The Concept Percolation Hypothesis: Analyzing the Emergence of Capabilities in Neural Networks Trained on Formal Grammars

Anonymous Authors[1]

Abstract

We analyze emergence of capabilities as a function of learning time, i.e., learning curve analysis. Training models on a well-defined, synthetic context-sensitive formal language, we find the existence of precise phases that separate the learning dynamics. Motivated by our results, we propose a qualitative theory grounded in the process of graph percolation that describes a mechanistic basis for how capabilities may be emerging in neural networks as they are trained on increasingly larger datasets.

1. Introduction

Modern neural networks, e.g., large language models (Gemini Team, 2023; OpenAI, 2023; Anthropic, 2023; Touvron et al., 2023) and large vision models (Yu et al., 2022; Betker et al., 2023), exhibit a broad spectrum of capabilities, allowing them to serve as the "foundation" for building several downstream, application-specific systems (Xu et al., 2023). As these models scale, either via addition of more data or via more compute, an intriguing behavior is at times observed: until a certain critical scale is reached, the model may not exhibit some specific capability; however, beyond this point, the capability suddenly "emerges" (Wei et al., 2022; Srivastava et al., 2022; Brown et al., 2020; Yu et al., 2022; Steinhardt, 2023; Pan et al., 2022; Rae et al., 2021). More specifically, the performance of the model on a task meant to evaluate a capability of interest (e.g., a benchmark) witnesses substantial and rapid performance growth as a function of scale, even though the broader training loss (e.g., the next token prediction objective in the case of language models) witnesses minimal, if any, differences (Arora and Goyal, 2023). Empirical evidence in fact suggests that several capabilities can emerge simultaneously (Wei et al., 2022; Wei, 2022).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

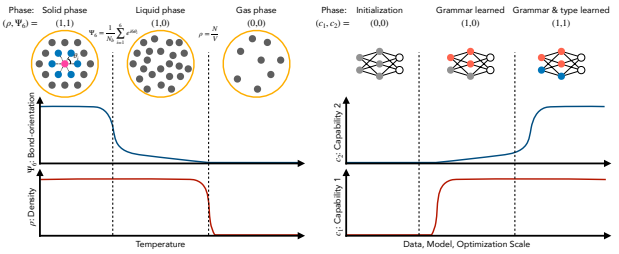

Figure 1: **Characterizing phases of a model by capability parameters.** In physics, defining a phase and detecting the associated transition often involves identifying an *order parameter*, which is typically zero in the absence of a particular structure and becomes finite abruptly during a phase transition. Similarly, in machine learning, it is crucial to design evaluation metrics specifically targeted at detecting the emergence of capabilities. This approach is essential for defining the *phases of a model* by the set of capabilities it possesses.

Beyond mere scientific curiosity for what causes such emergent capabilities to occur, understanding and "taming" emergent capabilities is an extremely important problem from a risk management perspective (Kaminski, 2023; Ganguli et al., 2022; Anwar et al., 2024). predicting what capabilities a model might possess at a certain scale can be difficult to assess until the model is developed, undermining risk-centric regulation frameworks for AI. Accordingly, a few recent works have made attempts at devising models to explain the factors in neural network training pipelines that yield emergent capabilities. For example, Okawa et al. (Okawa et al., 2023) and Arora et al. (Arora and Goyal, 2023) implicate the underlying compositional structure of the task and data as a cause of emergent curves for learning a capability, providing evidence for similar arguments made by Srivastava et al. (Srivastava et al., 2022) and Wei et al. (Wei et al., 2022). In the meantime, Schaeffer et al. (Schaeffer et al., 2023) argue emergent abilities are in fact a "mirage" and an artifact of poorly defined evaluations, claiming that the model is undergoing continuous, persistent improvements. Mechanistic interpretability analyses of seemingly emergent phenomena such as grokking (Nanda et al., 2022; Liu et al., 2023) provide credence to this claim by identifying "hidden

progress measures", albeit finding such progress measures in the first place can be exceedingly difficult (Barak et al., 2022; Anwar et al., 2024). Meanwhile, recent work by Chen et al. (Chen et al., 2024) demonstrates that even continuous metrics can demonstrate sudden changes in value, with such drops being correlated with the model learning a new capability. This undermines the claim that emergent capabilities are merely an artifact of our evaluation protocols. Taken together, these disparate results render the (in)existence of emergence as an extremely unclear phenomenon in machine learning. However, we argue that, at its core, the concept of emergence has never been defined in the context of machine learning. This has led to distinct mechanisms causing sudden shifts in the model performance to be all labeled as "emergence". What is the *phenomenology* that the term "emergence" is meant to capture? Is it merely a sudden increase in performance, or broader than that: e.g., are there truly systematic changes akin to *phase transitions* in physics occurring in a model's behavior? If so, is there an *order parameter* that captures these transitions, which potentially explains the seemingly distinct property changes?

