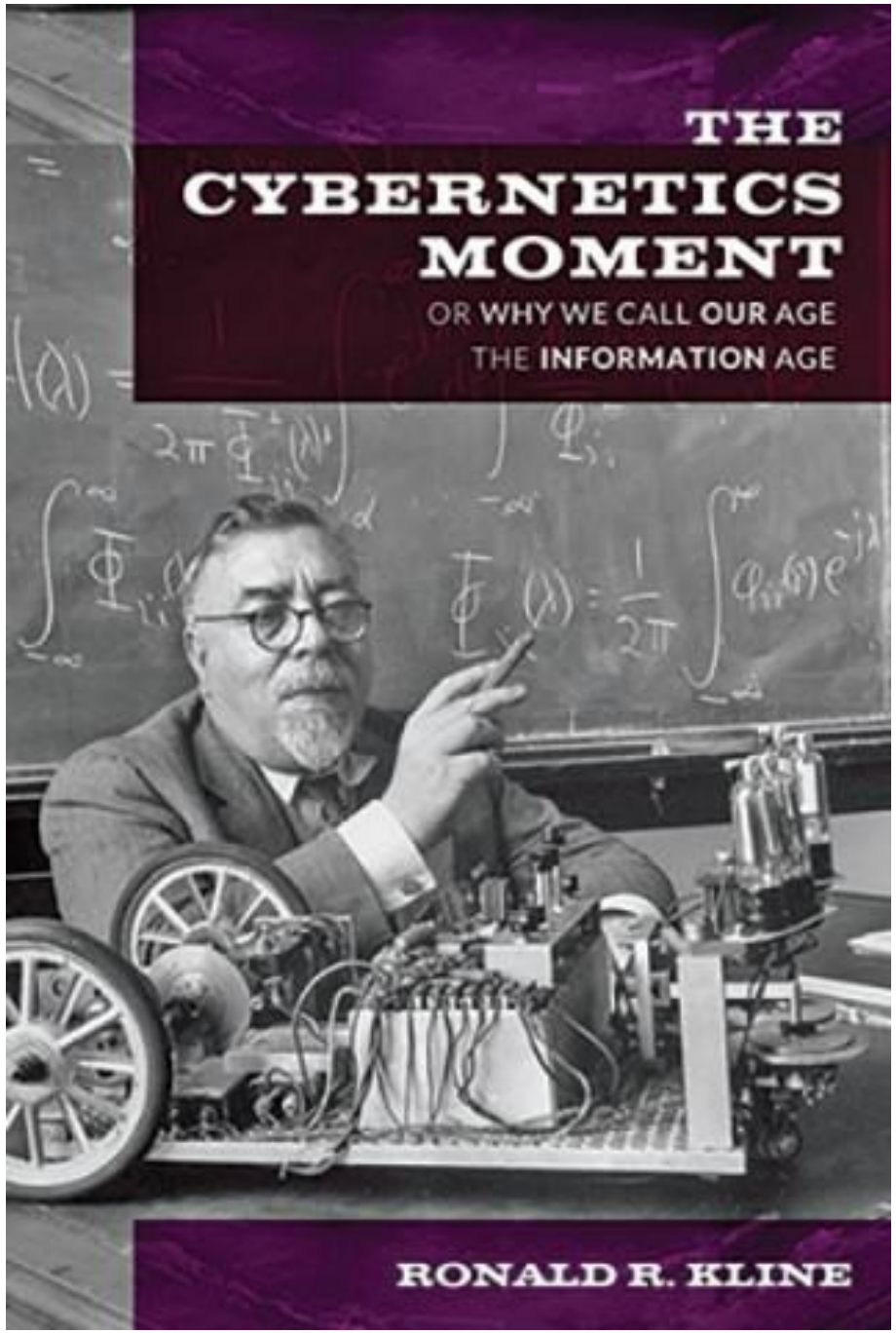
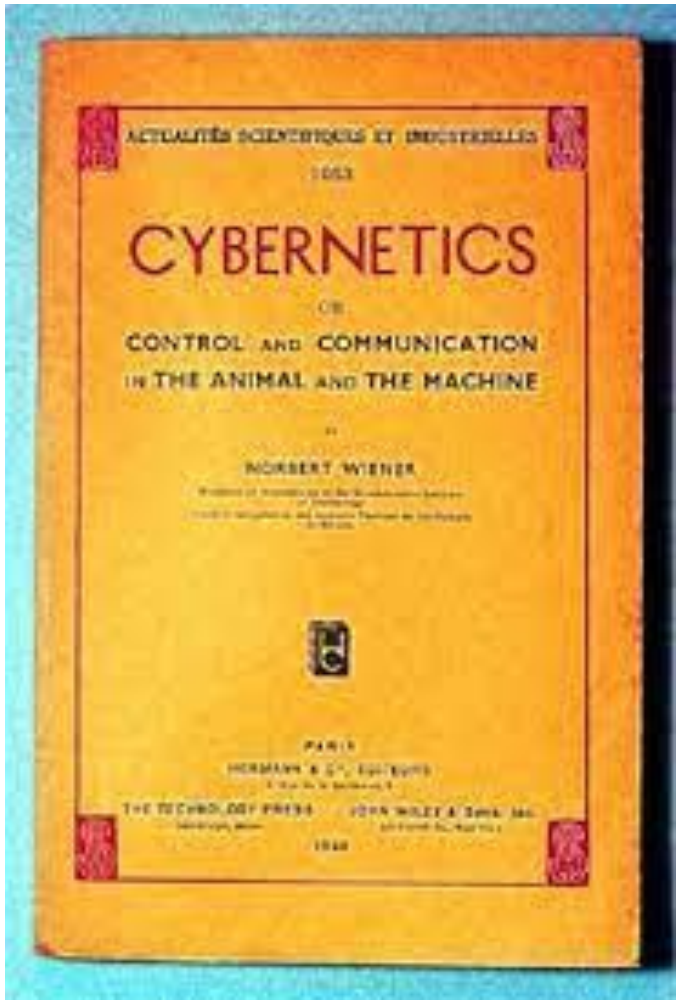
Representation of Information
ECE 598 LV – Lecture 22

Lav R. Varshney

9 April 2024


THE
CYBERNETICS
MOMENT
OR WHY WE CALL OUR AGE
THE INFORMATION AGE
RONALD R. KLINE


ACTUALITÉS SCIENTIFIQUES ET INDUSTRIELLES
CYBERNETICS
OR
CONTROL AND COMMUNICATION
IN THE ANIMAL AND THE MACHINE
NORBERT WIENER
PARIS
HERMANN & Cie, Editeurs
THE TECHNOLOGY PRESS
JOHN WILEY & SONS, Inc.
1948

3. The Series of Approximations to English

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English have been constructed and are given below. In all cases we have assumed a 27-symbol "alphabet," the 26 letters and a space.

1. Zero-order approximation (symbols independent and equiprobable).

   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZL-HJQD.

2. First-order approximation (symbols independent but with frequencies of English text).

   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.

3. Second-order approximation (digram structure as in English).

   ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TU-COOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.

4. Third-order approximation (trigram structure as in English).

   IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONS-TURES OF THE REPTAGIN IS REGOACTIONA OF CRE.

(Shannon, 1948)

5. First-order word approximation. Rather than continue with tetragram, . . . , *n*-gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

   REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

6. Second-order word approximation. The word transition probabilities are correct but no further structure is included.

   THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

The resemblance to ordinary English text increases quite noticeably at each of the above steps. Note that these samples have reasonably good structure out to about twice the range that is taken into account in their construction. Thus in (3) the statistical process insures reasonable text for two-letter sequences, but four-letter sequences from the sample can usually be fitted into good sentences. In (6) sequences of four or more words can easily be placed in sentences without unusual or strained constructions. The particular sequence of ten words "attack on an English writer that the character of this" is not at all unreasonable. It appears then that a sufficiently complex stochastic process will give a satisfactory representation of a discrete source.

(Shannon, 1948)

4. Graphical Representation of a Markoff Process

Stochastic processes of the type described above are known mathematically as discrete Markoff processes and have been extensively studied in the literature.[6] The general case can be described as follows: There exist a finite number of possible "states" of a system; $S_1, S_2, \ldots, S_n$. In addition there is a set of transition probabilities; $p_i(j)$ the probability that if the system is in state S_i it will next go to state S_j. To make this Markoff process into an information source we need only assume that a letter is produced for each transition from one state to another. The states will correspond to the "residue of influence" from preceding letters.

The situation can be represented graphically as shown in Figs. 3, 4 and 5. The "states" are the junction

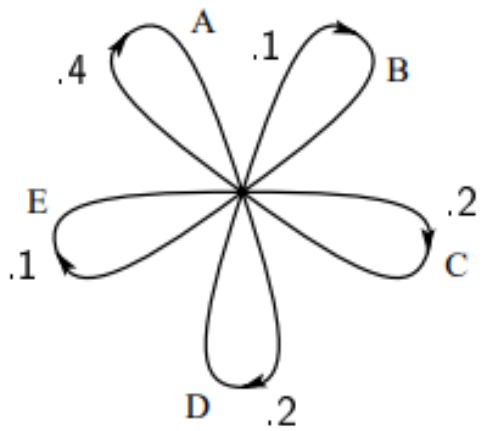


Fig. 3—A graph corresponding to the source in example B.

(Shannon, 1948)

