SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Threshold-like behavior in Large Language Models

av

Filippa Bjare

2026 - No K12

Threshold-like behavior in Large Language Models

Filippa Bjare

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Daniel Ahlberg

2026

Abstract

This paper presents deep neural networks with a focus on large language models and their rapid increase in performance. Many researchers in the field describe these improvements as demonstrating *emergent abilities* that exhibit *threshold-like* behavior. However, an alternative interpretation is offered by Schaeffer et al. (NeurIPS, 2023), where emergent abilities are seen to be induced by certain metrics. This paper provides a closer look at their arguments as well as an analogy to random-graphs, which will be rigorously studied in detail.

Sammanfattning

I denna uppsats presenteras djupa neurala nätverk med ett fokus på stora språkmodeller och deras drastiska förbättringar i prestation. Många forskare inom området beskriver dessa som *plötsliga förmågor* som uppvisar *tröskelbeteende*. Schaeffer et al. (NeurIPS, 2023) ifrågasätter dock detta och menar att uppvisandet av sådant beteende snarare är en konsekvens av utvärderingsmetoden. I denna uppsats tittar jag närmare på deras argument samt föreslår en analogi till slumpgrafer, vilka studeras rigoröst i detalj.

AI statement

I have used AI to generate graph simulations and assist with general LaTeX formatting, including specific symbol commands for mathematical definitions and theorems, as well as for grammar and spelling. I have also used AI to better understand some of the articles in section 5, but I have then presented the material in accordance with the original source.

AI användning

Jag har använt AI för att generera grafsimuleringar och assistera med generell LaTeX-formatering, inklusive specifika symbolkommandon för matematiska definitioner och satser, samt för grammatik och stavning. Jag har också använt AI för att bättre förstå några av artiklarna i avsnitt 5, men jag har sedan presenterat materialet i enlighet med originalkällan.

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Daniel Ahlberg, for his consistent support and for always being able to answer my questions, whether in person or by email.

Förord

Jag vill rikta ett varmt tack till min handledare Daniel Ahlberg som bistått med oersättlig stöttning och vägledning vid alla typer av frågor. Vare sig över mejl eller i person.

Contents

1 Introduction

Are you also one of those people who almost automatically turns to ChatGPT when your dishwasher breaks down or when you need to convince a family member about which movie to watch? Well, the rapid improvement of large language models is not only noticeable in the increase of use, but they also seem to drastically improve on established benchmarks such as modular arithmetic and word-scramble only by a small change in size. These models are a specific type of deep-neural network that, in short, aims to predict the next word.

Conventionally, the improvement of these models is understood to follow distinct *scaling laws*. That is, to gradually improve by either increasing the amount of parameters, training-data, or compute-budget. However, the observed drastic improvement of these models suggests otherwise and is, in particular, reminiscent of *thresholds*, that is; a considerable change of state in a system by a small change in a local parameter. For example, Wei et al. describe these improvements as *emergent abilities* [Wei+22], which they define as abilities apparent in larger models but not in smaller ones. However, this interpretation is criticized by Schaeffer et al. [SMK23] who argues that these findings are induced by non-linear and discontinuous metrics, rather than being representative of the actual improvement.

To better understand the debate, I will study large language models to some extent and reformulate Schaeffer's argument by drawing an analogy to random graphs, in which a rigorous analysis can be carried out. For random graphs, the transition of a graph having only small components to the appearance of a giant one is considered a *sharp threshold*, while its growth, which illustrates the model's predictive power, is considered to have a *coarse threshold*. However, with other metrics, its power can also be seen to increase instantly, which is illustrated by the sharp threshold of connectivity. In addition to this debate, I will look into findings of threshold-like behavior in neural networks where they are not necessarily described as emergent abilities. That is, Sebastian Bubeck provides a rigorous analysis demonstrating threshold neurons that seem to suddenly appear under a specific *learning rate*. [Bub23]

2 Deep neural network

In this section, we will have a closer look at deep neural networks inspired by the book *Deep Learning* by Good-fellow et. al [GBC16] as well as Mikael Nielsen book *Neural Networks and Deep learning*[Nie19]. Deep neural networks are a machine-learning algorithm inspired by the human brain. They consist of several artificial neurons that adjust and transfer information with the help of a large number of parameters. These networks are recognized for their many layers and the ability to extract patterns from data. The input of the network can be anything from a picture to a piece of text translated into a set of high-dimensional vectors.

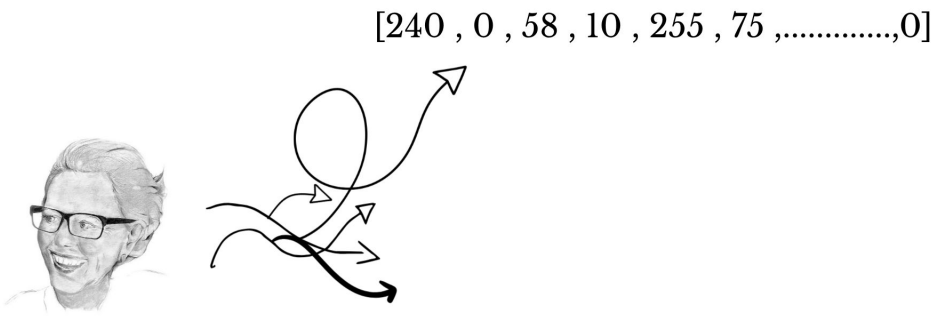

Figure 1: Vector-representation

The task of the network might be to answer a user's question or to categorize a picture provided by the final vector. In general, one can think of the network as a class of functions from one vector space to another, with a large number of parameters θ.

$$f_\theta : \mathbb{R}^n \to \mathbb{R}^m, \quad x \mapsto f(x;\theta)$$

To gain a better understanding of the structure of a network, we will consider a simplified one that recognizes facial expressions. Namely; the following is a way to conceptualize how the network is able to work as well as it does rather than being an actual description. Before we proceed, I would like to credit Grant Sanderson as

well as Michael Nielsen, for inspiration.

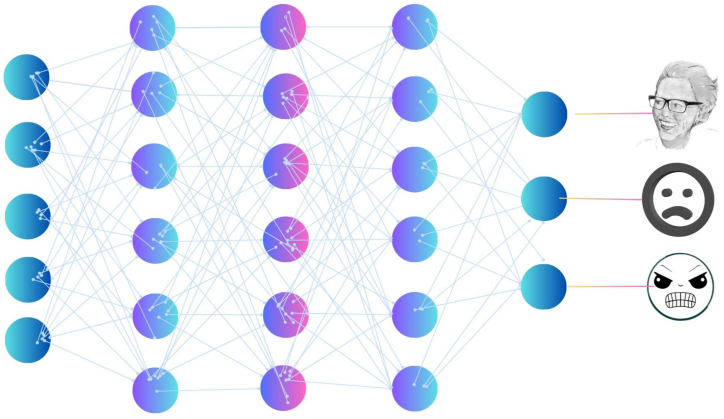

Figure 2: Deep Neural Network

For instance, in our network the picture is translated into a vector where each element represents the degree of grayness in each pixel. The goal of the network is to determine facial expression which requires different hidden layers to detect the necessary information. To better understand how the network detects those, one could think that the first hidden layer scans the edges of the face to determine what to analyze. The next layer might identify the expression of the eyes, whereas the last determines the shape of the mouth. All layers are needed to determine the correct facial expression offered by the final layer.

