Analysis of Statistical Language Models

N-gram Models, Neural Representations, and Transformer LMs

June 5, 2026

Seoul National University

Motivation: Statistical Language Modeling

Language Modeling in Everyday Life

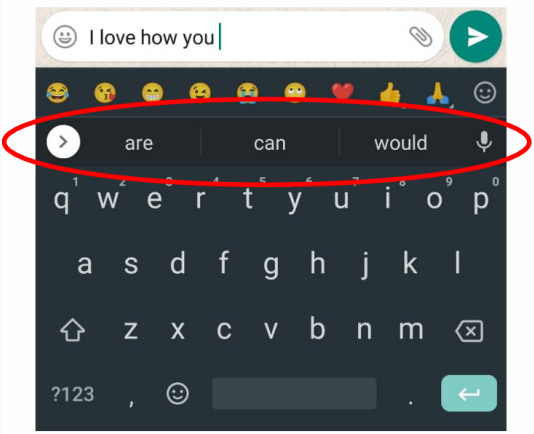


Figure 1: Keyboard autogeneration

What Is the Goal of Statistical Language Modeling?

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language.

[2], Bengio et al. (2003)

For a token sequence

$$x_{1:T} = (x_1, \ldots, x_T),$$

we want a probability model for

$$p(x_{1:T}).$$

A standard autoregressive form (chain rule) is

$$p(x_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_{<t}).$$

Estimating this joint probability is difficult because token sequences live in a large discrete space.

Why Is This Hard? Curse of Dimensionality

Let the vocabulary be

$$\mathcal{V} = \{\text{cat}, \text{dog}, \text{the}, \text{room}, \ldots\}.$$

Each token is a discrete random variable taking values in $\mathcal{V}$.

A length-m token sequence has

$$|\mathcal{V}|^m$$

possible configurations.

For example,

$$|\mathcal{V}| = 100{,}000, \qquad m = 10 \quad \Rightarrow \quad |\mathcal{V}|^{10} = 10^{50}.$$

A test sequence is likely to differ from every sequence observed during training.

Classical N-gram Models

An N-gram model uses only a finite history.

Model	Assumption	history
Unigram	$p(x_t \mid x_{<t}) \approx p(x_t)$	none
Bigram	$p(x_t \mid x_{<t}) \approx p(x_t \mid x_{t-1})$	previous token
Trigram	$p(x_t \mid x_{<t}) \approx p(x_t \mid x_{t-2}, x_{t-1})$	previous two tokens
N-gram	$p(x_t \mid x_{<t}) \approx p(x_t \mid x_{t-N+1:t-1})$	previous $N-1$ tokens

Instead of all $|\mathcal{V}|^m$ full sequences, an N-gram model only parameterizes the $|\mathcal{V}|^{N-1}$ possible histories.

There is a trade-off:

- larger N: more context, but $|\mathcal{V}|^{N-1}$ histories grow $\Rightarrow$ more sparsity;
- smaller N: less sparsity, but less context.

Count Table Parameterization

For history

$$h = x_{t-N+1:t-1},$$

a count-based N-gram estimate is

$$\widehat{p}(x_t = a \mid h) = \frac{c(h, a)}{c(h)}.$$

Example count table:

History h	Next token a	Count
`denied the`	`allegations`	5
`denied the`	`speculation`	2
`denied the`	`rumors`	1
`denied the`	`report`	1
`denied the`	`offer`	0
`denied the`	`loan`	0

Limitation 1: Zero Counts

From the previous table,

$$\widehat{p}(\texttt{offer} \mid \texttt{denied the}) = 0, \qquad \widehat{p}(\texttt{loan} \mid \texttt{denied the}) = 0.$$

But `denied the offer` is a possible phrase; it was simply unseen.

On the N-gram side, smoothing and backoff redistribute mass to unseen events,

$$\text{observed counts} \longrightarrow \text{probability mass for unseen events},$$

for example Katz backoff [4], Katz (1987).

Smoothing fixes zero counts, but a second, deeper problem remains.

Limitation 2: Similar Contexts

Example:

"The cat is walking in the bedroom."

A related sentence is

"A dog was running in a room."

Relevant similarities:

$$\text{cat} \approx \text{dog}, \qquad \text{walking} \approx \text{running}, \qquad \text{bedroom} \approx \text{room}.$$

A pure count table treats these as different surface forms.

