Transformers Can Represent n-gram Language Models

Anej Svete **Ryan Cotterell**
{asvete, ryan.cotterell}@inf.ethz.ch
ETH zürich

Abstract

Existing work has analyzed the representational capacity of the transformer architecture by means of formal models of computation. However, the focus so far has been on analyzing the architecture in terms of language *acceptance*. We contend that this is an ill-suited problem in the study of *language models* (LMs), which are definitionally *probability distributions* over strings. In this paper, we focus on the relationship between transformer LMs and n-gram LMs, a simple and historically relevant class of language models. We show that transformer LMs using the hard or sparse attention mechanisms can exactly represent any n-gram LM, giving us a concrete lower bound on their probabilistic representational capacity. This provides a first step towards understanding the mechanisms that transformer LMs can use to represent probability distributions over strings.

https://github.com/rycolab/transformer-ngrams

1 Introduction

Neural language models (LMs) have become the backbone of many NLP systems. Their widespread adoption has prompted a plethora of theoretical work investigating what they can and cannot do by studying their representational capacity. Most state-of-the-art LMs are based on the transformer architecture (Vaswani et al., 2017), whose theoretical abilities and limitations have been studied extensively; see, e.g., the survey by Strobl et al. (2023). But, many questions remain unanswered. Most existing work studies the architecture in terms of binary language recognition. This introduces a category error between the object of study—an LM, which is definitionally a *distribution* over strings—and the theoretical abstraction—a *set* of strings. To amend this discrepancy, we ask: What classes of probability distributions over strings can transformer LMs represent?

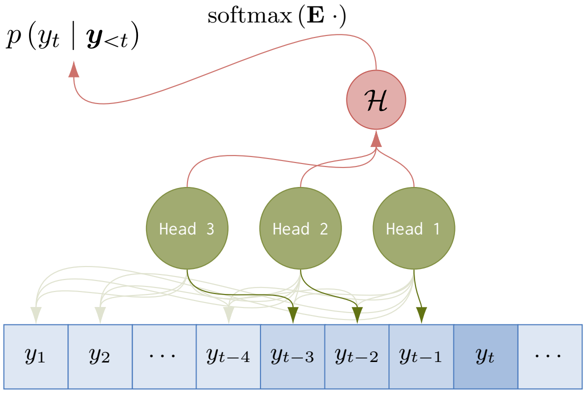

Figure 1: A transformer LM can simulate a 4-gram LM using 3 heads. The stronger arrows from the heads to the symbols show where the heads focus their attention.

Formal models of probabilistic computation provide a natural, well-understood, and precise framework for studying the classes of probability distributions language models can represent. Traditionally, the representational capacity of neural networks, both in terms of lower bounds (what they can provably do) as well as upper bounds (what they can provably *not* do), has been studied in terms of Boolean sequential models of computation, such as finite-state automata and Turing machines (e.g., Kleene, 1956; Minsky, 1954; Siegelmann and Sontag, 1992; Hao et al., 2018; Merrill, 2019; Merrill et al., 2020, 2022; Merrill and Tsilivis, 2022). Recent work has extended this paradigm to work with probabilistic models of computation (Svete and Cotterell, 2023; Nowak et al., 2023), but so far only for LMs based on recurrent neural networks.

However, the sequential nature of classical models makes the connection to the inherently *parallelizable* transformer architecture less straightforward and has resulted in a number of results upper-bounding their representational capacity (Hahn, 2020; Bhattamishra et al., 2020; Chiang and Cholak, 2022; Hao et al., 2022a; Merrill and Sabharwal, 2023c). We connect transformer LMs to a classical class of LMs that lend themselves particularly well to parallelized computations: n-gram LMs. We show that both hard as well as sparse attention transformer LMs can represent any n-gram LM (Theorems 3.1 and 4.1).[1] This gives

[1] An analysis completely analogous to practical implementations would also consider *soft* attention transformer LMs, whose full support when attending over the preceding symbols makes the analysis trickier. We, therefore, omit its analysis

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 6845–6881
June 16-21, 2024 ©2024 Association for Computational Linguistics

us a concrete lower bound on their probabilistic representational capacity. We also study the role of the number of heads (Theorem 3.1) and the number of layers (Theorem 3.2), illustrating a trade-off between the number of heads, layers, and the complexity of the non-linear transformations required for the simulation of n-gram LMs. Altogether, these results offer a step towards understanding the probabilistic representational capacity of transformer LMs and the mechanisms they might employ to implement formal models of computation.

2 Preliminaries

Let Σ be an alphabet, i.e., a finite, non-empty set of symbols, and Σ^* the (infinite) set of all strings formed from symbols of Σ. Most modern LMs define $p(\boldsymbol{y})$ for $\boldsymbol{y} \in \Sigma^*$ autoregressively—as a product of conditional probability distributions:

$$p(\boldsymbol{y}) \stackrel{\text{def}}{=} p(\textsc{eos} \mid \boldsymbol{y}) \prod_{t=1}^{|\boldsymbol{y}|} p(y_t \mid \boldsymbol{y}_{<t}). \quad (1)$$

Here, $\textsc{eos} \notin \Sigma$ is a distinguished end-of-string symbol. The $\textsc{eos}$ symbol enables us to define the probability of a string purely based on the conditional distributions. Such a factorization can be done without loss of generality (Du et al., 2023). We define $\overline{\Sigma} \stackrel{\text{def}}{=} \Sigma \cup \{\textsc{eos}\}$. Further, the conditional probability distributions $p(y_t \mid \boldsymbol{y}_{<t})$ are usually defined based on *vectorial* representations of $\boldsymbol{y}_{<t}$ computed by some function $\mathsf{enc}\colon \Sigma^* \to \mathbb{R}^D$. This leads us to the definition of representation-based LMs below.

Definition 2.1. *Let Σ be an alphabet and $\mathsf{enc}\colon \Sigma^* \to \mathbb{R}^D$ a **representation function** encoding strings as D-dimensional representations. Let $\mathbf{E} \in \mathbb{R}^{|\overline{\Sigma}| \times D}$ be an **output matrix**. A **representation-based** LM p defines the conditional probability distributions $p(y_t \mid \boldsymbol{y}_{<t})$ as*[2]

$$p(y_t \mid \boldsymbol{y}_{<t}) \stackrel{\text{def}}{=} \mathrm{softmax}(\mathbf{E}\, \mathsf{enc}(\boldsymbol{y}_{<t}))_{y_t}. \quad (2)$$

At a high level, we are interested in encoding an arbitrary n-gram LM using a transformer LM. To do so, we need a notion of equivalence between language models. In this paper, we will work with the following simple definition.

Definition 2.2. *Two LMs p and q over Σ^* are **weakly equivalent** if $p(\boldsymbol{y}) = q(\boldsymbol{y})$ for all $\boldsymbol{y} \in \Sigma^*$.*

This paper precisely explains and proves the following theorem, stated informally below.

Theorem 2.1 (Informal)**.** *For every n-gram LM, there exists a weakly equivalent ({hard, sparse} attention) transformer LM.*

2.1 An Aside about Boolean Recognition

Fundamentally, Theorem 2.1 is about weak equivalence (Definition 2.2) between two LMs. In this subsection, we make our case against treating LMs as *recognizers*. The most common manner of analyzing a language model as a recognizer is based on using its representations as an input to a classifier (Merrill, 2019; Merrill et al., 2020). We recapitulate a common definition below.

Definition 2.3. *Let p be a representation-based LM with the representation function $\mathsf{enc}\colon \Sigma^* \to \mathbb{R}^D$ and let $g\colon \mathbb{R}^D \to \{0,1\}$ be a classifier. The **binary language** of p with g is defined as*

$$\mathcal{L}_g(p) \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \Sigma^* \mid g(\mathsf{enc}(\boldsymbol{y})) = 1\}. \quad (3)$$

Related is the notion of truncated recognition.

Definition 2.4 (Hewitt et al. (2020), Definition 4)**.** *Let p be a language model over Σ^* and $\alpha > 0$. The α-**truncated language** of p is defined as*

$$\mathcal{L}_\alpha(p) \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \Sigma^* \mid p(\textsc{eos} \mid \boldsymbol{y}) \geqslant \alpha \quad (4)$$
$$\text{and } p(y_t \mid \boldsymbol{y}_{<t}) \geqslant \alpha \quad \forall t \in [|\boldsymbol{y}|]\}.$$

There are many results in the literature treating transformers' ability to recognize languages in the sense of the two definitions above. For instance, transformers are unable to recognize the Dyck language with more than one bracket type and the PARITY language in the sense of Definition 2.3 (Hahn, 2020), but *can* recognize *bounded* Dyck languages in the sense of Definition 2.4 (Yao et al., 2021). Our indictment of analyzing $\mathcal{L}_g(p)$ and $\mathcal{L}_\alpha(p)$ is that proceeding in such a manner disregards the probabilities assigned to strings by p, which we view as essential to language modeling. Moreover, Definition 2.3 depends on the form of the classifier g while Definition 2.4 depends on the hyperparameter α. For example, positively classified strings from a language could have their conditional probabilities only slightly above the classification threshold and the negatively classified ones only slightly below the threshold (Hahn, 2020), which hides the true distribution defined by the

here and reserve it for a separate treatment.

[2] One could, more generally, swap the softmax for any other normalization function, such as the sparsemax (Martins and Astudillo, 2016). Here, however, we focus on the softmax for conciseness.

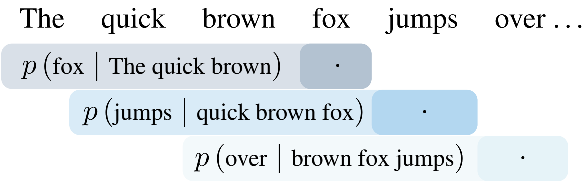
The quick brown fox jumps over . . .
p (fox | The quick brown) ·
p (jumps | quick brown fox) ·
p (over | brown fox jumps) ·

Figure 2: An illustration of how a 4-gram LM computes the probability of a string. All conditional probabilities can be computed in parallel and then multiplied into the probability of the entire string.

LM. In this context, our contention is that $\mathcal{L}_g(p)$ and $\mathcal{L}_\alpha(p)$ are not useful definitions for studying the representational capacity of LMs. Instead, we advocate for analyzing LMs as *probabilistic* formal languages.

2.2 Language Modeling with n-grams

The next-symbol probabilities in n-gram LMs are computed under the n-gram assumption.

Assumption 2.1. *The **n-gram assumption** states that the conditional probability of the symbol y_t given $\boldsymbol{y}_{<t}$ only depends on $n-1$ previous symbols $\boldsymbol{y}_{t-n+1}^{t-1} \overset{\text{def}}{=} y_{t-1}, \ldots, y_{t-n+1}$:*

$$p\left(y_t \mid \boldsymbol{y}_{<t}\right) = p\left(y_t \mid \boldsymbol{y}_{t-n+1}^{t-1}\right). \tag{5}$$

*We will refer to $\boldsymbol{y}_{t-n+1}^{t-1}$ as the **history** of y_t.*

Padding. Eq. (5) assumes the existence of $n-1$ preceding symbols that define the conditional distribution $p\left(y_t \mid \boldsymbol{y}_{<t}\right)$. To ensure this is the case even at the beginning of the string, it is standard to *pad* the input string with $n-1$ beginning-of-string symbols BOS. For ease of notation, we index the $n-1$ BOS tokens with indices $-n+2, \ldots, 0$ so that n-gram LMs conveniently fit the autoregressive factorization from Eq. (1). We also define $\underline{\Sigma} \overset{\text{def}}{=} \Sigma \cup \{\text{BOS}\}$.

Despite their simplicity, n-gram LMs have a storied place in language modeling (Shannon, 1948; Baker, 1975a,b; Jelinek, 1976; Bahl et al., 1983; Jelinek, 1990; Bengio et al., 2000, 2003, 2006; Schwenk, 2007; Heafield, 2011; Heafield et al., 2013). Because the conditional probabilities of n-gram LMs only depend on the previous $n-1$ symbols, different parts of the string can be processed independently, i.e., in parallel. This facilitates a natural connection to transformer LMs since parallelizability is a prevalent feature of the architecture and one of its main advantages over other neural LMs such as RNN LMs (Vaswani et al., 2017).

2.3 Transformer Language Models

Transformer LMs are LMs whose conditional distributions $p\left(y_t \mid \boldsymbol{y}_{<t}\right)$ are computed by a **transformer**. A transformer is a composition of multiple transformer **layers**, each of which implements the **attention mechanism**. We give definitions of these building blocks in what follows.

Notation. We use bold, unitalicized letters such as $\mathbf{x} \in \mathbb{R}^D$ to denote real-valued vectors and italicized letters $x_j \in \mathbb{R}$ for their entries. Capital bold letters such as $\mathbf{X} \in \mathbb{R}^{N \times D}$ denote matrices. All vectors are *column* vectors unless transposed. We define the vertically stacking operator $(\cdot\,; \cdots; \,\cdot)$, which denotes the vertical concatenation of the D-dimensional *column* vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ into a ND-dimensional vector $(\mathbf{x}_1; \cdots; \mathbf{x}_N) \in \mathbb{R}^{ND}$ and the concatenation of the D-dimensional *row* vectors $\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top$ into a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ with N rows and D columns. Given the matrix $\mathbf{X} = \left(\mathbf{x}_1^\top; \cdots; \mathbf{x}_N^\top\right)$, we write $\mathbf{X}_n = \left(\mathbf{x}_1^\top; \cdots; \mathbf{x}_n^\top\right)$ for the submatrix composed of the first n rows. We call a function $f\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ whose purpose is to evaluate the compatibility of two vectors a **scoring function**. A **normalization function** $\boldsymbol{\pi}\colon \mathbb{R}^N \to \boldsymbol{\Delta}^{N-1}$ maps vectors in $\mathbb{R}^N$ to N probabilities. Here, $\boldsymbol{\Delta}^{N-1} \overset{\text{def}}{=} \left\{\mathbf{x} \in [0,1]^N \mid \sum_{n=1}^N x_n = 1\right\}$ is the $(N-1)$-dimensional probability simplex. This notation is summarized in Tab. 1.

The Attention Mechanism. The attention mechanism works as follows. It takes a **q**uery vector $\mathbf{q} \in \mathbb{R}^D$ and two matrices: The matrix $\mathbf{K} \in \mathbb{R}^{N \times D}$ of **k**eys and the matrix $\mathbf{V} \in \mathbb{R}^{N \times D}$ of **v**alues and computes a weighted average of the value vectors based on the compatibilities of the key vectors and the query vector, as determined by a scoring function f. A formal definition is given below.

Definition 2.5 (Attention Mechanism)**.** *Let f be a scoring function and $\boldsymbol{\pi}$ a normalization function. Let $\mathbf{q} \in \mathbb{R}^D$ be a query vector and let $\mathbf{K} = \left(\mathbf{k}_1^\top; \cdots; \mathbf{k}_N^\top\right) \in \mathbb{R}^{N \times D}$ and $\mathbf{V} = \left(\mathbf{v}_1^\top; \cdots; \mathbf{v}_N^\top\right) \in \mathbb{R}^{N \times D}$ be matrices of keys and values, respectively. An **attention mechanism** $\texttt{Att}\colon \mathbb{R}^D \times \mathbb{R}^{N \times D} \times \mathbb{R}^{N \times D} \to \mathbb{R}^D$ is defined as*

$$\texttt{Att}\left(\mathbf{q}, \mathbf{K}, \mathbf{V}\right) \overset{\text{def}}{=} \sum_{n=1}^{N} s_n \mathbf{v}_n, \tag{6}$$

where

$$\mathbf{s} \overset{\text{def}}{=} \boldsymbol{\pi}\left(f\left(\mathbf{q}, \mathbf{k}_1\right), \ldots, f\left(\mathbf{q}, \mathbf{k}_N\right)\right) \tag{7}$$

Symbol	Type	Meaning
$[N]$	$\subset \mathbb{N}$	The set $\{1, \ldots, N\}$ for $N \in \mathbb{N}$.
y	$\in \Sigma$	A symbol, element of Σ.
$\Sigma, \underline{\Sigma}, \overline{\Sigma}$	alphabet	Σ is a set of symbols, $\underline{\Sigma} \overset{\text{def}}{=} \Sigma \cup \{\text{BOS}\}$, $\overline{\Sigma} \overset{\text{def}}{=} \Sigma \cup \{\text{EOS}\}$
$\boldsymbol{y}$	$\in \Sigma^*$	A string over Σ.
$\boldsymbol{y}_j^i$	$\in \Sigma^*$	A substring of $\boldsymbol{y}$, a string.
$[\![y]\!]$	$\in \{0,1\}^{\lvert\Sigma\rvert}$	One-hot encoding of the symbol $y \in \Sigma$.
D	$\in \mathbb{N}$	Size of the contextual representations in the transformer.
$\boldsymbol{\Delta}^{N-1}$	$\subseteq \mathbb{R}^N$	The $N-1$-dimensional probability simplex.
f	$\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$	A scoring function.
$\boldsymbol{\pi}$	$\mathbb{R}^N \to \boldsymbol{\Delta}^{N-1}$	A normalization function.
Q, K, V, O	$\mathbb{R}^D \to \mathbb{R}^D$	The query, key, value, and output functions.
F	$\mathbb{R}^D \to \mathbb{R}^D$	The final transformer LM transformation function.
enc	$\Sigma^* \to \mathbb{R}^D$	The string representation function.
$\mathbf{r}$	$\underline{\Sigma} \times \mathbb{N} \to \mathbb{R}^D$	The position-augmented representation function.
L	$\in \mathbb{N}$	Number of layers.
H	$\in \mathbb{N}$	Number of heads.
$\mathcal{H}$	$\mathbb{R}^{HD} \to \mathbb{R}^D$	The head combining function.
$(\cdot\,; \cdots ; \cdot)$		Vertical concatenation operator of vectors or matrices.