Without going into much detail, we will look into the architecture and internal processes of such networks. As seen in the figure above, each neuron is connected to every neuron in the previous layer. However, the information passed over is adjusted by the weights and biases of each connection.

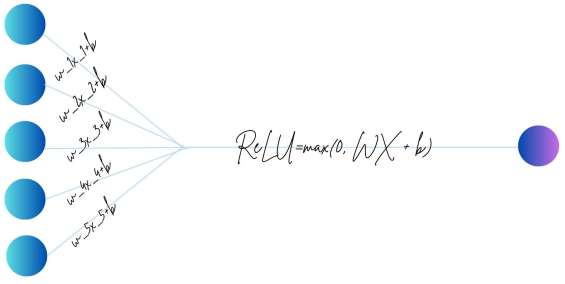

Figure 3: Neuron Activation

To clarify, the input X of a neuron is modified as a weighted sum, $WX + b$, where w_i and b are specific for each neuron. If the previous layer has dimension n and the later one m, then W is a matrix of size $m \times n$ and b a vector of dimension m. Once the input is modified, the neuron is activated by an activation function, where a common one is the ReLU function. Here x_i is the output of a neuron in the previous layer and w_i its associated weight:

$$\text{ReLU}(\sum_i w_i x_i + b) = \max(0, \sum_i w_i x_i + b)$$

In the very last layer, each alternative is assigned a score (z_i) that represents how much information the network has detected for that alternative. These scores are then processed through a function softmax(Z) = $[..., \frac{e^{z_j}}{\Sigma_i e^{z_i}}, ..]$, resulting in a probability distribution from which the final answer is sampled.

$$f(x;\theta) = [\frac{e^{z_1}}{\Sigma_i e^{z_i}}, ..., \frac{e^{z_n}}{\Sigma_i e^{z_i}}] = [0.002, ..., 0.3]$$

Since the prediction is based on the input x and the parameters θ of the network, it can be said to be a function of those $f(x;\theta)$.

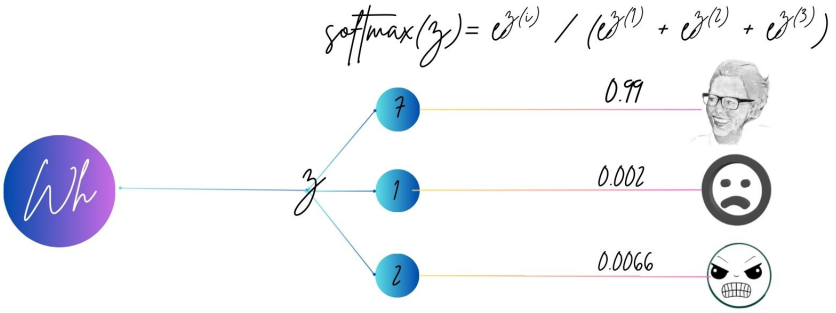

Figure 4: Probability Distribution

To get a better intuition of the weights, we will look at how they determine the activation of neurons in the next layer.

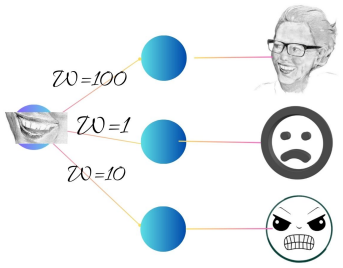

Figure 5: Weights

Suppose that the neuron representing a happy mouth is very active. Then, we do not want the angry or sad face to get a high score. In order to account for this, their corresponding weights have to be low. But since an angry face might come with an open mouth, it's corresponding weight has to be slightly higher, *10*, than the one associated with the sad face, *1*.

In modern deep neural networks, these models contain billions of weights and biases. At first, they are randomly distributed, but once the training progresses, they are updated by algorithms such as backpropagation and gradient descent.

2.1 Gradient descent

In order for the network to provide better answers the parameters have to be updated to better fit the training-data. To achieve this, one constructs a suitable *loss-function* which calculates the cost of the prediction. Namely; the predicted distribution $f(x;\theta)$ is compared to the real one $Y = [..., y^i, ..]$ defined by the training set. From this function, one finds the direction from which the parameters should adhere in order to minimize the cost.

In particular, one considers the average cost of all training samples:

$$\mathbb{E}_{(x,y)\sim\hat{p}_{data}} L(f(x;\theta), y) = \frac{1}{m}\sum_{i=1}^{m} L\left(f(x^{(i)};\theta), y^{(i)}\right)$$

[GBC16, chap. 8, p. 273]
Here L stands for the chosen loss-function that calculates how much the predictions cost.

In order to minimize the cost, we want to find the direction of a unit vector **u** in which the loss decreases the most. If we set alpha to zero, the derivative of such a vector can be written as follows:

$$\frac{\partial}{\partial \alpha} f(x + \alpha u) = u^\top \nabla_x f(x) = \|u\|_2 \, \|\nabla_x f(x)\|_2 \cos\theta$$

Since $\|u\|_2 = 1$ and $\|\nabla_x f(x)||$ is a constant, this is minimized when $cos\theta = -1$.

Therefore, the direction in which the function decreases the most is in the direction of the negative gradient. The parameters θ are adjusted accordingly, where ε is the learning rate:

$$\theta = \theta - \varepsilon \nabla_x f(x)$$

[GBC16, chap. 4, p. 83]
An iteration of this process is what constitutes the algorithm. However, in practice this algorithm can be quite time consuming, which is why one often uses the so called *stochastic* gradient descent.

2.2 Stochastic-gradient descent

Instead of calculating the gradient from all sample-data, one uses a mini-batch, namely; a random sample of the training-set. The estimated gradient of a batch of size w would be:

$$\hat{g} \leftarrow \frac{1}{w} \nabla_\theta \sum_{i=1}^{w} L\Big(f(x^{(i)}; \theta),\, y^{(i)}\Big)$$

The real gradient descent can use a constant learning rate since the gradient becomes smaller as it approaches the minimum. SGD does not necessarily become smaller because it includes some noise due to randomness, which is why one often takes smaller steps.
[GBC16, chap. 8, p. 278 & 290-291]

2.3 Challenges

Although gradient descent is a promising algorithm, it does come with some challenges. Since the loss-function typically cannot be expected to be convex, there are often several local minima, and it can be difficult to determine when one minimizes

the function. Therefore, we often aim for low values rather than minima. Furthermore, the landscape is often conflicting, and the gradient might oscillate, that is, taking som detours before arriving at the minimum as seen in figure 7.

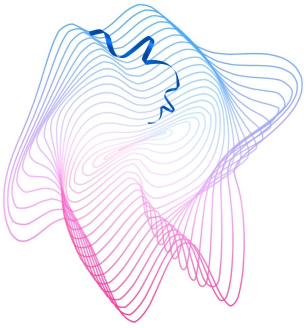

Figure 6: Conflicting curvatures - inspired by Deep Learning, p. 89

In addition, the stochastic gradient descent can be further delayed by the noise that comes with the variance.

Another problem which may arise is the risk of overfitting, namely when the gradient optimizes the training data to well, making it sensitive to details rather than picking up regularities. This results in the model performing badly on unseen data and is said to have poor generalization skills.