We aim to take a first step towards addressing the questions above by taking a model-experimental system approach. Specifically, following the practice in natural sciences studying complex systems, we design a simple but rich synthetic system that captures and demonstrates the characteristics often ascribed to the phenomenon of interest—emergence—in contemporary literature. Our goal is to show that even this simplistic pipeline can yield sufficient insight and viable hypotheses for explaining, at a mechanistic level, what factors cause emergence. In particular, our model experimental system is grounded in a probabilistic context-sensitive grammar (PCSG) with type constraints that allow only specific pairs of subjects or entities in a sentence to be seen in the context of certain concepts (e.g., a verb like "walking"). We define tasks to be performed upon the samples of this grammar, finding that a model trained to learn the grammar shows emergent learning curves across several metrics (including the continuous next token prediction loss) for individual tasks. Further, we show that we can ascribe vivid interpretations to each point of emergence as an explicit phase of learning. Herein, we find how a minimal change in loss (e.g., $<$0.005 nats) can have noticeable downstream effects, where the underlying causal factor for these effects is the model learning a new capability. Based on our findings, we finally formalize a theoretical model that qualitatively captures the phenomenology of emergence discovered in this work.

2. Related work

While emergence is not formally defined in machine learning, the term, at least in its recent use, is grounded in physics and complex systems (Anderson, 1972; Newman et al., 2001; Newman, 2003), where it describes a rapid change in a system's structural properties as a control parameter varies. For instance, as shown in Fig.1(a), a system of particles transitions through phases (solid, liquid, gas) as temperature increases. A crucial step in studying these transitions is defining an *order parameter*, a key indicator of structural change. The liquid-to-gas transition can be detected by a jump in particle density, and the formation of a crystalline structure can be identified by computing the bond-orientational order parameter. Similarly, we propose that one goal of modern deep learning research is to develop evaluation metrics that detect the emergence of fundamental capabilities, which together define the model's *state* from a capability-centric perspective. Emergence in machine learning has broadly come to imply a rapid change in a system's properties (specifically, its capabilities) as some relevant axis is scaled (e.g., amount of data or compute). We emphasize that for the purview of this paper, we focus on the effect of increasing data on a model's capabilities by training models from scratch in an online learning setting; often, such analyses are called "learning curves analysis" (Viering and Loog, 2022; Blumer et al., 1989; Bousquet et al., 2021; Seung et al., 1992; Watkin et al., 1993; Amari, 1993; Haussler et al., 1994).

Similar terms as emergence, e.g., breakthroughs (Srivastava et al., 2022; Cascella et al., 2024) and sudden drops (Chen et al., 2024), also commonly find use in literature. Emergence has especially seen a recent surge in its use in the machine learning community (Wei et al., 2022; Wei, 2022), where it is broadly used to mean rapid progress in model performance as a function of some scaling parameter (e.g., compute or data). A few recent works have also made attempts at explaining the factors underlying the emergence of capabilities in neural networks via empirical and theoretical approaches. Specifically, compositionality has been implicated for having a "multiplicative" effect on a performance measure, such that a model cannot perform well on a compositional task until the individual tasks in that composition are not well learned—this leads to a sudden growth in performance when all tasks improve (Okawa et al., 2023; Arora and Goyal, 2023; Yu et al., 2023; Srivastava et al., 2022; Wei et al., 2022). Schaeffer et al. (Schaeffer et al., 2023) argue emergent capabilities are in fact a consequence of poorly defined evaluation measures and go away once partial credit is given to the model. However, a continuous parameter is insufficient to measure valid progress in some cases. For example, one can check in the context of additions of two numbers (e.g., 61 + 62) whether the resulting output (say 321) contains the ground truth's digits (123). Hence, unless a measure captures the structure of the data, outputs of all orders will be deemed equally good, e.g., 123 and 321 will be given equal partial credit, which does not make sense.