Table 1: A summary of the notation used in the paper.

is the vector of normalized scores between the query $\mathbf{q}$ *and the keys in* $\mathbf{K}$.

The most standard implementation of the scoring function f is the (scaled) inner product $f(\mathbf{q}, \mathbf{k}) \overset{\text{def}}{=} \langle \mathbf{q}, \mathbf{k} \rangle$. Some of our results rely on this standard formulation. However, some also rely on the more general, but still simple and just as efficiently computable scoring functions.

The Transformer Architecture. A transformer layer uses the attention mechanism to compute augmented representations $\mathbf{z}_t = \mathtt{Att}(\mathbf{q}_t, \mathbf{K}_t, \mathbf{V}_t)$ of the input representations $\mathbf{X}_t = (\mathbf{x}_1; \cdots ; \mathbf{x}_t)$. The query $\mathbf{q}_t$, the keys $\mathbf{K}_t$, and values $\mathbf{V}_t$ are all transformations of the input representations $\mathbf{X}_t$.

Definition 2.6. *Let $Q, K, V, O \colon \mathbb{R}^D \to \mathbb{R}^D$ be the query, key, value, and* ***output*** *functions. A* ***transformer layer*** *is a function $\mathcal{L} \colon \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$ that computes*

$$\mathcal{L}\left(\mathbf{x}_1^\top; \ldots ; \mathbf{x}_T^\top\right) = \left(\mathbf{z}_1^\top; \ldots ; \mathbf{z}_T^\top\right) \in \mathbb{R}^{T \times D} \quad (8)$$

for $t \in [T]$ where

$$\mathbf{a}_t \overset{\text{def}}{=} \mathtt{Att}(\mathbf{q}_t, \mathbf{K}_t, \mathbf{V}_t) + \mathbf{x}_t \in \mathbb{R}^D \quad (9a)$$

$$\mathbf{z}_t \overset{\text{def}}{=} O(\mathbf{a}_t) + \mathbf{a}_t \in \mathbb{R}^D. \quad (9b)$$

Here, we define

$$\mathbf{q}_t \overset{\text{def}}{=} Q(\mathbf{x}_t) \in \mathbb{R}^D \quad (10a)$$

$$\mathbf{K}_t \overset{\text{def}}{=} \left(K(\mathbf{x}_1)^\top; \cdots ; K(\mathbf{x}_t)^\top\right) \in \mathbb{R}^{t \times D} \quad (10b)$$

$$\mathbf{V}_t \overset{\text{def}}{=} \left(V(\mathbf{x}_1)^\top; \cdots ; K(\mathbf{x}_t)^\top\right) \in \mathbb{R}^{t \times D}. \quad (10c)$$

Note: *For simplicity, we do not include layer normalization.*

Without further modification, the transformations applied by the transformer layer are position-invariant, which necessitates the addition of explicit positional information.

Definition 2.7. *A position-augmented symbol* ***representation function*** $\mathbf{r} \colon \Sigma \times \mathbb{N} \to \mathbb{R}^D$ *is a function representing symbols and their positions as D-dimensional vectors.*

Position-augmented symbol representation functions are often implemented as an addition or concatenation of separate symbol-only and position-only representation functions (Vaswani et al., 2017). Here, we define it more generally as any function of the symbol and its position.

Definition 2.8. *A* ***static encoding*** $\mathcal{R}$ *is a function $\mathcal{R} \colon \Sigma^T \to \mathbb{R}^{T \times D}$ defined for any $T \in \mathbb{N}$ as*

$$\mathcal{R}(\boldsymbol{y}) \overset{\text{def}}{=} \left(\mathbf{r}(y_1, 1)^\top; \cdots ; \mathbf{r}(y_T, T)^\top\right). \quad (11)$$

Multiple transformer layers are stacked into a transformer, which computes the (deep) contextual representations of all symbols in the string.

Definition 2.9. *For $L \in \mathbb{N}$, let $\mathcal{L}_\ell$ for $\ell \in [L]$ be transformer layers. Let $\mathcal{R}$ be a static encoding. An L-layer* ***transformer*** $\mathcal{T}$ *is defined as*

$$\mathcal{T}(\mathcal{R}) \overset{\text{def}}{=} \mathcal{L}_L \circ \cdots \circ \mathcal{L}_1 \circ \mathcal{R}. \quad (12)$$

A transformer computes the contextual representations of the symbols $\boldsymbol{y} = y_1 \cdots y_T$ as

$$\left(\mathbf{x}_1^{L\top}; \cdots ; \mathbf{x}_T^{L\top}\right) \overset{\text{def}}{=} \mathcal{T}(\mathcal{R})(\boldsymbol{y}). \quad (13)$$

If $\mathcal{R}$ is clear from the context or arbitrary, we will omit it as an argument to $\mathcal{T}$ and just write $\mathcal{T}(\boldsymbol{y})$.

Definition 2.10. *Let $\mathcal{T}$ be a transformer, $F\colon \mathbb{R}^D \to \mathbb{R}^D$ the final representation transformation function, and $\bar{\boldsymbol{y}} \in \Sigma^*$ with $|\boldsymbol{y}| = T$. We define*

$$\mathsf{enc}(\boldsymbol{y}) \overset{\text{def}}{=} F\left(\mathbf{x}_T^L\right), \tag{14}$$

where $\mathbf{x}_T^L$ is the representation of the T^{th} symbol in $\boldsymbol{y}$ computed by $\mathcal{T}$, i.e., $\left(\mathbf{x}_1^{L\top}; \cdots; \mathbf{x}_T^{L\top}\right) = \mathcal{T}(\boldsymbol{y})$.

Transformer Language Models. So far, we have only defined how the transformer architecture can be used to compute the contextual representations of the symbols. To complete the definition, we define a transformer *language model* as follows.

Definition 2.11. *A **transformer LM** $p_{\mathcal{T}}$ is the representation-based autoregressive LM with the representation function* enc *from Eq. (14). That is, $p_{\mathcal{T}}$ defines the conditional probability distributions*

$$p_{\mathcal{T}}(y_t \mid \boldsymbol{y}_{<t}) \overset{\text{def}}{=} \mathrm{softmax}(\mathbf{E}\,\mathsf{enc}(\boldsymbol{y}_{<t}))_{y_t}. \tag{15}$$

2.3.1 Variants of the Attention Mechanism

In this subsection, we discuss many common variants of the attention mechanism. First, **multi-headed** attention uses H **attention heads** to compute H representations of the symbols in the string. The representations constructed by the different attention heads are concatenated into a long vector and projected down to the output size of a single head with a head-combiner function $\mathcal{H}$.

Definition 2.12. *For $L, H \in \mathbb{N}$, let $\mathcal{L}_\ell^h\colon \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D}, \ell \in [L], h \in [H]$ be transformer layers. Define $\mathcal{L}_\ell\colon \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times HD}$ as*

$$\mathcal{L}_\ell(\mathbf{X}) \overset{\text{def}}{=} \left(\mathcal{L}_\ell^1(\mathbf{X})^\top; \cdots; \mathcal{L}_\ell^H(\mathbf{X})^\top\right)^\top. \tag{16}$$

Furthermore, let $\mathcal{H}\colon \mathbb{R}^{HD} \to \mathbb{R}^D$. An L-layer transformer with H heads computes:

$$\mathcal{T}(\mathcal{R}) \overset{\text{def}}{=} \mathcal{L}_L \circ \mathcal{H} \circ \cdots \circ \mathcal{H} \circ \mathcal{L}_1 \circ \mathcal{R}, \tag{17}$$

where $\mathcal{H}$ is applied row-wise *to project the representations of H heads to $\mathbb{R}^D$.*

Attention types. Attention weights are computed by normalizing the scores $f(\mathbf{q}, \mathbf{k}_1), \ldots, f(\mathbf{q}, \mathbf{k}_t)$. The choice of the projection function $\boldsymbol{\pi}$ determines the type of attention and has concrete implications on representational capacity (Hao et al., 2022a).

Definition 2.13. ***Hard attention** is computed with the* hardmax *projection function:*

$$\mathrm{hardmax}(\mathbf{x})_d \overset{\text{def}}{=} \begin{cases} \frac{1}{m} & \textbf{\textit{if}}\ d \in \mathrm{argmax}(\mathbf{x}) \\ 0 & \textbf{\textit{otherwise}} \end{cases} \tag{18}$$

for $d \in [D]$, where $\mathbf{x} \in \mathbb{R}^D$ and $m \overset{\text{def}}{=} |\mathrm{argmax}(\mathbf{x})|$ is the cardinality of the argmax set.

We also introduce *sparse* attention, which uses the sparsemax normalization function to compute the attention weights.

Definition 2.14. ***Sparse attention** is computed with the* sparsemax *projection function:*

$$\mathrm{sparsemax}(\mathbf{x}) \overset{\text{def}}{=} \underset{\mathbf{p}\in\boldsymbol{\Delta}^{D-1}}{\mathrm{argmin}} \|\mathbf{p} - \mathbf{x}\|_2^2. \tag{19}$$

3 Hard Attention Transformer LMs

This section presents a set of results describing the representational capacity of hard attention transformer LMs. Concretely, we show that transformer LMs with hard attention can represent n-gram LMs, either using $n-1$ heads (Theorem 3.1) or $n-1$ layers (Theorem 3.2). Simulation is possible even with a single head and a single layer (Theorem 3.3) but might require a more elaborate set of non-linear transformations and positional encodings whose precision scales linearly with the string length.

Theorem 3.1. *For any n-gram LM, there exists a weakly equivalent single-layer hard attention transformer LM with $n-1$ heads.*

Proof intuition. Given an n-gram LM p, we can construct a weakly equivalent LM $p_{\mathcal{T}}$ defined by a transformer $\mathcal{T}$ that looks back at the preceding $n-1$ positions using $n-1$ heads, each of them uniquely attending to exactly one position. The symbols attended to can be used to identify the full history, which can be used to access the conditional distribution over the next symbol. This is illustrated in Fig. 1. See Appendix B.2 for the full proof. ■

Theorem 3.1 shows that transformer LMs with hard attention can represent n-gram LMs, establishing, to the best of our knowledge, the first concrete relationship between transformer LMs and probabilistic languages. A natural follow-up question then is whether $n-1$ heads are *necessary* to correctly simulate an n-gram LM. Besides aiming to illuminate different mechanisms enabling the implementation of classical LMs, this question also follows the line of inquiry about the *uniqueness* and *interpretability* of the representations of formal models by neural LMs (Liu et al., 2023). The following two theorems show that the intuitive construction using $n-1$ heads is far from unique: Theorem 3.2 shows that a similarly simple simulation is possible with $n-1$ layers and a single head, while Theorem 3.3 shows that even a transformer LM with a single head and a single layer can simulate an n-gram LM, albeit with more complex position invariant transformation F. This suggests that there is no canonical way of determining whether a transformer LM has learned an n-gram LM by looking at individual components (e.g., positions attended to by the different heads).

Theorem 3.2. *For any n-gram LM, there exists a weakly equivalent $(n-1)$-layer hard attention transformer LM with a single head.*

Proof intuition. Whereas the transformer LM constructed in Theorem 3.1 used $n-1$ heads to look at all the $n-1$ positions of interest, an $n-1$-layer transformer LM can use the $n-1$ layers to look back at the *immediately* preceding position and copy it forward $n-1$ times (keeping the current symbol there as well). After $n-1$ layers of such transformations, the entire history can be read from the current contextual representation. See Appendix B.2 for the full proof. ■

Apart from using hard attention, both transformer LMs used in Theorems 3.1 and 3.2 rely on modeling assumptions often found in practical implementations of the transformer: The transformations Q, K, and V are linear functions, the scoring function is implemented as a dot-product and positional encodings are bounded. This makes the results comparable to practical implementations. The following theorem, in contrast, shows that if we permit the use of less standard components, transformer LMs can identify the history of interest using only a single head and a single layer.

Theorem 3.3. *For any n-gram LM, there exists a weakly equivalent single-layer hard attention transformer LM with a single head.*

Proof intuition. The bulk of this construction lies in the encoding $\boldsymbol{y}_{t-n+1}^{t-1}$ in a vector that can be constructed by a single attention head in one layer. This is done by an attention head that *(1)* puts non-zero attention on only the previous $n-1$ symbols and *(2)* encodes the identities and the positions of symbols in a $|\Sigma|$-dimensional value vector. The value vector can then be decoded into a one-hot encoding of $\boldsymbol{y}_{t-n+1}^{t-1}$ by an $n-1$-layer MLP that defines F, which allows us to match the conditional probabilities of the n-gram LM as in Theorems 3.1 and 3.2. See Appendix B.2 for the full proof. ■

4 Sparse Attention Transformer LMs

While the results in §3 concretely characterize the abilities of hard attention transformer LMs, the assumption of hard attention is somewhat removed from practical implementations of the model. Those most often rely on differentiable normalization functions, such as the softmax.[3] However, the full support of the softmax function makes the connection to formal models of computation difficult (Hahn, 2020). To bring the theoretical models closer to practical implementations yet still be able to make clear analogies to formal models of computation, we now consider *sparse* attention transformers, which use the sparsemax normalization function. The sparsity allows sparse attention transformers to simulate n-gram LMs just like hard attention transformers while relying on differentiable operations.

Theorem 4.1. *For any n-gram LM, there exists a weakly equivalent single-layer sparse attention transformer LM with $n-1$ heads.*

Proof intuition. The intuition behind the simulation with sparse attention is similar to the hard attention one; each head attends to a single position, as illustrated in Fig. 1. Effectively, the construction results in a sparse attention transformer that simulates hard attention. In contrast to Theorems 3.1 and 3.2, we here require a model with *unbounded* positional encodings and a non-linearly transformed dot-product scoring function. Intuitively, the unbounded positional encodings are required to scale the unnormalized attention scores to differ enough for the sparsemax to focus on a single position. The rest of the proof follows that of Theorem 3.1; see Appendix C for the details. ■

Theorem 4.1 (representing an n-gram LM with $n-1$ heads) could naturally be extended to analogs

[3] As noted in §1, the analysis of soft attention transformers requires a different type of analysis in terms of approximation of the probabilities. A complete study would have to consider the approximation over *arbitrarily* long strings (since Σ^* is an infinite set), which is difficult by simply scaling model parameters to a large constant. We focus on exact simulation here, but conjecture that soft attention transformers can approximate LMs whose *average* string length is finite.

of Theorem 3.2 (representing an n-gram LM with $n-1$ layers) and Theorem 3.3 (representing an n-gram LM with a single head and a single layer) using a similar adaptation of the construction from the hard attention case to the sparse attention one as in Theorem 4.1.

5 Space Complexity

In §3 and §4, we describe lower bounds that tell us what types of probability distributions transformer LMs *can* represent, but do not say how *efficiently* they can do so. The space complexity of simulating n-gram LMs is discussed in this section. We focus on hard-attention transformer LMs with multiple heads or multiple layers (Theorems 3.1 and 3.2) since their modeling assumptions (bar hard attention) are closest to practical implementations. The constructive proofs of Theorems 3.1 and 3.2 allow us to directly analyze the space requirements for the simulation of n-gram LMs, both in terms of *(1)* the size of the contextual representations $\mathbf{X}_{\ell}^{h}$ as well as *(2)* the number of bits required to represent the individual entries of the vectors $\mathbf{x}_{\ell,t}^{h}$.

5.1 Scaling with Respect to the Number of Computational Steps

We first address the second point. Specifically, we are interested in how the number of bits scales with respect to t, the number of computational steps performed during the generation of a string $\boldsymbol{y} \in \Sigma^*$. As summarized by Tab. 2, the models constructed in the proofs of Theorems 3.1 and 3.2 use positional encodings with entries of the form $\sqrt{\frac{1}{t}}$ for $t \in \mathbb{N}$, i.e., they contain square roots of rational numbers. This makes the scaling of the space complexity difficult, as square roots of rational numbers are not in general representable with a finite number of bits. While this might seem discouraging, we emphasize that these specific positional encodings were only used to keep the contextual representations bounded and the scoring function in line with the original formulation (Vaswani et al., 2017) and concurrent work (Merrill and Sabharwal, 2023a). A closer look at the constructions in the proof of Theorems 3.1, 3.2 and 4.1 reveals that simpler (but unbounded) positional encodings with a less standard scoring function can be used to the same effect. In particular, we can use positional encodings that contain entries of the form t, which only require a *logarithmic* number of bits with respect to t. Since such scaling is required to uniquely encode the positional information in general (Merrill and Sabharwal, 2023c), this represents an asymptotically optimal scaling of the space complexity of the contextual representations.[4]

Importantly, n-gram LMs are *real-time*: they, by definition, generate a symbol at each step of the computation. This means that the scaling of the space complexity with respect to the number of computation steps coincides with its scaling with respect to the length of the generated string $\boldsymbol{y}$—the scaling is logarithmic in $|\boldsymbol{y}|$. This is in contrast to non-real-time models of computation which might not generate a symbol at each step of the computation; while those might still require an asymptotically optimal scaling with respect to t, the additional computational steps that do not emit any symbol might mean that the space complexity is *unbounded* with respect to the length of the generated string. An example of such a model is a transformer LM simulating a (probabilistic) Turing machine, which would require the model to not emit symbols at some points of the computation (Nowak et al., 2023, 2024).

5.2 The Dimensionality of the Contextual Representations

We now discuss the size of the contextual representations required for the simulation of n-gram LMs. From a high level, we have to consider two stages: *(1)* the contextual representations $\mathbf{x}_{\ell,t}^{h}$ of the different layers and heads and *(2)* the size of the final representation $\mathsf{enc}(\boldsymbol{y})$. The contextual representations $\mathbf{x}_{\ell,t}^{h}$ in stage *(1)* are composed of the symbol and positional encodings. The symbol representations include (two copies of) the one-hot encodings while the positional encodings include between two and $2n$ dimensions encoding positional information. This means that the per-head and per-layer space complexity scales with $|\Sigma|$ and n. For stage *(2)* we use the one-hot encodings of the entire *history* $\boldsymbol{y}_{t-n+1}^{t-1}$, with which we index the matrix of $|\underline{\Sigma}|^{n-1}$ conditional probabilities defined by the n-gram LM. While the size of $\mathsf{enc}(\boldsymbol{y})$ is theoretically only lower-bounded by $|\overline{\Sigma}|$ (Yang et al., 2018; Svete and Cotterell, 2023),[5] reducing its size requires a lower-

[4] The construction in Theorem 3.3, in contrast, relies on encoding the entire preceding string in a single dimension with one digit per position in the string, which requires a number of bits that scales linearly with respect to the string length. A simplification to a logarithmic number of bits does not seem as straightforward.