3 Large language models

Many neural-networks of today are designed to communicate with a user, which is the case for large language models. These models belong to a class of neural networks that, in short, aims to predict the next word. In order for the model to mathematically process the input, the text-string is divided into tokens which are represented as high-dimensional vectors. The text can then flow through the network to produce a probability distribution over the vocabulary from which the next word is sampled. The generation of text is simply this process over and over again. The following description focuses on decoders that analyze tokens from left to right.

3.1 The internal structure

By drawing on Daniel Jurafskys and James H. Martins book "Speech and Language Processing"[JM26], we will go through the very construction of these probability distributions. Despite predictions becoming more accurate with training, the construction itself remains the same regardless of how much training is given. At first, the model receives a string of text which is divided into tokens.

$$\text{'The bus was late'} \rightarrow \text{['The', 'bus', 'was', 'late']}$$

The tokens are then *embedded* into vector-representations that capture their meaning. Firstly, each token receives an index with a corresponding one-hot vector. By assembling these vectors in a matrix $N \times d$ (N being the number of tokens and d the dimension of the embeddings) and multiplying it by the embedding matrix, one obtains vector-representations for each token.

As the model produces a probability distribution across the whole vocabulary based on these vector-representations, the network can be seen as a class of functions between vector spaces.

$$R^{N \times d} \rightarrow R^{|V|}$$

The embeddings, along with the parameters in later layers, are updated when the model is trained to better predict the next word. Once a token has received its initial representation, it is updated by a series of transformer blocks, where it is said to flow through a residual-stream [chap. 8, p.1][JM26]. The two main layers

within each block are a multi-head attention layer and a feedforward layer. In the attention layer, the tokens interact with each other to gather contextual information. To illustrate, let us consider the sequence *'The painting did not dry in the studio because it was –'*. 'It' could refer to different objects depending on the continuation of the sentence:

'The painting did not dry in the studio because it was foggy'

'The painting did not dry in the studio because it was made of plastilina'

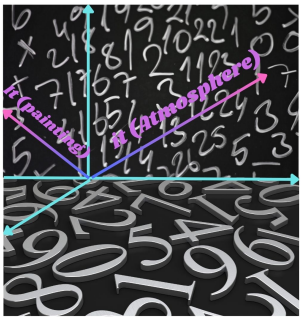

Figure 7: Different representation for 'it'

Such context sensitivity is captured by the "attention-heads", where each token plays three roles. Namely; as the element being processed (query), as a comparing token (key), and as a value to update the given token (value). To determine the query, key, and value vectors, each token is projected onto the weight-matrices W_q, W_k, W_v.

$$q_i = x_i W_Q \quad k_i = x_i W_K \quad v_i = x_i W_V$$

To produce the attention-head for a single token one calculates the scores between the given query and each key to determine their similarity.

$$\text{score}(\text{query}_{\text{it}}, x_j) = \frac{q_{it} \cdot k_j}{\sqrt{d_k}}, \forall j \leq 10$$

These scores are then processed through a softmax function to produce a distribution which determines how much each value vector should contribute to the attention-head of the given token.

$$\alpha_{10,j} = \text{softmax}\Big(\text{score}(\text{query}_{\text{it}}, x_j)\Big), \quad \forall\, j \leq 10$$

The attention head for the given token is calculated by adding the value vectors in proportion to their share in the distribution.

$$\text{head}_{\text{it}} = \sum_{j \leq 10} \alpha_{10,j} v_j$$

Lastly, one reshapes the attention-head to the same dimension as the initial representation of the token by projecting it onto the matrix W_{O}:

$$a_{\text{it}} = \text{head}_{\text{it}} W_O$$

The attention-head is then added to the vector-representation, making it contextualized. However, since there are many contextual relationships to consider, we need several of these processes, which is the idea behind multi-head attention. These are computed in parallel and gathered to produce the final attention of a token:

$$a_i = (\text{head}_i^1 \oplus \text{head}_i^2 \oplus \cdots \oplus \text{head}_i^A)\, W_O$$

[JM26, chap. 8, p. 6] Once the attention-heads are added to the initial-embeddings, the tokens with their new meanings are processed individually in a feedforward-layer where the token-representations are further updated.

$$\text{FFN}(x_i) = \text{ReLU}(x_i W_1 + b_1) W_2 + b_2$$

As the large language model consists of several transformer blocks, this process is repeated until it reaches its last block. Lastly, the tokens enter the Language Modeling Head where they are *unembedded* in order to create a probability distribution over the vocabulary.

3.2 Generation of text

One way to generate a text from the probability distribution is by *greedy decoding*, where one always chooses the token with the highest probability. However, this often results in repetitive responses. Instead, we often use *random sampling*, where words are sampled from the distribution. To reduce randomness, the higher probabilities can be enhanced, using *temperature-sampling*:

$$y = \text{softmax}\left(\frac{u}{\tau}\right) = \frac{e^{\frac{u}{t}}}{\sum_k e^{\frac{i}{t}}}$$

[JM26, chap. 7, p. 20]