A phenomenon related to emergence is grokking (Power et al., 2022; Liu et al., 2023; Žunkovič and Ilievski, 2022; Murty et al., 2023; Barak et al., 2022; Edelman et al., 2023; Nanda et al., 2022), or delayed generalization, wherein a model's test performance on a task rapidly improves long after it has fit the train data. Since we are in a more realistic, online learning setting wherein multiple passes through the data are not allowed, we argue an emergent capability and grokking, albeit behaviorally related, are different concepts.

3. Towards a Phenomenological Understanding of Emergence

To investigate emergence, we must first establish what we mean by the term in the first place. For this purpose, we define emergence in a phenomenological manner, i.e., by assembling the specific behaviors in the current literature generally associated with the term emergence. We emphasize the definition above is merely *a* definition for emergence, and does not necessarily represent all possible perspectives; e.g., it does not sufficiently accommodate a notion of phase transitions that is often deeply intertwined with emergence in complex systems (Anderson, 1972). However, qualitatively, the definition does capture the spirit of phase transitions as well: discontinuous, rapid change in a system's properties as the relevant order parameter is changed. Our goal will be to design an experimental system where such order parameters can be precisely constructed.

Definition 3.1. (Emergence.) We say a capability is emergent upon *an increase in data* if its learning curve shows the following three characteristics:

1. nonlinear progression of generalization performance as a function of scale;
2. simultaneous emergence of multiple capabilities; and
3. the nonlinear progression is evident when the network undergoes a structural change in its computation, and the evaluation metric faithfully captures this computational structure.

3.1. Data Generating Process: Probabilistic Context-Sensitive Grammars

To design a model experimental system that can enable a rich investigation of emergence, we propose the use of probabilistic context-sensitive grammars (PCSGs), a formal model of language that is meant to capture a minimal notion of semantics, hence making it more sophisticated than context-free grammars, which can only capture syntactic properties of language (Wikipedia, 2023; Sipser, 1996; Chomsky, 1956). Specifically, the grammar (abstractly shown in Fig 2) possesses a set of terminal and non-terminal symbols that represent the syntactic rules (e.g., the rules for constructing noun phrases and verb phrases). To embed a minimal notion of semantics into this grammar, we impose "type constraints", specifically constraints that specify how probable it is that a certain subject or object can be seen in the context of a certain property. These type constraints can be formalized as a bipartite graph, as shown next.

Definition 3.2. (Type Constraints Graph.) Let a **property** p denote a discrete variable that takes values from a predefined set V_p. Given the set of all properties P, a **concept class** C is defined via the set $P_C \subset P$ that denotes which properties are valid for that class. When the properties in P_C take specific values, we get an entity e and write $e \in C$ to denote that e is a member of the class C. Properties that describe an entity are called **descriptive properties** (e.g., `age`) and ones that represent an action a subject can take or an action that can be taken upon an object are called **relative properties** (e.g., `walk`). The **type constraints graph** $\mathcal{G} := (N, N', E)$ is a bipartite graph with a set of nodes N that represent all entities from all concept classes, a set of nodes N' that represent all properties, and a set of edges E connecting nodes from $e \in N$ to $p \in N'$ if the entity e has the property p.

The type constraints graph is abstractly depicted in Fig. 2 (b). As an example, consider a concept class of `Humans`, who are connected to properties `age`, `name`, `job`, `walk` etc. This means a `human` will take on some value for each of these properties, e.g., they will have an `age`, a `name`, a `job`, and the ability to walk on objects that can be walked upon. Overall, the type constraints graph, when sampling sentences from the grammar, will be used to restrict which tokens can be seen in the same context / sentence.[1] Thus, a neural network trained via the usual next token prediction objective on this grammar will not only be expected to learn the syntactic rules of the grammar, but the broader structure of individual concept classes, i.e., their type constraints, as well. We expect the model will struggle on any downstream task grounded in this grammar until the grammar itself is learned, yielding a rapid growth in performance after. Further, multiple capabilities corresponding to each task and their interplay with the specific type constraints will have their own rich learning dynamics, showing different phases of learning depending on when the grammar is learned and when the relevant type constraints are internalized.

3.2. Data Generating Process: Learning Tasks

In this work, we consider two variants of the general pipeline above: (i) **simple grammar**, wherein relatively simple sentences of lengths 4—6 that describe an entity and its properties can be sampled, and (ii) **complex grammar**, wherein

[1] We note that this is technically equivalent to the normal form for a PCSG, wherein rules are made context-dependent by making left non-terminals have multiple symbols involved (e.g., AB $\rightarrow$ CD).