[5] This is because the (logits of the) conditional probabilities can span a $|\overline{\Sigma}|$-dimensional space.

rank decomposition of the matrix of conditional probabilities and a corresponding reparametrization of the contextual symbol representation. This might require a blow-up in the number of bits required to represent individual dimensions, or, in general, result in a real-valued vector that could not be represented on a finite-precision system. Altogether, most of the space complexity of the contextual representations comes from the one-hot encodings in $\mathsf{enc}(\boldsymbol{y})$, which require $|\Sigma|^{n-1}$ dimensions.

The discussion in this section can be summarized by the following theorem.

Theorem 5.1. *Let p be an n-gram LM over the alphabet Σ. There exists a weakly equivalent hard-attention transformer LM with contextual representations $\mathbf{x}$ of size $\mathcal{O}(n|\Sigma|)$ and representation $\mathsf{enc}(\boldsymbol{y})$ of size $\mathcal{O}\left(|\Sigma|^{n-1}\right)$. Each of $\mathbf{x}$'s entries can be represented with $\mathcal{O}(\log_2(|\boldsymbol{y}|))$ bits for $\boldsymbol{y} \in \Sigma^*$.*

6 Discussion and Related Work

To the best of our knowledge, §3 and §4 provide the first results on the probabilistic representational capacity of transformer LMs.

The relevance of n-gram LMs to modern LMs. One might rightfully question the utility of connecting the state-of-the-art language modeling architecture to n-gram LMs. LMs based on the n-gram assumption indeed constitute some of the simplest and least expressive classes of probability distributions. Nevertheless, n-gram LMs provide a useful playground and theoretical foundation for contextualizing and interpreting the inner workings of modern LMs. For example, n-gram LMs have been found to comprise a crucial component of *in-context learning*, where attention heads in different layers of the model together identify individual n-grams and base their predictions on the presence of such n-grams (Olsson et al., 2022; Akyürek et al., 2024). The presence of specific n-grams might therefore present the foundations of in-context-based knowledge of transformer LMs. As such, understanding the requirements for correct simulation of n-gram LMs is important for a thorough grasp of the abilities of LMs to learn in context. Encouraging n-gram-LM-based learning has also been observed to aid in-context learning abilities (Akyürek et al., 2024). Existing work has also linked n-gram LMs to other neural network architectures such as one-dimensional convolutional neural networks (Merrill, 2019). Moreover, the theoretical grounding in classical formal language theory makes precise statements about n-gram LMs possible while their simplicity makes them inherently interpretable and easy to analyze. n-gram LMs also have a cognitive interpretation (Jäger and Rogers, 2012). Most importantly, however, n-gram LMs lend themselves to *paralellized* processing, which affords a succinct and natural connection to transformer LMs and has been suggested to be a crucial part of any theoretical treatment of transformer LMs (Strobl et al., 2023; Merrill and Sabharwal, 2023c).

Understanding the limitations and abilities of hard attention transformer LMs. Our results are the strongest in the hard attention setting, which follows the trend of using hard attention in theoretical treatments. Hard attention makes singling out the important aspects of the string (in our case, the history) possible. Concretely, we showcase three different mechanisms that make it possible for transformer LMs to implement n-gram LMs with hard attention and investigate the role of the number of heads and layers. Particularly, our constructions suggest a possible mechanism in which the different transformer heads or layers can specialize in focusing on different positions in the string, a feature that has previously been suggested as an explanation of how transformer LMs process strings (Elhage et al., 2021) and has been observed in trained transformer LMs (Olsson et al., 2022; Akyürek et al., 2024).[6] Our presentation of multiple orthogonal mechanisms that can simulate n-gram LMs equivalently well is another confirmation of the observation that algorithmic principles learned or implemented by neural LMs do not always correspond to a *single* intuitive implementation of a formal model of computation, which has concrete implications on interpretability methods for the architecture. This has been observed in practical scenarios and warns us that focusing on individual components of the model (that is, using myopic interpretability methods) might result in misleading interpretability results (Wen et al., 2023).

Probabilistic representational capacity. We augment existing literature by providing an explicit connection between transformer LMs and n-gram

[6] Note that the constructions presented in this paper are purely meant to showcase the existence of a mechanism that can be used to simulate n-gram LMs; we do not suggest that the same mechanisms will be employed by models used in practice, which is also why do not present any empirical results. For example, the use of the sparse one-hot encodings differs from the standard dense representations of symbols.

language models. In line with work comparing transformers to circuits (Merrill and Sabharwal, 2023c,a; Weiss et al., 2021; Hao et al., 2022b; Merrill and Sabharwal, 2023b; Merrill et al., 2022; Chiang et al., 2023; Angluin et al., 2023), we also show the utility of analyzing transformers with parallelizable models of computation, which go hand in hand with the parallelizable nature of the transformer architecture. Moreover, the formalization of an LM with the output matrix indexed by the contextual representation (cf. Definition 2.11) makes it easy to connect existing results on the expressivity of the transformer architecture with the probabilistic setting by defining a mapping from the contextual representation to the conditional probability distribution through the output matrix. We suppose other simple and parallelizable classes of distributions might lend themselves well to similar probabilistic treatments; probabilistic analysis may be particularly interesting in the context of circuit complexity with sigmoid-activated circuits (Maass et al., 1991), (Smolensky et al., 1996, Chapter 4).

Connection to (sub-)regular languages This work focuses on connecting transformer LMs to n-gram LMs, which are a special instance of the more general class of *sub-regular LMs*. In words, sub-regular LMs are LMs that can be described without using the full power of the probabilistic finite-state automata (Jäger and Rogers, 2012). In this sense, sub-regular LMs fall below regular languages in the Chomsky hierarchy and themselves define their own hierarchy (Jäger and Rogers, 2012; Heinz and Rogers, 2013; Avcu et al., 2017). For example, some classes of sub-regular languages do not require sequential processing of the input string that is usually required by finite-state automata for correct recognition. The intuitive connection between transformer LMs and n-gram LMs encourages further work on the connections between other classes of (sub-)regular LMs and transformer LMs. Transformers have been linked to (sub)-regular languages by Yao et al. (2021) and Liu et al. (2023). Yao et al. (2021) study the ability of transformers to generate bounded hierarchical languages using a transformer but do not extend their analysis to the fully probabilistic case.[7] Liu et al. (2023) connect transformers with both hard and soft attention to general finite-state automata (FSAs), which define a strictly larger set of languages than n-gram languages. This additional generality, however, comes with some caveats. Liu et al. (2023, Theorem 1)—their most general result—relies on a model whose depth *scales* (logarithmically) with the string length. As such, no particular finite-size transformer construction can simulate FSAs on strings of *arbitrary* length (which is required for equivalence). This is in contrast to our results, which feature transformers of *fixed size* with respect to the string length. While Liu et al. (2023, Theorem 2) also provides a result using a finite-depth transformer for a subset of FSAs, that construction results in a network much bigger and deeper than ours. For example, their construction would result in $\left(|\Sigma|^{n-1}\right)^2 (n-1)\log|\Sigma|$ layers in contrast to $n-1$ layers with a single head in our case. Our focus on n-gram LMs thus allows for a more compact and simpler representation. Worth noting is that despite being close to our analysis, none of the related work treats probability distributions over strings but rather focuses on the binary decision of whether a string belongs to a language or not based on the conditional probabilities output by the model, or, in the case of Liu et al. (2023), only the computation of state sequences. Transformer LMs are studied probabilistically by Xie et al. (2022), who provide a Bayesian interpretation of their ability to learn *in context* by studying the learning abilities of hidden Markov models (HMMs). While HMMs are equivalent to probabilistic finite-state automata, Xie et al. (2022) do not connect the in-context learning ability to concrete representational capacity results. Rather, they seek to *explain* the behavior of in-context learning as implicit Bayesian inference.

7 Conclusion

We study the representational capacity of transformer LMs with n-gram LMs. We show how the parallelizable nature of n-gram LMs is easy to capture with the transformer architecture and provide multiple lower bounds on the probabilistic representational capacity of transformer LMs. Concretely, we show that transformer LMs can represent n-gram LMs both with hard and sparse attention, exhibiting multiple mechanisms transformer LMs can employ to simulate n-gram LMs. Altogether, our results reinforce the utility of non-sequential models of computation for the study of transformers,

[7] The natural connection of the transformer architecture to both bounded hierarchical languages as well as to n-gram LMs is interesting since those two classes of LMs can both be represented particularly *efficiently* by *recurrent* neural LMs (Svete et al., 2024).

particularly in the language modeling setting.

Limitations

We connect transformer LMs to n-gram LMs because of their parallelizable nature and their traditional popularity in NLP. However, n-gram LMs describe a very simple class of LMs, meaning that the lower bounds are somewhat less relevant than the characterization in terms of more expressive formal models of computation would be. Accordingly, we expect that the lower bounds are somewhat loose and that transformer LMs can represent more than n-gram LMs, which is also in line with the empirical success of transformer LMs. We leave it to future work to tighten the established lower bounds.

As with most theoretical investigations of transformers, our results are strongest and the most precise in the hard attention setting. However, hard attention is not used in practice, which limits the applicability of the results. The constructions presented in this paper are also purely meant to showcase the existence of a mechanism that can be used to simulate n-gram LMs. They do not suggest that the same mechanisms will be learned by models used in practice. Indeed, the very sparse representations are not in line with the common dense contextual representations usually learned by trained models.

We also only focus on *lower bounds* of the representational capacity. We do not consider any upper bounds and existing results for similar models to ours suggest that the lower bound is indeed somewhat loose (Yao et al., 2021).[8] That is, we expect that transformer LMs can represent much more than n-gram LMs, and expect that many of the existing results on the computational power of such models can be extended to the probabilistic setting.

While we present a comprehensive analysis of transformer LMs in the context of n-gram LMs, we do not consider various aspects of the relationship that could be interesting. This is done to keep the presentation focused and concise. For example, we do not consider whether such simulations can be *learned* from data, an interesting avenue for future research. Lastly, note that while we focus here specifically on the commonly deployed transformer-based *language models*, there are many other interesting applications of transformers, such as encoder-only acceptors of unweighted languages. These applications are better covered by existing work.

Ethics Statement

The paper provides a way to theoretically analyze language models. To the best knowledge of the authors, there are no ethical implications of this paper.

Acknowledgements

Ryan Cotterell acknowledges support from the Swiss National Science Foundation (SNSF) as part of the "The Forgotten Role of Inductive Bias in Interpretability" project. Anej Svete is supported by the ETH AI Center Doctoral Fellowship. We thank Tim Vieira and Shannon Veitch for helpful discussions and comments on the manuscript.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. 2024. In-context language learning: Architectures and algorithms. *arXiv preprint arXiv:2401.12973*.

Dana Angluin, David Chiang, and Andy Yang. 2023. Masked hard-attention transformers and boolean rasp recognize exactly the star-free languages. *arXiv preprint arXiv:2310.13897*.

Enes Avcu, Chihiro Shibata, and Jeffrey Heinz. 2017. Subregular complexity and deep learning. *arXiv preprint arXiv:1705.05940*.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190.

J. Baker. 1975a. The DRAGON system–An overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29.

James K. Baker. 1975b. *Stochastic Modeling as a Means of Automatic Speech Recognition.* Ph.D. thesis, Carnegie Mellon University, USA. AAI7519843.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. *Neural Probabilistic Language Models*, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg.

[8] For example, we cannot say that the lower bounds imply any limitations of hard attention transformer LMs.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7654–7664, Dublin, Ireland. Association for Computational Linguistics.

David Chiang, Peter Cholak, and Anand Pillay. 2023. Tighter bounds on the expressivity of transformer encoders. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Li Du, Lucas Torroba Hennigen, Tiago Pimentel, Clara Meister, Jason Eisner, and Ryan Cotterell. 2023. A measure-theoretic characterization of tight language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9744–9770, Toronto, Canada. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Yiding Hao, Dana Angluin, and Robert Frank. 2022a. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

Yiding Hao, Dana Angluin, and Robert Frank. 2022b. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

Yiding Hao, William Merrill, Dana Angluin, Robert Frank, Noah Amsel, Andrew Benz, and Simon Mendelsohn. 2018. Context-free transductions with neural stacks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 306–315, Brussels, Belgium. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria. Association for Computational Linguistics.

Jeffrey Heinz and James Rogers. 2013. Learning subregular classes of languages with factored deterministic automata. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 64–71, Sofia, Bulgaria. Association for Computational Linguistics.

John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D. Manning. 2020. RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, Online. Association for Computational Linguistics.

Gerhard Jäger and James Rogers. 2012. Formal language theory: Refining the Chomsky hierarchy. *Philos Trans R Soc Lond B Biol Sci*, 367(1598):1956–1970.

F. Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

F. Jelinek. 1990. *Self-Organized Language Modeling for Speech Recognition*, page 450–506. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

S. C. Kleene. 1956. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies. (AM-34), Volume 34*, pages 3–42. Princeton University Press, Princeton.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. Transformers learn shortcuts to automata. In *International Conference on Learning Representations*.

W. Maass, G. Schnitger, and E. D. Sontag. 1991. On the computational power of sigmoid versus boolean threshold circuits. In *Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 767–776.

André F. T. Martins and Ramón F. Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1614–1623.

William Merrill. 2019. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, Florence. Association for Computational Linguistics.

William Merrill and Ashish Sabharwal. 2023a. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*.

William Merrill and Ashish Sabharwal. 2023b. A logic for expressing log-precision transformers. *arXiv preprint arXiv:2210.02671*.

William Merrill and Ashish Sabharwal. 2023c. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545.

William Merrill, Ashish Sabharwal, and Noah A. Smith. 2022. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856.

William Merrill and Nikolaos Tsilivis. 2022. Extracting finite automata from RNNs using state merging. *arXiv preprint arXiv:2201.12451*.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of RNN architectures. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 443–459, Online. Association for Computational Linguistics.

Marvin Lee Minsky. 1954. *Neural Nets and the Brain Model Problem*. Ph.D. thesis, Princeton University.

Franz Nowak, Anej Svete, Alexandra Butoi, and Ryan Cotterell. 2024. On the representational capacity of neural language models with chain-of-thought reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Franz Nowak, Anej Svete, Li Du, and Ryan Cotterell. 2023. On the representational capacity of recurrent neural language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7011–7034, Singapore. Association for Computational Linguistics.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. 2021. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Hava T. Siegelmann and E. D. Sontag. 1992. On the computational power of neural nets. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 440–449, New York, NY, USA. Association for Computing Machinery.

Paul Smolensky, Michael C. Mozer, and David E. Rumelhart. 1996. *Mathematical Perspectives on Neural Networks*. Psychology Press.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2023. Transformers as recognizers of formal languages: A survey on expressivity. *arXiv preprint arXiv:2311.00208*.

Anej Svete, Robin Shing Moon Chan, and Ryan Cotterell. 2024. A theoretical result on the inductive bias of RNN language models. *arXiv preprint arXiv:2402.15814*.

Anej Svete and Ryan Cotterell. 2023. Recurrent neural language models as probabilistic finite-state automata. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8069–8086, Singapore. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR.

Kaiyue Wen, Yuchen Li, Bingbin Liu, and Andrej Risteski. 2023. Transformers are uninterpretable with myopic methods: A case study with bounded Dyck grammars. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3770–3785, Online. Association for Computational Linguistics.

A Modelling Assumptions

As encouraged by Strobl et al. (2023), we provide in Tab. 2 a short summary of the assumptions behind the specific transformer architecture we consider in this work. This aims to facilitate the placement of the results in the context of existing work and to make the results more accessible to the reader.

Lower bound	PE	Precision	Attention	Attention	Architecture	Notes
∋ n-gram LMs	$\begin{pmatrix} \sqrt{\frac{1}{t+k}} \\ \sqrt{1-\frac{1}{t+k}} \end{pmatrix}_{k=0,\dots,n-1}$	$\mathbb{Q}$, $\mathbb{R}$	hard	decoder-only	$n-1$ heads, 1 layer	Theorem 3.1
∋ n-gram LMs	$\begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix}$	$\mathbb{Q}$, $\mathbb{R}$	hard	decoder-only	1 head, $n-1$ layers	Theorem 3.2
∋ n-gram LMs	$1, t, 10^{-t}$	$\mathbb{Q}$, $\mathbb{R}$	hard	decoder-only	1 head, 1 layer	Theorem 3.3
∋ n-gram LMs	$1, t$	$\mathbb{Q}$, $\mathbb{R}$	sparse	decoder-only	$n-1$ heads, 1 layer	Theorem 4.1

Table 2: A summary of the main assumptions about the models in the style of Strobl et al. (2023, Table 1). Hard attention here refers to average-hard in the vocabulary of Strobl et al. (2023).

B Proofs: Hard Attention

This section provides detailed proofs of all theorems about the representational capacity of hard-attention transformer LMs stated in the main part of the paper.