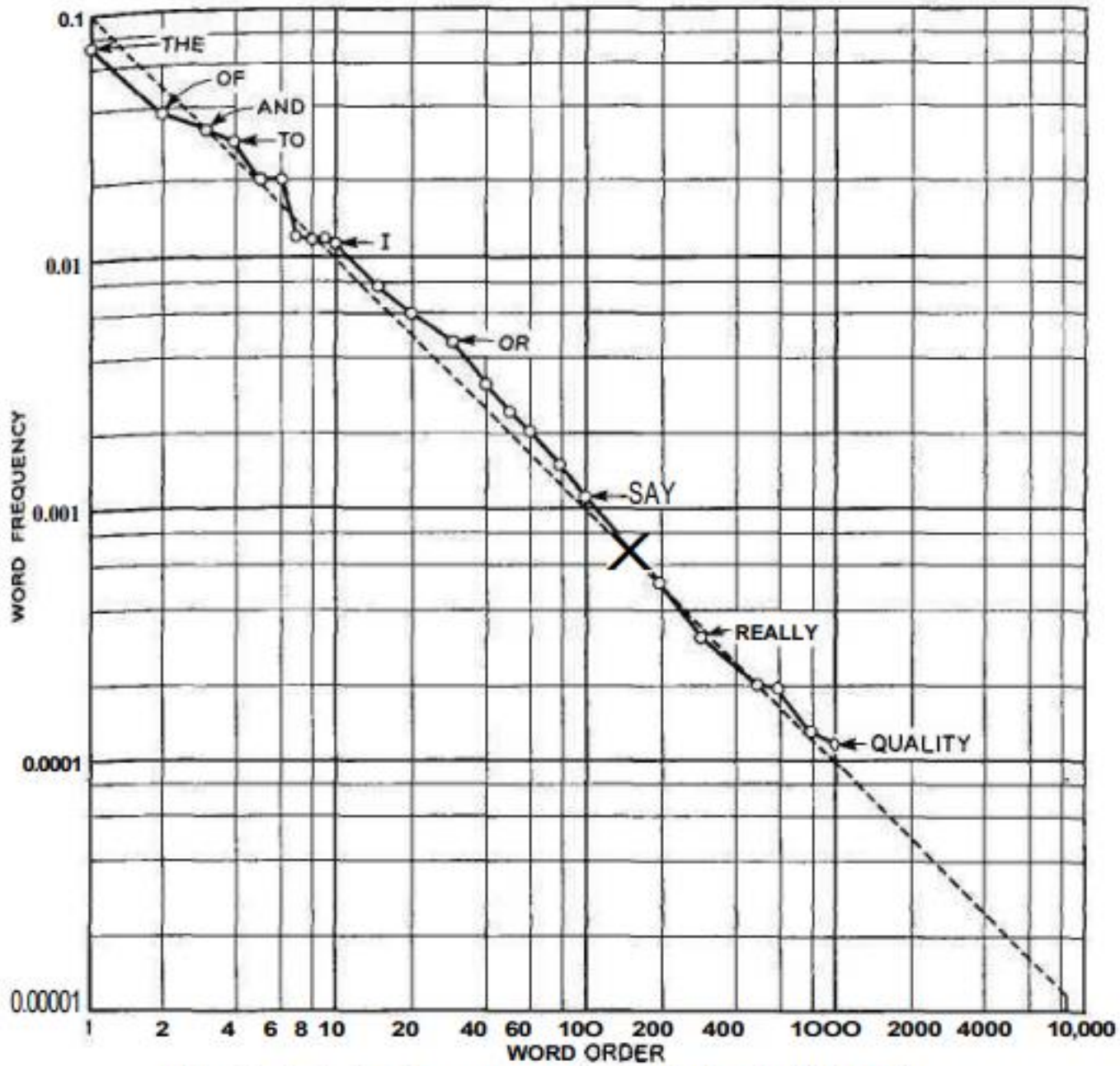


Fig. 1—Relative frequency against rank for English words.

(Shannon, 1951)

In the footnote to this conclusion he considers the possibility of a useful probabilistic/statistical model, saying "I would certainly not care to argue that ... is unthinkable, but I know of no suggestion to this effect that does not have obvious flaws." The main "obvious flaw" is this: Consider:

1. **I** never, ever, ever, ever, ... **fiddle** around in any way with electrical equipment.
2. **She** never, ever, ever, ever, ... **fiddles** around in any way with electrical equipment.
3. * **I** never, ever, ever, ever, ... **fiddles** around in any way with electrical equipment.
4. * **She** never, ever, ever, ever, ... **fiddle** around in any way with electrical equipment.

No matter how many repetitions of "ever" you insert, sentences 1 and 2 are grammatical and 3 and 4 are ungrammatical. A probabilistic Markov-chain model with *n* states can never make the necessary distinction (between 1 or 2 versus 3 or 4) when there are more than *n* copies of "ever." Therefore, a probabilistic Markov-chain model cannot handle all of English.

This criticism is correct, but it is a criticism of Markov-chain models—it has nothing to do with probabilistic models (or trained models) at all. Moreover, since 1957 we have seen many types of probabilistic language models beyond the Markov-chain word models. Examples 1-4 above can in fact be distinguished with a finite-state model that is not a chain, but other examples require more sophisticated models. The best studied is probabilistic context-free grammar (PCFG), which operates over trees, categories of words, and individual lexical items, and has none of the restrictions of finite-state models. We find that PCFGs are state-of-the-art for parsing performance and are easier to learn from data than categorical context-free grammars. Other types of probabilistic models cover semantic

https://norvig.com/chomsky.html

Generative Grammar

Context-free grammar

From Wikipedia, the free encyclopedia

In formal language theory, a **context-free grammar** (**CFG**) is a formal grammar whose production rules are of the form

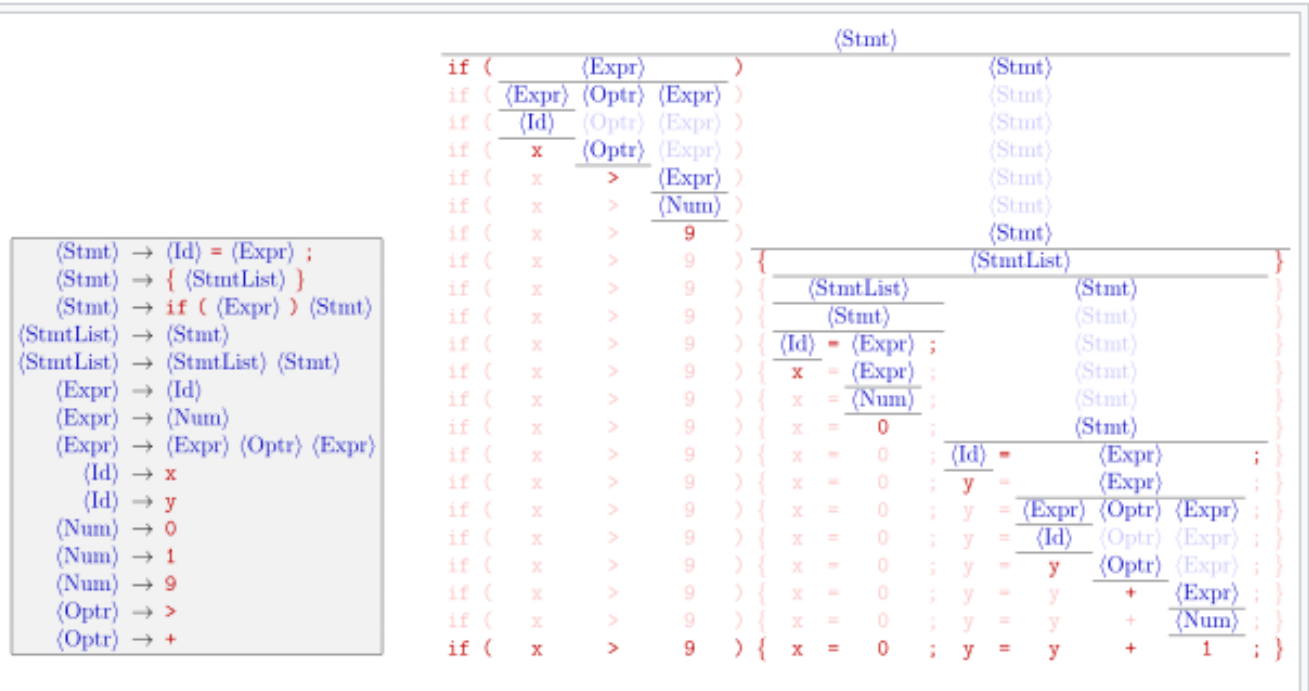
$$A \to \alpha$$

with A a *single* nonterminal symbol, and α a string of terminals and/or nonterminals (α can be empty). A formal grammar is "context free" if its production rules can be applied regardless of the context of a nonterminal. No matter which symbols surround it, the single nonterminal on the left hand side can always be replaced by the right hand side. This is what distinguishes it from a context-sensitive grammar.

A formal grammar is essentially a set of production rules that describe all possible strings in a given formal language. Production rules are simple replacements. For example, the first rule in the picture,

$$\langle \text{Stmt} \rangle \to \langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$$

replaces $\langle \text{Stmt} \rangle$ with $\langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$. There can be multiple replacement rules for a given nonterminal symbol. The language generated by a grammar is the set of all strings of terminal symbols that can be derived, by repeated rule applications, from some particular nonterminal symbol ("start symbol"). Nonterminal symbols are used during the derivation process, but do not appear in its final result string.



Simplified excerpt of the formal grammar[1] for the C programming language (left), and a derivation of a piece of C code (right) from the nonterminal symbol $\langle \text{Stmt} \rangle$. Nonterminal and terminal symbols are shown in blue and red, respectively.

What did Chomsky mean, and is he right?

I take Chomsky's points to be the following:

A. Statistical language models have had engineering success, but that is irrelevant to science.
B. Accurately modeling linguistic facts is just butterfly collecting; what matters in science (and specifically linguistics) is the underlying principles.
C. Statistical models are incomprehensible; they provide no insight.
D. Statistical models may provide an accurate simulation of some phenomena, but the simulation is done completely the wrong way; people don't decide what the third word of a sentence should be by consulting a probability table keyed on the previous two words, rather they map from an internal semantic form to a syntactic tree-structure, which is then linearized into words. This is done without any probability or statistics.
E. Statistical models have been proven incapable of learning language; therefore language must be innate, so why are these statistical modelers wasting their time on the wrong enterprise?

https://norvig.com/chomsky.html

Is he right? That's a long-standing debate. These are my answers:

A. I agree that engineering success is not the goal or the measure of science. But I observe that science and engineering develop together, and that engineering success shows that something is working right, and so is evidence (but not proof) of a scientifically successful model.
B. Science is a combination of gathering facts and making theories; neither can progress on its own. I think Chomsky is wrong to push the needle so far towards theory over facts; in the history of science, the laborious accumulation of facts is the dominant mode, not a novelty. The science of understanding language is no different than other sciences in this respect.
C. I agree that it can be difficult to make sense of a model containing billions of parameters. Certainly a human can't understand such a model by inspecting the values of each parameter individually. But one can gain insight by examing the *properties* of the model—where it succeeds and fails, how well it learns as a function of data, etc.

https://norvig.com/chomsky.html

D. I agree that a Markov model of word probabilities cannot model all of language. It is equally true that a concise tree-structure model without probabilities cannot model all of language. What is needed is a probabilistic model that covers words, trees, semantics, context, discourse, etc. Chomsky dismisses all probabilistic models because of shortcomings of particular 50-year old models. I understand how Chomsky arrives at the conclusion that probabilistic models are unnecessary, from his study of the generation of language. But the vast majority of people who study *interpretation* tasks, such as speech recognition, quickly see that interpretation is an inherently probabilistic problem: given a stream of noisy input to my ears, what did the speaker most likely mean? Einstein said to make everything as simple as possible, but no simpler. Many phenomena in science are stochastic, and the simplest model of them is a probabilistic model; I believe language is such a phenomenon and therefore that probabilistic models are our best tool for representing facts about language, for algorithmically processing language, and for understanding how humans process language.

https://norvig.com/chomsky.html

Learning Bounded Context-Free-Grammar via LSTM and the Transformer: Difference and Explanations