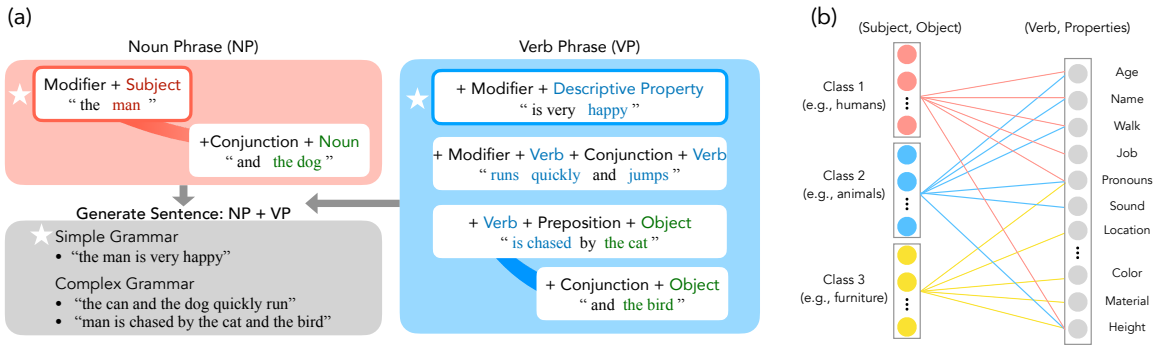

Figure 2: **Data generation process (DGP) by the probabilistic context-sensitive grammar (PCSG).** (a) Schematic of the sentence generation process using PCSG. For simple grammar, our DGP combines lists of "Modifier + Subject" and "Modifier + Descriptive Property," highlighted by the star symbol. For complex grammar, the DGP includes all possible sampling methods. (b) We apply type constraints between (Subject, Object) and (Verb, Properties), where the type restricts which sets of nodes can be connected, thereby introducing class structure into our DGP.

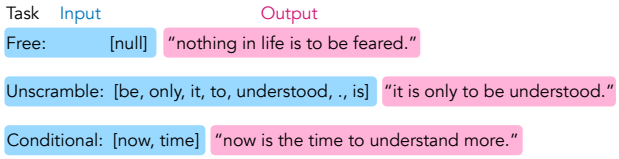

Figure 3: **Task definitions.** Our model is trained and evaluated on three types of tasks. [top] Free generation task: The model generates sentences with correct grammar. [middle] Unscrambling task: The model is provided with a set of words and must reorder them to form grammatically correct sentences. [bottom] Conditional generation task: The model is given two words and must generate grammatically correct sentences using those two words.

sentences are much more complex, range from lengths 4—75, and can have multiple phrases combined together via conjunctions and relative properties. Sentences from these grammars are restructured into a format that is fed into the model for training, as shown in Fig. 3. The reformatted representation explicitly describes a task the model is expected to perform. Specifically, we consider the following tasks with 80/10/10% splits in the training data respectively.

- **Free generation:** Produce a sentence that respects the structure of the grammar, i.e., syntactic rules and semantic type constraints.
- **Unscrambling:** A sentence is sampled from the grammar and randomly permuted. The model is expected to unscramble the sentence. Note that this is a task known to show emergent behavior in real language datasets with large-scale models (Wei et al., 2022). For complex grammar, since sentences can be fairly long, this task inherently involves a test of model's ability to length generalize—a standard compositional generalization task (Hupkes et al., 2020).
- **Conditional Generation:** A set of tokens corresponding to entities and properties are shown to the model, which is expected to generate a sentence combining these tokens in a valid manner. Note that this task is inherently compositional in nature (specifically it tests systematicity (Hupkes et al., 2020)), and can be used to probe the model's ability to respect type constraints in the grammar.

3.3. Experimental Setup

Before proceeding further, we briefly describe our experimental setup. We train a two-block Transformer based on the nanoGPT architecture (Andrej Karpathy, 2023) using the next token prediction objective with Adam optimizer, 10^{-3} learning rate, and 10^{-4} weight decay for $70K$ iterations. The sentences are sampled "online", i.e., we sample a fresh batch of data every iteration by following the rules of the grammar. Since the grammar is probabilistically defined, the odds of seeing the same sample multiple times are exceedingly low. Unless otherwise stated, we have 1000 entities and 2000 properties uniformly distributed over 10 classes in the complex grammar, and a single class in the simple grammar, with edges connecting properties sparsely and randomly distributed over valid properties for a given object (specifically, only 15% connections are made). We emphasize that our results remain consistent upon ablations of number of classes, number of properties, and the connection prior.