B.1 Computing Logical AND with an MLP

Definition B.1. *A* ReLU*-activated* ***multi-layer-perceptron*** *(MLP)* $\mathrm{MLP}\colon \mathbb{R}^N \to \mathbb{R}^M$ *is a function defined as the composition of functions* $\boldsymbol{f}_1, \cdots, \boldsymbol{f}_L$

$$\mathrm{MLP}\left(\mathbf{x}\right) \overset{\text{def}}{=} \left(\boldsymbol{f}_L \circ \boldsymbol{f}_{L-1} \circ \cdots \circ \boldsymbol{f}_1\right)\left(\mathbf{x}\right) \tag{20}$$

where each $\boldsymbol{f}_\ell$ *for* $\ell \in [L]$ *is defined as*

$$\boldsymbol{f}_\ell\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathrm{ReLU}\left(\mathbf{W}_\ell \mathbf{x} + \mathbf{b}_\ell\right) \quad \ell \in [L-1] \tag{21a}$$

$$\boldsymbol{f}_L\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathbf{W}_L \mathbf{x} + \mathbf{b}_L \tag{21b}$$

where $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times M_\ell}$ *is a weight matrix with dimensions* N_ℓ *and* M_ℓ *specific to layer* ℓ*,* $\mathbf{b}_\ell \in \mathbb{R}^{M_\ell}$ *is a bias vector. We refer to MLPs by the number of hidden layers, e.g., a one-layer-MLP is an MLP with one* hidden *layer.*

In our construction, simulating a n-gram LM with a transformer LM requires a component of the transformer to perform the logical AND operation between specific entries of binary vectors $\mathbf{x} \in \mathbb{B}^D$. The following lemma shows how this can be performed by an MLP with appropriately set parameters.

Lemma B.1. *Consider* m *indices* $i_1, \dots, i_m \in [D]$ *and vectors* $\mathbf{x}, \mathbf{v} \in \mathbb{B}^D$ *such that*

$$v_i = \mathbb{1}\left\{i \in \{i_1, \dots, i_m\}\right\}, \tag{22}$$

i.e., with entries 1 *at indices* $i_1, \dots, i_m$*. Then, it holds for the MLP* $\mathrm{MLP}\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathrm{ReLU}\left(\mathbf{v}^\top \mathbf{x} - (m-1)\right)$ *that*

$$\mathrm{MLP}\left(\mathbf{x}\right) = 1 \textit{ if and only if } x_{i_k} = 1 \textit{ for all } k = 1, \dots, m. \tag{23}$$

In other words,

$$\mathrm{MLP}\left(\mathbf{x}\right) = x_{i_1} \wedge \cdots \wedge x_{i_m}. \tag{24}$$

Proof. By the definition of $\mathbf{v}$, $\mathbf{v}^\top \mathbf{x} \leqslant m$ for all $\mathbf{x} \in \mathbb{B}^D$. Furthermore, $\mathbf{v}^\top \mathbf{x} = m$ if and only if $x_{i_k} = 1$ for all $k = 1, \dots, m$. The ReLU function maps all other values to 0 and does not change the output value 1 in this case. ■

B.2 Proofs of the Hard Attention Case

This subsection contains all proofs of the representational capacity of hard attention transformer LMs. We tackle three cases: the simulation with $n-1$ heads and a single layer, with $n-1$ layers and a single head, and lastly, the simulation with a single head and a single layer. All sections first outline the intuition behind the proofs and then provide the details in the form of finer-grained lemmata.

B.2.1 Simulation with $n-1$ Heads: The Intuition

We now outline the intuition behind the construction of a hard attention transformer LM simulating an n-gram LM, as first presented in Fig. 1.[9] To ease the exposition, we start with the final step of the construction: Assuming we have identified the appropriate history $\boldsymbol{y}_{t-n+1}^{t-1}$ after combining the head values using the head combining function $\mathcal{H}$, we show how $p_{\mathcal{T}}$ can encode the conditional probability distribution $p\left(y_t \mid \boldsymbol{y}_{t-n+1:t-1}\right)$. The intuition of this step is simple: Knowing what the individual $p\left(y_t \mid \boldsymbol{y}_{t-n+1}^{t-1}\right)$ for $y_t \in \overline{\Sigma}$ are, we can simply put their logits into a vector and combine the constructed vectors for all possible histories into the output matrix $\mathbf{E}$:[10,11]

$$E_{y,\boldsymbol{y}_{t-n+1}^{t-1}} \overset{\text{def}}{=} \log p\left(y_t \mid \boldsymbol{y}_{t-n+1}^{t-1}\right) \tag{25}$$

In the following, we write $\mathsf{enc}\left(\boldsymbol{y}_{<t}\right)$ as shorthand notation for $\mathsf{enc}\left(\boldsymbol{y}_{<t}\right) \overset{\text{def}}{=} F\left(\mathbf{x}_{t-1}^{L}\right)$ (i.e., the representation which is linearly transformed by $\mathbf{E}$ to compute $p\left(y_t \mid \boldsymbol{y}_{<t}\right)$ after normalization) where $\mathbf{X}^{L} = \mathcal{T}\left(\mathcal{R}\right)\left(\boldsymbol{y}_{<t}\right)$. If we one-hot encode the identified history with $\mathcal{T}$ as

$$\mathsf{enc}\left(\boldsymbol{y}_{<t}\right) \overset{\text{def}}{=} [\![\boldsymbol{y}_{t-n+1}^{t-1}]\!] \tag{26}$$

we can then, using the formulation of the transformer LM from Definition 2.11, use the $\mathsf{enc}\left(\boldsymbol{y}_{<t}\right)$ to look up the appropriate column in $\mathbf{E}$ containing the logits of the conditional probabilities given the identified history for all possible $y_t \in \overline{\Sigma}$, i.e., $\left(\mathbf{E}\,\mathsf{enc}\left(\boldsymbol{y}_{<t}\right)\right)_y = \log p\left(y \mid \boldsymbol{y}_{t-n+1}^{t-1}\right)$.

We now consider the preceding step of the simulation: Identifying the history given that the $n-1$ heads identified the symbols $y_1, \ldots, y_{n-1}$ in the positions they attended to. If we concatenate the values of the $n-1$ heads into a vector $\mathbf{v}$, this vector of size $\left(n-1\right)|\overline{\Sigma}|$ will contain the **multi-hot** representation of the history of interest:

$$\mathbf{v} = \begin{pmatrix} [\![y_1]\!] \\ \vdots \\ [\![y_{n-1}]\!] \end{pmatrix} \tag{27}$$

and $\mathbf{v}_{i|\overline{\Sigma}|+j} = 1$ if and only if $m\left(y_i\right) = j$ for a bijection $m \colon \overline{\Sigma} \to \left[|\overline{\Sigma}|\right]$ that determines the indices of the one-hot representations of the symbols. We would then like to transform this vector into a vector $\mathbf{u} \in \mathbb{R}^{|\Sigma|^{n-1}}$ such that

$$u_i = 1 \iff i = s\left(y_1, \ldots, y_{n-1}\right) \tag{28}$$

for a bijection $s \colon \underbrace{\overline{\Sigma} \times \cdots \times \overline{\Sigma}}_{n-1 \text{ times}} \to \left[|\overline{\Sigma}|^{n-1}\right]$. This can be equivalently written as

$$u_i = 1 \iff v_{j|\overline{\Sigma}|+m(y_j)} = 1 \text{ for all } j = 1, \ldots, n-1 \tag{29}$$

where $i = s\left(y_1, \ldots, y_{n-1}\right)$. This is an instance of performing the logical AND operation, which can be implemented by an MLP as described in Lemma B.1. This MLP will form the transformation $\mathcal{H}$ combining the information obtained from all the heads of the transformer.

This brings us to the final part of the proof: Identifying the symbols at the previous $n-1$ positions by the $n-1$ transformer heads. To show how this can be done, let us consider the parameters we can still set to define a transformer:

[9] For simplicity, we disregard the role of residual connections in the following outline. Residual connections are, however, considered in the full proof later.

[10] To be able to take the log of 0 probabilities, we work over the set of *extended* reals $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$.

[11] Throughout the paper, we implicitly index the matrices directly with symbols and histories. We assume that the symbols and histories are ordered in some way and that the matrices are ordered accordingly.

- The position-augmented symbol representations $\mathbf{r}$. Inspired by concurrent work from Merrill and Sabharwal (2023a), we use the representations of the form

$$\mathbf{r}\left(y_t, t\right) = \begin{pmatrix} [\![y_t]\!] \\ \sqrt{\frac{1}{t}} \\ \sqrt{1 - \frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1 - \frac{1}{t+1}} \\ \vdots \\ \sqrt{\frac{1}{t+n-1}} \\ \sqrt{1 - \frac{1}{t+n-1}} \end{pmatrix} \in \mathbb{R}^{2n} \tag{30}$$

  This results in vectors $\mathbf{r}\left(y_t, t\right)$ of size $|\Sigma| + 2 + 2\left(n-1\right)$. Note that such a symbol representation function can be implemented by concatenating or adding symbol- ($[\![y_t]\!]$) and position- ($\sqrt{\frac{1}{t}}, \ldots, \sqrt{1 - \frac{1}{t+n-1}}$) specific components, which is in line with most practical implementations of the transformer architecture.
- The attention scoring function f. We will use the standard dot-product scoring function

$$f\left(\mathbf{q}, \mathbf{k}\right) \overset{\text{def}}{=} \langle \mathbf{q}, \mathbf{k} \rangle. \tag{31}$$

  f will, together with the positional encodings, allow us to easily single out the relevant positions in the string.
- The parameters of each of the attention heads, that is, the transformations Q, K, and V. Each of those will take the form of a linear transformation of the symbol (and positional) representations. We describe them and their roles in more detail below.

The parameters of all the heads will be identical, with the only difference being a single parameter that depends on the "index" of the head, h. In the following, we describe the construction of a single head. At any time step t (i.e., when modeling the conditional distribution $p\left(y_t \mid \boldsymbol{y}_{<t}\right)$), the head h will attend to the symbol at position $t-h$, y_{t-h}. In Fig. 1, for example, Head 3 attends to the position $t-3$, which is denoted by the stronger arrow to that position. We now describe the individual transformations Q_h, K_h, V_h, and O_h of the head h. All of them will be *affine* transformations. Since we are considering only the first layer of the transformer, we can think of the inputs to the layer as the original symbol representations together with their position encodings (rather than some contextual representations at higher levels). As mentioned, the head h will be responsible for identifying the symbol at position $t-h$. Therefore, we want it to put all its attention to this position. In other words, given the query $\mathbf{q}_{t-1}$, we want the attention function in Eq. (31) to be uniquely maximized by the key of the symbol at position $t-h$. Notice that, therefore, the key does not have to depend on the identity of the symbol at position $t-h$—only the positional information matters. Let us then consider the following query and key transformations for head h:

$$Q_h \colon \mathbf{r}\left(y_t, t\right) \mapsto \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1 - \frac{1}{t}} \end{pmatrix} \tag{32}$$

$$K_h \colon \mathbf{r}\left(y_t, t\right) \mapsto \begin{pmatrix} \sqrt{\frac{1}{t+h}} \\ \sqrt{1 - \frac{1}{t+h}} \end{pmatrix}. \tag{33}$$

Given such a query and such keys, the scoring function computes

$$f\left(\mathbf{q}_t, \mathbf{k}_j\right) = \left\langle \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1 - \frac{1}{t}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1 - \frac{1}{j+h}} \end{pmatrix} \right\rangle. \tag{34}$$

Eq. (34) is an inner product between two unit vectors, and is therefore maximized if and only if they are the same, that is, if $j = t - h$. This is exactly the position that we want the head h to attend to.[12] Intuitively, both transformations keep only the positional information. The query transformation "injects" the knowledge of which position should maximize the attention score, while the key transformation simply "exposes" the positional information about the symbol. The constants 1 and -1 and the index of the position ensure that the inner product simply computes the difference between the position of the symbol and the position of interest.

This leaves us with the question of how to use the position of the symbol of interest $(t - h)$ to extract the one-hot encoding of the symbol at that position. Due to the information contained in the symbol representations $\mathbf{r}\left(y_j\right)$, this is trivial:

$$V \colon \mathbf{r}\left(y_t, t\right) \mapsto [\![y_j]\!]. \tag{35}$$

With this, the identity of the symbol is carried forward through the attention mechanism. Notice that the only head-depend transformation is the query transformation—it depends on the index of the head, determining the position of interest, meaning that every head defines a different query transformation, while the keys and values transformations are the same among all heads. This concludes the outline of the proof.

B.2.2 Simulation with $n - 1$ Heads: Proofs

This subsection formally proves the construction intuited in Appendix B.2.1 by proving a sequence of lemmata that formalize each of the steps described in the intuition. Specifically,

1. Lemma B.2 shows how the one-hot encodings of individual symbols in the history can be combined into the one-hot encoding of the history.

2. Lemma B.3 shows that the scoring function is maximized at the position of interest.

3. Lemma B.4 shows how the one-hot encodings of the symbols in the history can be identified by the hard attention mechanism.

4. The proof of Theorem 3.1 shows how the construction of the one-hot encoding of the current history allows us to define the appropriate next-symbol conditional distribution of the n-gram LMs.

Lemma B.2. *Let $\mathcal{T}$ be a transformer with $H = n - 1$ heads. Let $\mathbf{z}_h = [\![y_{t-h}]\!]$ be the output of the h^{th} head at time t. Then, there exists a function $\mathcal{H}$ implemented by a single-layer MLP such that*

$$\mathcal{H}\left(\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}\right) = [\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]. \tag{36}$$

Proof. Let $\boldsymbol{y} = y_1 \ldots y_{n-1} \in \underline{\Sigma}^{n-1}$ and let i be the index in $\left[|\underline{\Sigma}|^{n-1}\right]$ that corresponds to $\boldsymbol{y}$. Furthermore, let $i_1, \ldots, i_{n-1}$ be the indices corresponding to the symbols $y_1, \ldots, y_{n-1}$ in $\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}$. Then, we have

$$[\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]_i = 1 \iff z_{1,i_1} = 1 \wedge \cdots \wedge z_{n-1,i_{n-1}} = 1 \tag{37}$$

Eq. (36) is an instance of the logical AND operation on the indices encoding the individual histories, which can be implemented by an MLP as shown in Lemma B.1. ■

The next lemma presents a useful equality about the standard dot-product attention scoring function: For unit vectors, the attention score is maximized if and only if the vectors are identical. We will use this fact in our construction to attend to particular positions in the string.

Lemma B.3. *Given a fixed $t \in \mathbb{N}$, f, define*

$$g\left(j\right) \stackrel{\text{def}}{=} \left\langle \begin{pmatrix} \sqrt{\frac{1}{t-1}} \\ \sqrt{1 - \frac{1}{t-1}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1 - \frac{1}{j+h}} \end{pmatrix} \right\rangle \tag{38}$$

[12] Note that while the choice of the positional encodings in this construction is uncommon in practice, the popular sinusoidal positional encodings (Vaswani et al., 2017) have also been linked to the ability of the transformer-based models to attend to specific positions of interest based on linear transformations of the positional encodings (Vaswani et al., 2017).

for $j \in [t-1]$. Then, g is maximized at $j = t-1-h$:

$$\underset{j \in [t-1]}{\operatorname{argmax}} \left(\left\langle \begin{pmatrix} \sqrt{\frac{1}{t-1}} \\ \sqrt{1-\frac{1}{t-1}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \right\rangle \right) = t-1-h. \tag{39}$$

Proof. The two arguments to the inner product in Eq. (39) are unit vectors. Inner products of unit vectors are at most 1, with the maximum achieved only if the two vectors are identical. This means that the function in Eq. (39) is maximized if and only if

$$\begin{pmatrix} \sqrt{\frac{1}{t-1}} \\ \sqrt{1-\frac{1}{t-1}} \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \iff \tag{40a}$$

$$\sqrt{\frac{1}{t-1}} = \sqrt{\frac{1}{j+h}} \iff \tag{40b}$$

$$\frac{1}{t-1} = \frac{1}{j+h} \iff \tag{40c}$$

$$j = t-1-h. \tag{40d}$$

■

The following lemma presents the core of the proof of Theorem 3.1, exhibiting the construction of a transformer head that can single out and one-hot encode a symbol at a specific position in the input string. The lemma relies on a simple pre-processing of the input string, where the string is prepended (padded) with $n-1$ beginning of string symbols $y_1 = \ldots = y_{n-1} \stackrel{\text{def}}{=} \text{BOS}$, which is common practice in language modeling literature, especially when talking about n-gram LMs. We will denote $\underline{\Sigma} \stackrel{\text{def}}{=} \Sigma \cup \{\text{BOS}\}$. This enables a cleaner presentation of the concrete construction of the attention mechanism.

The idea of the proof. Lemma B.4 contains a number of technical definitions of the parameters of a transformer layer (cf. Definition 2.6. Together, they describe a single head of a transformer layer (which will contain $H = n-1$ such heads) that is able to extract the one-hot encoding of a particular symbol in the history. The layer of $n-1$ heads will then be able to extract the $n-1$ symbols, as required by Lemma B.2. We now describe a single head more formally. Define the following position-augmented symbol representation function of the transformer head h:

$$\mathbf{r}(y, t) = \begin{pmatrix} [\![y]\!] \\ \mathbf{0}_{|\underline{\Sigma}|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \\ \vdots \\ \sqrt{\frac{1}{t+n-1}} \\ \sqrt{1-\frac{1}{t+n-1}} \end{pmatrix} \in \mathbb{R}^{2|\underline{\Sigma}|+2n}. \tag{41}$$

Here, $[\![\cdot]\!] \in \{0,1\}^{|\underline{\Sigma}|}$ one-hot encodes symbols over $\underline{\Sigma}$. This means that the entire static representations contain multiple components:

- two $|\underline{\Sigma}|$-dimensional slots for *symbol* representations and
- n 2-dimensional slots for head-specific *positional* representations.