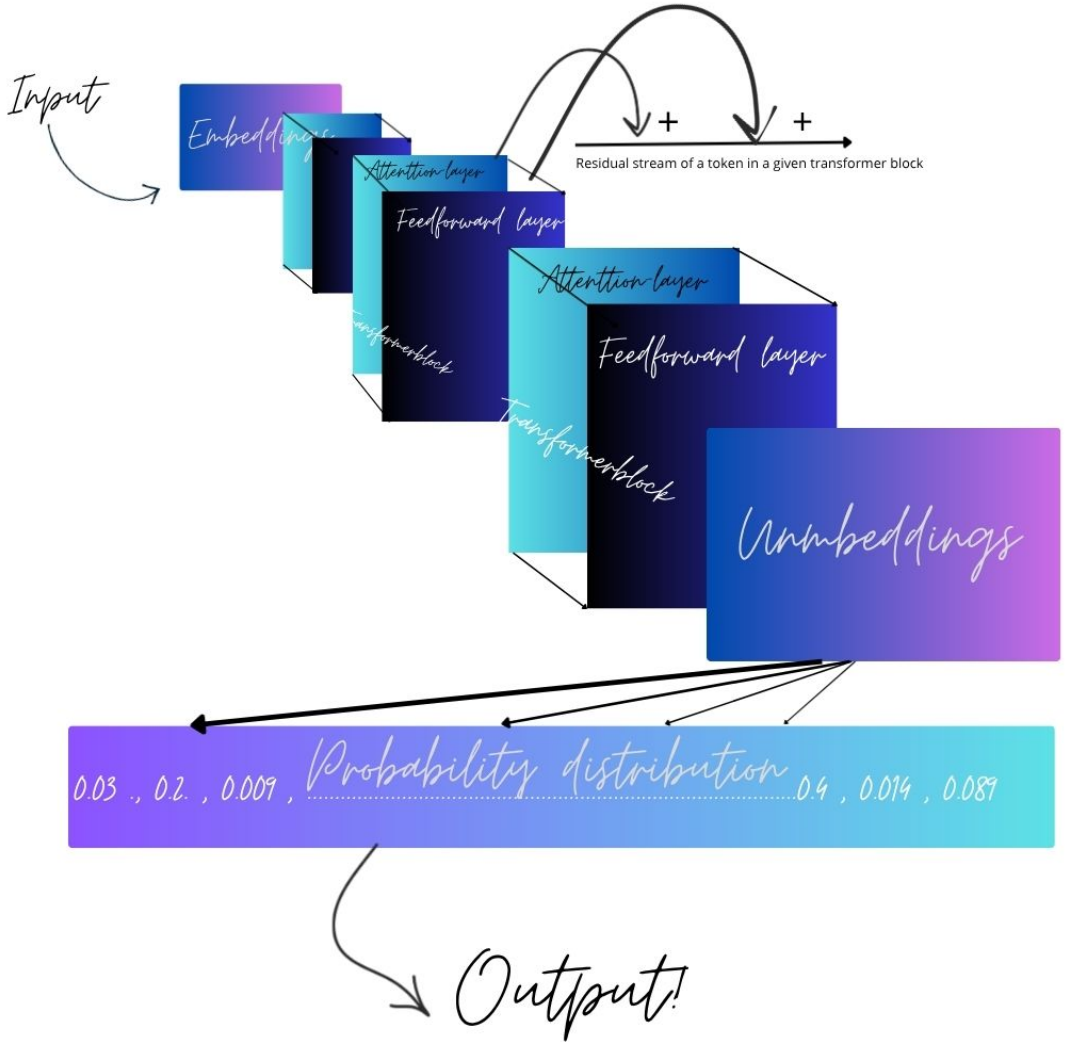

Figure 8: The internal structure - inspired by Grant Sanderson & Daniel Jurafsky et al.

3.3 Training of large language models

The training of large language models can be divided into three stages; pretraining, fine-tuning, and alignment. Initially, the parameters are set at random, but as

training progresses, they are tuned to better predict the next word. This regards both the weights and biases within the transformer-blocks as well as the embeddings.

In the first step, the model is provided with semantics by letting it predict words over and over again. The training is self-supervised, meaning that its prediction is not compared to any installed label, but rather to the word that actually follows in the given sequence. In particular, the model reads a sequence of words and, at each word, the next is predicted and compared with the following. The chosen loss-function, often the *cross-entropy*, calculates the cost of each prediction. Then these are summarized and divided by the number of words. With algorithms such as gradient descent and backpropagation, the parameters are updated in order to lower the average cost and make better predictions.

$$L = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{y}_t[w_{t+1}]$$

[chap. 7, p.15-16][JM26]
In general terms, the pretraining phase is just the model reading text. However, the model can be further specialized in the fine-tuning step, where it is trained on additional data focused on certain domains. It can also be trained to align with different values in the alignment phase. Here, the network is trained on text combined with labels of accepted or rejected responses.

3.4 Scaling Laws

Based on empirical observations, the performance of these models is said to be determined by three factors. Namely; the amount of non-embedding parameters N (weights and biases), the size of the dataset size D, and the compute budget C [Kap+20, p. 4]. As the number of these factors scales, the loss function is said to decrease by the following laws:

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$$

$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

Where N, D, C are the sizes and $N_c, D_c, C_C, \alpha_N, \alpha_D, \alpha_C$, constant values based on the unique structure of the network. This implies that one should be able to predict the loss for certain values of these contributing factors. [chap. 8, p. 22][JM26]

3.5 A semantical interpretation

As discussed above, the input of a large language model is translated to a set of high-dimensional vectors. These can be seen as directions in the embedding space that capture the meaning of the word. Furthermore; the embedding space of models such as WordVec has been detected to capture relational directions. For example, a linear combination of representations of *'King'*, *'Man'* and *'Woman'* results in the vector-representation of *'Queen'*:

$$\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} = \overrightarrow{Queen}$$

[chap. 5, p. 17][JM26]
Nevertheless, with techniques such as autoencoders, interpretable activation patterns denoted as features have been detected in ChatGPT 4. Examples of such where "Price Increases" and "Human Imperfection". [Gao+24].

4 Random graphs and thresholds

In this section, we will study random graphs using the Béla Bollobás and Robert Morris book *Basic Graph Theory* [BM26] and the Goeffrey Grimmets book *Probability on graphs* [Gri10]. The study of random graphs examines the behaviors of graphs in which nodes are connected through random processes. In this paper, we will focus on the Erdős–Rényi graph and study the occurrence and growth of a giant component.

4.1 Erdös-Renyi graph

An Erdös-Renyi graph, denoted as G(n,p), is a probability distribution in which each potential edge is an independent Bernoulli variable with parameter $p = \frac{\lambda}{n}$, where λ is a constant and n the number of nodes. This is also known as the sparse regime, where the expected number of neighbors, $(np = \lambda)$, is limited. Furthermore, the probability that the graph has a certain property, such as a giant component or a triangle, varies with the local parameter λ.

4.2 Sharp and coarse thresholds

A particular area of interest is the study of thresholds, namely; when the probability of the graph having a property seems to suddenly increase only by a small change in the local parameter. If the pace of this transition is rather fast, one often talks about a sharp-threshold, while if it is a bit slower, it is often the case of a coarse threshold. In particular, a coarse and a sharp threshold can be separated by the following definitions:

Definition 4.1. Let $\mathcal{P} \subseteq \mathcal{G}$ be a graph property. We say that a function $f(n)$ is a *threshold* for $\mathcal{P}$ if

$$\lim_{n\to\infty} \mathbb{P}\Big(G(n,p) \in \mathcal{P}\Big) = \begin{cases} 0 & \text{if } p \ll f(n), \\ 1 & \text{if } p \gg f(n). \end{cases}$$

[BM26, p. 181] Whereas a sharp threshold is defined as:

Definition 4.2. Let $\mathcal{P} \subseteq \mathcal{G}$ be a graph property and let $f(n)$ be a function. We say

that $\mathcal{P}$ has a *sharp threshold* at $f(n)$ if, for every $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\Big(G(n,p) \in \mathcal{P}\Big) = \begin{cases} 0 & \text{if } p \le (1-\varepsilon)f(n), \\ 1 & \text{if } p \ge (1+\varepsilon)f(n). \end{cases}$$

[BM26, p. 186] To further understand, we will consider examples of both kinds. An example of a coarse threshold is the appearance of a triangle K_3:

Theorem 4.3 (5.1.1)**.** *Let $G \sim G(n,p)$. Then*

$$\mathbb{P}(K_3 \subset G(n,p)) \longrightarrow \begin{cases} 0, & \text{if } p \ll \dfrac{1}{n}, \\ 1, & \text{if } p \gg \dfrac{1}{n}, \end{cases} \qquad \text{as } n \to \infty.$$

[BM26, p. 175] The proof is divided into two steps regarding the sub- and supercritical regimes.

Proof. In order to prove the subcritical regime, we use the Markov's inequality to construct an upper bound for the probability of the existence of a triangle. Leaving X be the number of triangles in the graph and noting that there are $\binom{n}{3}$ triangles that all exist with probability p^3, the expected number of triangles is $E[X] = p^3\binom{n}{3}$. By Markov's inequality it follows that:

$$P(X \ge 1) \le E[X] = p^3 \binom{n}{3}$$

Furthermore, since:

$$p^3 \binom{n}{3} \le (pn)^3 \to 0$$

as $n \to 0$, a triangle exists with low probability.