Metrics: For the unscrambling task, we compute two metrics. Specifically, the *per-token accuracy*, which assesses at a given token from the ground truth sentence whether

the generated sentence from the model corresponding to the scrambled version of the ground truth matches the ground truth at that token's location. We also evaluate an *exact match accuracy*, which assesses whether every token at every location is the model generation matches the ground truth sentence. We note that the per token accuracy is an approximately continuous version of the exact match accuracy, and was first introduced in the paper by Schaeffer et al. (Schaeffer et al., 2023) to claim that emergent abilities can lose their abrupt nature upon change of metric. As we show, even the per token accuracy shows precise transitions that mark specific behavioral changes in the model. Relatedly, for the conditional generation task, we use *constraint accuracy* as a metric. Specifically, herein we evaluate whether an operand the model is supposed to include in its produced sentence, i.e., the conditioning information, is present in the sentence at all—that is, at any location. We compute accuracy as sum of indicator variables that repeat this process for all tokens in the conditioning information.

4. Results

Having defined our experimental setup, we now intend to evaluate (i) whether our proposed phenomenological definition of emergence is captured by the setup and (ii) whether we can extract useful insights into the mechanisms of emergence based upon our setup.

4.1. Demonstrating Emergence and Phases of Language Acquisition in Transformers

As per our definition, emergent properties show rapid, nonlinear growth in performance, with several capabilities emerging simultaneously, especially when the capability is compositional in nature (Okawa et al., 2023; Arora and Goyal, 2023). We find our empirical setup does exhibit this phenomenology for both the simple and complex grammar settings (see Fig. 4). *More importantly, for the two tasks that we evaluate, i.e., unscrambling and conditional generation show precise and large changes in their performance at literally the same training step.* That said, we do see a specific transition / emergence point in the unscrambling task in the complex grammar wherein the model suddenly substantially improves in its exact match accuracy (see Fig. 4b). As we show next, this point corresponds to the model learning how to generalize to longer length sentences. Since sentences in the simple grammar are shorter, this transition point does not occur in the plots in Fig. 4a.

4.2. Length Generalization

To further analyze precisely what is happening at the extra transition point at ∼15K training steps in the complex grammar, we evaluate the model generations at a more microscopic level. Specifically, we decompose the inputs to the model according to their lengths and evaluate the model's performance on the unscrambling task. Interestingly, we see that for sentences the model first learns the sentences with lengths 4–6, and then suddenly at iteration 15K improves in its performance to exactly solve the task by ∼40–50%. *Evaluating the model generations by eye, we see the model improves in its ability to produce relative properties, i.e., verbs, which help connect a subject phrase with an object phrase, hence explaining the cause behind this length generalization* We furthermore find that the per-token accuracy barely, but noticeably improves. This improvement marks getting literally one more token right, which is the relative verb token. Overall, this transition explains how the model improves in its length generalization by essentially learning an integral rule of the grammar that enables connecting different noun phrases, i.e., ones with subjects to ones with objects.

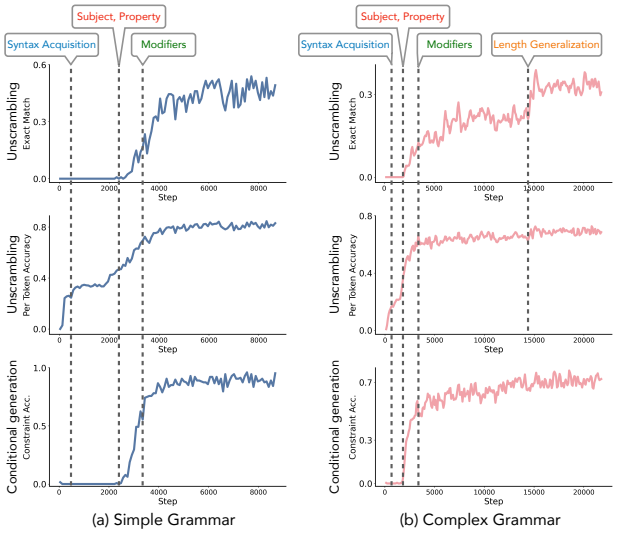

Figure 4: **Learning dynamics of evaluation metrics capture multiple interpretable phase changes.** (a) Time course of scores for the unscrambling task and the conditional generation task in the simple grammar setup. Distinct parameters indicate the onset of capability acquisition. (b) Same plot for the complex grammar setup. An additional jump, observed most clearly in the unscrambling task score for exact matches, corresponds to length generalization.