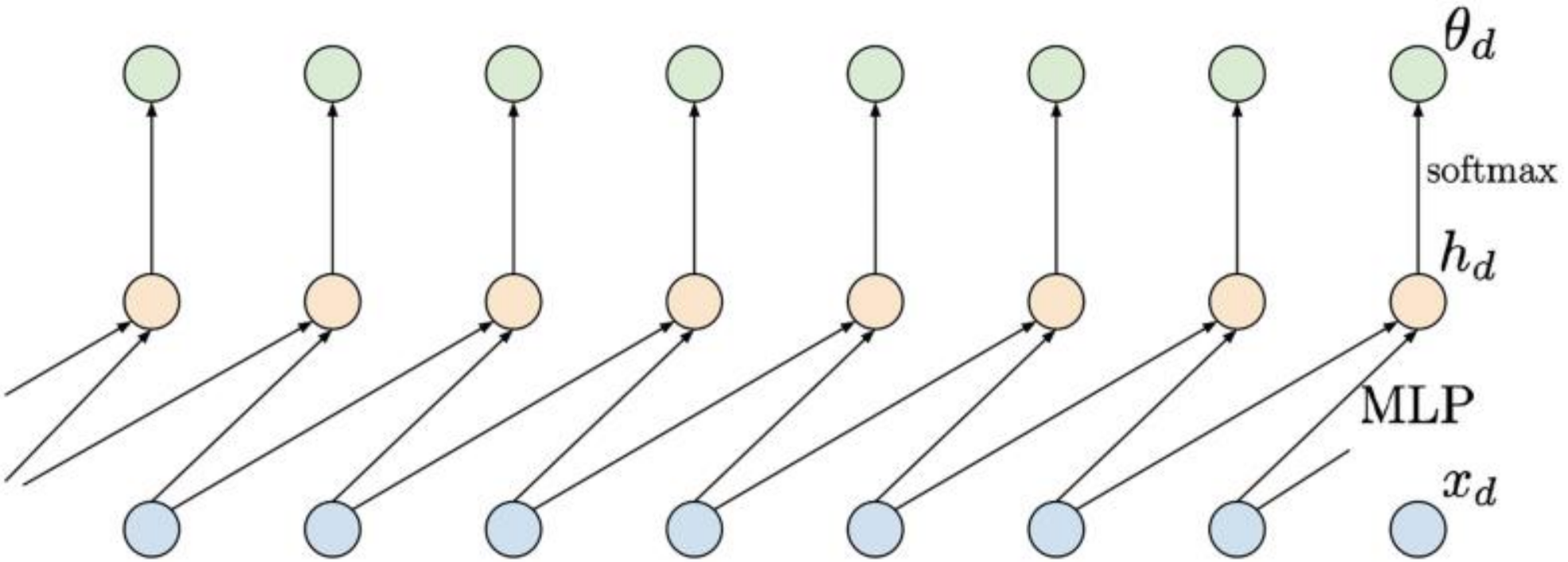
Hui Shi [1], **Sicun Gao** [1], **Yuandong Tian** [2], **Xinyun Chen** [3], **Jishen Zhao**[1]
[1]University of California San Diego, [2]Facebook AI Research, [3]University of California, Berkeley
{hshi, jzhao, sicung}@ucsd.edu, yuandong@fb.com, xinyun.chen@berkeley.edu

Evaluating the Ability of LSTMs to Learn Context-Free Grammars

Luzi Sennhauser
Federal Institute of Technology
Zurich, Switzerland
Massachusetts Institute of Technology
Cambridge, MA, USA
luzis@student.ethz.ch

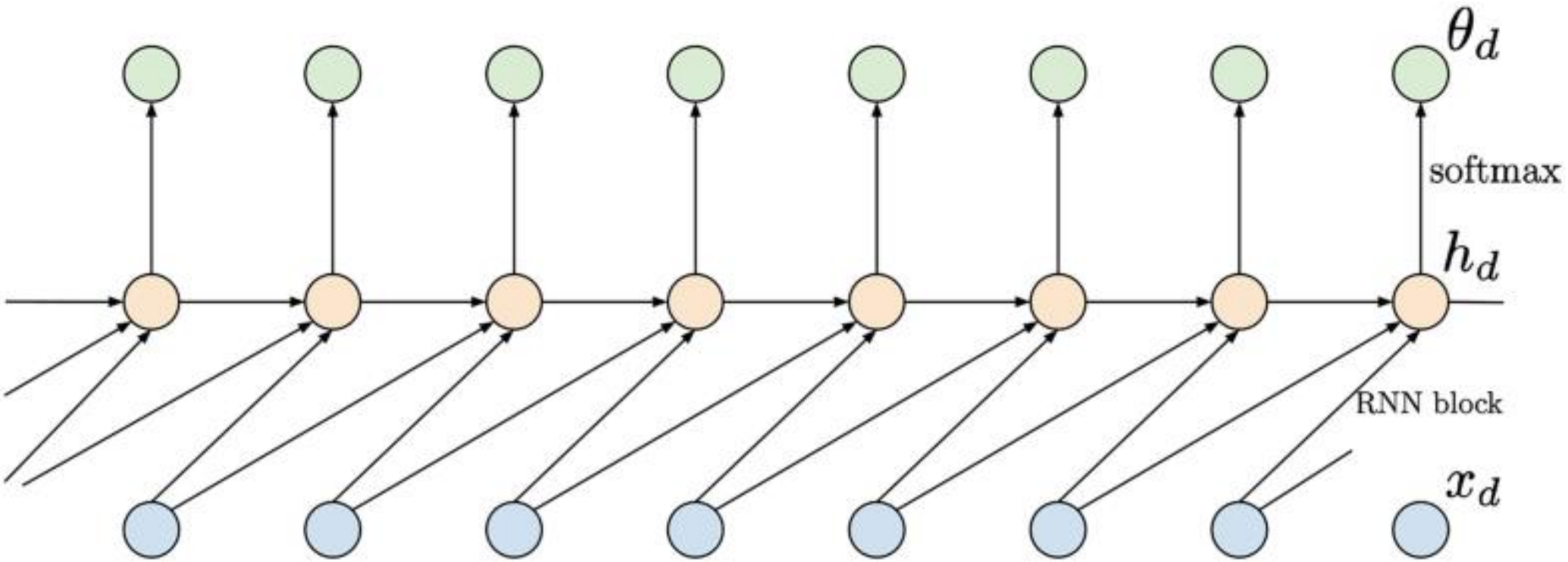
Robert C. Berwick
LIDS, Room 32-D728
Massachusetts Institute of Technology
Cambridge, MA, USA
berwick@csail.mit.edu

Can one learn a language model for a (probabilistic) context-free grammar source and do information-theoretic probing of what rules are learned?



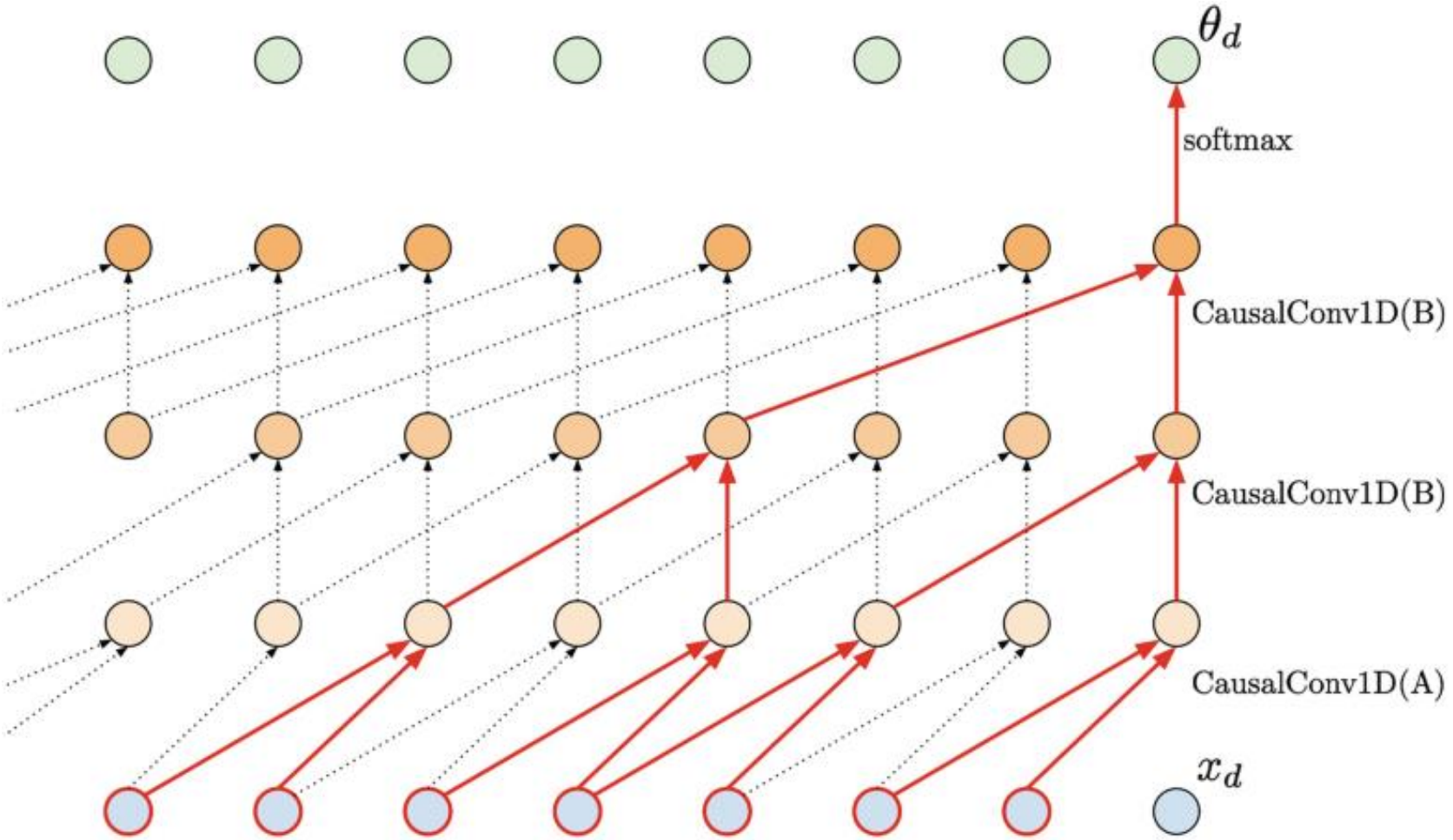
An example of applying a shared MLP depending on two last inputs. Inputs are denoted by blue nodes (bottom), intermediate representations are denoted by orange nodes (middle), and output probabilities are denoted by green nodes (top). Notice that a probability θ_d is not dependent on x_d

[J. M. Tomczak, *Deep Generative Modeling*, Springer, 2022.]



An example of applying an RNN depending on two last inputs. Inputs are denoted by blue nodes (bottom), intermediate representations are denoted by orange nodes (middle), and output probabilities are denoted by green nodes (top). Notice that compared to the approach with a shared MLP, there is an additional dependency between intermediate nodes h_d

[J. M. Tomczak, *Deep Generative Modeling*, Springer, 2022.]



An example of applying causal convolutions. The kernel size is 2, but by applying dilation in higher layers, a much larger input could be processed (red edges), thus, a larger memory is utilized. Notice that the first layers must be option A to ensure proper processing

[J. M. Tomczak, *Deep Generative Modeling*, Springer, 2022.]