To prove the supercritical regime, we use the Chebyshev inequality and find an upper-bound for the variance. First, we note that:

$$\text{Var(X)} = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \text{ and that } X^2 = \Big(\sum_{T\in\mathcal{T}_n} \mathbf{1}_{\{T\subset G(n,p)\}}\Big)^2 = \sum_{S,T\in\mathcal{T}_n} \mathbf{1}_{\{S\cup T\subset G(n,p)\}}.$$

With this in mind, we can begin by finding an upper bound for $\mathbb{E}[X^2]$ by considering the different ways in which two triangles can intersect. They can be either the same

triangle, share an edge, or share a node. The first case corresponds to $\mathbb{E}[X]$. For the second, this means that the two triangles occupy 4 nodes but 5 edges, leaving us with a sum of at most p^5n^4. Since the set of all disjoint triangles includes those that share a node, the sum of the last case is at most $p^6\binom{n}{3} = \mathbb{E}[X]^2$.

By linearity of expectation it follows that:

$$\mathbb{E}[X^2] \leq (\mathbb{E}[X])^2 + p^5n^4 + \mathbb{E}[X]$$

This, combined with our conditions for p, leaves us with the following:

$$\mathrm{Var}(X) \leq \mathbb{E}[X] + p^5n^4 \ll \mathbb{E}[X]^2$$

If we let $t = \frac{\mathbb{E}[X]}{2}$, it follows from the Chebyshev inequality that:

$$\mathbb{P}(X = 0) \leq \mathbb{P}\left(X \leq \frac{\mathbb{E}[X]}{2}\right) \leq \mathbb{P}\Big(|X - \mathbb{E}[X]| \geq \frac{\mathbb{E}[X]}{2}\Big) \leq 4 \cdot \frac{\mathrm{Var}(X)}{\mathbb{E}[X]^2} \to 0$$

Since $\mathbb{E}[X] = p^3\binom{n}{3} \to \infty$, when $n \to \infty$ given our conditions for p.

Hence, there is a high probability that the graph has a triangle under the given conditions for p. □

An example of sharp thresholds is the property of the graph being connected:

Theorem 4.4 (Connectivity threshold)**.** *For any function $\omega = \omega(n) \gg 1$,*

$$\mathbb{P}\Big(G(n,p) \text{ is connected}\Big) \to \begin{cases} 0 & \text{if } p \leq \dfrac{\log n - \omega}{n}, \\ 1 & \text{if } p \geq \dfrac{\log n + \omega}{n}, \end{cases}$$

as $n \to \infty$.

[BM26, p. 184] As for the theorem above, the proof is divided into two steps for each regime.

Proof. When trying to prove the subcritical regime, we use the fact that a connected graph implies that no vertex is isolated. Namely; we construct an upper bound for the probability that there are no isolated vertices. If we let X denote the number of

isolated vertices, the expected number of such vertices, with our conditions for p, is:

$$\mathbb{E}[\mathbb{X}] = n \cdot (1-p)^{n-1}$$

$$\mathbb{E}[\mathbb{X}] = n(1-p)^{n-1} \geq n \cdot e^{-pn+o(1)} \to \infty \ (1^*)$$

as $n \to \infty$. Due to the fact that $e^{-x} = 1 - x + O(x^2) \quad$ as $x \to 0$. In order to use the Chebyshev inequality, we construct an upper bound for the variance by first considering pairs:

$$\mathbb{E}[X^2] = \sum_{u,v \in V(K_n)} \mathbb{P}(u \text{ and } v \text{ are both isolated})$$

To determine the sum, we will have to consider two cases. Either $u = v$ which is the same as $\mathbb{E}[X]$, or $u \neq v$. Since the probability of two separate vertices being isolated is $(1-p)^{2n-3}$ and there are $\binom{n}{2}$ many pairs, we obtain an upper bound for $\mathbb{E}[X^2]$:

$$\mathbb{E}[X^2] \leq \mathbb{E}[X] + n^2(1-p)^{2n-3}$$

Hence, we can restrict the variance:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \leq \mathbb{E}[X] + \frac{p}{1-p}\,\mathbb{E}[X]^2 \ll \mathbb{E}[X]^2$$

Where the last inequality is given by (1^*). If we apply the Chebyshev inequality with $t = \frac{\mathbb{E}[X]}{2}$, we obtain:

$$\mathbb{P}(X = 0) \leq \mathbb{P}\left(X \leq \frac{\mathbb{E}[X]}{2}\right) \leq \frac{4\,\mathrm{Var}(X)}{\mathbb{E}[X]^2} \to 0 \quad \text{as } n \to \infty.$$

Hence, the probability of connectivity is low.

To prove the supercritical range, we use the fact that if G is not connected, then at least one of the sets Y_k, containing all components of size k for $k = 1, 2, \ldots, \frac{n}{2}$, will be non-empty. Firstly, note that if A is a component of size k, then there are no edges between A and A^c, which could appear in $k(n-k)$ many ways. And since we can have $\binom{n}{k}$ components of size k, we have that:

$$E[Y_k] \leq \binom{n}{k}(1-p)^{k(n-k)}$$

Furthermore, since $(1-x) \leq e^{-x}$ we have the following.

$$\binom{n}{k}(1-p)^{k(n-k)} \leq \Big(\frac{en}{k} \times e^{-p(n-k)}\Big)^k$$

We can also see that $\frac{e^{pk}}{k}$ is maximized when $k = 1$ or $k = n/2$, and it follows that:

$$E[Y_k]^{\frac{1}{k}} \leq \max(en \times e^{-p(n-1)}, 2e \times e^{-pn/2}) \leq e^{-\omega+2}$$

Since $pn \geq logn + \omega$, and assuming that $\omega = \omega(n)$ does not grow too quickly with n. By Markov's inequality it follows that:

$$\mathbb{P}(\text{G(n,p) is not connected}) \leq \sum_{k=1}^{n/2} \mathbb{E}[Y_k] \leq \sum_{k=1}^{n/2} (e^{-\omega+2})^k \to 0$$

as $n \to \infty$ □

Another example of a sharp threshold is the occurrence of a giant component.

Theorem 4.5. *11.1 Theorem [82] We have that*

$$\frac{1}{n}X_n = \begin{cases} o_p(1), & \text{if } \lambda \leq 1, \\ \alpha(\lambda)\Big(1 + o_p(1)\Big), & \text{if } \lambda > 1, \end{cases}$$

where $\alpha(\lambda)$ is the survival probability of a branching process with a single progenitor and a family-size distribution $\mathrm{Po}(\lambda)$.

[Gri10, p. 207] For convenience, we will substitute α with δ, as the latter notation is used throughout the paper. In this section, we will avoid studying the proof in great detail. Instead, we will distinguish the occurrence of a giant component with the growth of the giant after a certain threshold. Although the giant component does occur sharply, its growth is considered a coarse threshold and approximately linear as $\lambda \to 1^+$.