We further define

$$f\left(\mathbf{q}, \mathbf{k}\right) \overset{\text{def}}{=} \langle \mathbf{q}, \mathbf{k} \rangle, \tag{42}$$

$$\begin{aligned}
Q\left(\mathbf{x}\right) &\overset{\text{def}}{=} \mathbf{Q}\mathbf{x}, & \mathbf{Q} &\in \mathbb{R}^{2\times(2|\Sigma|+2n)}, & \text{(43a)}\\
K\left(\mathbf{x}\right) &\overset{\text{def}}{=} \mathbf{K}\mathbf{x}, & \mathbf{K} &\in \mathbb{R}^{2\times(2|\Sigma|+2n)}, & \text{(43b)}\\
V\left(\mathbf{x}\right) &\overset{\text{def}}{=} \mathbf{V}\mathbf{x}, & \mathbf{V} &\in \mathbb{R}^{(2|\Sigma|+2n)\times(2|\Sigma|+2n)}, & \text{(43c)}\\
O\left(\mathbf{x}\right) &\overset{\text{def}}{=} \mathbf{O}\mathbf{x}, & \mathbf{O} &\in \mathbb{R}^{(2|\Sigma|+2n)\times(2|\Sigma|+2n)}, & \text{(43d)}
\end{aligned}$$

$$\begin{aligned}
\mathbf{Q}_{:,2|\Sigma|+1:2|\Sigma|+2} &\overset{\text{def}}{=} \mathbf{I}_2 & \text{(44a)}\\
\mathbf{K}_{:,2|\Sigma|+2h+1:2|\Sigma|+2h+2} &\overset{\text{def}}{=} \mathbf{I}_2, & \text{(44b)}\\
\mathbf{V}_{|\Sigma|+1:2|\Sigma|,1:|\Sigma|} &\overset{\text{def}}{=} \mathbf{I}_{|\Sigma|} & \text{(44c)}\\
\mathbf{O}_{1:|\Sigma|,1:|\Sigma|} &\overset{\text{def}}{=} -\mathbf{I}_{|\Sigma|} & \text{(44d)}
\end{aligned}$$

We can visualize these matrices as

$$\mathbf{Q} = \begin{array}{ccccc}
\overbrace{\hphantom{xxxxxxx}}^{\substack{|\Sigma| \\ 1^{\text{st}}\text{ symbol slot}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{|\Sigma| \\ 2^{\text{nd}}\text{ symbol slot}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{2 \\ 0^{\text{th}}\text{ position slot}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{(n-1)2 \\ n-1\text{ position slots}}} & \\
\Big(& & \mathbf{I}_2 & \Big) & 2
\end{array} \tag{45a}$$

$$\mathbf{K} = \begin{array}{cccccc}
\overbrace{\hphantom{xxxxxxx}}^{\substack{|\Sigma| \\ 1^{\text{st}}\text{ symbol slot}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{|\Sigma| \\ 2^{\text{nd}}\text{ symbol slot}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{2h \\ h+1\text{ position slots}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{2 \\ h^{\text{th}}\text{ position slot}}} & \overbrace{\hphantom{xxxxxxx}}^{\substack{(n-1-h)2 \\ n-h\text{ position slots}}} & \\
\Big(& & & \mathbf{I}_2 & \Big) & 2
\end{array} \tag{45b}$$

$$\mathbf{V} = \begin{array}{c}
\begin{array}{ccc}
\overbrace{\hphantom{xxxxxxx}}^{1^{\text{st}}\text{ symbol slot}} & \overbrace{\hphantom{xxxxxxx}}^{2^{\text{nd}}\text{ symbol slot}} & \overbrace{\hphantom{xxxxxxx}}^{n\text{ position slots}}
\end{array} \\
\left(\begin{array}{ccc}
 & & \\
\mathbf{I}_{|\Sigma|} & & \\
 & &
\end{array}\right)
\begin{array}{l}
1^{\text{st}}\text{ symbol slot } (|\Sigma|) \\
2^{\text{nd}}\text{ symbol slot } (|\Sigma|) \\
n\text{ position slots } (2n)
\end{array}
\end{array} \tag{45c}$$

$$\mathbf{O} = \begin{array}{c}
\begin{array}{ccc}
\overbrace{\hphantom{xxxxxxx}}^{1^{\text{st}}\text{ symbol slot}} & \overbrace{\hphantom{xxxxxxx}}^{2^{\text{nd}}\text{ symbol slot}} & \overbrace{\hphantom{xxxxxxx}}^{n\text{ position slots}}
\end{array} \\
\left(\begin{array}{ccc}
-\mathbf{I}_{|\Sigma|} & & \\
 & & \\
 & &
\end{array}\right)
\begin{array}{l}
1^{\text{st}}\text{ symbol slot } (|\Sigma|) \\
2^{\text{nd}}\text{ symbol slot } (|\Sigma|) \\
n\text{ position slots } (2n)
\end{array}
\end{array} \tag{45d}$$

where $\mathbf{0}_N$ is a N-dimensional vector of zeros, $\mathbf{I}_N$ is the N-dimensional identity matrix and the unspecified elements of $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ are 0.

Lemma B.4. *Let Σ be an alphabet and $\boldsymbol{y} \in \Sigma^*$. For any $t = 1, \ldots, |\boldsymbol{y}|$, the h^{th} transformer head ($h \in [n-1]$) defined with the parameters specified in Eq. (41) to Eq. (44d) outputs*

$$\mathbf{z}_{t-1} = \begin{pmatrix} \mathbf{0}_{|\Sigma|} \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_{2n} \end{pmatrix}. \tag{46}$$

In particular, this means that the output $\mathbf{z}_{t-1}$ at time step $t-1$ contains the one-hot encoding of the symbol at position $t-h-1$.[13]

[13] For technical reasons—the residual connections—the output is not $[\![y_{t-1-h}]\!]$ but larger (with additional zeros), as shown in Eq. (46). This, however, is equivalent for the purposes of Lemma B.2 and later Theorem 3.1.

Proof. Fix $t \in [|\boldsymbol{y}|]$. We compute the representation of $\boldsymbol{y}_{<t}$ computed by the head. First, observe that the matrix defining the query function Q projects onto the first two components of the positional encoding from the representation of y_{t-1}:

$$Q\left(\mathbf{r}\left(y_{t-1}\right)\right) = \mathbf{Q}\,\mathbf{r}\left(y_{t-1}\right) = \begin{pmatrix} \sqrt{\frac{1}{t-1}} \\ \sqrt{1-\frac{1}{t-1}} \end{pmatrix}. \tag{47}$$

Similarly, the matrix defining the key transformation projects onto the h^{th} positional encoding slot, i.e., the dimensions $2|\Sigma| + 2h + 1$ and $2|\Sigma| + 2h + 2$:

$$K\left(\mathbf{r}\left(y_j\right)\right) = \mathbf{K}\,\mathbf{r}\left(y_j\right) = \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \tag{48}$$

for $j = -n+2, \ldots, t-1$.

This results in the scoring function

$$f\left(\mathbf{q}_{t-1}, \mathbf{k}_j\right) \overset{\text{def}}{=} \langle \mathbf{q}_{t-1}, \mathbf{k}_j \rangle = \left\langle \begin{pmatrix} \sqrt{\frac{1}{t-1}} \\ \sqrt{1-\frac{1}{t-1}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \right\rangle. \tag{49}$$

In particular, as shown in Lemma B.3, f is maximized for

$$j = t-1-h. \tag{50}$$

This means that

$$\operatorname{hardmax}\left(f\left(\mathbf{q}_{t-1}, \mathbf{k}_1\right), f\left(\mathbf{q}_{t-1}, \mathbf{k}_2\right), \ldots, f\left(\mathbf{q}_{t-1}, \mathbf{k}_{t-1}\right)\right)_j = \mathbb{1}\left\{j = t-1-h\right\}. \tag{51}$$

The definition of $\mathbf{V}$ further means that

$$V\left(\mathbf{r}\left(y_j\right)\right) = \mathbf{V}\,\mathbf{r}\left(y_j\right) = \begin{pmatrix} \mathbf{0}_{|\Sigma|} \\ [\![y_j]\!] \\ \mathbf{0}_{2n} \end{pmatrix}, \tag{52}$$

giving us, by Eq. (9a),

$$\mathbf{a}_{t-1} = \mathbf{v}_{t-1-h} + \mathbf{x}_{t-1} = \begin{pmatrix} [\![y_{t-1}]\!] \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_{2n} \end{pmatrix}. \tag{53}$$

The definition of O then gives us

$$\begin{aligned}
\mathbf{z}_{t-1} &= O\left(\mathbf{a}_{t-1}\right) + \mathbf{a}_{t-1} && \text{(54a)} \\
&= \mathbf{O}\mathbf{a}_{t-1} + \mathbf{a}_{t-1} && \text{(54b)} \\
&= \begin{pmatrix} -\mathbf{I}_{|\Sigma|} & & \\ & & \\ & & \end{pmatrix} \begin{pmatrix} [\![y_{t-1}]\!] \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_{2n} \end{pmatrix} + \begin{pmatrix} [\![y_{t-1}]\!] \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_{2n} \end{pmatrix} && \text{(54c)} \\
&= \begin{pmatrix} -[\![y_{t-1}]\!] \\ \mathbf{0}_{|\Sigma|} \\ \mathbf{0}_{2n} \end{pmatrix} + \begin{pmatrix} [\![y_{t-1}]\!] \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_{2n} \end{pmatrix} && \text{(54d)} \\
&= \begin{pmatrix} \mathbf{0}_{|\Sigma|} \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_{2n} \end{pmatrix}, && \text{(54e)}
\end{aligned}$$

which is what we wanted to prove. ■

Lemmas B.2 and B.4 show that we can define a transformer that one-hot encodes the history of interest $\boldsymbol{y}_{t-n+1}^{t-1}$. We now show how to define an output matrix $\mathbf{E}$ to define a weakly equivalent transformer LM. Concretely, we define $\mathbf{E} \in \mathbb{R}^{|\overline{\Sigma}| \times |\Sigma|^{n-1}}$ with

$$E_{y, \boldsymbol{y}_{t-n+1}^{t-1}} \stackrel{\text{def}}{=} \log p\left(y_t \mid \boldsymbol{y}_{t-n+1}^{t-1}\right). \tag{55}$$

Theorem 3.1. *For any n-gram LM, there exists a weakly equivalent single-layer hard attention transformer LM with $n-1$ heads.*

Proof. Let $\mathcal{T}$ be a transformer LM with $n-1$ heads defined with the parameters specified in Eq. (41) to Eq. (44d) ($h \in [n-1]$). Let $\boldsymbol{y} \in \{\text{BOS}\}^{n-1}\Sigma^*$ with $|\boldsymbol{y}| = T$ be a string and $\mathbf{X}^L = \left(\mathbf{x}_{-n+2}^{L\top}; \ldots; \mathbf{x}_T^{L\top}\right) = \mathcal{T}(\boldsymbol{y})$. We derive:

$$\begin{aligned}
p_{\mathcal{T}}(\boldsymbol{y}) &= p_{\mathcal{T}}(\text{EOS} \mid \boldsymbol{y}) \prod_{t=1}^{T} p_{\mathcal{T}}(y_t \mid \boldsymbol{y}_{<t}) && \text{(56a, Autoregressive LM.)} \\
&= \operatorname{softmax}\left(\mathbf{E}\, F\left(\mathbf{x}_T^L\right)\right)_{\text{EOS}} \prod_{t=1}^{T} \operatorname{softmax}\left(\mathbf{E}\, F\left(\mathbf{x}_{t-1}^L\right)\right)_{y_t} && \text{(56b, Definition 2.11.)} \\
&= \operatorname{softmax}(\mathbf{E}\, [\![\boldsymbol{y}_{T-n+2:T}]\!])_{\text{EOS}} \prod_{t=1}^{T} \operatorname{softmax}\left(\mathbf{E}\, [\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]\right)_{y_t} && \text{(56c, Lemma B.2.)} \\
&= \frac{\exp\left(\mathbf{E}\, [\![\boldsymbol{y}_{T-n+2:T}]\!]\right)_{\text{EOS}}}{\sum_{y \in \overline{\Sigma}} \exp\left(\mathbf{E}\, [\![\boldsymbol{y}_{T-n+2:T}]\!]\right)_y} \prod_{t=1}^{T} \frac{\exp\left(\mathbf{E}\, [\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]\right)_{y_t}}{\sum_{y \in \overline{\Sigma}} \exp\left(\mathbf{E}\, [\![\boldsymbol{y}_{T-n+1:t-1}]\!]\right)_y} && \text{(56d, Definition of softmax.)} \\
&= \frac{\exp\left(\log p\left(\text{EOS} \mid \boldsymbol{y}_{t-n+2}^{T}\right)\right)}{\sum_{y \in \overline{\Sigma}} \exp\left(\log p\left(y \mid \boldsymbol{y}_{t-n+2}^{T}\right)\right)} \prod_{t=1}^{T} \frac{\exp\left(\log p\left(y_t \mid \boldsymbol{y}_{t-n+1}^{t-1}\right)\right)}{\sum_{y \in \overline{\Sigma}} \exp\left(\log p\left(y \mid \boldsymbol{y}_{t-n+1}^{t-1}\right)\right)} && \text{(56e, Eq. (55).)} \\
&= p\left(\text{EOS} \mid \boldsymbol{y}_{T-n+2:T}\right) \prod_{t=1}^{T} p\left(y_t \mid \boldsymbol{y}_{t-n+1}^{t-1}\right) = p(\boldsymbol{y}) && \text{(56f, The } n\text{-gram LM is autoregressive.)}
\end{aligned}$$

■

B.2.3 Simulation with $n-1$ Layers: The Intuition

This section presents the construction of a transformer LM with a *single* head but $n-1$ layers that is weakly equivalent to a given n-gram LM. We again outline the intuition first before giving the technical details below as part of Lemma B.5. The construction we present resembles the one from the proof of Theorem 3.1. The main difference is that, instead of using $n-1$ attention heads to identify the history, we instead use $n-1$ transformer *layers*. Intuitively, each of the $n-1$ layers iteratively adds one of the $n-1$ symbols needed to identify the history, resulting in the identification of the full history after the $(n-1)^{\text{st}}$ layer. Once the history is identified, the $(n-1)^{\text{st}}$ layer can compute the conditional distribution over the next symbol as in the proof of Theorem 3.1. This can be illustrated as the following sequence of transformations:[14]

$$\mathbf{X}^1\colon \begin{pmatrix} \llbracket y_1 \rrbracket & \llbracket y_2 \rrbracket & \llbracket y_3 \rrbracket & \cdots & \llbracket y_t \rrbracket & \cdots & \llbracket y_T \rrbracket \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} \end{pmatrix} \tag{57a}$$

$$\mathbf{X}^2\colon \begin{pmatrix} \llbracket y_1 \rrbracket & \llbracket y_2 \rrbracket & \llbracket y_3 \rrbracket & \cdots & \llbracket y_t \rrbracket & \cdots & \llbracket y_T \rrbracket \\ \mathbf{0}_{|\Sigma|} & \llbracket y_1 \rrbracket & \llbracket y_2 \rrbracket & \cdots & \llbracket y_{t-1} \rrbracket & \cdots & \llbracket y_{T-1} \rrbracket \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} & \cdots & \mathbf{0}_{|\Sigma|} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \cdots & \cdots & \cdots & \mathbf{0}_{|\Sigma|} \end{pmatrix} \tag{57b}$$

$$\vdots$$

$$\mathbf{X}^{n-1}\colon \begin{pmatrix} \llbracket y_1 \rrbracket & \llbracket y_2 \rrbracket & \llbracket y_3 \rrbracket & \cdots & \llbracket y_t \rrbracket & \cdots & \llbracket y_T \rrbracket \\ \mathbf{0}_{|\Sigma|} & \llbracket y_1 \rrbracket & \llbracket y_2 \rrbracket & \cdots & \llbracket y_{t-1} \rrbracket & \cdots & \llbracket y_{T-1} \rrbracket \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \llbracket y_1 \rrbracket & \cdots & \llbracket y_{t-2} \rrbracket & \cdots & \llbracket y_{T-1} \rrbracket \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \mathbf{0}_{|\Sigma|} & \cdots & \llbracket y_{t-n+1} \rrbracket & \cdots & \llbracket y_{T-n+2} \rrbracket \end{pmatrix} \tag{57c}$$

Such representations can be constructed by starting with the initial symbol encoding ($\mathbf{X}^1$) and then *passing over* the information from the t^{th} symbol to the contextual representations of the $(t+1)^{\text{st}}$ symbol in $\mathbf{X}^2$ by adding a *shifted* the contextual representation of the t^{th} symbol. This is performed $n-1$ times, resulting in contextual representations of the form $\mathbf{X}^{n-1}$ where the t^{th} contextual representation contains the (ordered) information about the preceding $n-1$ symbols, i.e., the required history. Intuitively, the ℓ^{th} transformation of the contextual representation $\mathbf{x}_t^{\ell-1}$ should be of the form

$$\mathbf{x}_t^{\ell} = \underbrace{\mathbf{x}_t^{\ell-1}}_{\substack{\text{The previous } \ell-1 \\ \text{symbols.}}} + \underbrace{\downarrow \mathbf{x}_{t-1}^{\ell-1}}_{\substack{\text{The symbol } \ell \text{ symbols back} \\ \text{shifted one ``cell'' downward.}}} \tag{58}$$

The first term is simply the residual connection of the transformer layer. The second term is the shifted representation of the symbol $y_{t-\ell}$, which can be performed by a simple linear transformation in the value transformation V.

