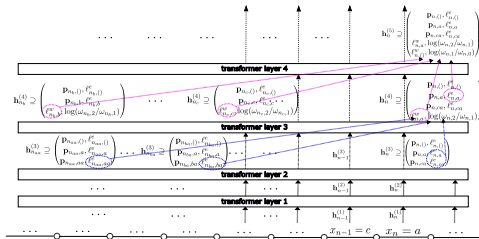
Question 3: Why Do 2-layer Transformers Work So Well?

The forward + backward statistics give $\mathbf{p}_{n,s}$

- The optimal weights $\omega_{n,i}$'s need to be obtained through CTW-like steps;



Question 3: Why Do 2-layer Transformers Work So Well?

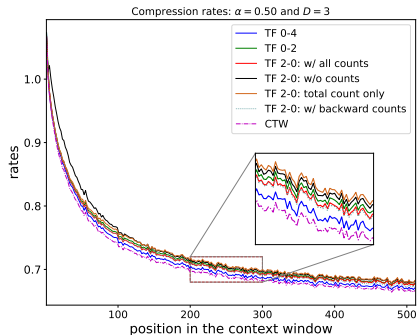


The forward + backward statistics give $\mathbf{p}_{n,s}$

- The optimal weights $\omega_{n,i}$'s need to be obtained through CTW-like steps;
- The FF layer can learn to approximate these weights $\omega_{n,i}$'s?
- We verify it using synthetic and conventional transformer layers;
- Synthetic layers: based on the constructions above or other variants.



Question 3: Why Do 2-layer Transformers Work So Well?

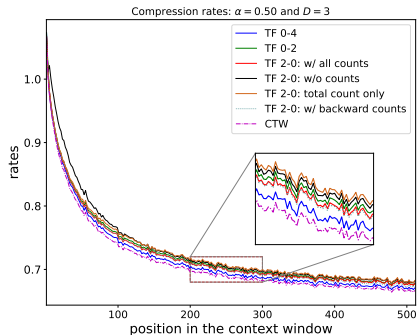


Hybrid transformers: $n - k$ means n synthetic layers & k transformer layers

- $2 - 0$ = the constructed version; $2 - 0$ w/ all counts = artificially extracted counts;
- FF layer approximately computes the counts, and is likely the mechanism in the 2-layer transformers;



Question 3: Why Do 2-layer Transformers Work So Well?

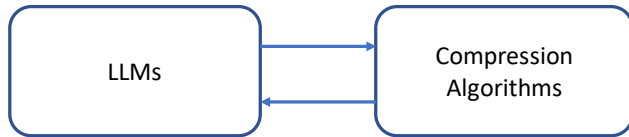


Hybrid transformers: $n - k$ means n synthetic layers & k transformer layers

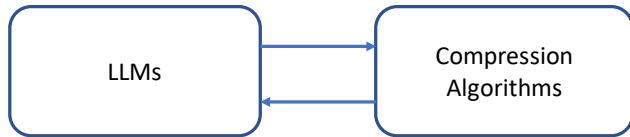
- $2 - 0$ = the constructed version; $2 - 0$ w/ all counts = artificially extracted counts;
- FF layer approximately computes the counts, and is likely the mechanism in the 2-layer transformers;
- $2 - 0$ is similar to but slightly better than $0 - 2$;
- $2 - 0$ w/o counts is not good, particularly at the beginning; trading-off within the context window;
- $2 - 0$ w/ total counts is another tradeoff within the context window;



LLMs vs. Compression Algorithms



LLMs vs. Compression Algorithms

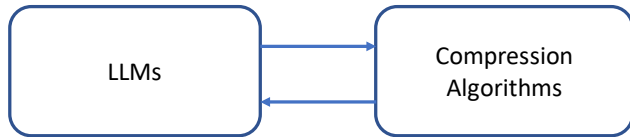


Many questions naturally arise, for example:

- ← LLMs are black boxes mostly. Can we extract components similar to those in compression algorithms, in order to interpret the underlying mechanism?



LLMs vs. Compression Algorithms

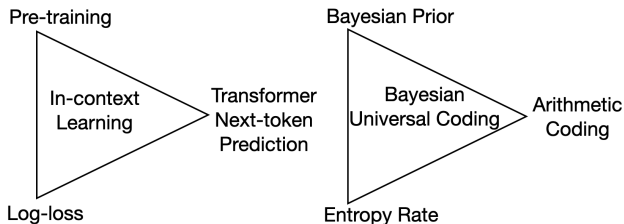


Many questions naturally arise, for example:

- ← LLMs are black boxes mostly. Can we extract components similar to those in compression algorithms, in order to interpret the underlying mechanism?
- → LLMs can compress much better. Can we design new compression algorithms that utilize some mechanisms in LLMs?



LLM ICL vs. Bayesian Universal Compression



- Pre-training is to use data generated from the prior distribution to train the model;
- Arithmetic coding: using the estimated probability distribution to compress;
- The log-loss function is in fact exactly the entropy rate: the rates over the whole context window rate are important



Conclusion

We study how transformers learn variable-order Markov chains in context

- This perspective allows us to use mature (optimal) compression algorithms as references;
- Transformers can capture patterns in a more flexible manner, beyond simple suffixes;
- **FF layers are important** in more complex settings, in contrast to in ICL of linear regression.

We have leveraged deep IT results in the study of transformers for the complex task of ICL-VOMCs.



Thank You!