Furthermore, the probability of extinction $\eta(\lambda) = 1 - \delta(\lambda)$ is the smallest non-negative root of equation $s = \mathbb{E}[s^X]$, see the proof [GS01, p.173].

$$s = \mathbb{E}[s^X] = \sum_{k=0}^{\infty} s^k \, \mathbb{P}(X = k) = \sum_{k=0}^{\infty} s^k \, e^{-\lambda} \frac{\lambda^k}{k!} = e^{-\lambda} \sum_{k=0}^{\infty} \frac{(\lambda s)^k}{k!} = e^{-\lambda} e^{\lambda s} = e^{\lambda(s-1)}.$$

By substituting s with $1-\delta$, we will get the equation for the survival probability and hence the density of the giant:

$$1 - e^{-\lambda\delta} = \delta.$$

In particular, the growth of the giant is a coarse threshold in the sense that the density of the giant going from ϵ to $1-\epsilon$ requires λ to go from close to 1 to a large constant C dependent on epsilon:

$$1 - e^{-C(1-\epsilon)} = 1 - \epsilon \Leftrightarrow e^{-C(1-\epsilon)} = \epsilon \Leftrightarrow -C(1-\epsilon) = log(\epsilon) \Leftrightarrow C = \frac{log(\frac{1}{\epsilon})}{(1-\epsilon)}$$

With help from R, we can also solve $1 - e^{-\lambda\delta} = \delta$ numerically for λ within the interval (1-6] with a step size 0.05 and illustrate the gradual growth of δ with λ.

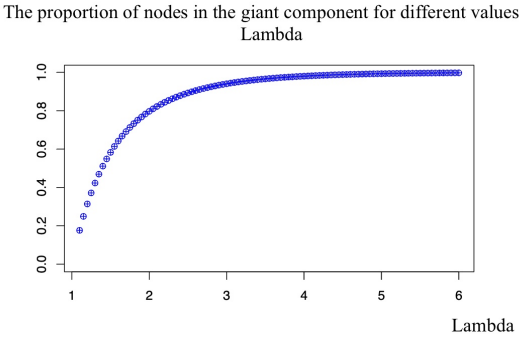

Figure 9: Growth of the giant

It might be difficult to see from the graph, but the growth of the giant is approximately linear when λ is close to 1. Namely; by using taylor-expansion for small x, we can approximate δ with:

$$\delta = 1 - e^{-\lambda\delta} = \lambda\delta - (\lambda\delta)^2(1/2 + o(1)) \Leftrightarrow \lambda - 1 = \delta\lambda^2\left(\frac{1}{2} + o(1)\right) \Leftrightarrow$$

$$\delta = 2(\lambda - 1)(1 + o(1)) \text{ using the fact that } \lambda^2 \approx 1 \text{ when } \lambda \to 1^+$$

Nevertheless, as $\delta \to 0$ once $\lambda \to 1^+$, we can approximate the derivative and obtain:

$$\delta' \approx \frac{\delta}{(\lambda - 1)} = 2(1 + o(1)) \approx 2$$

Hence, the growth is approximately linear when $\lambda \to 1^+$.

5 Can we expect threshold-like behavior in large language models?

As thresholds are well documented in random graphs, similar behavior has been detected in large language models. What has attracted great attention is the instant improvement on certain benchmarks where the models are described as demonstrating emergent abilities that exhibit threshold-like behavior.

5.1 Emergent abilities or a mirage?

In particular Wei. et al claim that these models exhibit emergent abilities, which they define as the following:

"An ability is emergent if it is not present in smaller models but is present in larger models."[Wei+22, p. 2], which they define as abilities apparent in larger models but not in smaller ones. However, this interpretation is criticized by Schaeffer et al. [SMK23]

Namely, the abilities seem to go from non-existing to existing by scaling the model, and cannot be predicted by extrapolating on smaller models with scaling laws. By gathering results from several Large Language Models, such as LAMBDA, GPT-3 and Gopher, tested on different benchmarks, they managed to show emergent abilities according to their definition. In particular, they detected a sudden increase in performance while scaling the amount of computation measured in FLOP.

As a response, Schaeffer et al. question whether emergent abilities appear at all or whether they are a measurement artifact. Their critique relies on the above definition of emergent abilities, as well as descriptions of subsequent work. From these, they claim that these definitions identify two defining properties; sharpness and unpredictability. Furthermore, their hypothesis is that these abilities increase gradually and predictably by scaling laws and that the sharp improvement is induced by certain metrics. They also suggested that smaller models are underestimated, since they are tested on too little data. [SMK23, p. 2].

Firstly, they support their thesis with a theoretical illustration. By assuming the

cross-entropy to follow a scaling law in regard to the amount of parameters, they show how non-linear metrics such as “Accuracy" (1 if all tokens are correct and 0 otherwise) make the improvement seem emergent when it is, in fact, gradual.

Namely, by assuming;

$$L_{CE}(N) = -\log \hat{p}_N(v^*) = (\frac{N}{c})^{\alpha}$$

They find that the probability of getting a single token correct changes gradually with scale:

$$p(\text{single token correct}) = \exp(-\text{LCE}(N)) = \exp\left(-\left(\frac{N}{c}\right)^{\alpha}\right)$$

Requiring all tokens to be correct (accuracy) would require a large single token probability, displaying a sharp and unpredictable improvement:

$$\text{Accuracy}(N) \approx (p_N(\text{single token correct}))^L = \exp\left(-\left(\frac{N}{c}\right)^{\alpha}\right)^L$$

[SMK23, p. 3-4]
In contrast, with linear measurement such as “Token Edit Distance”, which measures the expected number of incorrect tokens, the improvement is displayed as smooth and predictable.

$$\text{Token Edit Distance}(N) \approx L(\text{1-single token correct}) = L(1 - \exp\left(-\left(\frac{N}{c}\right)^{\alpha}\right)$$

[SMK23, p. 3-4]
Secondly, they test their hypothesis with experiments on InstructGPT/GPT-3 models that solve arithmetic tasks. They show that emergent abilities appear with non-linear metrics (accuracy) when testing on longer sequences but not with the linear metric- Token Edit Distance. However, when adding test-data to the smaller model, the emergent abilities seem to evaporate even with metrics such as accuracy.

Since other model families are not publicly query-able, they analyze already published results from BIG-bench. They found that model-tasks which show emergent abilities are tied to non-linear/discontinuous metrics, and disappear under linear- and/or continuous metrics. Lastly, the authors show how emergent abilities can be

induced in other neural networks that have not yet demonstrated emergent abilities. They focused on those who reconstruct images (CIFAR-100) and recognize signs (Omniglot).

In conclusion, they claim that "emergent abilities may be creations of the researcher's choices, not a fundamental property of the model family on the specific task"[SMK23, p. 9]. However, they admit that large language models might exhibit emergent abilities, but argue that those found are mirages induced by the research's choice of measurement.

5.2 Documented threshold-like behavior

Nevertheless, there are documented threshold-like behaviors in neural networks that are not targeted by the above debate, as they concern learning rate rather than size. Namely; in the article "Learning Threshold Neurons via the 'Edge of Stability" Bubeck et al. manage to rigorously show that threshold neurons crucial for generalization appear under a specific learning rate. [Bub23]

In particular, they analyze the gradient descent for simplified models of two layer neural networks for which they provably show a phase-transition for when threshold neurons are learned. In addition, they demonstrate the *edge-of-stability phenomenon*; when the gradient descent is unstable and the loss decreases non-monotonically due to a high learning rate. As they explain how threshold neurons appear within this regime, they challenge traditional views of low learning rates close to the gradient flow.