Language Model

$$P_{(w_1,w_2,\ldots,w_n)} = p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)\ldots p(w_n|w_1,w_2,..,w_{n-1})$$
$$= \prod_{i=1}^{n} p(w_i|w_1,\ldots,w_{i-1}) \quad (1)$$

S = Where are we going

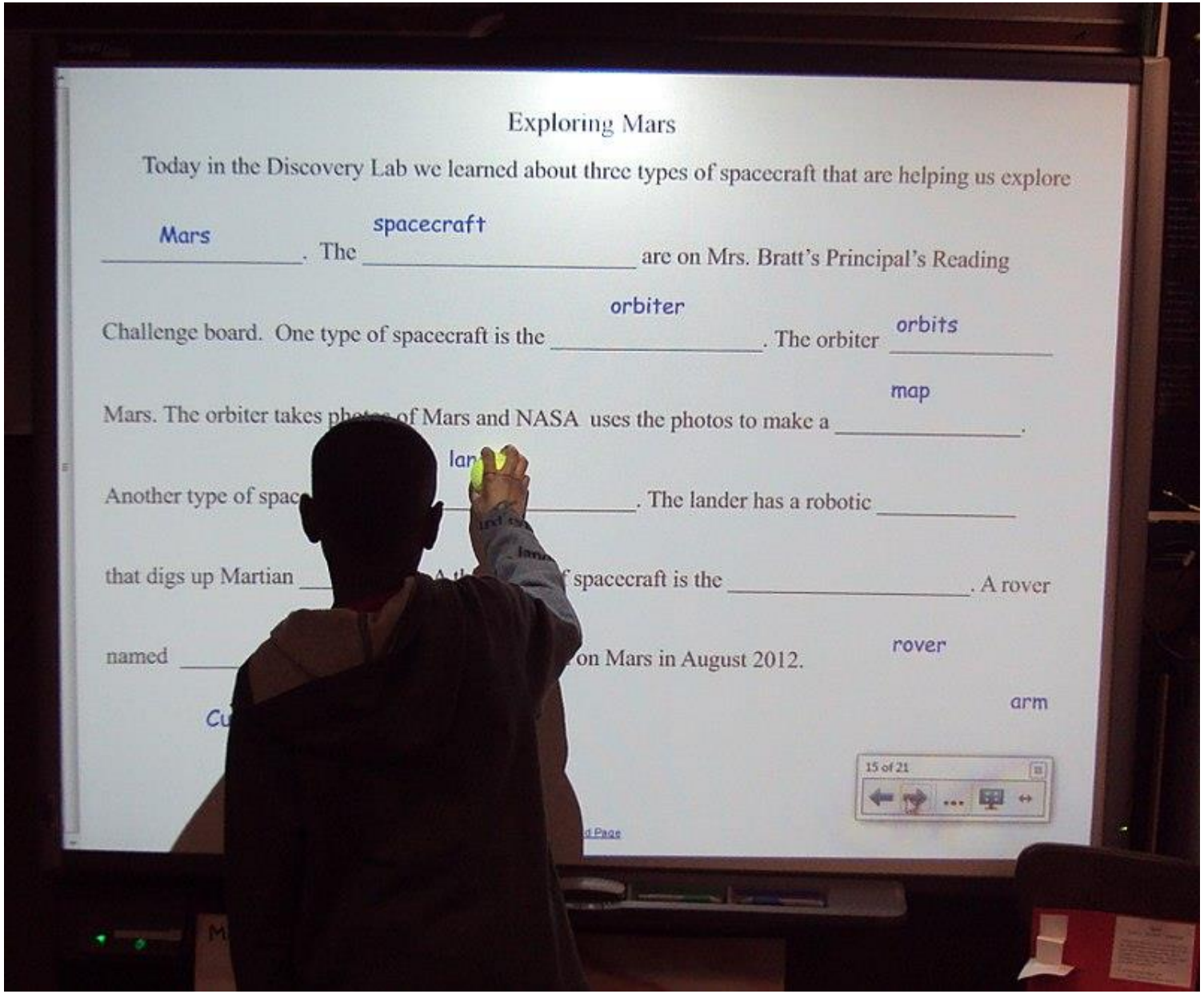
Previous words (Context)

Word being predicted

P(S) = P(Where) x P(are | Where) x P(we | Where are) x P(going | Where are we)

https://thegradient.pub/understanding-evaluation-metrics-for-language-models/

Self-Supervision and Cloze Task



https://transformer.huggingface.co/

CTRL: A Conditional Transformer Language Model for Controllable Generation

Nitish Shirish Keskar[*], **Bryan McCann**[*], **Lav R. Varshney**, **Caiming Xiong**, **Richard Socher**
Salesforce Research[†]

Abstract

Large-scale language models show promising text generation capabilities, but users cannot easily control particular aspects of the generated text. We release CTRL, a 1.63 billion-parameter conditional transformer language model, trained to condition on control codes that govern style, content, and task-specific behavior. Control codes were derived from structure that naturally co-occurs with raw text, preserving the advantages of unsupervised learning while providing more explicit control over text generation. These codes also allow CTRL to predict which parts of the training data are most likely given a sequence. This provides a potential method for analyzing large amounts of data via model-based source attribution. We have released multiple full-sized, pretrained versions of CTRL at `https://github.com/salesforce/ctrl`.

Conditional Generation (CTRL Codes)

Links https://www.cnn.com/2007/09/20/us-president-meets-british-pm
LONDON, England (CNN) – U.S. President George W. Bush met with British Prime Minister Tony Blair on Monday to discuss the war in Iraq, according to a statement from Blair's office.\n\nThe meeting was held at 10 Downing Street and lasted about an hour.\n\n"The two leaders discussed Iraq and other international issues of mutual concern," said Blair.\n\nBush, who is scheduled to meet Wednesday with Russian President Vladimir Putin, will also visit Germany for talks later this week.\n\nIn his statement, Blair said, "We agreed that we should continue our efforts together to bring peace and stability to Iraq. We both reaffirmed our commitment to working closely together, as well as to continuing to work constructively toward achieving lasting security and prosperity throughout the Middle East region."\n\nBush's trip comes after he visited Britain last week where he spoke out against terrorism while visiting Buckingham Palace.\n\nHe has been criticized by some lawmakers over what they say are insufficient military resources being devoted to fighting terrorism.

Data

We train on 140 GB of text drawing from a wide variety of domains: Wikipedia (En, De, Es, Fr), Project Gutenberg[1], submissions from 45 subreddits, OpenWebText[2], a large collection of news data (Hermann et al., 2015; Barrault et al., 2019; Sandhaus, 2008; Grusky et al., 2018), Amazon Reviews (McAuley et al., 2015), Europarl and UN data from WMT (En-De, En-Es, En-Fr) (Barrault et al., 2019), question-answer pairs (no context documents) from ELI5 (Fan et al., 2019) and the MRQA shared task[3], which includes the Stanford Question Answering Dataset (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018), and Natural Questions (Kwiatkowski et al., 2019). A full account of training data and associated control codes can be found in Table 7 in the Appendix.

[1]We use a modified version of `https://github.com/chiphuyen/lazynlp`
[2]We use a modified version of `https://github.com/jcpeterson/openwebtext.git`
[3]`https://github.com/mrqa/MRQA-Shared-Task-2019`

Tokenization

We learn BPE (Sennrich et al., 2015) codes and tokenize the data using fastBPE[4], but we use a large vocabulary of roughly 250K tokens. This includes the sub-word tokens necessary to mitigate problems with rare words, but it also reduces the average number of tokens required to generate long text by including most common words. We use English Wikipedia and a 5% split of our collected OpenWebText data for learning BPE codes. We also introduce an `unknown` token so that during preprocessing we can filter out sequences that contain more than 2 unknown tokens. This, along with the compressed storage for efficient training (TFRecords) (Abadi et al., 2016), reduces our training data to 140 GB from the total 180 GB collected. Data was treated as a single stream of tokens with non-domain control codes inserted where appropriate (often at document boundaries).

Tokenization for Controllable Generation

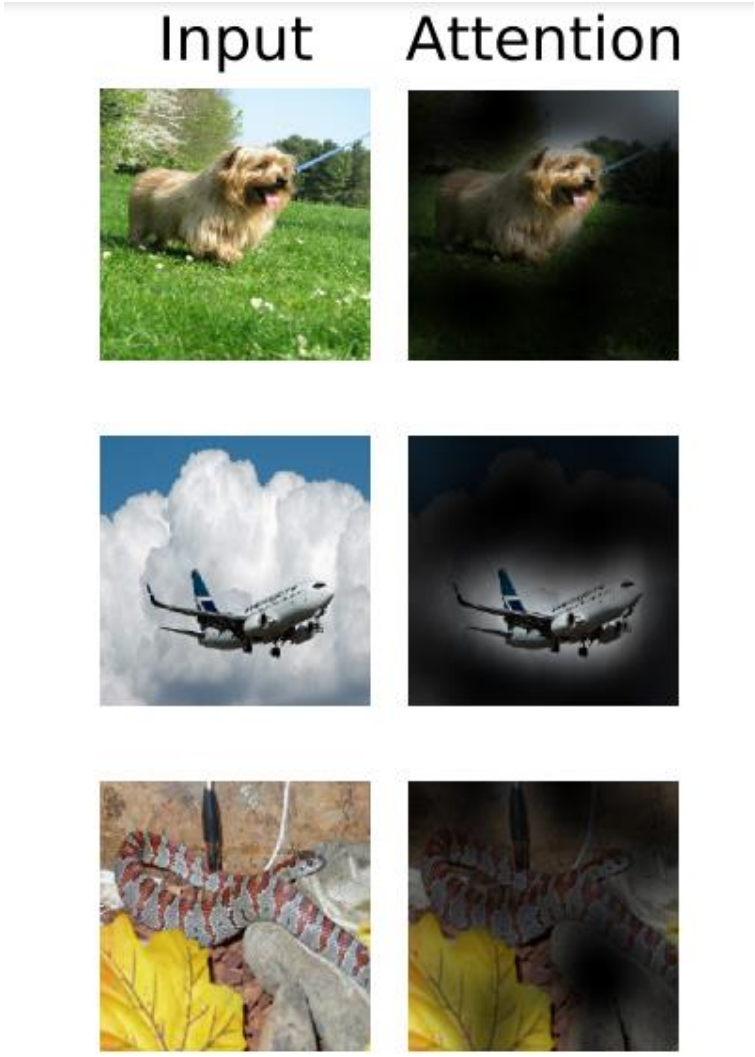
The stream was chunked into contiguous sequences of tokens. Each sequence originated from a domain, and it has the corresponding domain control code prepended as the first token in the sequence. In this way, domain control codes receive special treatment (Kobus et al., 2016). They are propagated to all text in the domain as the first token. This is similar to how codes and natural language sequences have been used in multi-task settings (Wu et al., 2016; Johnson et al., 2017; McCann et al., 2018) to control conditional language models. All other control codes are injected into the data without such special treatment (Moryossef et al., 2019; Caswell et al., 2019). We experimented with sequence lengths of 256 and 512 due to memory and optimization constraints. Despite training on relatively short sequences compared to other approaches, we found that a sliding-window approach allows for generation beyond these windows, and we also found little difference in quality between the two models within the first 256 tokens. Further, we note that our vocabulary is approximately 4 times larger than similar approaches, hence the effective sequence length in characters is comparable.

Architecture and Training Algorithms/Infrastructure

CTRL has model dimension $d = 1280$, inner dimension $f = 8192$, 48 layers, and 16 heads per layer. Dropout with probability 0.1 follows the residual connections in each layer. Token embeddings were tied with the final output embedding layer (Inan et al., 2016; Press & Wolf, 2016).

CTRL was implemented in TensorFlow (Abadi et al., 2016) and trained with a global batch size of 1024 distributed across 256 cores of a Cloud TPU v3 Pod for 800k iterations. Training took approximately 2 weeks using Adagrad (Duchi et al., 2011) with a linear warmup from 0 to 0.05 over 25k steps. The norm of gradients were clipped to 0.25 as in (Merity et al., 2017). Learning rate decay was not necessary due to the monotonic nature of the Adagrad accumulator. We compared to the Adam optimizer (Kingma & Ba, 2014) while training smaller models, but we noticed comparable convergence rates and significant memory savings with Adagrad. We also experimented with explicit memory-saving optimizers including SM3 (Anil et al., 2019), Adafactor (Shazeer & Stern, 2018), and NovoGrad (Ginsburg et al., 2019) with mixed results.