Smoothing does not capture this similarity. Distributed representations are designed to share mass across similar words and similar contexts.

Neural Probabilistic Language Model

Two objects are learned:

1. an embedding for each token,
$$C_{\text{emb}}(a) \in \mathbb{R}^d,$$
2. a neural probability function over token sequences.

For a fixed window,

$$f(i, x_{t-1}, \ldots, x_{t-n+1}) = g_\theta(i, C_{\text{emb}}(x_{t-1}), \ldots, C_{\text{emb}}(x_{t-n+1})),$$

where

$$f(i, x_{t-1}, \ldots, x_{t-n+1}) = \widehat{p}(x_t = i \mid x_{t-1}, \ldots, x_{t-n+1}).$$

Similar tokens get similar embeddings, so mass is shared across similar contexts.

Notation: C_{emb}, g_θ, f, and θ

Symbol	Meaning
C_{emb}	embedding lookup map $\mathcal{V} \to \mathbb{R}^d$; implemented as a matrix
$C_{\text{emb}}(a)$	vector representation of token a
g_θ	neural network mapping context vectors to a next-token distribution
f	full language-model function obtained by composing C_{emb} and g_θ
θ	all trainable parameters

Classical N-gram: parameters are count-table probabilities.

Neural LM: parameters are embeddings and neural-network weights.

Neural Probabilistic LM: Architecture

Let

$$z = (C_{\mathrm{emb}}(x_{t-1}), C_{\mathrm{emb}}(x_{t-2}), \ldots, C_{\mathrm{emb}}(x_{t-n+1})).$$

A one-hidden-layer model computes logits

$$\ell = b + Wz + U \tanh(d + Hz).$$

The next-token distribution is

$$\widehat{p}_\theta(x_t = i \mid x_{t-n+1:t-1}) = \frac{\exp(\ell_i)}{\sum_{j \in \mathcal{V}} \exp(\ell_j)}.$$

From Neural LM to Transformer LM

Modern Transformer LMs keep the same probability modeling goal:

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}).$$

The context function changes:

$$\text{fixed-window neural network} \longrightarrow \text{causal Transformer representation function.}$$

This representation function is the decoder-only causal stack, not the encoder block of the original encoder–decoder Transformer.

Transformers Can Represent n-gram LMs, Svete & Cotterell

Transformers Can Represent n-gram LMs

[6], Svete and Cotterell (2024)

Question:

What probability distributions over strings can Transformer LMs represent?

Main result:

Hard/sparse attention Transformer LMs can exactly represent any n-gram LM.

Here *hard* and *sparse* attention place all attention mass on a single position, rather than spreading it over all previous tokens as standard softmax does. Formal definitions follow.

Language Models over Strings

Let Σ be a finite alphabet (here, the token vocabulary) and $\Sigma^\star$ the set of all finite strings over Σ.

A string $x \in \Sigma^\star$ is a finite token sequence $x = x_1 \cdots x_{|x|}$.

Example: $\Sigma = \{a, b\}$,

$$\Sigma^\star = \{\varepsilon,\ a,\ b,\ aa,\ ab,\ ba,\ bb,\ aaa, \ldots\},$$

where ε is the empty string.

With an end-of-string symbol EOS, a language model assigns each string a probability

$$p(x) = p(\mathrm{EOS} \mid x) \prod_{t=1}^{|x|} p(x_t \mid x_{<t}).$$

Definition 2.1: Representation-based LM

Definition 1 (Representation-based LM)

Let $\mathrm{enc} : \Sigma^\star \to \mathbb{R}^D$ be a representation function and $E \in \mathbb{R}^{|\Sigma| \times D}$ an output matrix. A representation-based LM defines

$$p(x_t \mid x_{<t}) = \mathrm{softmax}(E\,\mathrm{enc}(x_{<t}))_{x_t}.$$

A Transformer LM is the case where enc is computed by a Transformer: the decoder-only causal stack mapping the prefix $x_{<t}$ to the contextual representation of its last token.

Definition 2.2: Weak Equivalence

Definition 2 (Weak equivalence)

Two LMs p and q over $\Sigma^\star$ are weakly equivalent if

$$p(x) = q(x) \qquad \forall x \in \Sigma^\star.$$

This compares full string probabilities, not only one next-token distribution.