4.3. Overgeneralization results

Up to this point, our results focused on a setting wherein two classes never share a property, i.e., their entities never share a property. We now extend our results to allow classes to share properties, such that beyond learning a singular concept class, the model can see shared properties to possibly "overgeneralize" to similar classes. Specifically, we make 2 out of 10 concept classes share 10% properties now and

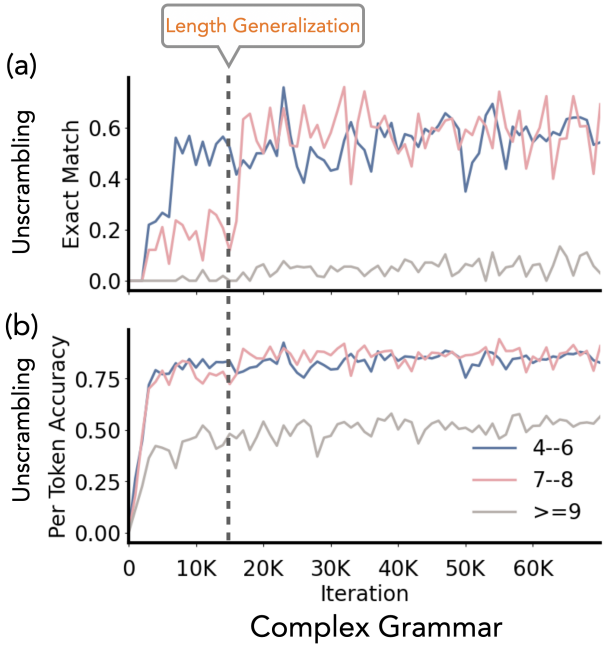

Figure 5: **Length Generalization.** (a) There exists a clear improvement in the exact match score for the unscrambling task. (b) This improvement is only barely observed in the per token accuracy since the learned property is compositional.

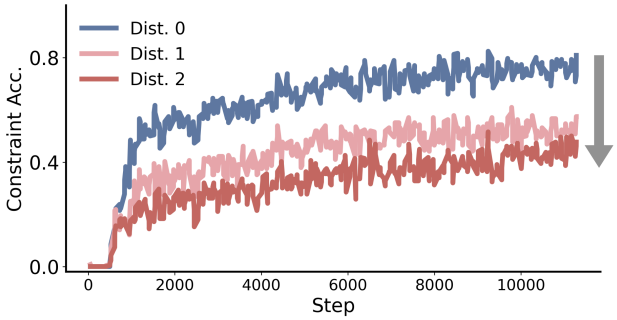

Figure 6: **Overgeneralization Dynamics Resemble an Inter-Class Percolation Process.** Accuracy of the compositional generalization task, where the model must compose verb and noun phrases from (i) the same class, (ii) two classes that are sparsely connected, and (iii) two classes with far fewer connections. The model demonstrates overgeneralization in this task, and notably, the order of generalization resembles the percolation process on a graph, where the nodes are noun phrases (NP) and verb phrases (VP), and the edges represent their composition.

perform evaluations for the model's ability to perform conditional generation for objects from a class and the properties from same/another class. Precisely, we define a notion of "distance" as follows: (i) *Distance 0*, evaluations within the same class; (ii) *Distance 1*, evaluations of objects with properties of the class that shares a property; and (iii) *Distance 2*, all remaining classes. Results are shown in Fig. 6. We find the model first learns to generalize to within the same class, then class at distance 1, and then at distance 2. For clarity purposes, the figure was pruned at 10K steps, but the trend continues to persist across training.

5. Matrix representation of data and learning compositions

In this section, we propose a framework for modeling the emergence of capabilities through learning compositions. Specifically, we consider the learning of a concept class, where each entity in the class is expected to have common properties. The entities and properties may each be single words, such as a noun and a verb, or multiple words, as in the case of objective and subjective phrases that we observed in the length generalization setup. The question is whether upon sub-sampling pairs of entities and properties from a concept class, can the model learn that, in fact, all pairs of entities and properties are valid and compose the concept class.