Attention Mechanisms



https://towardsdatascience.com/deconstructing-bert-distilling-6-patterns-from-100-million-parameters-b49113672f77

Attention in Sequence-to-Sequence Models, e.g. for Translation

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Attention in the Transformer Architecture

https://jalammar.github.io/illustrated-transformer/

Transformers as Universal over Domains

https://www.youtube.com/watch?v=Elxn8rS88bI

Transformers as Universal Predictors?

1258 IEEE TRANSACTIONS ON INFORMATION THEORY VOL. 38, NO. 4, JULY 1992

Universal Prediction of Individual Sequences

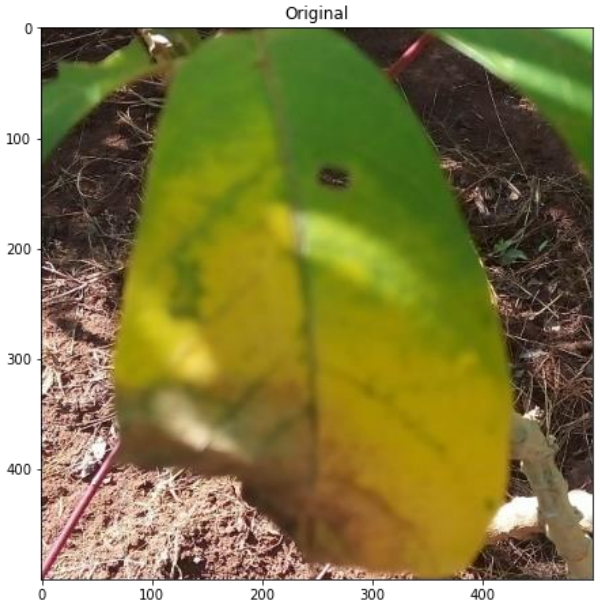
Meir Feder, *Member, IEEE,* Neri Merhav, *Member, IEEE,* and
Michael Gutman, *Member, IEEE*

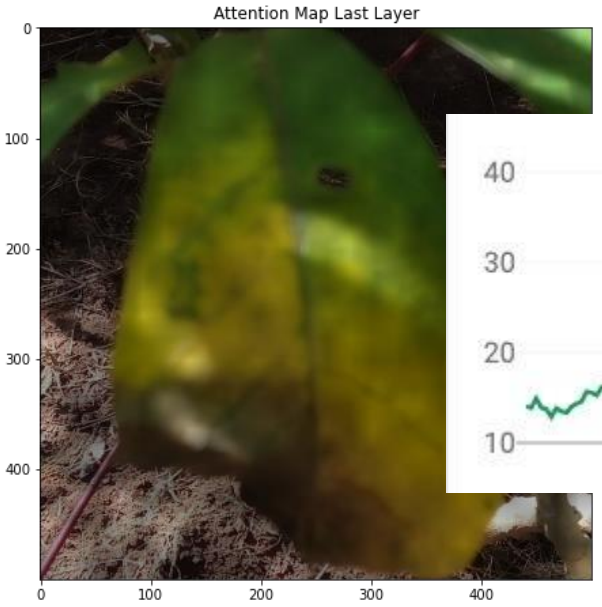
1506 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 50, NO. 7, JULY 2004

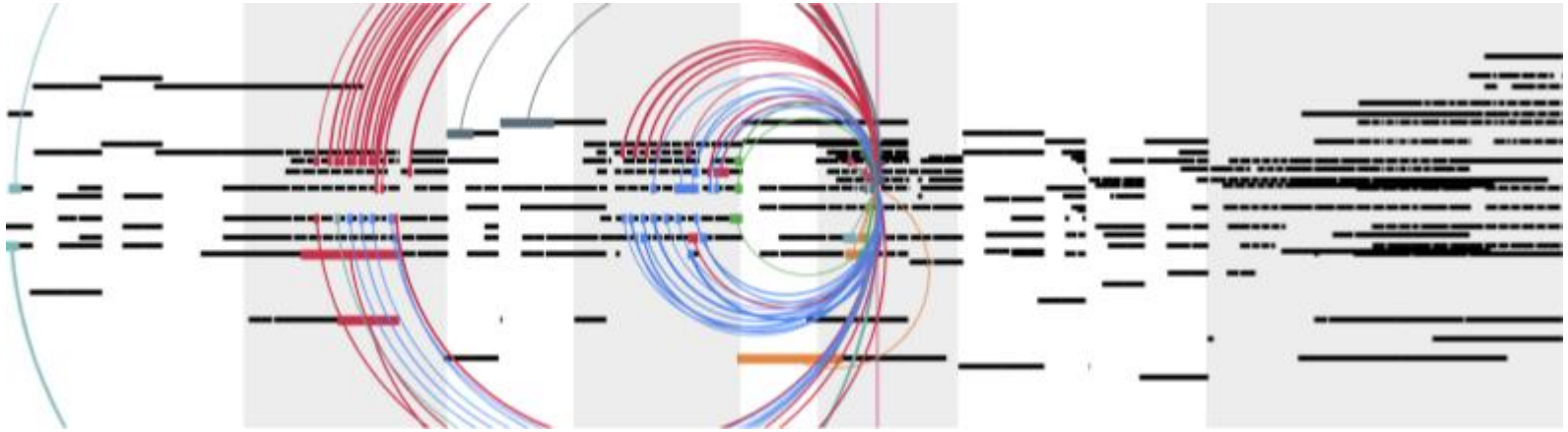
Finite-Memory Universal Prediction of Individual Sequences

Eado Meron and Meir Feder, *Fellow, IEEE*

Transformers as Universal over Domains







[https://magenta.tensorflow.org/music-transformer, https://www.kaggle.com/piantic/vision-transformer-vit-visualize-attention-map, https://neuravest.net/how-transformers-with-attention-networks-boost-time-series-forecasting/]

Transformers as Universal Approximators

2 Transformer networks

A Transformer block is a sequence-to-sequence function mapping $\mathbb{R}^{d\times n}$ to $\mathbb{R}^{d\times n}$. It consists of two layers: a self-attention layer and a token-wise feed-forward layer, with both layers having a skip connection. More concretely, for an input $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ consisting of d-dimensional embeddings of n tokens, a Transformer block with *multiplicative* or *dot-product* attention [Luong et al., 2015] consists of the following two layers[1]:

$$\mathrm{Attn}(\boldsymbol{X}) = \boldsymbol{X} + \sum_{i=1}^{h} \boldsymbol{W}_O^i \boldsymbol{W}_V^i \boldsymbol{X} \cdot \sigma[(\boldsymbol{W}_K^i \boldsymbol{X})^T \boldsymbol{W}_Q^i \boldsymbol{X}], \tag{1}$$

$$\mathrm{FF}(\boldsymbol{X}) = \mathrm{Attn}(\boldsymbol{X}) + \boldsymbol{W}_2 \cdot \mathrm{ReLU}(\boldsymbol{W}_1 \cdot \mathrm{Attn}(\boldsymbol{X}) + \boldsymbol{b}_1 \boldsymbol{1}_n^T) + \boldsymbol{b}_2 \boldsymbol{1}_n^T, \tag{2}$$

where $\boldsymbol{W}_O^i \in \mathbb{R}^{d\times m}, \boldsymbol{W}_V^i, \boldsymbol{W}_K^i, \boldsymbol{W}_Q^i \in \mathbb{R}^{m\times d}, \boldsymbol{W}_2 \in \mathbb{R}^{d\times r}, \boldsymbol{W}_1 \in \mathbb{R}^{r\times d}, \boldsymbol{b}_2 \in \mathbb{R}^d, \boldsymbol{b}_1 \in \mathbb{R}^r$, and $\mathrm{FF}(\boldsymbol{X})$ is the output of the Transformer block. The number of heads h and the head size m are two main parameters of the attention layer; and r denotes the hidden layer size of the feed-forward layer.

Notation. Given a matrix $\boldsymbol{A}$, we use $\|\boldsymbol{A}\|_p$ to denote the entry-wise ℓ^p norm of $\boldsymbol{A}$. Let $\sigma[\cdot]$ be the softmax operator, which takes a matrix as input and applies softmax operation to each column of the matrix, which results in a column stochastic matrix, i.e., a matrix that has non-negative entries with all columns summing to 1. We use $\boldsymbol{1}_n$ to denote a vector of length n whose entries are all 1. We use d and n to denote the embedding dimension and the sequence length, respectively. We assume throughout that $n \geq 2$, as the Transformers reduce to residual networks when $n = 1$.

Transformers as Universal Approximators

We define the Transformer networks as the composition of Transformer blocks. The family of the sequence-to-sequence functions corresponding to the Transformers can be defined as:

$$\mathcal{T}^{h,m,r} := \{g : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n} \mid g \text{ is a composition of Transformer blocks } t^{h,m,r}\text{'s}\}. \quad (3)$$

where $t^{h,m,r} : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ denotes a Transformer block defined by an attention layer with h heads of size m each, and a feed-forward layer with r hidden nodes.

We say that a function $f : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ is *permutation equivariant* if for any permutation matrix $\boldsymbol{P}$, we have $f(\boldsymbol{X}\boldsymbol{P}) = f(\boldsymbol{X})\boldsymbol{P}$; i.e., if we permute the columns of $\boldsymbol{X}$, then the columns of $f(\boldsymbol{X})$ are permuted in the same way. A Transformer block is permutation equivariant, which we formally prove in Section B. This consequently establishes the permutation equivariance of the class $\mathcal{T}^{h,m,r}$.

Claim 1. *A Transformer block $t^{h,m,r}$ defines a permutation equivariant map from $\mathbb{R}^{d\times n}$ to $\mathbb{R}^{d\times n}$.*

Transformers as Universal Approximators

3 Transformers are universal approximators of seq-to-seq functions

In this section, we present our results showing that the Transformer networks are universal approximators of sequence-to-sequence functions. Let us start by defining the target function class $\mathcal{F}_{\mathrm{PE}}$, which consists of all continuous permutation equivariant functions with compact support that map $\mathbb{R}^{d\times n}$ to $\mathbb{R}^{d\times n}$. Here, continuity is defined with respect to any entry-wise ℓ^p norm, $1 \leq p < \infty$. Given two functions $f_1, f_2 : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$, for $1 \leq p < \infty$, we define a distance between them as

$$\mathsf{d}_p(f_1, f_2) := \Big(\int \|f_1(\boldsymbol{X}) - f_2(\boldsymbol{X})\|_p^p \, d\boldsymbol{X} \Big)^{1/p}.$$

The following result shows that a Transformer network with a constant number of heads h, head size m, and hidden layer of size r can approximate any function in $\mathcal{F}_{\mathrm{PE}}$.