Definition 2.11: Transformer LM

Definition 3 (Transformer LM)

A Transformer LM p_T is the representation-based autoregressive LM with representation function enc computed by a Transformer:

$$p_T(x_t \mid x_{<t}) = \mathrm{softmax}(E\,\mathrm{enc}(x_{<t}))_{x_t}.$$

Hard and Sparse Attention

Definition 4 (Hard attention)

Hard attention uses hardmax: attention mass is placed only on positions with maximum score.

Definition 5 (Sparse attention)

Sparse attention uses

$$\text{sparsemax}(x) = \arg\min_{p \in \Delta^{D-1}} \|p - x\|_2^2.$$

The exact theorem is for hard/sparse attention. It does not claim that standard soft attention implements the same construction exactly.

Svete and Cotterell: Representability

Main Result: Transformer LMs Represent n-gram LMs

Theorem 6 (Svete–Cotterell, informal)

For every n-gram LM, there exists a weakly equivalent hard-attention or sparse-attention Transformer LM.

In symbols:

$$p_T(x) = p_{n\text{-gram}}(x) \qquad \forall x \in \Sigma^\star.$$

This is a lower bound on probabilistic representational capacity:

$$\{n\text{-gram LMs}\} \subseteq \{\text{Transformer-representable LM distributions}\}.$$

4-gram Simulation

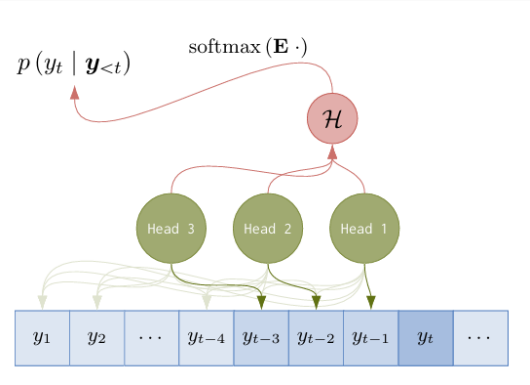


Figure 2: A transformer LM can simulate a 4-gram LM using 3 heads. The stronger arrows from the heads to the symbols show where the heads focus their attention.

Output Matrix as Lookup Table

Suppose the Transformer has identified the history

$$h = x_{t-n+1:t-1}.$$

Encode the history as a one-hot vector:

$$\mathrm{enc}(x_{<t}) = [\![h]\!].$$

Define the output matrix by storing conditional log probabilities:

$$E_{a,h} = \log p_{n\text{-gram}}(a \mid h).$$

Then

$$\mathrm{softmax}(E[\![h]\!])_a = p_{n\text{-gram}}(a \mid h).$$

The output matrix is used as a conditional-probability lookup table.

Three Mechanisms and Non-uniqueness

The $n-1$ heads construction is one of three:

Thm	Resource	Mechanism
3.1	$n-1$ heads, 1 layer	each head reads one history position in parallel
3.2	1 head, $n-1$ layers	each layer copies one symbol forward (shift register)
3.3	1 head, 1 layer	single head reads all positions via digit-coded values; MLP decodes

$$\#\text{ heads} \longleftrightarrow \#\text{ layers} \longleftrightarrow \text{nonlinearity complexity.}$$

All three define the same string distribution. So there is no canonical map from an n-gram LM to transformer components: inspecting which positions a single head attends to cannot certify that the model "is" an n-gram LM.

What Svete and Cotterell Establish

Proposition 7 (Representability)

Transformer LMs have enough probabilistic representational capacity to exactly represent any n-gram LM under hard/sparse attention constructions.

What this says:

- Transformer LMs can represent classical n-gram distributions.
- The relationship is about full string probabilities.
- There are multiple mechanisms: heads, layers, or digit encoding.

What this does not say:

- It does not prove that trained Transformers actually learn these constructions.
- It does not analyze standard soft attention exactly.

Bridge: Representability vs. Learned Behavior

Svete–Cotterell answer:

Can a Transformer LM represent an n-gram LM?

Answer:

Yes, under hard/sparse attention constructions.

Next question:

Do trained Transformers actually produce predictions describable by N-gram statistics?

This is Nguyen's question.

Nguyen: Learned Predictions and N-gram Statistics

Paper: Understanding Transformers via N-gram Statistics

Paper:

Understanding Transformers via N-gram Statistics

Question:

How does a trained Transformer use its context when predicting the next token?