B.2.4 Simulation with $n-1$ Layers: Proofs

We now make the intuition presented in the previous section more formal. Concretely, we only investigate how to identify the correct $n-1$ symbols in the history with the $n-1$ layers. We then rely on Lemma B.2 and the derivation from the proof of Theorem 3.1 again to convert this information into a weakly equivalent transformer LM. Let $\ell \in [n-1]$. Define the following parameters of the attention head of the ℓ^{th} transformer layer:

$$\mathbf{r}(y,t) \overset{\text{def}}{=} \begin{pmatrix} \llbracket y \rrbracket \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} \in \mathbb{R}^{(n-1)|\Sigma|+4}, \tag{59}$$

[14] We leave out the positional encodings for clarity.

$$f\left(\mathbf{q},\mathbf{k}\right) \stackrel{\text{def}}{=} \langle \mathbf{q},\mathbf{k}\rangle, \tag{60}$$

$$\begin{aligned}
Q\left(\mathbf{x}\right) &\stackrel{\text{def}}{=} \mathbf{Q}\mathbf{x}, \quad \mathbf{Q} \in \mathbb{R}^{2\times((n-1)|\Sigma|+4)} && \text{(61a)}\\
K\left(\mathbf{x}\right) &\stackrel{\text{def}}{=} \mathbf{K}\mathbf{x}, \quad \mathbf{K} \in \mathbb{R}^{2\times((n-1)|\Sigma|+4)} && \text{(61b)}\\
V\left(\mathbf{x}\right) &\stackrel{\text{def}}{=} \mathbf{V}\mathbf{x}, \quad \mathbf{V} \in \mathbb{R}^{((n-1)|\Sigma|+4)\times((n-1)|\Sigma|+4)}, && \text{(61c)}\\
O\left(\mathbf{x}\right) &\stackrel{\text{def}}{=} \mathbf{O}\mathbf{x}, \quad \mathbf{O} \in \mathbb{R}^{((n-1)|\Sigma|+4)\times((n-1)|\Sigma|+4)}, && \text{(61d)}
\end{aligned}$$

$$\begin{aligned}
\mathbf{Q}_{:,(n-1)|\Sigma|+1:(n-1)|\Sigma|+2} &\stackrel{\text{def}}{=} \mathbf{I}_2, && \text{(62a)}\\
\mathbf{K}_{:,(n-1)|\Sigma|+3:(n-1)|\Sigma|+4} &\stackrel{\text{def}}{=} \mathbf{I}_2, && \text{(62b)}\\
\mathbf{V}_{1+\ell|\Sigma|:1+(\ell+1)|\Sigma|,1+(\ell-1)|\Sigma|:1+\ell|\Sigma|} &\stackrel{\text{def}}{=} \mathbf{I}_{|\Sigma|}, && \text{(62c)}
\end{aligned}$$

where the unspecified elements of $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are 0. Schematically, $\mathbf{V}$ looks as follows:

$$\mathbf{V} = \begin{array}{c} \\ \left(\begin{array}{cccc} \overbrace{\qquad\qquad}^{\substack{(\ell-1)|\Sigma| \\ (\ell-1)\text{ symbol slots}}} & \overbrace{\qquad\qquad}^{\substack{|\Sigma| \\ \ell^{\text{th}}\text{ symbol slot}}} & \overbrace{\qquad\qquad}^{\substack{(n-1-\ell)|\Sigma| \\ (n-1-\ell)\text{ symbol slots}}} & \overbrace{\qquad\qquad}^{\substack{4 \\ 4\text{ position slots}}} \\ & & & \\ & \mathbf{I}_{|\Sigma|} & & \\ & & & \\ & & & \end{array}\right) \end{array} \begin{array}{c} \\ \\ \ell|\Sigma| \\ |\Sigma| \\ (n-1-(\ell+1))|\Sigma| \\ 4 \end{array} \tag{63}$$

That is, $\mathbf{I}_{|\Sigma|}$ occupies the "cell" ℓ cells down and $\ell - 1$ right of the top-left corner. Moreover, the matrix $\mathbf{O}$ is a matrix of all zeros, meaning that $O\left(\mathbf{x}\right) = \mathbf{0}$ for all $\mathbf{x}$. Again, notice that the position-augmented symbol representation function $\mathbf{r}$ can be implemented by concatenating or summing a symbol- and a position-specific component.

Lemma B.5. *With the parameters defined above, it holds that*

$$\mathbf{x}_{t-1}^{\ell} = \begin{pmatrix} [\![y_{t-1}]\!] \\ [\![y_{t-1-1}]\!] \\ [\![y_{t-1-2}]\!] \\ \vdots \\ [\![y_{t-1-\ell}]\!] \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} \tag{64}$$

Proof. We prove the lemma by induction. As in the proof of Lemma B.4, we pad the input string with $n - 1$ BOS symbols to resolve the case when $t - 1 < n - 1$.

Base Case. For $\ell = 1$, the claim follows from the definition of the symbol representation function $\mathbf{r}$.

Inductive Step. For $\ell > 1$, we assume that the claim holds for $\ell - 1$. We then prove that it holds for ℓ as well. By the construction of the keys and values matrices, as in the proof of Lemma B.4, it holds that the attention head puts all its attention for query $\mathbf{q}^{\ell}_{t-1}$ on the key $\mathbf{k}^{\ell}_{t-2}$. This means that

$$\mathbf{a}^{\ell}_{t-1} = \mathbf{x}^{\ell-1}_{t-1} + \mathbf{v}^{\ell}_{t-2}. \tag{65}$$

By the induction hypothesis, we have that

$$\mathbf{x}^{\ell-1}_{t-1} = \begin{pmatrix} [\![y_{t-1}]\!] \\ [\![y_{t-2}]\!] \\ [\![y_{t-3}]\!] \\ \vdots \\ [\![y_{t-1-(\ell-1)}]\!] \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} \tag{66}$$

$$\mathbf{x}^{\ell-1}_{t-2} = \begin{pmatrix} [\![y_{t-2}]\!] \\ [\![y_{t-3}]\!] \\ [\![y_{t-4}]\!] \\ \vdots \\ [\![y_{t-2-(\ell-1)}]\!] \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix}. \tag{67}$$

Furthermore, by definition of $\mathbf{V}$, we have that

$$\mathbf{v}^{\ell}_{t-2} = V\left(\mathbf{x}^{\ell-1}_{t-2}\right) \tag{68a}$$

$$= \mathbf{V}\mathbf{x}^{\ell-1}_{t-2} \tag{68b}$$

$$= \begin{pmatrix} \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ [\![y_{t-2-(\ell-1)}]\!] \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} \tag{68c}$$

Inserting this into Eq. (65), we get that $\mathbf{a}_{t-1}^{\ell}$ satisfies the required equality:

$$\mathbf{a}_{t-1}^{\ell} = \mathbf{x}_{t-1}^{\ell-1} + \mathbf{v}_{t-2}^{\ell} \tag{69a}$$

$$= \begin{pmatrix} \llbracket y_{t-1} \rrbracket \\ \vdots \\ \llbracket y_{t-1-(\ell-1)} \rrbracket \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} + \begin{pmatrix} \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \llbracket y_{t-2-(\ell-1)} \rrbracket \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} = \begin{pmatrix} \llbracket y_{t-1} \rrbracket \\ \vdots \\ \llbracket y_{t-1-(\ell-1)} \rrbracket \\ \llbracket y_{t-2-(\ell-1)} \rrbracket \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} = \begin{pmatrix} \llbracket y_{t-1} \rrbracket \\ \vdots \\ \llbracket y_{t-1-(\ell-1)} \rrbracket \\ \llbracket y_{t-1-\ell} \rrbracket \\ \mathbf{0}_{|\Sigma|} \\ \vdots \\ \mathbf{0}_{|\Sigma|} \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \end{pmatrix} \tag{69b}$$

Since the computation of $\mathbf{x}_{t-1}^{\ell} \stackrel{\text{def}}{=} \mathbf{z}_{t-1}^{\ell} = O\left(\mathbf{a}_{t-1}^{\ell}\right) + \mathbf{a}_{t-1}^{\ell} = \mathbf{a}_{t-1}^{\ell}$ with the definition of O as the zero function gives us $\mathbf{z}_{t-1}^{\ell} = \mathbf{a}_{t-1}^{\ell}$, we get the required equality, which finishes the proof. ■

This allows us to prove the following theorem.

Theorem 3.2. *For any n-gram LM, there exists a weakly equivalent $(n-1)$-layer hard attention transformer LM with a single head.*

Proof. Lemma B.5 shows that the $n-1$-layer transformer can identify the history of interest. Applying Lemma B.2 and the same derivation as in the proof of Theorem 3.1 shows that we can construct a weakly equivalent hard attention transformer. ■

B.2.5 Simulation with a Single Layer and a Single Head: Intuition

While the construction presented here is considerably less intuitive than that of Theorem 3.1, the steps of the proof remain the same—they include the identification of the individual symbols and their positions in the history, combining them into the one-hot encoding of the entire history, and using that to compute the correct next-symbol conditional distributions. This proof focuses on encoding the entire history of interest $\boldsymbol{y}_{t-n+1}^{t-1}$ into a single vector in a way that can be decoded to index the conditional probability distribution as in Definition 2.11. This can then be used to index the appropriate conditional probability distributions as in the proof of Theorem 3.1.

Again, we outline the intuition first. Fix $t \leqslant |\boldsymbol{y}|$. We decompose the computation of the representation $\mathsf{enc}\left(\boldsymbol{y}_{<t}\right)$ constructed by a single-layer-single-head transformer network as follows.

1. Attending exactly to the history of interest with a single head and a single layer. This can be done by assigning the same attention *score* with the scoring function to all positions within the history $\boldsymbol{y}_{t-n+1}^{t}$ and a lower score to all other positions. Keeping the definitions of Q and V from Theorem 3.1, we can achieve that by defining

$$f\left(\mathbf{q}_{t-1}, \mathbf{k}_j\right) = -\mathrm{ReLU}(\langle \mathbf{q}_{t-1}, \mathbf{k}_j \rangle) \tag{70}$$

which assigns positions in the history score 0 and others negative scores. This is illustrated in Fig. 3. Concretely, the scoring function f together with hard attention results in attention weights of the form

$$s_j = \frac{1}{n-1} \mathbb{1}\left\{j \geqslant t-n+1\right\}. \tag{71}$$

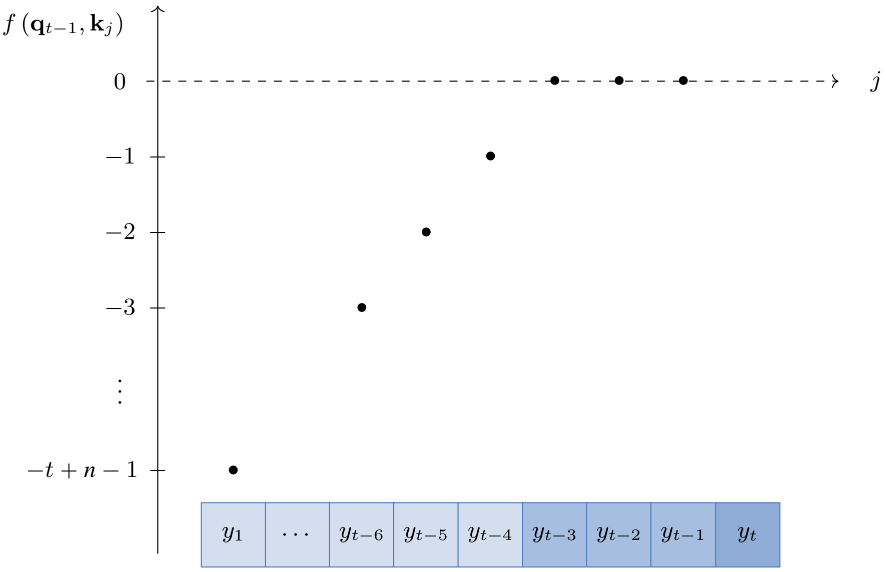

Figure 3: An illustration of the attention scores in a single-layer-single-head transformer network for a history $\boldsymbol{y}_{t-3:t-1} = y_{t-3}y_{t-2}y_{t-1}$ when determining the conditional distribution $p(y_t \mid \boldsymbol{y}_{<t})$.

2. Storing the order of the symbols in the history. We will define the position-augmented representations as position-scaled one-hot encodings of the input symbols. In particular, define

$$\mathbf{r}(y_{t-1}, j) \overset{\text{def}}{=} \begin{pmatrix} 10^{-j} \cdot [\![y_{t-1}]\!] \\ t \\ 1 \end{pmatrix} \in \mathbb{R}^{|\Sigma|+2} \tag{72}$$

This effectively stores the information about both the position of the current symbol (with the magnitude) as well as the identity of the symbol y_j (with the index of the non-zero entry). Unlike in the multi-head or multi-layer case, note that in this case, the function $\mathbf{r}$ is not a concatenation (or addition) of two separate components (one for symbols and one for positions).

Ignoring residual connections, Eq. (71) then implies that

$$\mathbf{a}_{t-1} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j} [\![y_j]\!]. \tag{73}$$

For simplicity, we now write $\mathbf{a} \overset{\text{def}}{=} \mathbf{a}_{t-1}$. The individual entries of $\mathbf{a}$ will correspond to symbols in $\underline{\Sigma}$. The *digits* of these symbols will encode the positions of the symbols in the history. Concretely, $\mathbf{a}$ will have a non-zero digit in the i^{th} position if and only if the symbol y appears in the string $\boldsymbol{y}_{<t}$ at position i for $i \in [t-1]$. For example, in a 5-gram LM over the alphabet $\Sigma = \{a, b\}$, the contextual representation $\mathbf{a}$ for the history $\boldsymbol{y}_{t-4:t-1} = abaa$ will be

$$\mathbf{a} = \frac{1}{4} \begin{pmatrix} 10^{t-1} + 10^{t-3} + 10^{t-4} \\ 10^{t-2} \end{pmatrix} = \frac{1}{4} 10^{-t} \begin{pmatrix} 0.1011 \\ 0.0100 \end{pmatrix}. \tag{74}$$

Such representations therefore uniquely encode the history of interest.

Decoding the representations of the history. The representations $\mathbf{a}$, therefore, contain the information about the history of interest compactly represented in a $|\underline{\Sigma}|$-dimensional vector $\mathbf{a}$. We now want to construct a function that transforms the constructed vector $\mathbf{a}$ into a one-hot encoding of the history. To make $\mathbf{a}$ invariant with respect to t, we first scale it by $\frac{n-1}{10^{n-t}}$ and define

$$\mathbf{a}' = \frac{n-1}{10^{n-t}} \mathbf{a}. \tag{75}$$

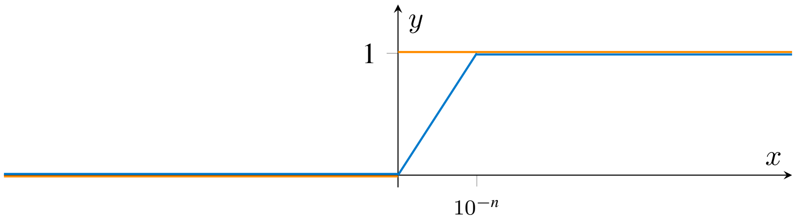

Figure 4: The step function $\mathbb{1}\left\{x\right\}$ and the approximated step function that matches $\mathbb{1}\left\{x\right\}$ outside of $[0, 10^{-n}]$.

Then

$$a'_y = \sum_{i=1}^{n-1} \mathbb{1}\left\{y = y_{i+t-n}\right\} 10^{-i}. \tag{76}$$

In words, the $|\Sigma|$ entries of $\mathbf{a}'$ are of the form $a'_y = 0.d_1 \ldots d_{n-1}$, where $d_i = 1$ if and only if y appears in the history $\boldsymbol{y}_{<t}$ at position $t - n + i$.

We now focus on a specific symbol $y \in \Sigma$ with the entry a'_y in the vector $\mathbf{a}'$ and decode it into a vector that can be used to construct a one-hot encoding of the relevant history $\boldsymbol{y}_{t-n+1}^{t-1}$ with F. The "decoding function" will take the form of a $n - 1$-layer ReLU-activated MLP. Intuitively, each of the $n - 1$ layers will contain $|\Sigma|$ neurons, where each of them will compute the values d_i for a particular $y \in \Sigma$. Now, d_1 can be computed as

$$d_1 = \mathbb{1}\left\{10^1 \cdot a_y - 1 + 10^{-n} > 0\right\}. \tag{77}$$

Since $\mathbb{1}\left\{\cdot\right\}$ is not a continuous function, and, unlike in Appendix B.1, the arguments can now take arbitrary real values, it cannot be implemented using a composition of ReLU functions. We can, however, simulate the discontinuous function with a linear combination of two ReLU functions that together define the same function as $\mathbb{1}\left\{\cdot\right\}$ on a subset of $\mathbb{R}$ relevant for our purposes. Notice that, as long as $d_1 = 1$, we have that $10^1 \cdot a_y - 1 + 10^{-n} \geqslant 10^{-n}$ while we have $10^1 \cdot a_y - 1 + 10^{-n+1} = 10^1 \cdot 0 - 1 + 10^{-n} < 0$ if $d_1 = 0$. This means that our approximation of $\mathbb{1}\left\{\cdot\right\}$ only has to map values greater or equal to 10^{-n} to 1, rather than all positive values. This allows us to continuously transition from 0 to 1 as the input to the ReLU function increases from 0 to 10^{-n}. Such a piecewise linear approximation of $\mathbb{1}\left\{\cdot\right\}$ can be easily implemented by a linear combination of ReLU functions, i.e., with an MLP. See Fig. 4 for an illustration of the approximation.

d_2 can then be computed as

$$d_2 = \mathbb{1}\left\{10^2 \cdot a_y - 10^1 d_1 - 1 + 10^{-n} > 0\right\} \tag{78}$$

and, in general,

$$d_i = \mathbb{1}\left\{10^i \cdot a_y - \sum_{j=1}^{i-1} 10^{i-j} d_j - 1 + 10^{-n} > 0\right\}. \tag{79}$$

The computation of d_i requires i layers (since d_j for $j < i$ have to be computed first), meaning that $n - 1$ layers are required in total. Altogether, these layers compute the values d_i for a single $y \in \Sigma$. Replicating this computation for every $y \in \Sigma$ and concatenating the results gives us the desired contextual representation $\mathbf{z}$.