Theorem 2. *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{\mathrm{PE}}$, there exists a Transformer network $g \in \mathcal{T}^{2,1,4}$, such that $\mathsf{d}_p(f, g) \leq \epsilon$.*

Transformers as Universal Approximators

3.1 Transformers with trainable positional encodings

In order to endow the Transformer networks with the ability to capture the information about the position of tokens in the input sequence, it is a common practice to add positional encodings $\boldsymbol{E} \in \mathbb{R}^{d\times n}$ to the input sequence before feeding it to the Transformer network [Vaswani et al., 2017, Devlin et al., 2018]. Consider the functions represented by Transformers with positional encodings:

$$\mathcal{T}_{\mathrm{P}}^{h,m,r} := \{g_{\mathrm{P}}(\boldsymbol{X}) = g(\boldsymbol{X}+\boldsymbol{E}) \mid g \in \mathcal{T}^{h,m,r} \text{ and } \boldsymbol{E} \in \mathbb{R}^{d\times n}\}. \tag{4}$$

Here we show that if $\boldsymbol{E}$ is trainable, these positional encodings are sufficient to remove the permutation equivariance restriction of the Transformers. Towards this, we define $\mathcal{F}_{\mathrm{CD}}$ to be the set of all continuous functions that map a compact domain in $\mathbb{R}^{d\times n}$ to $\mathbb{R}^{d\times n}$. Note that $\mathcal{F}_{\mathrm{CD}}$ does not have the restriction of permutation equivariance as in $\mathcal{F}_{\mathrm{PE}}$, but any $f \in \mathcal{F}_{\mathrm{CD}}$ is defined on a compact domain instead of the whole $\mathbb{R}^{d\times n}$. The following result states that, equipped with the trainable positional encodings, Transformers can approximate any sequence-to-sequence function in $\mathcal{F}_{\mathrm{CD}}$.

Theorem 3. *Let $1 \le p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{\mathrm{CD}}$, there exists a Transformer network $g \in \mathcal{T}_{\mathrm{P}}^{2,1,4}$ such that we have $\mathrm{d}_p(f,g) \le \epsilon$.*

Transformers as Universal Approximators

4 Conclusion

In this paper, we prove that Transformer networks are universal approximators of any continuous and permutation equivariant sequence-to-sequence functions, which shed light on the expressive power of Transformer networks. We also theoretically validate the use of additive positional encodings in Transformers, as they can remove the permutation equivaraince restriction and make Transformers universal approximators of arbitrary continuous sequence-to-sequence functions.

In the supplementary material, we present the proofs of our theorems, which reveal that self-attention layers in Transformer networks can compute *contextual mappings*; this is one of the crucial components that make Transformer networks universal. We also discuss and experiment with other simpler layers that can implement weaker forms of contextual mappings.

Transformers as Universal Approximators

C Proof of Theorem 2

Recall that we want to show that given a function $f \in \mathcal{F}_{\mathrm{PE}}$, we can find a Transformer network $g \in \mathcal{T}^{2,1,4}$ such that $\mathsf{d}_p(f, g) \leq \epsilon$. Without loss of generality, we can assume that the compact support of f is contained in $[0, 1]^{d \times n}$. We achieve our desired objective in three key steps:

Step 1. Approximate $\mathcal{F}_{\mathrm{PE}}$ with piece-wise constant functions. We first use (a variant of) the classical result that any continuous function can be approximated up to arbitrary accuracy by piece-wise constant functions. For $\delta > 0$, we define the following class of piece-wise constant functions.

$$\overline{\mathcal{F}}_{\mathrm{PE}}(\delta) := \left\{ f : \boldsymbol{X} \mapsto \sum\nolimits_{\boldsymbol{L} \in \mathbb{G}_\delta} \boldsymbol{A}_{\boldsymbol{L}} \mathbb{1}\left\{\boldsymbol{X} \in \mathbb{S}_{\boldsymbol{L}}\right\} \mid f \text{ is permutation equivariant, } \boldsymbol{A}_{\boldsymbol{L}} \in \mathbb{R}^{d \times n} \right\},$$

where $\mathbb{G}_\delta := \{0, \delta, \ldots, 1 - \delta\}^{d \times n}$ and, for a grid point $\boldsymbol{L} \in \mathbb{G}_\delta$, $\mathbb{S}_{\boldsymbol{L}} := \prod_{j=1}^{d} \prod_{k=1}^{n} [L_{j,k}, L_{j,k} + \delta) \subset [0, 1]^{d \times n}$ denotes the associated cube of width δ.

The following result states that the underlying function $f \in \mathcal{F}_{\mathrm{PE}}$ can be approximated using the function class $\overline{\mathcal{F}}_{\mathrm{PE}}(\delta)$.

Lemma 4. *For any given $f \in \mathcal{F}_{\mathrm{PE}}$ and $1 \leq p < \infty$, one can find a $\delta^* > 0$ such that $\exists\, \overline{f} \in \overline{\mathcal{F}}_{\mathrm{PE}}(\delta^*)$ which satisfies $\mathsf{d}_p(f, \overline{f}) \leq \epsilon/3$.*

Transformers as Universal Approximators

Step 2. Approximate $\overline{\mathcal{F}}_{\mathrm{PE}}(\delta)$ with *modified* Transformers. We then consider a slightly modified architecture for Transformer networks, where the softmax operator $\sigma[\cdot]$ and $\mathrm{ReLU}(\cdot)$ are replaced by the hardmax operator $\sigma_{\mathrm{H}}[\cdot]$ and an activation function $\phi \in \Phi$, respectively. Here, the set of allowed activations Φ consists of all piece-wise linear functions with at most three pieces, where at least one piece is constant. Let $\overline{\mathcal{T}}^{h,m,r}$ denote the function class corresponding to the sequence-to-sequence functions defined by the modified Transformer networks. The following result establishes that the modified Transformer networks in $\overline{\mathcal{T}}^{2,1,1}$ can closely approximate functions in $\overline{\mathcal{F}}_{\mathrm{PE}}(\delta)$.

Proposition 5. *For each* $\overline{f} \in \overline{\mathcal{F}}_{\mathrm{PE}}(\delta)$ *and* $1 \leq p < \infty$, $\exists\, \overline{g} \in \overline{\mathcal{T}}^{2,1,1}$ *such that* $\mathrm{d}_p(\overline{f}, \overline{g}) = O(\delta^{d/p})$.

Transformers as Universal Approximators

Step 3. Approximate modified Transformers with (original) Transformers. Finally, we show that $\overline{g} \in \overline{\mathcal{T}}^{2,1,1}$ can be approximated by $\mathcal{T}^{2,1,4}$. Let $g \in \mathcal{T}^{2,1,4}$ be such that $\mathsf{d}_p(\overline{g}, g) \leq \epsilon/3$.

The following result relies on the connection between the softmax operator and the hardmax operator; and the fact each activation $\phi \in \Phi$ can approximated by the sum of four ReLU's.

Lemma 6. *For each* $\overline{g} \in \overline{\mathcal{T}}^{2,1,1}$ *and* $1 \leq p < \infty$, $\exists\, g \in \mathcal{T}^{2,1,4}$ *such that* $\mathsf{d}_p(\overline{g}, g) \leq \epsilon/3$.

Theorem 2 now follows from these three steps, because we have

$$\mathsf{d}_p(f, g) \leq \mathsf{d}_p(f, \overline{f}) + \mathsf{d}_p(\overline{f}, \overline{g}) + \mathsf{d}_p(\overline{g}, g) \leq 2\epsilon/3 + O(\delta^{d/p}).$$

Choosing $\delta \leq \delta^*$ small enough ensures that $\mathsf{d}_p(f, g) \leq \epsilon$. □

Allometric Scaling

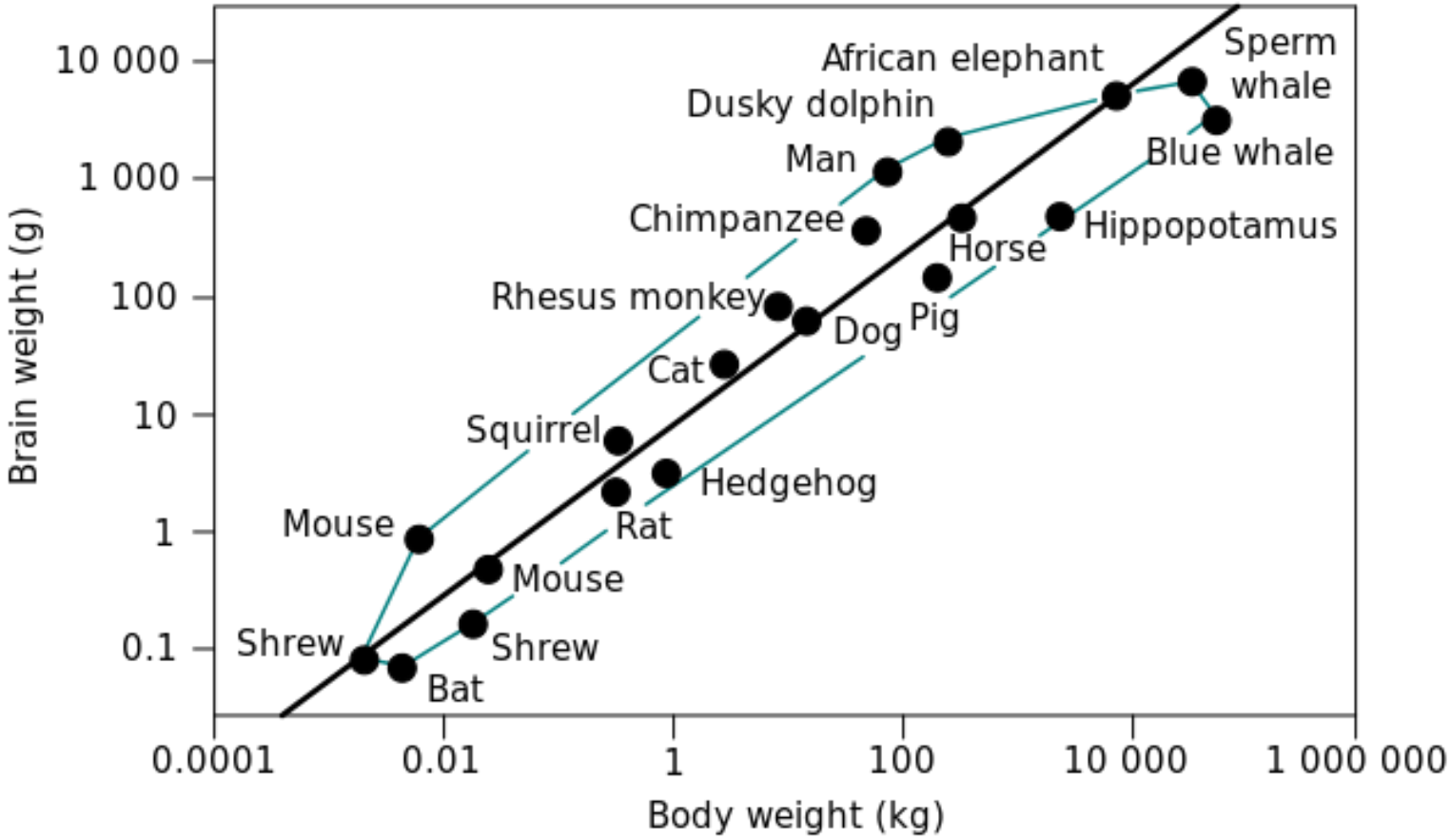
- Allometry studies the relationship between body size to shape. Goes back to D'Arcy Thompson's *On Growth and Form* (1917)
- In neurobiology, one can look at allometric scaling relationships:
  - across different species with similar brain architectures [evolution],
  - scaling relationships for different individuals of same species [growth],
  - properties of the brain within the same individual [structure]
- The relationship between the two measured quantities is usually expressed as a power law equation:

$$y = kx^{\alpha}$$

where α is the scaling exponent of the law.