Approach:

$$p_\theta(t \mid C) \approx R_\sigma(t \mid C),$$

where R_σ is a rule built from training-data N-gram statistics.

This is a descriptive approximation of model outputs, not a proof of internal mechanism.

Form vs. Selection

Nguyen separates two questions.

Form

$$p_\theta(t \mid C) \approx R(t \mid C)$$

- Is the prediction rule-like?
- Is it a function of training-data statistics?

Selection

$$C \mapsto R_C^\star$$

- Which rule is selected?
- Why that rule for this context?

The paper primarily measures form. Selection remains the harder explanatory problem.

Rule Notation: Empirical N-gram Rule

Given a rule context α, define

$$R_\alpha(t) = \frac{\#(\alpha t)}{\#(\alpha *)}.$$

- $\#(\alpha t)$: number of times pattern α is followed by token t in training data.
- $\#(\alpha *)$: number of times α is followed by any token.

Vanilla N-gram prediction is a special case where α is an ordinary suffix context.

Keep, Drop, Marginalize

Let the context be

$$C = C_{-N} \cdots C_{-2} C_{-1}.$$

Nguyen uses three token-level operations:

$$+ : \text{keep}, \qquad - : \text{drop}, \qquad * : \text{marginalize}.$$

For a symbol $\sigma \in \{+, -, *\}$, define

$$S_\sigma(t) = \begin{cases} t, & \sigma = +, \\ \epsilon, & \sigma = -, \\ *, & \sigma = *. \end{cases}$$

For an operation sequence σ, the rule is

$$R_\sigma(t \mid C) = R_{S_\sigma(C)}(t).$$

Example: Rule Construction

Let

$$C = C_{-4}C_{-3}C_{-2}C_{-1}, \qquad \sigma = + - * + .$$

Then

$$S_\sigma(C) = C_{-4} * C_{-1}.$$

The rule distribution is

$$R_\sigma(t \mid C) = \frac{\#(C_{-4} * C_{-1}t)}{\#(C_{-4} * C_{-1}*)}.$$

Interpretation:

- keep C_{-4} and C_{-1};
- drop C_{-3};
- marginalize over C_{-2}.

Worked Example: Rules on `the big brown dog`

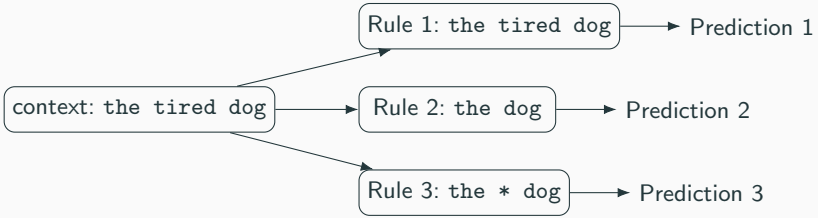
Predict the token after $C =$ `the big brown dog` ($C_{-4}C_{-3}C_{-2}C_{-1}$).

σ	$S_\sigma(C)$	rule $R_\sigma(t \mid C)$
$++++$	`the big brown dog`	full 4-gram suffix
$---+$	`dog`	$\#(\texttt{dog}\ t)/\#(\texttt{dog}\ *)$: bigram $p(t \mid \texttt{dog})$
$+--+$	`the dog`	$\#(\texttt{the dog}\ t)/\#(\texttt{the dog}\ *)$: contiguous trigram
$+**+$	`the * * dog`	$\#(\texttt{the}* *\texttt{dog}\ t)/\#(\texttt{the}* *\texttt{dog}\ *)$

Drop vs. marginalize: $+--+$ collapses to the contiguous context `the dog`; $+**+$ keeps four slots and sums counts over the two middle tokens. Same kept endpoints, different count query.

Nguyen Figure 1: Rule Approximation

For context `the tired dog`, different rules give different next-token distributions.



Schematic version of Nguyen's rule approximation idea.

Rule Families

Nguyen groups rules by increasing flexibility.