For instance, in the case of a concept class such as "human", the set of entities can include names of humans as well as human-associated entities such as an astronaut. The corresponding properties for the "human" concept class will be, for example, name, age, height, as well as associations with possible verbs. An astronaut, being human, is expected to have all these properties, although data specifying the properties of an astronaut will be rare or even absent in the training data.

We are interested in the case where the data, such as sentences, includes examples of these pairs of entities and properties. This could be represented by a matrix where rows and columns represent the entities and the properties, and the matrix values indicate the quantity or density of data available for each composition, such as a *noun-verb* pairing. We define this matrix as the **concept density matrix**:

Definition 5.1. (Concept Density Matrix.) Let D be an $N \times M$ matrix with real-valued entries between 0 and 1, inclusive. Each entry D_{nm} represents the density for the entity and property pair (n, m) (e.g., the amount of data that represents the specific composition), where $n \in \{1, ..., N\}$ and $m \in \{1, ..., M\}$ are the indices of the entities and properties, respectively.

For example, consider the case where there are three values of entities and properties ($N = M = 3$), with the nouns (rows) being {"Alice", "astronaut", "telephone"}, and the properties (columns) being verbs, such as {"walking","horse-riding","ringing"}. The corresponding

D will be:

$$D = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (1)$$

A common composition such as "Alice walking" will lead to a value of 1 at the intersection of "Alice" and "walking", i.e., $D_{11} = 1$. Conversely, a highly unlikely composition like "astronaut horse-riding" will be absent in the dataset, and will be represented by a zero at the respective matrix position, i.e., $D_{22} = 0$. We can also assume for example that Alice ringing or a telephone walking are rare, which yields $D_{13} = D_{31} = 0$.

We introduce the **concept propagation matrix** to model the inference of novel entity-feature combinations from the incomplete data represented in D.

Definition 5.2. Concept Propagation Matrix. An n-th order concept propagation matrix ($n \geq 0$) is defined as $T^{(n)} = (DD^T)^n D = C^n D$, where $C := DD^T$.

The concept propagation matrix can be intuitively understood using a bipartite graph, as shown in Figure 7. A bipartite graph in this case is a sub-graph of the **type constraints graph**, where one set of nodes represents entities while the other represents properties, with edges indicating the presence of entity-feature pairings in the training data. The strength of connectivity of the graph directly corresponds to the values in the concept composition propagation matrix, $T^{(n)}$.

Specifically, if two concepts are connected by a path of minimal length $2k + 1$ (i.e., the shortest path between them alternates between the two sets k times), the corresponding entry in $T^{(n)}$ becomes non-zero only for $n \geq k$. In other words, the number of propagation steps n required for the object and feature pair to be associated is determined by the minimal number of hops needed to connect the two nodes in the graph. Conversely, if two nodes belong to disconnected regions of the graph, their composition remains fundamentally unlearnable, regardless of the order of propagation, indicating that composition is not valid as a class. This is reflected by the corresponding entry in $T^{(n)}$ remaining zero for all n. For example, in the case where the concepts represented by the first and third rows belong to disconnected regions of the graph, and consequently, their composition (e.g., astronaut ringing) cannot be achieved even after an infinite number of hops between nodes.

With this first-order propagation, we now have a finite score for $T^{(1)}_{22}$, which corresponds to "astronaut horse-riding". It is clear from the form of the matrix that $T^{(n)}_{11}, T^{(n)}_{12}, T^{(n)}_{21}, T^{(n)}_{22}, T^{(n)}_{33}$ will be finite for all $n \geq 1$, whereas $T^{(n)}_{13}, T^{(n)}_{12}, T^{(n)}_{31}, T^{(n)}_{32}$ will remain zero for all n. This suggests that our model of concept propagation will never allow the composition of certain compositions of concept values such as "Alice ringing" or "telephone walking". In the bipartite graph, this amounts to having two distinct clusters that are connected within themselves but not across each other. Such a situation corresponds to the learning of a concept; Alice and an astronaut are both humans, whereas telephones are not.