At first, they provide the reader with empirical findings to support their hypotheses. Namely, they use a two layered ReLU network to solve a simplified sparse coding problem. That is, to find the sign of the signal in a sparse vector x:

$$x^{(i)} = \lambda y^{(i)} e_{j^{(i)}} + \xi^{(i)} \in \mathbb{R}^d$$

Where $y \in [-1, 1]$ is the sign, $\lambda > 1$ the strength of the signal, $j(i) \in [d]$ a random index, and $\xi^{(i)}$ is Gaussian noise. The task is to filter out the noise to find the signal sign. They consider a simplified case with a known basis for a two layered network to solve:

$$f(x; a^-, a^+, b) = a^- \sum_{i=1}^{d} \text{ReLU}(-x[i] + b) + a^+ \sum_{i=1}^{d} \text{ReLU}(x[i] + b).$$

[Bub23, p. 2]

In order to identify the correct sign of the signal, the bias needs to threshold out the noise from non-relevant coordinates, and a^-, a^+ needs to produce the correct sign. Hence, threshold neurons are ReLU-neurons with a negative bias b in the first layer.

Using gradient descent on a logistic loss function, they train the model with learning rates;

$$n_1 = 2.5x10^{-5}, n_2 = 2.5x10^{-4}, n_3 = 2.5x10^{-3},$$

The result shows that a higher learning rate might lead to oscillation in the parameters and a non-monotonic loss function during training, but that it is crucial for threshold neurons to appear.

The threshold units are also coupled with a higher test accuracy. In fact, there seems to be a phase-transition at $n \approx 0.0006$, where both threshold neurons appear as well as a sudden increase in test-accuracy.[Bub23, p. 2]

The observations above are on a simple ReLU network, but they argue that "these findings are indicative of behaviors observed in practical neural network training settings" [Bub23, p. 3]. That is, they tested different learning rates on a deep neural network trained on CIFAR-10, as well as the full sparse coding problem on a two-layer ReLU network, with an unknown basis. The same result appears; a non-monotonic loss function and a decrease in bias.

In addition to these empirical findings, they provide rigorous results on simplified models which they label as "single-neuron linear network" and "mean model".

The first model takes two parameters:

$$f(x, y) = l(x \times y)$$

[Bub23, p. 6]

By studying the gradient descent dynamics, they managed to show that there are

two regimes. In the first regime, the GD remains close to the gradient flow, whereas in the second it enters the edge of stability, diverging from the gradient flow. In the latter regime, they managed to show that x oscillates wildly as y decreases, which resembles the oscillation of the weights $(a^- + a^+)$ while b decreases in a ReLU network.

Then, they use their insights on a so-called mean-model which they argue the ReLU network to be well-approximated by. They managed to prove a critical value for when threshold neurons are learned:
For any $\delta > 0$,

- if $\eta \leq (8-\delta)\pi/d^2$, then the mean model does not learn threshold neurons: the limiting bias satisfies $|b_\infty| = \mathcal{O}_\delta(1/d^2)$.

- if $\eta \geq (8+\delta)\pi/d^2$, then the mean model enters the EoS and learns threshold neurons: the limiting bias satisfies $b_\infty \leq -\Omega_\delta(1)$.

[Bub23, p. 5]
Even though their analysis regards simple models and hence does not specifically address large language models, they argue that the "underlying principles can potentially be applied to more general settings". [Bub23, p. 10]

5.3 Analogies with random-graphs

To better understand Schaeffer's critique of how the real gradual improvement might be manipulated by certain metrics we will provide an analogy to random-graphs. As a large language model cannot be seen as a graph, this will be solely for illustrative purposes.

If we consider the graph $G(n,p)$ where $p = \frac{\lambda}{n}$, and let $\lambda \geq 1$, the graph can illustrate a trained large language model where the parameter λ corresponds to the size of the network. Furthermore, let us illustrate the goal of the large language model, to predict the next word, with the goal of connecting as many nodes as possible. In the following, we will see how measurements such as accuracy and token-edit distance display how we might get closer to this goal by increasing the value of λ.

Let us choose L nodes uniformly and independently and let X be the number of

nodes that are not in the giant component. Then the accuracy-metrics would correspond to

$$\mathbb{P}(X = 0) = \mathbb{P}(\text{every node in giant}) = \mathbb{P}(\text{chosen node in giant})^L = \delta(n,p)^L$$

and the token-edit distance:

$$E[X] = L(1 - \mathbb{P}(\text{chosen node in giant})) = L(1 - \delta(n,p))$$

Where, $\delta(n,p)$ stands for the density of the giant, but since we chose a large value for n, it hardly depends on n. Furthermore, increasing δ to the power L (1000) will move the threshold to the right, and in order to better compare the two thresholds, we will need to rescale the x-axis. In particular, we would like both δ and δ^L, to reach $\frac{1}{2}$ by, say, the first quarter of the x-axis.

For δ this happens at:

$$\frac{1}{2} = 1 - e^{-\frac{1}{2}\lambda} \Leftrightarrow log2 = \frac{1}{2}\lambda \Leftrightarrow \lambda_* = log4$$

And for δ^L:

$$\frac{1}{2} = (1-e^{-\delta\lambda})^L \Leftrightarrow \lambda \approx \frac{1}{\delta}\log\left(\frac{L}{2}\right) \quad \textit{using the fact that } \log(1+x) = x+o(x) \textit{ for small } x$$

With $\delta \approx 1$, this leaves us with $\lambda_* \approx \log\left(\frac{L}{2}\right)$. Rescaling the two x-axes by letting the max value for x be 4 times its corresponding λ_* leaves us with two better comparable graphs.

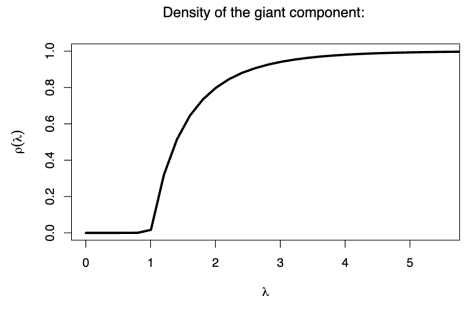

Figure 10: δ

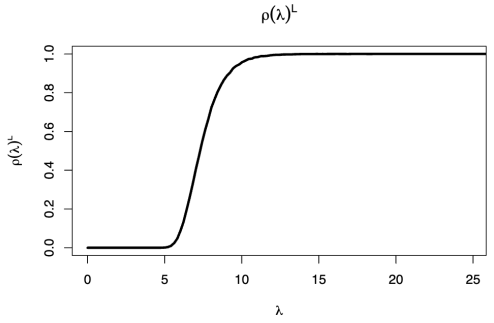

Figure 11: Illustrating accuracy

Furthermore, the negative token-edit distance displays a curve with the same shape

as the growth of the giant component:

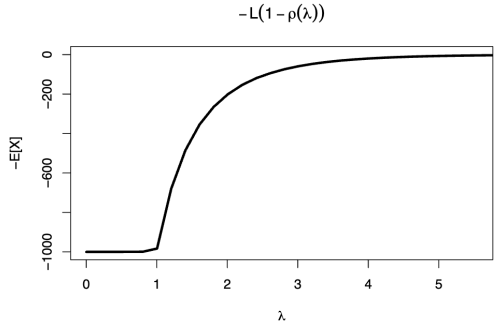

Figure 12: Illustrating TED

To further demonstrate, we can compare the metrics with even higher values of L (10 000). Even though this might not correspond well to the number of tokens, it serves an illustrative purpose and successfully demonstrates our goal of reaching connectivity.