Ruleset	Meaning
$\mathcal{R}_M^{\text{suffix}}$	ordinary suffix rules; only recent contiguous context
$\mathcal{R}_M^{\text{subgram}}$	keep/drop rules; non-contiguous context allowed
$\mathcal{R}_M^{\text{all}}$	keep/drop/marginalize rules; most flexible family

$$\mathcal{R}_M^{\text{suffix}} \subset \mathcal{R}_M^{\text{subgram}} \subset \mathcal{R}_M^{\text{all}}.$$

Measuring Rule Approximation

Nguyen uses variational distance:

$$d(p, q) = \frac{1}{2} \sum_{t \in \mathcal{V}} |p(t) - q(t)|.$$

For context C, the optimal rule is

$$r^\star(C) = \arg\min_{r \in \mathcal{R}} d(p_\theta(\cdot \mid C), p_r(\cdot \mid C)).$$

The optimal rule distance is

$$\mathrm{ORD}(C) = \min_{r \in \mathcal{R}} d(p_\theta(\cdot \mid C), p_r(\cdot \mid C)).$$

Model Variance

Train multiple models with different dataset shuffles. For the same context C, compare their predictions:

$$p_\theta^{(i)}(\cdot \mid C), \qquad p_\theta^{(j)}(\cdot \mid C).$$

Model variance is

$$\mathrm{MV}(C) = \mathrm{avg}_{i,j}\, d(p_\theta^{(i)}(\cdot \mid C), p_\theta^{(j)}(\cdot \mid C)).$$

Low $\mathrm{MV}(C)$ means predictions are stable across training runs.

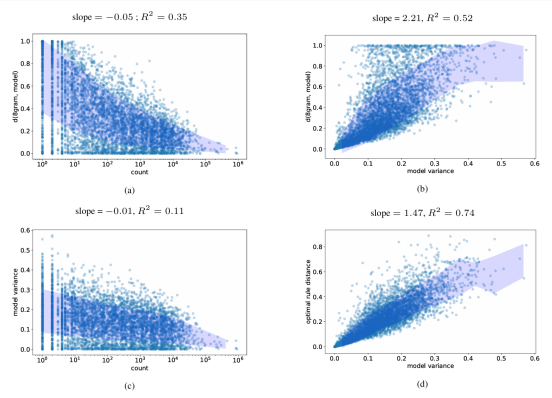
Finding 1: Approximation Criterion

Main empirical criterion:

$$\mathrm{MV}(C) \text{ small} \quad \Longrightarrow \quad \mathrm{ORD}(C) \text{ small.}$$

Interpretation:

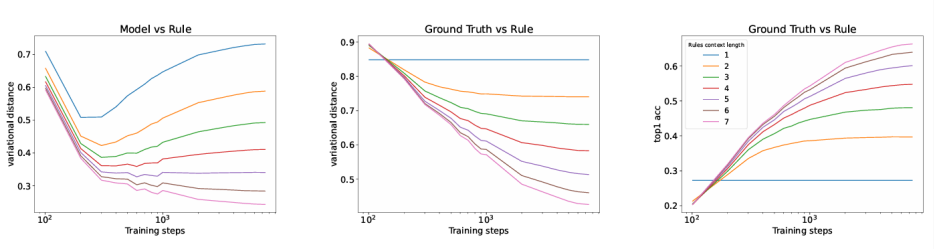
- stable predictions tend to be rule-like;
- high count helps, but count alone is not the strongest signal;
- rare contexts can still be rule-like if model variance is low.



Finding 2: Curriculum Learning Dynamics

During training, Transformer predictions move from simple rules to more complex rules.

unigram/bigram → short suffix rules → longer context rules → subgram/marginalized rules.



This gives a statistical form of curriculum learning.

Finding 3: Overfitting Criterion

For maximum context length n, define

$$p_n(t \mid C) = p(t \mid C_{-n} \cdots C_{-1}).$$

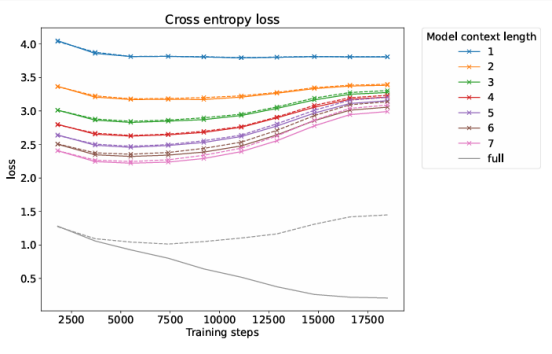
Overfitting picture:

memorize full context

at the expense of

subcontext generalization.

Short-context train performance can indicate overfitting without a holdout set.



Finding 4: Approximation Strength

Larger rule family and larger context each raise top-1 agreement with model predictions.