With this construction, it holds for every $y \in \Sigma$ that $d_i = 1$ if and only if the symbol y appears in the history $\boldsymbol{y}_{t-n+1}^{t-1}$ at position $t - n + i$. This, therefore, gives us enough information to reconstruct the multi-hot encoding of the history of interest. As in Theorem 3.1, this can then be converted into a one-hot encoding using another ReLU layer to implement the logical `AND` operation. This intuition is formalized in the following section.

B.2.6 Simulation with a Single Layer and a Single Head: Proofs

We define the following parameters of the transformer head.[15]

[15] For simplicity, we ignore residual connections in this section since we do not require them and the omission makes the presentation cleaner. By duplicating the representations as in Theorem 3.1, residual connections could easily be added back to

- Static encodings

$$\mathbf{r}\left(y_{t-1}, j\right) \stackrel{\text{def}}{=} \begin{pmatrix} 10^{-j} \cdot [\![y_{t-1}]\!] \\ 1 \\ t \end{pmatrix} \in \mathbb{R}^{|\Sigma|+3} \tag{80}$$

- Query, key, value, and output transformations

$$Q\left(\mathbf{x}\right) \stackrel{\text{def}}{=} \mathbf{Qx} + \mathbf{b}_Q, \qquad \mathbf{Q} \in \mathbb{R}^{2\times(|\Sigma|+2)}, \mathbf{b}_Q \in \mathbb{R}^2, \tag{81a}$$

$$K\left(\mathbf{x}\right) \stackrel{\text{def}}{=} \mathbf{Kx}, \qquad \mathbf{K} \in \mathbb{R}^{2\times(|\Sigma|+2)}, \tag{81b}$$

$$V\left(\mathbf{x}\right) \stackrel{\text{def}}{=} \mathbf{Vx}, \qquad \mathbf{V} \in \mathbb{R}^{|\Sigma|\times(|\Sigma|+2)}, \tag{81c}$$

$$O\left(\mathbf{x}\right) \stackrel{\text{def}}{=} \mathbf{I}_{|\Sigma|}\mathbf{x}, \tag{81d}$$

$$\mathbf{Q}_{:,|\Sigma|+1:|\Sigma|+2} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{82a}$$

$$\mathbf{b}_Q \stackrel{\text{def}}{=} \begin{pmatrix} -\left(n-2\right) \\ 0 \end{pmatrix} \tag{82b}$$

$$\mathbf{K}_{:,|\Sigma|+1:|\Sigma|+2} \stackrel{\text{def}}{=} \mathbf{I}_2 \tag{82c}$$

$$\mathbf{V}_{:,1:|\Sigma|} \stackrel{\text{def}}{=} \mathbf{I}_{|\Sigma|} \tag{82d}$$

where the unspecified elements of $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ are 0.

A large part of the proof of correctness will rely on identifying the digits of the contextual representations of strings. We will rely heavily on the following definition.

Definition B.2. *Let $x \in \mathbb{Q} \cap \left(10^{-(N+1)}, 10^{-1}\right]$ be a rational-valued number with at most N digits in its decimal representation. We define $d_i\left(x\right)$ as the i^{th} digit of x for $i \in [N]$. We also group these N values into the vector $\mathbf{d}\left(x\right)$:*

$$\mathbf{d}\left(x\right) \stackrel{\text{def}}{=} \begin{pmatrix} d_1\left(x\right) \\ \vdots \\ d_N\left(x\right) \end{pmatrix}. \tag{83}$$

Lemma B.6. *A transformer with the parameters and functions defined in Eq. (80)–Eq. (82d) computes for string $\boldsymbol{y} \in \Sigma^*$*

$$\mathbf{a}_{t-1} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j} [\![y_j]\!]. \tag{84}$$

Proof. By construction, we have

$$\mathbf{q}_{t-1} = \begin{pmatrix} t-1-\left(n-2\right) \\ -1 \end{pmatrix} \tag{85a}$$

$$\mathbf{k}_j = \begin{pmatrix} 1 \\ j \end{pmatrix} \tag{85b}$$

$$\mathbf{v}_j = 10^{-j} \cdot [\![y_{t-1}]\!], \tag{85c}$$

make this setting closer to the general transformer setting.

thus

$$
\begin{aligned}
f\left(\mathbf{q}_{t-1}, \mathbf{k}_j\right) &= -\mathrm{ReLU}(\langle \mathbf{q}_{t-1}, \mathbf{k}_j \rangle) \tag{86a}\\
&= -\mathrm{ReLU}\left(\left\langle \begin{pmatrix} t-1-(n-2) \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ j \end{pmatrix} \right\rangle\right) \tag{86b}\\
&= -\mathrm{ReLU}(t-1-n+2-j) \tag{86c}\\
&= -\mathrm{ReLU}(t-n+1-j) \tag{86d}\\
&= \begin{cases} 0 & \textbf{if } j \geqslant t-n+1 \\ < 0 & \textbf{otherwise} \end{cases} \tag{86e}
\end{aligned}
$$

meaning that

$$
s_j = \frac{1}{n-1} \mathbb{1}\left\{j \geqslant t-n+1\right\}. \tag{87}
$$

This means that

$$
\mathbf{a}_{t-1} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} \mathbf{v}_j = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j} [\![y_j]\!] \tag{88}
$$

which is what we needed to prove. ■

Lemma B.7. *Define*

$$
\mathbf{a}' \stackrel{\text{def}}{=} \frac{n-1}{10^{n-t}}\, \mathbf{a}_{t-1} \tag{89}
$$

with $\mathbf{a}_{t-1}$ *from Lemma B.6. Indexing the* $|\underline{\Sigma}|$ *elements of* $\mathbf{a}'$ *directly with* $y \in \underline{\Sigma}$, *it holds that*

$$
d_i\left(a'_y\right) = 1 \iff y_{t-n+i} = y \tag{90}
$$

for all $i \in [N]$ *and* $d_i\left(x\right) = 0$ *for all* $i > N$.

Proof. By Lemma B.6, $\mathbf{a}$ contains entries of the form

$$
\begin{aligned}
a_y &= \frac{1}{n-1} \sum_{j=t-n+1}^{t-1} \mathbb{1}\left\{y = y_j\right\} 10^{-j} \tag{91a}\\
&= \frac{1}{n-1} \sum_{j'=1}^{t-1-(t-n)} \mathbb{1}\left\{y = y_{j'+t-n}\right\} 10^{-(j'+t-n)} \tag{91b, Change of variables.}\\
&= \frac{1}{n-1} \sum_{j=1}^{n-1} \mathbb{1}\left\{y = y_{j+t-n}\right\} 10^{n-t-j} \tag{91c, Change of variables.}\\
&= \frac{10^{n-t}}{n-1} \sum_{j=1}^{n-1} \mathbb{1}\left\{y = y_{j+t-n}\right\} 10^{-j} \tag{91d, Distributivity.}
\end{aligned}
$$

for $y \in \underline{\Sigma}$. Then

$$
\begin{aligned}
a'_y &\stackrel{\text{def}}{=} \frac{n-1}{10^{n-t}} a_y \tag{92a}\\
&= \frac{n-1}{10^{n-t}} \frac{10^{n-t}}{n-1} \sum_{j=1}^{n-1} \mathbb{1}\left\{y = y_{j+t-n}\right\} 10^{-j} \tag{92b}\\
&= \sum_{j=1}^{n-1} \mathbb{1}\left\{y = y_{j+t-n}\right\} 10^{-j}. \tag{92c}
\end{aligned}
$$

This implies that $d_i\left(a'_y\right) = 1 \iff y_{t-n+i} = y$ for $i \in [N]$ and that $d_i\left(x\right) = 0$ for $i > N$, which is what we wanted to prove. ■

Lemma B.8. *Let $x \in \mathbb{Q} \cap \left(10^{-(N+1)}, 10^{-1}\right]$ with $d_i\left(x\right) \in \{0, 1\}$ for $i \in [N]$. Then, $d_i\left(x\right)$ satisfy the equality*

$$d_i\left(x\right) = \mathbb{1}\left\{10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j\left(x\right) - 1 + 10^{-(N+1)} > 0\right\}. \tag{93}$$

for all $i \in [N]$.

Proof. Let $x \in \mathbb{Q} \cap \left(10^{-(N+1)}, 10^{-1}\right]$ with $d_i\left(x\right) \in \{0, 1\}$ for $i \in [N]$. Then, by definition of $d_i\left(x\right)$, we have that

$$x = \sum_{j=1}^{N} d_j\left(x\right) 10^{-j}. \tag{94}$$

This means that

$$\begin{aligned} 10^i x &= 10^i \sum_{j=1}^{N} d_j\left(x\right) 10^{-j} && \text{(95a)}\\ &= \sum_{j=1}^{N} d_j\left(x\right) 10^{i-j} && \text{(95b)}\\ &= \sum_{j=1}^{i-1} d_j\left(x\right) 10^{i-j} + d_i\left(x\right) + \sum_{j=i+1}^{N} d_j\left(x\right) 10^{i-j}, && \text{(95c)} \end{aligned}$$

implying that

$$10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j\left(x\right) = d_i\left(x\right) + \sum_{j=i+1}^{N} d_j\left(x\right) 10^{i-j}. \tag{96}$$

Suppose now that $d_i\left(x\right) = 1$. Then

$$\begin{aligned} 10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j\left(x\right) - 1 + 10^{-(N+1)} &= d_i\left(x\right) + \sum_{j=i+1}^{N} d_j\left(x\right) 10^{i-j} - 1 + 10^{-(N+1)} && \text{(97a)}\\ &= 1 + \sum_{j=i+1}^{N} d_j\left(x\right) 10^{i-j} - 1 + 10^{-(N+1)} && \text{(97b)}\\ &= \sum_{j=i+1}^{N} d_j\left(x\right) 10^{i-j} + 10^{-(N+1)} > 0, && \text{(97c)} \end{aligned}$$

meaning that

$$\mathbb{1}\left\{10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j\left(x\right) - 1 + 10^{-(N+1)}\right\} = 1 = d_i\left(x\right). \tag{98}$$

Suppose, on the contrary, that $d_i(x) = 0$. Then

$$\begin{aligned}
10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j(x) - 1 + 10^{-(N+1)} &= d_i(x) + \sum_{j=i+1}^{N} d_j(x) 10^{i-j} - 1 + 10^{-(N+1)} && \text{(99a)}\\
&= 0 + \sum_{j=i+1}^{N} d_j(x) 10^{i-j} - 1 + 10^{-(N+1)} && \text{(99b)}\\
&= \sum_{j=i+1}^{N} d_j(x) 10^{i-j} + 10^{-(N+1)} - 1 && \text{(99c)}\\
&= \sum_{j'=1}^{N-(i+1)} d_{j'+i+1}(x) 10^{i-j'-i-1} + 10^{-(N+1)} - 1 && \text{(99d)}\\
&= \underbrace{\sum_{j'=1}^{N-(i+1)} d_{j'+i+1}(x) 10^{-j'-1} + 10^{-(N+1)}}_{<1} - 1 < 0 && \text{(99e)}
\end{aligned}$$

meaning that

$$\mathbb{1}\left\{10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j(x) - 1 + 10^{-(N+1)}\right\} = 0 = d_i(x). \tag{100}$$

This finishes the proof. ■

Lemma B.9. *Let $x \in \mathbb{Q} \cap \left(10^{-(N+1)}, 10^{-1}\right]$ with $d_i(x) \in \{0, 1\}$ for $i \in [N]$. Given $d_j(x)$ for $j < i$, $d_i(x)$ can be computed by a single layer MLP.*

Proof. By Lemma B.8, we can use the knowledge of $d_j(x)$ for $j < i$, $d_i(x)$ to implement the function

$$d_i(x) = \mathbb{1}\left\{10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j(x) - 1 + 10^{-(N+1)} > 0\right\}. \tag{101}$$

with an MLP. For any $i \in [N]$, the inner function

$$\begin{aligned}
\begin{pmatrix} d_1(x) \\ \vdots \\ d_{i-1}(x) \\ x \end{pmatrix} &\mapsto 10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j(x) - 1 + 10^{-(N+1)} && \text{(102a)}\\
&= \left\langle \underbrace{\begin{pmatrix} -10^{i-1} \\ \vdots \\ -10^1 \\ 10^i \end{pmatrix}}_{\mathbf{W}^\top}, \begin{pmatrix} d_1(x) \\ \vdots \\ d_{i-1}(x) \\ x \end{pmatrix} \right\rangle \underbrace{-1 + 10^{-(N+1)}}_{\mathbf{b}} && \text{(102b)}
\end{aligned}$$

is an affine transformation. The indicator function in Eq. (101), however, is discontinuous and can thus not be implemented by a composition of continuous ReLU MLPs. Here, we take advantage of the fact that

$$10^i \cdot x - \sum_{j=1}^{i-1} 10^{i-j}\, d_j(x) - 1 + 10^{-(N+1)} \in \underbrace{(-\infty, 0] \cup \left[10^{-(N+1)}, \infty\right)}_{\mathcal{I}}. \tag{103}$$

The MLP

$$\mathrm{MLP}_{\mathcal{I}}(z) = 10^{N+1}\left(\mathrm{ReLU}(z) - \mathrm{ReLU}\left(z - 10^{-(N+1)}\right)\right) \tag{104a}$$

$$= \begin{pmatrix}10^{N+1} & -10^{N+1}\end{pmatrix}\mathrm{ReLU}\left(\begin{pmatrix}1\\1\end{pmatrix}z + \begin{pmatrix}0\\-10^{-(N+1)}\end{pmatrix}\right) \tag{104b}$$

matches $\mathbb{1}\{\cdot > 0\}$ on $\mathcal{I}$, as we show next. See Fig. 4 for an illustration of the approximation. First, assume that $z \leqslant 0$. Then

$$\mathrm{MLP}_{\mathcal{I}}(z) = 10^{N+1}\left(\mathrm{ReLU}(z) - \mathrm{ReLU}\left(z - 10^{-(N+1)}\right)\right) = 10^{N+1}(0-0) = 0 = \mathbb{1}\{0 > 0\}. \tag{105}$$

On the contrary, assuming $z \geqslant 10^{-(N+1)}$, we have that

$$\mathrm{MLP}_{\mathcal{I}}(z) = 10^{N+1}\left(\mathrm{ReLU}(z) - \mathrm{ReLU}\left(z - 10^{-(N+1)}\right)\right) \tag{106a}$$

$$= 10^{N+1}\left(z - \left(z - 10^{-(N+1)}\right)\right) \tag{106b}$$

$$= 10^{N+1}\left(z - z + 10^{-(N+1)}\right) \tag{106c}$$

$$= 10^{N+1}\left(10^{-(N+1)}\right) \tag{106d}$$

$$= 1 = \mathbb{1}\{z > 0\}. \tag{106e}$$

We can therefore construct the MLP MLP computing Eq. (101) on $\mathcal{I}$ by a composition of Eq. (102b) (computing z in Eq. (104b)) and the MLP $\mathrm{MLP}_{\mathcal{I}}$ from Eq. (104b). ■

Lemma B.10. *Let $N \in \mathbb{N}$. There exists an MLP* MLP *such that*

$$\mathrm{MLP}(x) = \mathbf{d}(x) \tag{107}$$

for all $x \in \mathbb{Q} \cap \left(10^{-(N+1)}, 10^{-1}\right]$ with $d_i(x) \in \{0,1\}$ for $i \in [N]$.

Proof. At a very high level, we will construct an N-layer MLP performing the transformations

$$x \mapsto \begin{pmatrix}x\\x\\\vdots\\x\end{pmatrix} \mapsto \begin{pmatrix}d_1(x)\\x\\\vdots\\x\end{pmatrix} \mapsto \begin{pmatrix}d_1(x)\\d_2(x)\\\vdots\\x\end{pmatrix} \mapsto \cdots \mapsto \begin{pmatrix}d_1(x)\\d_2(x)\\\vdots\\d_N(x)\end{pmatrix} = \mathbf{d}(x). \tag{108}$$

By Lemma B.9, all individual transformations can be performed exactly by a single-layer MLP. The composition of the N layers results in the vector $\mathbf{d}(x)$.