- How should we interpret superlinear ($\alpha > 1$) or sublinear ($\alpha < 1$) scaling?

Allometric Scaling



Encephalization quotient

$E = CS^2$, where E and S are body and brain weights

[https://en.wikipedia.org/wiki/Brain-to-body_mass_ratio]

Allometric Scaling

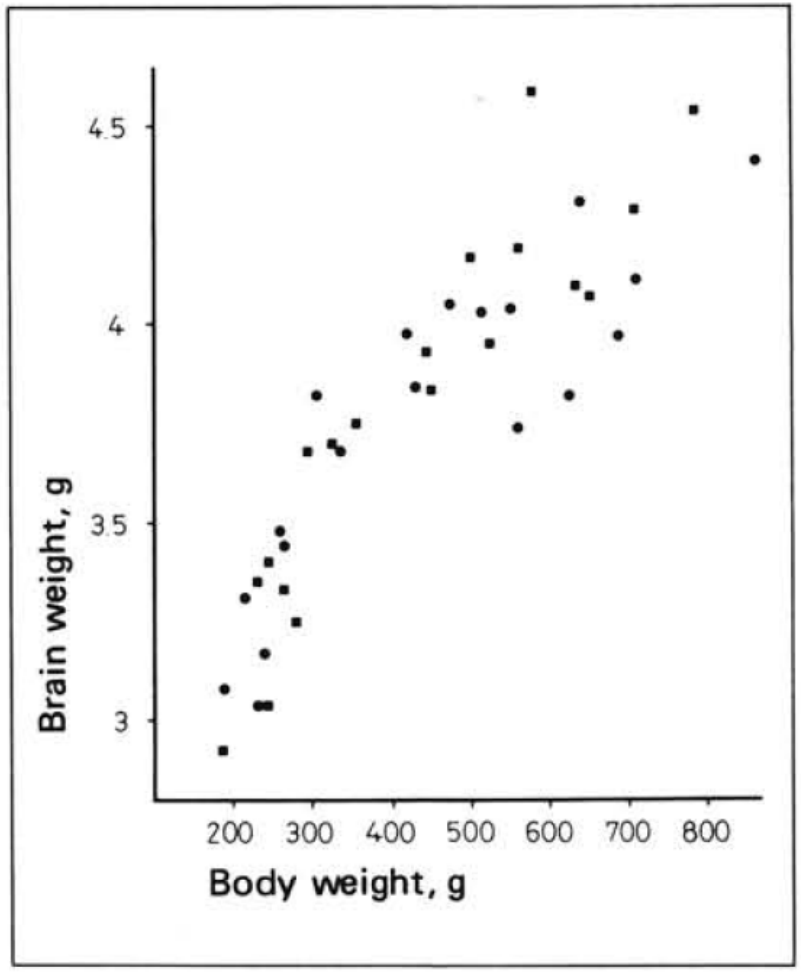
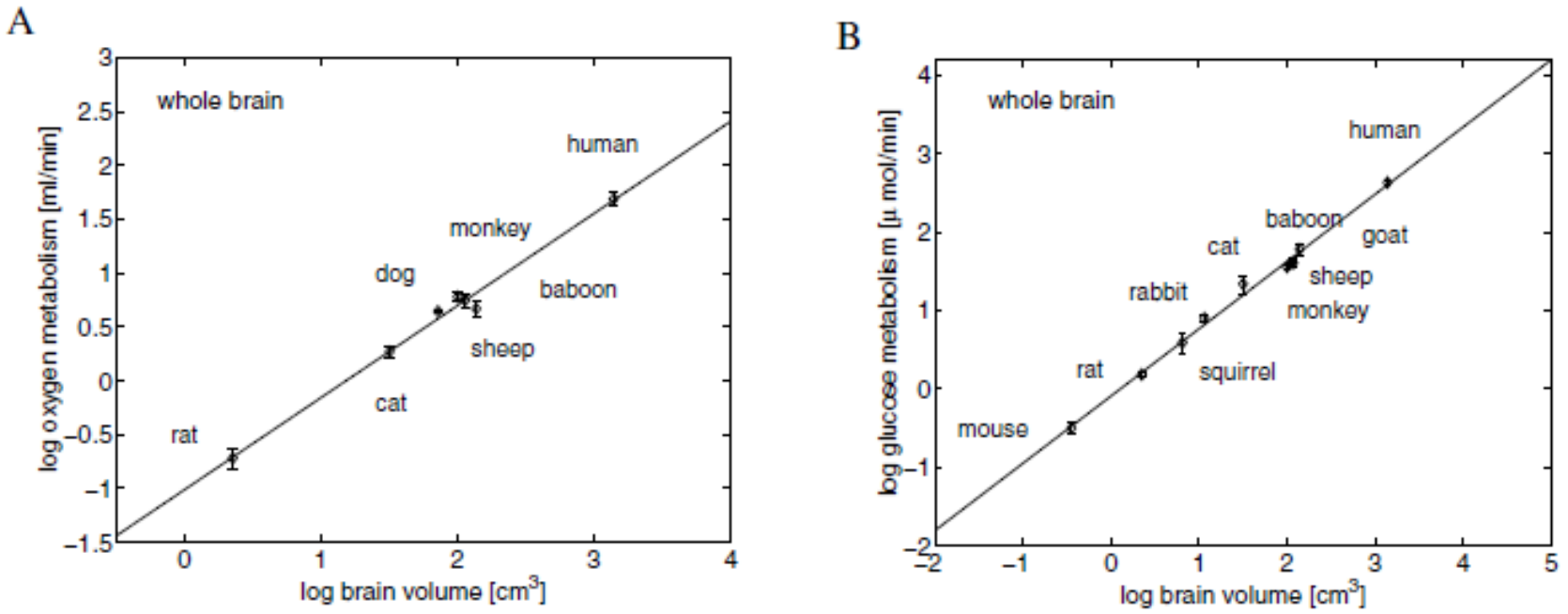


Fig. 1. Brain weights of guinea pigs *(Cavia cobaya)*

[Heinz Stephan, Heiko Frahm, and Georg Baron, "New and Revised Data on Volumes of Brain Structures in Insectovores and Primates," Folia Primatol., vol. 35, pp. 1-29, 1981.]

Allometric Scaling



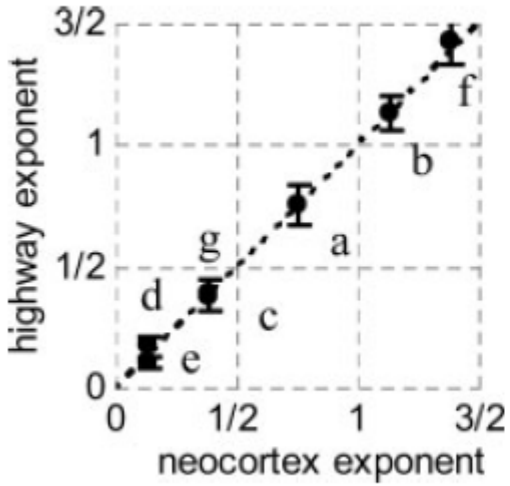
Scaling of the total basal cerebral metabolism with brain volume. The least-square fit line for the log – log plot yields the following. (**A**) For the total oxygen consumption
rate, the scaling exponent was 0.86 ± 0.04 ($y = 0.86x - 1.02$, $R^2 = 0.989$, $p < 10^{-4}$, $n = 7$), and its 95% confidence interval was 0.75 to 0.96. (**B**) For the total glucose utilization rate, an identical exponent 0.86 ± 0.03 was found ($y = 0.86x - 0.09$, $R^2 = 0.994$, $p < 10^{-4}$, $n = 10$) and its 95% confidence interval was 0.80 to 0.91.

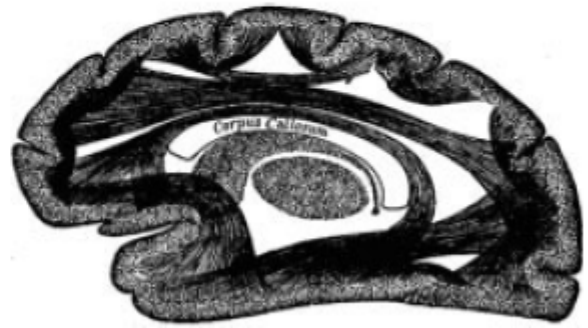
[Jan Karbowski, "Global and regional brain metabolic scaling and its functional consequences," *BMC Biology*, 2007, 5:18.]

Are there common allometric scalings among different kinds of networks?

Are there common allometric scalings among different kinds of networks?

Comparison of City Highway System and Neocortex Exponents for Quantities as a Function of Surface Area





Generic Name	Variable for City Highways	City Highway System Exponent	Variable for Neocortex	Neocortex Exponent
Surface area	Land area	1	Total convoluted surface area	1
(a) No. of conduits	No. of highways	0.759 (±0.083)	No. of pyramidal neurons	3/4 = 0.75
(b) Total no. of leaves	Total no. exits	1.138 (±0.072)	Total no. of synapses	9/8 = 1.125
(c) No. of leaves per conduit	No. of exits per highway	0.379 (±0.064)	No. of synapses per neuron	3/8 = 0.375
(d) Diameter of conduit	No. of highway lanes	0.174 (±0.038)	Diameter of white matter axon	1/8 = 0.125
(e) Propagation velocity	Velocity of cross-city travel	0.108 (±0.021)	Propagation velocity of white matter axon	1/8 = 0.125
(f) Total surface area of conduits	Total surface of highways	1.433 (±0.096)	Total surface area of white matter axons	11/8 = 1.375
	Population	1.462 (±0.141)		
			Total volume of white matter axons	3/2 = 1.5
(g) No. of compartments	No. of concentric ring regions	0.390 (±0.055)	No. of cortical areas	3/8 = 0.375

A General Model for the Origin of Allometric Scaling Laws in Biology

Geoffrey B. West, James H. Brown,* Brian J. Enquist

Allometric scaling relations, including the 3/4 power law for metabolic rates, are characteristic of all organisms and are here derived from a general model that describes how essential materials are transported through space-filling fractal networks of branching tubes. The model assumes that the energy dissipated is minimized and that the terminal tubes do not vary with body size. It provides a complete analysis of scaling relations for mammalian circulatory systems that are in agreement with data. More generally, the model predicts structural and functional properties of vertebrate cardiovascular and respiratory systems, plant vascular systems, insect tracheal tubes, and other distribution networks.

The Origins of Scaling in Cities

Luís M. A. Bettencourt

Despite the increasing importance of cities in human societies, our ability to understand them scientifically and manage them in practice has remained limited. The greatest difficulties to any scientific approach to cities have resulted from their many interdependent facets, as social, economic, infrastructural, and spatial complex systems that exist in similar but changing forms over a huge range of scales. Here, I show how all cities may evolve according to a small set of basic principles that operate locally. A theoretical framework was developed to predict the average social, spatial, and infrastructural properties of cities as a set of scaling relations that apply to all urban systems. Confirmation of these predictions was observed for thousands of cities worldwide, from many urban systems at different levels of development. Measures of urban efficiency, capturing the balance between socioeconomic outputs and infrastructural costs, were shown to be independent of city size and might be a useful means to evaluate urban planning strategies.