Ruleset	max context length M						
	1	2	3	4	5	6	7
$\mathcal{R}_M^{\text{all}}$	30.0	45.1	55.0	63.2	69.6	75.0	**78.9**
$\mathcal{R}_M^{\text{subgram}}$	30.0	44.7	53.5	60.4	65.1	68.5	71.2
$\mathcal{R}_M^{\text{suffix}}$	30.0	44.5	52.6	58.0	61.4	63.6	65.2
backoff$_M$	30.0	42.5	48.7	52.7	54.6	55.9	56.7

TinyStories, L_∞-selected, 160M.

Column: backoff $<$ suffix $<$ subgram $<$ all. Row: accuracy rises with M.

$\mathcal{R}_7^{\text{all}}$ reaches 78.9% (TinyStories) and 67.7% (Wikipedia, 1.4B). The model's own top-1 vs. ground truth is 69.6%, so the rule tracks the model's greedy choice more often than the model tracks the data.

Svete + Nguyen: Combined Message

Svete–Cotterell

Transformer LMs can represent

n-gram LMs.

- theory
- model class
- representability

Nguyen

Trained predictions are often

approximated by N-gram rules.

- empirical analysis
- trained models
- learned behavior

capacity to represent + evidence of rule-like behavior

Appendix

Appendix: Why the Count Ratio Is an MLE

For a fixed history h, let

$$\theta_{a|h} = P(a \mid h), \qquad \sum_{a \in \mathcal{V}} \theta_{a|h} = 1.$$

The likelihood for observed counts is

$$L(\theta_{\cdot|h}) = \prod_{a \in \mathcal{V}} \theta_{a|h}^{c(h,a)}.$$

Maximizing under the simplex constraint gives

$$\widehat{\theta}_{a|h} = \frac{c(h,a)}{c(h)}.$$

Thus the count ratio is the MLE for each multinomial row.

Appendix: Where Earlier Papers Fit

Paper	Role in this talk
Shannon [5]	language as stochastic source
Katz [4]	sparse count-based probability estimation
Bengio et al. [2]	neural representation-based smoothing
Vaswani et al. [7]	Transformer as attention-based context representation
Yang et al. [9]	output softmax as probability-family constraint

Appendix: Output Map and the Softmax Bottleneck

Svete–Cotterell's construction uses the output matrix as a lookup table:

$$E_{a,h} = \log p_{n\text{-gram}}(a \mid h).$$

Yang et al. ask whether practical softmax output maps are expressive enough.

If

$$L_\theta = H_\theta W_\theta^\top,$$

then

$$\operatorname{rank}(L_\theta) \leq d.$$

This is the softmax bottleneck: even a powerful context representation can be limited by the output probability map.

Appendix: Theorem 3.1 — Statement

Theorem 8 (Svete–Cotterell, Theorem 3.1)

For any n-gram LM, there exists a weakly equivalent single-layer hard attention transformer LM with $n-1$ heads.

Construction. Store the n-gram log-probabilities in the output matrix (paper, Eq. 55):

$$E_{y,\, y_{t-n+1}^{t-1}} \stackrel{\text{def}}{=} \log p\Big(y \mid y_{t-n+1}^{t-1}\Big)\,.$$

Build the transformer so that the representation one-hot encodes the history:

$$\operatorname{enc}(y_{<t}) \;=\; [\![y_{t-n+1}^{t-1}]\!].$$

Appendix: Lemmas B.3–B.4 — Heads Identify Symbols

With unit-vector positional encodings and a dot-product score, head h attends to exactly one position.

Lemma 9 (Svete–Cotterell, Lemma B.3)

For head h, the score $\langle q, k_j \rangle$ between the query at the prediction step and the key at position j is uniquely maximized at the targeted position h steps back.

Why. The query and keys are unit vectors, so $\langle q, k_j \rangle \leq 1$ with equality iff they coincide; equality of the position-encoded entries forces j to be the targeted position.

Lemma 10 (Svete–Cotterell, Lemma B.4)

Head h outputs $z_h = [\![y_{t-h}]\!]$, the one-hot of the symbol h positions back. For $h = 1, \ldots, n-1$ these are the history symbols $y_{t-1}, \ldots, y_{t-n+1}$.

Why. B.3 makes hard attention select that position; the value map sends the symbol there to its one-hot, and the output map cancels the residual copy.