The transformation $x \mapsto \begin{pmatrix}x\\x\\\vdots\\x\end{pmatrix}$ is a simple linear transformation. We now construct the ℓ^{th} layer $\boldsymbol{f}_\ell$ of the MLP with parameters $\mathbf{W}_\ell$ and $\mathbf{b}_\ell$ (cf. Definition B.1), assuming that it has the input $\begin{pmatrix}d_1(x)\\d_2(x)\\\vdots\\d_{\ell-1}(x)\\x\\\vdots\\x\end{pmatrix}$. The layer $\boldsymbol{f}_\ell$ has to satisfy

$$\boldsymbol{f}\left(\begin{pmatrix} d_1(x) \\ d_2(x) \\ \vdots \\ d_{\ell-1}(x) \\ x \\ x \\ \vdots \\ x \end{pmatrix}\right) = \begin{pmatrix} d_1(x) \\ d_2(x) \\ \vdots \\ d_{\ell-1}(x) \\ d_\ell(x) \\ x \\ \vdots \\ x \end{pmatrix}, \tag{109}$$

i.e., it must copy all $n-1$ entries apart from the ℓ^{th} one. Thus, $\mathbf{W}_{k,k'} = \mathbb{1}\{k = k'\}$ for $k \neq \ell$ and $\mathbf{b}_k = 0$ for all $k \neq \ell$ (here, we write $\mathbf{W}$ and $\mathbf{b}$ for $\mathbf{W}_\ell$ and $\mathbf{b}_\ell$ to avoid clutter). To define the remaining ℓ^{th} row, we use Lemma B.9. It tells us that defining

$$\mathbf{W}_{\ell,:} \overset{\text{def}}{=} \begin{pmatrix} 10^{\ell-1} & \cdots & 10^1 & 10^\ell & 0 & \cdots & 0 \\ 10^{\ell-1} & \cdots & 10^1 & 10^\ell & 0 & \cdots & 0 \end{pmatrix} \tag{110a}$$

$$b_\ell \overset{\text{def}}{=} \begin{pmatrix} -1+10^{-n} \\ -1+10^{-n}-10^{-n} \end{pmatrix} = \begin{pmatrix} -1+10^{-n} \\ -1 \end{pmatrix} \tag{110b}$$

will result in the ℓ^{th} row of $\boldsymbol{f}_\ell$ computing exactly $d_\ell(x)$ after being multiplied by the matrix

$$\mathbf{W}' \overset{\text{def}}{=} \begin{pmatrix} 10^n & 10^n \end{pmatrix}. \tag{111}$$

This is what Eq. (109) requires. The parameters $\mathbf{W}_{\ell,:}$ and b_ℓ represent the parameters of the affine transformation in Lemma B.9. Note that the matrix $\mathbf{W}'$ is not part of the original definition of the MLP. However, it can easily be absorbed into the matrix $\mathbf{W}_{\ell+1}$ in the actual implementation (at the cost of duplicating the size of the hidden state). We keep it here to make the presentation more intuitive. Since any layer $\boldsymbol{f}_\ell$ can be defined like this, and the final MLP is their composition, this finishes the proof. ■

Lemma B.11. *Given* $\mathbf{a}_{t-1}$ *from Lemma B.6, it holds that*

$$\|\mathbf{a}_{t-1}\|_1 = \frac{10^{n-1-t}}{n-1}\frac{1-\frac{1}{10^{n-1}}}{1-\frac{1}{10}}. \tag{112}$$

Proof. By Lemma B.7, we have that

$$\mathbf{a} \overset{\text{def}}{=} \mathbf{a}_{t-1} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j} [\![y_j]\!] \tag{113}$$

We compute:

$$\|\mathbf{a}\|_1 = \frac{1}{n-1} \sum_{j=t-n+1}^{t-1} 10^{-j} \tag{114a}$$

$$= \frac{1}{n-1} \sum_{j'=0}^{t-1-(t-n+1)} 10^{-(j'+t-n+1)} \tag{114b}$$

$$= \frac{1}{n-1} 10^{-(t-n+1)} \sum_{j=0}^{n-2} \frac{1}{10^j} \tag{114c}$$

$$= \frac{10^{-(t-n+1)}}{n-1} \frac{1-\frac{1}{10^{n-1}}}{1-\frac{1}{10}} \tag{114d}$$

$$= \frac{10^{n-1-t}}{n-1}\frac{1-\frac{1}{10^{n-1}}}{1-\frac{1}{10}}. \tag{114e}$$

■

Corollary B.1. *Given* $\mathbf{a} \overset{\text{def}}{=} \mathbf{a}_{t-1}$ *from Lemma B.6 and* $\mathbf{a}' \overset{\text{def}}{=} \mathbf{a}'_{t-1}$ *from Lemma B.7, it holds that*

$$\frac{1}{\|\mathbf{a}\|_1} \cdot 10 \cdot \frac{1 - \frac{1}{10^{n-1}}}{1 - \frac{1}{10}} \cdot \mathbf{a} = \mathbf{a}'. \tag{115}$$

Proof. By Lemma B.11, we can write

$$\|\mathbf{a}\|_1 = \frac{10^{n-1-t}}{n-1} \frac{1 - \frac{1}{10^{n-1}}}{1 - \frac{1}{10}} = \frac{10^{n-t}}{n-1} \cdot Z \tag{116}$$

and

$$\frac{1}{\|\mathbf{a}\|_1} = \frac{n-1}{10^{n-t}} \cdot \frac{1}{Z} \tag{117}$$

where $Z \overset{\text{def}}{=} \frac{1}{10} \cdot \frac{1-\frac{1}{10^{n-1}}}{1-\frac{1}{10}}$ is a constant independent of t. Then

$$\begin{aligned}
\frac{1}{\|\mathbf{a}\|_1} \cdot \frac{1}{10} \cdot \frac{1 - \frac{1}{10^{n-1}}}{1 - \frac{1}{10}} \cdot \mathbf{a} &= \frac{n-1}{10^{n-t}} \cdot \frac{1}{Z} \cdot \underbrace{\frac{1}{10} \cdot \frac{1 - \frac{1}{10^{n-1}}}{1 - \frac{1}{10}}}_{Z} \cdot \mathbf{a} && \text{(118a)} \\
&= \frac{n-1}{10^{n-t}} \cdot \frac{1}{Z} \cdot Z \cdot \mathbf{a} && \text{(118b)} \\
&= \frac{n-1}{10^{n-t}} \cdot \mathbf{a} && \text{(118c)} \\
&= \mathbf{a}' && \text{(118d)}
\end{aligned}$$

■

Lemma B.12. *Let* Σ *be an alphabet. Given the transformer parameters and functions defined in Eq. (80)–Eq. (82d), there exists an MLP* F *(whose inputs are* $\|\cdot\|_1$*-normalized) that maps the contextual representations* $\mathbf{a}$ *of* $\boldsymbol{y}_{<t}$ *into a one-hot encoding of* $\boldsymbol{y}_{t-n+1}^{t-1}$ *for any string* $\boldsymbol{y} \in \Sigma^*$ *and any* $t \in [|\boldsymbol{y}|]$.

Proof. By Lemma B.7, we have that

$$\mathbf{a} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j} [\![y_j]\!] \tag{119}$$

and that $d_i\left(a'_y\right) = 1 \iff y_{t-n+i} = y$ for $\mathbf{a}' \overset{\text{def}}{=} \frac{n-1}{10^{n-t}}\, \mathbf{a}$ (where $a'_i = 0$ for $i > n-1$, from the same lemma). We can express the entries of the one-hot encoding of the history $\boldsymbol{y}_{t-n+1}^{t-1}$, $[\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]$, as

$$[\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]_{y_{t-n+1}\cdots y_{t-1}} = 1 \iff d_1\left(a'_{y_{t-n+1}}\right) \wedge \cdots \wedge d_{n-1}\left(a'_{y_{t-1}}\right). \tag{120}$$

By Lemma B.10, the vectors $\mathbf{d}\left(a'_{y_{t-n+1}}\right), \ldots, \mathbf{d}\left(a'_{y_{t-1}}\right)$ can be computed by an $n-1$-layer MLP. Each of these vectors is of size $n-1$ and, among others, contains the values $d_1\left(a'_{y_{t-n+1}}\right), \ldots, d_{n-1}\left(a'_{y_{t-1}}\right)$. Since the entries $[\![\boldsymbol{y}_{t-n+1}^{t-1}]\!]_{y_{t-n+1}\cdots y_{t-1}}$ can be expressed as the results of the logical AND operation, their computation can be performed by a single-layer ReLU MLP as per Lemma B.1. The MLP F can therefore be constructed as a composition of three functions:

1. The scaling $\mathbf{a} \mapsto \frac{1}{\|\mathbf{a}\|_1} \cdot \frac{1}{10} \cdot \frac{1-\frac{1}{10^{n-1}}}{1-\frac{1}{10}} \cdot \mathbf{a}$, which results in $\mathbf{a}'$ by Corollary B.1.
2. The concatenation of the $|\underline{\Sigma}|$ $n-1$-layer MLPs computing $\mathbf{d}\left(a'_y\right)$ for all $y \in \underline{\Sigma}$. This results in $(n-1)\,|\underline{\Sigma}|$ binary values altogether.

3. The MLP performing the AND operation between the entries of $\mathbf{d}\left(a'_y\right)$.

This finishes the proof. ■

Theorem 3.3. *For any n-gram LM, there exists a weakly equivalent single-layer hard attention transformer LM with a single head.*

Proof. To show that there exists a weakly equivalent single-layer-single-head transformer LM to any n-gram LM, we combine the lemmata in this section. Let $\mathcal{T}$ be a transformer LM defined with the parameters and functions defined in Eq. (80)–Eq. (82d). By Lemma B.6, the representations $\mathbf{a}_{t-1} = \sum_{j=t-n+1}^{t-1} \frac{1}{n-1} 10^{-j} [\![y_j]\!]$ computed by $\mathcal{T}$ contain information about the symbols and their positions in the history $\boldsymbol{y}_{t-n+1}^{t-1}$. By Lemma B.12 then, $\mathbf{a}$ can be mapped to the one-hot encoding of the history with a $n-1$-layer MLP F. This one-hot encoding can then be used to index (the logits of) the probabilities stored in the output matrix $\mathbf{E}$ defining a weakly equivalent transformer. ■

C Sparse Attention

We now prove a lemma analogous to Lemma B.4. It shows that a sparse attention transformer head can isolate a particular symbol in the string. First, define the following position-augmented symbol representation function of the transformer head h:

$$\mathbf{r}\left(y,t\right) \overset{\text{def}}{=} \begin{pmatrix} [\![y]\!] \\ \mathbf{0}_{|\Sigma|} \\ 1 \\ t \end{pmatrix} \in \{0,1\}^{2|\Sigma|+2} \tag{121}$$

and the scoring function

$$f\left(\mathbf{q},\mathbf{k}\right) \overset{\text{def}}{=} -\left|\mathbf{q}^\top \mathbf{k}\right|. \tag{122}$$

Here, the position-augmented symbol representation function $\mathbf{r}$ can again be implemented by concatenating or summing a symbol- and a position-specific component. Lastly, we define the transformation matrices

$$Q\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathbf{Q}\mathbf{x} + \mathbf{b}_Q, \qquad \mathbf{Q} \in \mathbb{R}^{2\times(2|\Sigma|+2)}, \mathbf{b}_Q \in \mathbb{R}^2, \tag{123a}$$

$$K\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathbf{K}\mathbf{x}, \qquad \mathbf{K} \in \mathbb{R}^{2\times(2|\Sigma|+2)}, \tag{123b}$$

$$V\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathbf{V}\mathbf{x}, \qquad \mathbf{V} \in \mathbb{R}^{(2|\Sigma|+2)\times(2|\Sigma|+2)}, \tag{123c}$$

$$O\left(\mathbf{x}\right) \overset{\text{def}}{=} \mathbf{O}\mathbf{x}, \qquad \mathbf{O} \in \mathbb{R}^{(2|\Sigma|+2)\times(2|\Sigma|+2)}, \tag{123d}$$

$$\mathbf{Q}_{:,2|\Sigma|+1:2|\Sigma|+2} \overset{\text{def}}{=} \mathbf{I}_2 \tag{124a}$$

$$\mathbf{b}_Q \overset{\text{def}}{=} \begin{pmatrix} 0 \\ -h \end{pmatrix} \tag{124b}$$

$$\mathbf{K}_{:,2|\Sigma|+1:2|\Sigma|+2} \overset{\text{def}}{=} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{124c}$$

$$\mathbf{V}_{|\Sigma|+1:2|\Sigma|,1:|\Sigma|} \overset{\text{def}}{=} \mathbf{I}_{|\Sigma|} \tag{124d}$$

$$\mathbf{O}_{1:|\Sigma|,1:|\Sigma|} \overset{\text{def}}{=} -\mathbf{I}_{|\Sigma|} \tag{124e}$$

where the unspecified elements of $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ are 0.

Lemma C.1. *Let Σ be an alphabet and $\boldsymbol{y} \in \Sigma^*$. For any $t \in [|\boldsymbol{y}|]$, a transformer head defined with the parameters above outputs*

$$\mathbf{z}_{t-1} = \begin{pmatrix} \mathbf{0}_{|\Sigma|} \\ [\![y_{t-1-h}]\!] \\ \mathbf{0}_2 \end{pmatrix}. \tag{125}$$

In particular, this means that the output $\mathbf{z}_{t-1}$ at time step $t-1$ contains the one-hot encoding of the symbol at position $t-h-1$.

Proof. The construction is largely identical to the one in Theorem 3.1, with one crucial difference: It relies on simpler, but *unbounded* positional encodings and a less-standard, but still easily implementable attention scoring function in the form of an MLP.

It is easy to see that the query and value transformations result in:

$$\mathbf{q}_{t-1} = Q\left(\mathbf{r}\left(y_t, t\right)\right) = \begin{pmatrix} 1 \\ t-1-h \end{pmatrix} \tag{126a}$$

$$\mathbf{k}_j = K\left(\mathbf{r}\left(y_j, j\right)\right) = \begin{pmatrix} -j \\ 1 \end{pmatrix}. \tag{126b}$$

Thus, we get that

$$\begin{aligned} f\left(\mathbf{q}_{t-1}, \mathbf{k}_j\right) &= -\left|\mathbf{q}_{t-1}^\top \mathbf{k}_j\right| & (127a) \\ &= -\left|\begin{pmatrix} 1 \\ t-1-h \end{pmatrix}^\top \begin{pmatrix} -j \\ 1 \end{pmatrix}\right| & (127b) \\ &= -|j-(t-h-1)|. & (127c) \end{aligned}$$

f clearly has a unique maximum at $j^* = t-1-h$. Moreover, by construction, $f\left(\mathbf{q}_t, \mathbf{k}_{j^*}\right) \geqslant f\left(\mathbf{q}_t, \mathbf{k}_j\right)+1$ for any $j \neq j^*$. This is a crucial property of the scoring function and one that allows sparsemax to *uniquely* attend to j^*; by Lemma C.2, it holds that

$$\operatorname{sparsemax}\left(f\left(\mathbf{q}_{t-1}, \mathbf{k}_1\right), \ldots, f\left(\mathbf{q}_{t-1}, \mathbf{k}_{t-1}\right)\right) = \operatorname{hardmax}\left(f\left(\mathbf{q}_{t-1}, \mathbf{k}_1\right), \ldots, f\left(\mathbf{q}_{t-1}, \mathbf{k}_{t-1}\right)\right), \tag{128}$$

meaning that

$$\operatorname{sparsemax}\left(f\left(\mathbf{q}_{t-1}, \mathbf{k}_1\right), \ldots, f\left(\mathbf{q}_{t-1}, \mathbf{k}_{t-1}\right)\right)_j = \mathbb{1}\left\{j = t-1-h\right\} \tag{129}$$

as in the proof of Lemma B.4. This is exactly the same as Eq. (51). Since O and V result in the same vectors as in Lemma B.4, the remainder of the proof is the same as in Lemma B.4. ■

Lemma C.2. *Let* $\mathbf{x} \in \mathbb{R}^D$. *If* $\max_{d=1}^{D} x_d \geqslant \max_{\substack{d=1 \\ d \notin \operatorname{argmax}(\mathbf{x})}}^{D} x_d + 1$, *then* $\operatorname{sparsemax}(\mathbf{x}) = \operatorname{hardmax}(\mathbf{x})$.

Proof. Let $\mathbf{x} \in \mathbb{R}^D$ and let $x_{(1)} \geqslant x_{(2)} \geqslant \ldots \geqslant x_{(D)}$ be the non-increasing entries of $\mathbf{x}$. Due to the additive invariance of the softmax (Martins and Astudillo (2016, Proposition 2)), we can assume that $x_{(1)} = 0$ and $x_{(2)} \leqslant -1$. By Martins and Astudillo (2016, Proposition 1),

$$\operatorname{sparsemax}(\mathbf{x})_d = \max\left(0, x_d - \tau(\mathbf{x})\right), \tag{130}$$

where

$$\tau(\mathbf{x}) \overset{\text{def}}{=} \frac{\sum_{j=1}^{k(\mathbf{x})} x_{(j)} - 1}{k(x)} \tag{131}$$

and

$$k(\mathbf{x}) \overset{\text{def}}{=} \max\left(k \in [D] \mid 1 + k x_{(k)} > \sum_{j=1}^{k} x_{(j)}\right). \tag{132}$$

It suffices to show that $k(\mathbf{x}) = 1$. For $k = 1$, we get

$$1 + 1 \cdot x_{(1)} = 1 + 1 \cdot 0 = 1 = 1 \geqslant 0 = x_{(1)}. \tag{133}$$

For $k = 2$, we get

$$\begin{aligned} 1 + 2 \cdot x_{(2)} &= 1 + x_{(2)} + x_{(2)} && (134a) \\ &\leqslant 1 - 1 + x_{(2)} && (134b) \\ &= x_{(2)} && (134c) \\ &= x_{(1)} + x_{(2)} && (134d) \\ &= \sum_{j=1}^{2} x_{(j)}, && (134e) \end{aligned}$$

meaning that the condition from Eq. (132) is not fulfilled for $k = 2$. This implies that $k(\mathbf{x}) = 1$ and $\tau(\mathbf{x}) = x_{(1)} - 1$. Thus, we get that

$$\text{sparsemax}(\mathbf{x})_{(1)} = \max\left(0, x_{(1)} - x_{(1)} + 1\right) = 1 \tag{135}$$

and

$$\text{sparsemax}(\mathbf{x})_{(d)} = \max\left(0, x_{(d)} - x_{(1)} + 1\right) = \max\left(0, \underbrace{x_{(1)}}_{\leqslant -1} -0 + 1\right) = 0 \tag{136}$$

for $d > 1$. ■

This allows us to prove the main theorem for sparse-attention transformer LMs.

Theorem 4.1. *For any n-gram LM, there exists a weakly equivalent single-layer sparse attention transformer LM with $n - 1$ heads.*

Proof. Lemma C.1 shows how individual heads of the transformer can identify the symbols in the position of interest. $n - 1$ of them can identify the entire history. The proof then follows the same reasoning as that of Theorem 3.1. ■

Adapting the same proof strategy to Theorem 3.2 would naturally result in an analogous result for $n - 1$ layers and a single head.

Notice that Lemma C.1 requires different and less standard positional encodings, which are, crucially, unbounded. Constructing a sparse attention transformer with bounded positional encodings seems more difficult; the contextual representations would in that case either converge or be non-unique with $t \to \infty$ and since the sparsemax always contracts (Martins and Astudillo, 2016, Proposition 2), attending to individual positions would be difficult. While the positional encodings and the scoring function used in the proof of Lemma C.1 are somewhat less standard than those used in Lemma B.4, similar positional encodings and the same scoring function have been used in theoretical analyses before and even in practical implementations (Pérez et al., 2021).