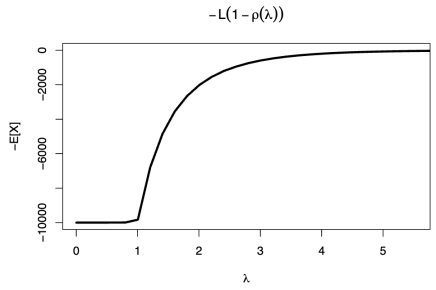

Figure 13: Illustrating TED

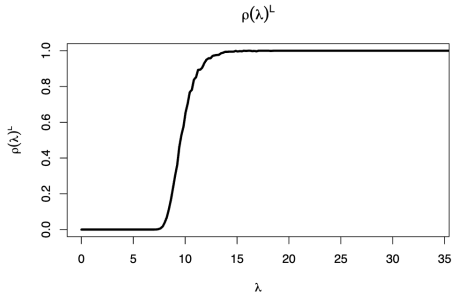

Figure 14: Illustrating accuracy

As in the case of Schaeffer, the metrics of choice do indeed have an effect on the displayed threshold. The first graph (figure 11) suggests that we achieved our goal of connecting as many nodes as possible in a more dramatic way than if we had used the measurement in the latter graph (figure 12). Namely, the fact that the accuracy displays a sharp improvement is illustrated by the phenomenon of a graph being connected having a sharp threshold. In contrast, the fact that the token-edit distance displays a gradual change is illustrated by the coarse growth of the giant component.

6 Discussion

Although large language models are networks, they are too complex to allow for a rigorous analysis similar to that of random graphs. However, there are indeed behaviors of these models that are reminiscent of those in random graphs. Such as instant improvement on certain benchmark only by a small change in size which gives the impression of a sharp threshold. With Schaeffers definition of emergent abilities as unpredictable and sharp, he manages to show how metrics indeed have an effect of the displayed threshold, which is further verified by our analogy. In particular, he explains how metrics such as accuracy, requiring several tokens to be correct, manipulates the loss-function, which is said to follow distinct scaling laws. At the same time, Bubeck et al. manage to rigorously show threshold-like behavior in neural networks. However, as these findings relate to *learning rate* rather than *size*, and that the study does not specifically address large language models, they are not subject to the above debate. What might not also be covered in the above debate is that even though an ability might gradually improve, this does not rule out the possibility of it sharply coming to existence. For example, despite the gradual growth of the giant component, it does indeed come into existence sharply. In this sense, there is the possibility that both conclusions are correct.

7 References

[BM26] Béla Bollobás and Robert Morris. *Basic Graph Theory*. Cambridge University Press, 2026.

[Bub23] Kwangjun Ahn, Sébastien Bubeck, Sinho Chewi, Yin Tat Lee, Felipe Suarez, and Yi Zhang. "Learning Threshold Neurons via the Edge of Stability". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 36. 2023.

[Gao+24] Leo Gao, Jeffrey Wu, Tom Dupré la Tour, and Henk Tillman. *Extracting Concepts from GPT-4*. Tech. rep. OpenAI, 2024. URL: https://openai.com/index/extracting-concepts-from-gpt-4/.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://deeplearningbook.org.

[Gri10] Geoffrey Grimmett. *Probability on Graphs: Random Processes on Graphs and Lattices*. Vol. 1. 3rd printing with corrections, 2012. Cambridge University Press, 2010.

[GS01] Geoffrey Grimmett and David R. Stirzaker. *Probability and Random Processes*. 3rd ed. Oxford University Press, 2001.

[JM26] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. Online manuscript released January 6, 2026. 2026. URL: https://web.stanford.edu/~jurafsky/slp3/.

[Kap+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling Laws for Neural Language Models". In: *arXiv preprint arXiv:2001.08361* (2020).

[Nie19] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2019. URL: http://neuralnetworksanddeeplearning.com.

[SMK23] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. "Are Emergent Abilities of Large Language Models a Mirage?" In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 36. 2023.

[Wei+22] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. "Emergent Abilities of Large Language Models". In: *Transactions on Machine Learning Research (TMLR)* (2022).

Fel:

1. s. 6

$$\theta = \theta - \varepsilon \nabla_x f(x)$$

Fel
Det ska stå:
\(\theta^{'}= \theta - \varepsilon \nabla_\theta L\)

2. s.13

Furthermore, since:

$$p^3 \binom{n}{3} \leq (pn)^3 \to 0$$

as $n \to 0$, a triangle exists with low probability.

Fel! Det ska stå n → \infinity

3. s.14

triangle, share an edge, or share a node. The first case corresponds to $\mathbb{E}[X]$. For the second, this means that the two triangles occupy 4 nodes but 5 edges, leaving us with a sum of at most p^5n^4. Since the set of all disjoint triangles includes those that share a node, the sum of the last case is at most $p^6\binom{n}{3} = \mathbb{E}[X]^2$.

Detta är fel!
det ska stå:
With this in mind, we can begin by finding an upper bound for \(\mathbb{E}[X^2]\) by considering the different ways in which two triangles can intersect. They can be either the same triangle, share an edge, or be disjoint. The first case corresponds to \(\mathbb{E}[X]\). For the second, this means that the two triangles occupy 4 nodes but 5 edges, leaving us with a sum of at most \(p^5n^4\). The sum of the last case is at most \(p^6\binom{n}{3}^2=\mathbb{E}[X]^2\).

4. s.16

$$\mathrm{Var}(X) \leq \mathbb{E}[X] + p^5 n^4 \ll \mathbb{E}[X]^2$$

If we let $t = \frac{\mathbb{E}[X]}{2}$, it follows from the Chebyshev inequality that:

$$\mathbb{P}(X = 0) \leq \mathbb{P}\left(X \leq \frac{\mathbb{E}[X]}{2}\right) \leq \mathbb{P}\left(|X - \mathbb{E}[X]| \geq \frac{\mathbb{E}[X]}{2}\right) \leq 4 \cdot \frac{\mathrm{Var}(X)}{\mathbb{E}[X]^2} \to 0$$

Since $\mathbb{E}[X] = p^3\binom{n}{3} \to \infty$, when $n \to \infty$ given our conditions for p.

Hence, there is a high probability that the graph has a triangle under the given conditions for p. □

Fel!

det ska stå:
\[\operatorname{Var}(X) \leq \mathbb{E}[X] + p^5 n^4 \ll \mathbb{E}[X]^2 \text{ ,} \text{ \((1^*)\)}\]
If we let \(t=\frac{\mathbb{E}[X]}{2}\), it follows from the Chebyshev inequality that:
\[
\mathbb{P}(X = 0)
\leq \mathbb{P}\!\left(X \leq \frac{\mathbb{E}[X]}{2}\right)
\leq \mathbb{P}\bigl(|X - \mathbb{E}[X]| \ge \frac{\mathbb{E}[X]}{2}\bigr) \leq 4 \cdot
\frac{\operatorname{Var}(X)}{\mathbb{E}[X]^2}
\to 0
\]
Due to \((1^*)\). Hence, there is a high probability that the graph has a triangle under the given conditions for p.
\end{proof}