Appendix: Lemmas B.1–B.2 — Combine into History Code

Lemma 11 (Svete–Cotterell, Lemma B.1)

For $x, v \in \{0,1\}^D$ with $v_i = \mathbf{1}\{i \in \{i_1, \ldots, i_m\}\}$,

$$\mathrm{ReLU}\big(v^\top x - (m-1)\big) = x_{i_1} \wedge \cdots \wedge x_{i_m}.$$

Why. On $\{0,1\}^D$, $v^\top x \leq m$ with equality iff every selected entry is 1; ReLU sends all smaller values to 0.

The $n-1$ head outputs $[\![y_{t-1}]\!], \ldots, [\![y_{t-n+1}]\!]$ concatenate into a multi-hot vector. B.1 with $m = n-1$ (offset $n-2$) fires only when all history tokens match.

Lemma 12 (Svete–Cotterell, Lemma B.2)

A single-layer ReLU MLP $\mathcal{H}$ satisfies

$$\mathcal{H}\big([\![y_{t-1}]\!], \ldots, [\![y_{t-n+1}]\!]\big) = [\![y_{t-n+1}^{t-1}]\!].$$

Why. The history index is the logical AND of the per-position symbol indicators (an instance of B.1).

Appendix: Theorem 3.1 — Proof Skeleton

Three steps (paper, Lemmas B.2 and B.4):

1. Each of the $n-1$ heads attends to exactly one previous position and outputs the one-hot encoding of the symbol there (Lemma B.4).
2. A ReLU MLP head-combiner $\mathcal{H}$ turns the $n-1$ one-hot symbols (a multi-hot vector) into the one-hot history code, via a logical AND (Lemma B.2):

$$\mathcal{H}(\llbracket y_1 \rrbracket, \ldots, \llbracket y_{n-1} \rrbracket) = \llbracket y_{t-n+1}^{t-1} \rrbracket.$$

3. The output matrix E maps this one-hot history to the n-gram log-probabilities.

Appendix: Theorem 3.1 — Proof (line by line)

Let $\mathcal{T}$ be the transformer LM above, $y \in \{\mathrm{BOS}\}^{n-1}\Sigma^\star$ with $|y| = T$. Then

$$
\begin{aligned}
p_{\mathcal{T}}(y) &= p_{\mathcal{T}}(\mathrm{EOS} \mid y) \prod_{t=1}^{T} p_{\mathcal{T}}(y_t \mid y_{<t}) && \text{(autoregressive LM)} \\
&= \mathrm{softmax}\big(E\, F(x_T^L)\big)_{\mathrm{EOS}} \prod_{t=1}^{T} \mathrm{softmax}\big(E\, F(x_{t-1}^L)\big)_{y_t} && \text{(Def. 2.11)} \\
&= \mathrm{softmax}\big(E\, [\![y_{T-n+2:T}]\!]\big)_{\mathrm{EOS}} \prod_{t=1}^{T} \mathrm{softmax}\big(E\, [\![y_{t-n+1}^{t-1}]\!]\big)_{y_t} && \text{(Lemma B.2)} \\
&= \frac{\exp\big(E\, [\![y_{T-n+2:T}]\!]\big)_{\mathrm{EOS}}}{\sum_{y'} \exp\big(E\, [\![y_{T-n+2:T}]\!]\big)_{y'}} \prod_{t=1}^{T} \frac{\exp\big(E\, [\![y_{t-n+1}^{t-1}]\!]\big)_{y_t}}{\sum_{y'} \exp\big(E\, [\![y_{t-n+1}^{t-1}]\!]\big)_{y'}} && \text{(softmax)} \\
&= \frac{\exp\big(\log p(\mathrm{EOS} \mid y_{T-n+2}^{T})\big)}{\sum_{y'} \exp\big(\log p(y' \mid y_{T-n+2}^{T})\big)} \prod_{t=1}^{T} \frac{\exp\big(\log p(y_t \mid y_{t-n+1}^{t-1})\big)}{\sum_{y'} \exp\big(\log p(y' \mid y_{t-n+1}^{t-1})\big)} && \text{(Eq. 55)} \\
&= p(\mathrm{EOS} \mid y_{T-n+2:T}) \prod_{t=1}^{T} p(y_t \mid y_{t-n+1}^{t-1}) \;=\; p(y). && (n\text{-gram is autoregressive})
\end{aligned}
$$

Appendix: Theorem 3.2

Theorem 3.2 trades heads for layers:

$$\text{one head} + (n-1) \text{ layers.}$$

Intuition:

$$\begin{aligned} \text{layer } 1: &\ \text{copy } x_{t-1}, \\ \text{layer } 2: &\ \text{copy } x_{t-2}, \\ &\dots \\ \text{layer } n-1: &\ \text{copy } x_{t-n+1}. \end{aligned}$$

After $n-1$ layers, the current representation contains the entire $n-1$-token history. Head parallelism and layer recursion gather the same history with different resources — the basis for the non-uniqueness message.