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range.

Scaling Laws for Transfer

Danny Hernandez*

Jared Kaplan†‡ Tom Henighan† Sam McCandlish†

When we train increasingly large neural networks from-scratch on a fixed-size dataset, they eventually become data-limited and stop improving in performance (cross-entropy loss). When we do the same for models pre-trained on a large language dataset, the slope in performance gains is merely reduced rather than going to zero. We calculate the effective data “transferred” from pre-training by determining how much data a transformer of the same size would have required to achieve the same loss when training from scratch. In other words, we focus on units of data while holding everything else fixed. We find that the effective data transferred is described well in the low data regime by a power-law of parameter count and fine-tuning dataset size. We believe the exponents in these power-laws correspond to measures of the generality of a model and proximity of distributions (in a directed rather than symmetric sense). We find that pre-training effectively multiplies the fine-tuning dataset size.

Scaling Laws Under the Microscope: Predicting Transformer Performance from Small Scale Experiments

Maor Ivgi
Tel-Aviv University

Yair Carmon
Tel-Aviv University

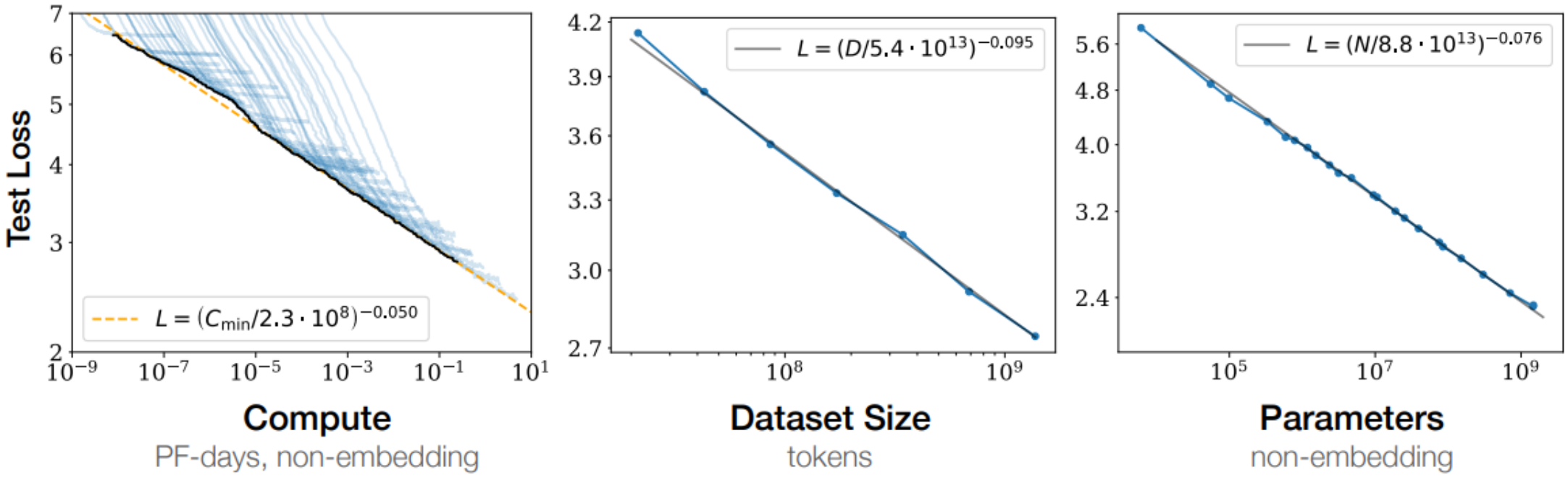
Jonathan Berant
Tel-Aviv University

Neural scaling laws define a predictable relationship between a model's parameter count and its performance after training in the form of a power law. However, most research to date has not explicitly investigated whether scaling laws can be used to accelerate model development. In this work, we perform such an empirical investigation across a wide range of language understanding tasks, starting from models with as few as 10K parameters, and evaluate downstream performance across 9 language understanding tasks. We find that scaling laws emerge at finetuning time in some NLP tasks, and that they can also be exploited for debugging convergence when training large models. Moreover, for tasks where scaling laws exist, they can be used to predict the performance of larger models, which enables effective model selection.

Scaling Laws for Transformers

Model performance depends most strongly on scale, which consists of three factors: the number of model parameters N (excluding embeddings), the size of the dataset D, and the amount of compute C used for training. Within reasonable limits, performance depends very weakly on other architectural hyperparameters such as depth vs. width.

Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two, with trends spanning more than six orders of magnitude



https://arxiv.org/pdf/2001.08361.pdf

Scaling Laws for Transformers

- **Universality of overfitting**: Performance improves predictably as long as we scale up N and D in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases. The performance penalty depends predictably on the ratio $N^{0.74}/D$, meaning that every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty.

[Biological ¾ law?]

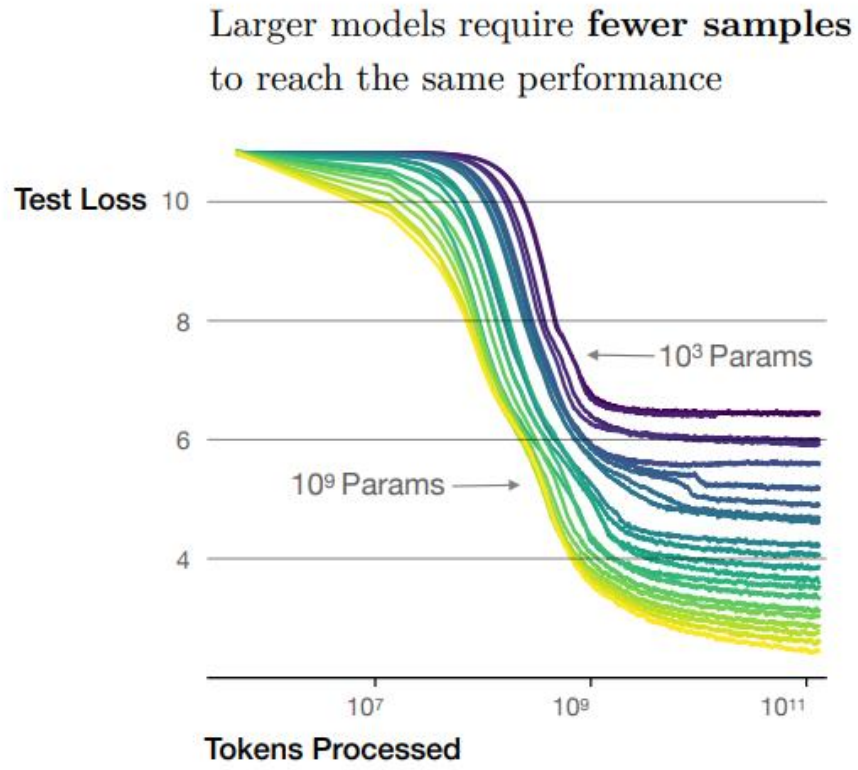
- **Universality of training**: Training curves follow predictable power-laws whose parameters are roughly independent of the model size. By extrapolating the early part of a training curve, we can roughly predict the loss that would be achieved if we trained for much longer.

- **Transfer improves with test performance**: When we evaluate models on text with a different distribution than they were trained on, the results are strongly correlated to those on the training validation set with a roughly constant offset in the loss – in other words, transfer to a different distribution incurs a constant penalty but otherwise improves roughly in line with performance on the training set.

https://arxiv.org/pdf/2001.08361.pdf

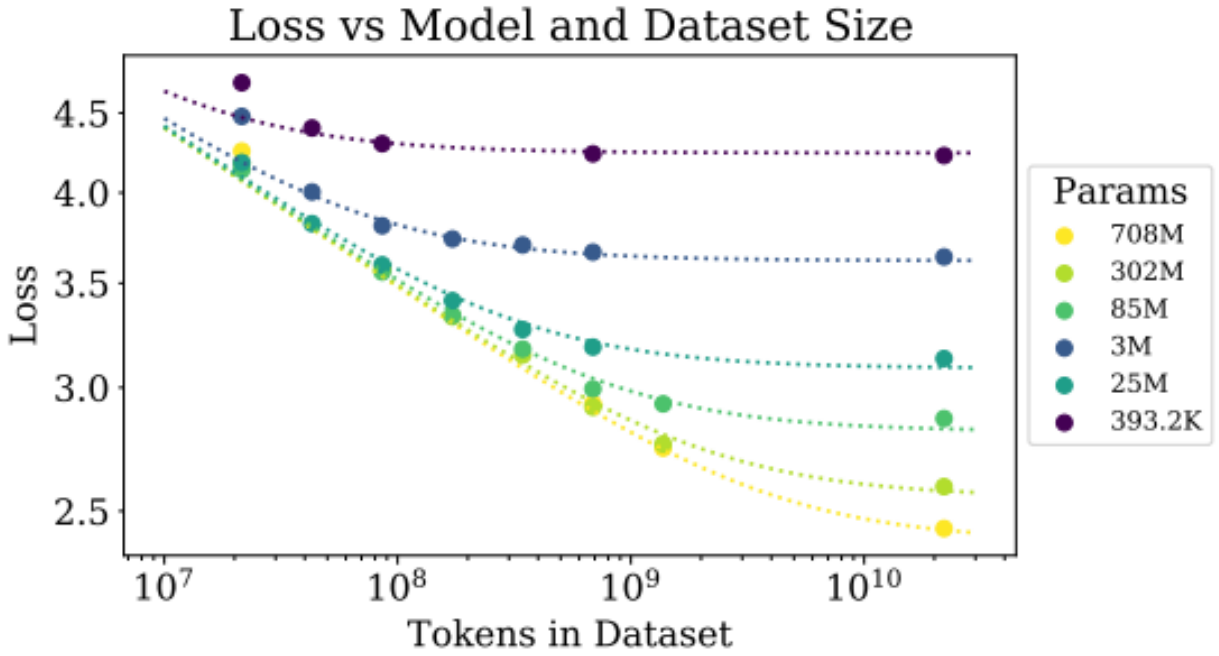
Scaling Laws for Transformers

- **Sample efficiency**: Large models are more sample-efficient than small models, reaching the same level of performance with fewer optimization steps and using fewer data points.



- **Convergence is inefficient**: When working within a fixed compute budget C but without any other restrictions on the model size N or available data D, we attain optimal performance by training very large models and stopping significantly short of convergence. Maximally compute-efficient training would therefore be far more sample efficient than one might expect based on training small models to convergence, with data requirements growing very slowly as $D \sim C^{0.27}$ with training compute.

https://arxiv.org/pdf/2001.08361.pdf



$$\alpha_N \sim 0.076,$$
$$\alpha_D \sim 0.095,$$

as we increase the model size, we should increase the dataset size sublinearly according to $D \propto N^{\{\alpha_N/\alpha_D\}} \sim N^{0.74}$

$$L(N, D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D} \tag{1.5}$$

with fits pictured on the left in figure 4. We conjecture that this functional form may also parameterize the trained log-likelihood for other generative modeling tasks.