Appendix: Theorem 3.3

Theorem 3.3 is more compressed but less intuitive.

A single head attends to all $n-1$ previous positions and encodes symbols using position-scaled digits:

$$r(x,j) = \begin{pmatrix} 10^{-j}[\![x]\!] \\ j \\ 1 \end{pmatrix}.$$

The aggregation contains position information in decimal digits:

$$a_{t-1} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j}[\![x_j]\!].$$

A final MLP decodes the digits and reconstructs the history code.

Appendix: ICL as an Extension

This talk focuses on next-token probability and training-data statistics.

In-context learning extends the same statistical view:

Paper	Role
Brown et al. [3]	prompt examples act like an in-context dataset
Xie et al. [8]	ICL as implicit Bayesian inference over latent concepts
Bai et al. [1]	Transformers implement statistical algorithms in context

Pretraining statistics come from the corpus. ICL also uses statistics supplied inside the prompt.

Appendix: Unified Statistical View

Pretraining level:

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}).$$

In-context level:

$$p_\theta(y \mid x, D_{\text{context}}).$$

Overall:

LLM = autoregressive probability estimator + context-conditioned inference mechanism.

Appendix: Distance Metrics

Three quantities compare two next-token distributions p, q over $\mathcal{V}$.

Variational distance (used for ORD, curriculum):

$$d(p, q) = \tfrac{1}{2} \sum_{t \in \mathcal{V}} |p(t) - q(t)|.$$

Top-1 accuracy:

$$\text{top-1-acc}(p, q) = \frac{|\arg\max(p) \cap \arg\max(q)|}{|\arg\max(p) \cup \arg\max(q)|}.$$

With singleton argmaxes, this is agreement of greedy predictions.

L_∞ distance (used to select rules in the top-1 tables):

$$d_{L_\infty}(p, q) = \max_{t \in \mathcal{V}} |p(t) - q(t)|.$$

Appendix: Why Variational Distance vs. L_∞

d is symmetric and bounded, so it needs no direction choice (unlike KL, infinite when $q(t) = 0 < p(t)$) and outliers do not dominate the token-averaged distance.

L_∞ gives slightly higher rule-approximation numbers for well-trained models, so the top-1 tables (Tables 5, 7) select rules under L_∞.

L_∞ fails for two high-entropy distributions: if all $p(t), q(t)$ are small, d_{L_∞} stays small even when d is large. At initialization the model is near-uniform, so it would sit close to many bad high-entropy rules. The curriculum analysis (§6.1) therefore uses d throughout.

References

[1] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. **"Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection".** In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023. URL: https://arxiv.org/abs/2306.04637.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. **"A Neural Probabilistic Language Model".** In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155. URL: https://www.jmlr.org/papers/v3/bengio03a.html.

References ii

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. **"Language Models Are Few-Shot Learners".** In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[4] Slava M. Katz. **"Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer".** In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35.3 (1987), pp. 400–401. DOI: 10.1109/TASSP.1987.1165125. URL: https://www.cis.upenn.edu/~danroth/Teaching/CS598-05/Papers/Katz87.pdf.

References iii

[5] Claude E. Shannon. **"A Mathematical Theory of Communication".** In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. URL: https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.

[6] Anej Svete and Ryan Cotterell. **"Transformers Can Represent *n*-gram Language Models".** In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Association for Computational Linguistics, 2024, pp. 6845–6881. URL: https://aclanthology.org/2024.naacl-long.381/.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. **"Attention Is All You Need".** In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.

[8] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. **"An Explanation of In-context Learning as Implicit Bayesian Inference".** In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=RdJVFCHjUMI.

[9] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. **"Breaking the Softmax Bottleneck: A High-Rank RNN Language Model".** In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HkwZSG-CZ.