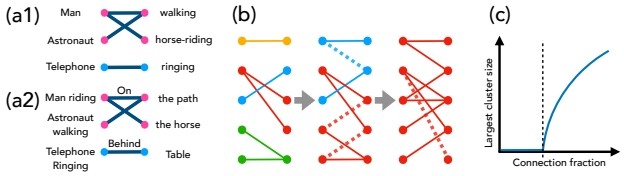

Figure 7: **Emergence of compositional abilities as a percolation phase transition in a bipartite concept graph.** (a1) The dataset is represented as a bipartite graph with subjects on the left and verbs on the right. Compositional generalization can occur for two nodes that are not directly connected (e.g., "astronaut" and "horse-riding") if they can be connected by stitching edges. (b) When only a fraction of the concept classes are included in the dataset and the edge density is low, concept variables (nodes) form many disconnected clusters, indicated by different colors (left). As more concept classes are added (dashed edges) to the bipartite graph, the small clusters begin to merge (middle). With a sufficient number of edges, a macroscopic number of nodes can be connected, forming a single cluster (right). This suggests that a model can compositionally generalize to most concept classes with enough edges and inference steps. (c) Our formalism establishes this transition as a second-order phase transition, where the size of the largest cluster increases non-linearly as the fraction of connected node pairs is scaled.

6. Percolation Transition on Concept Graphs

Using the bipartite graph framework, the generalization, or the learning of the concept class, can be defined as the situation where a large cluster of entity-feature connected pairs arises despite the sparse concept density matrix. A critical aspect to examine is the proportion of the inference matrix values where $T^{(\infty)}_{nm}$ is non-zero, out of the total possible pairs $N \times M$. This particular scenario aligns with the bond percolation problem on a bipartite graph.

In bond percolation, we investigate how the largest connected cluster's size varies with the probability p of each edge (bond) being present. In a typical setting, there exists a critical threshold value, $p = p_c$, called the percolation threshold. Below this threshold ($p < p_c$), the graph typically exhibits a disconnected phase characterized by the absence of extensively connected clusters, with most nodes

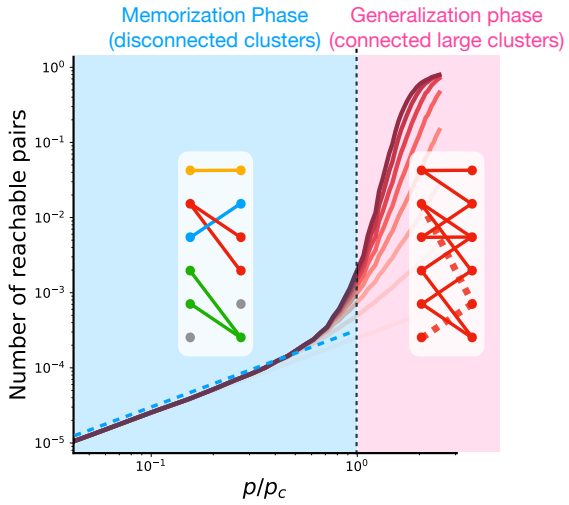

Figure 8: **Memorization and generalization in the bipartite graph percolation simulation.** The transition from the memorization phase with disconnected clusters to the generalization phase with connected large clusters appears at the percolation threshold. Distinct colors in the lines correspond to the number of steps corresponding to n in the concept propagation matrix, with the darkest line corresponding to $n \to \infty$.

either isolated or part of smaller clusters. Above this threshold ($p > p_c$), the graph transitions to a connected phase, significantly increasing the likelihood of a vast connected component spanning a large portion of the graph. This shift from a predominantly disconnected state to one with a macroscopic cluster is a defining characteristic of the percolation process, and this transition sharpens as the number of components in the system increases. See Fig. 1 for the schematic.

Consider a simple model situation with $N = 1000$ nodes as entities and $M = 1000$ nods as properties in the concept class of our interest. As we progressively increase the fraction of connecting edges in the graph, where the edges are randomly chosen, the largest cluster's relative size (compared to $N+M$) transitions from nearly zero to a significant number.

As seen in Fig. 8, the number of learned compositions scales more or less linearly as a function of the number of added edges in the regime of $p < p_c$, indicating that the model can essentially only mimic the data in the training set. In scenarios where connecting edges are selected randomly on the graph with probability p, the percolation threshold is obtained as $p_c \simeq \sqrt{1/NM}$ for large N and M (Newman et al., 2001). For $p > p_c$, the number of nodes included in the connected cluster will become macroscopic, meaning that the probability that a randomly selected pair of an object and a feature will be finite. We present in Appendix A the derivation of the percolation threshold for bipartite graphs that are uncorrelated, and how the cluster size (i.e., number of nodes in the largest connected graph) scales as $\sim (p - p_c)^\beta$ with $\beta = 1$ for usual cases.

We posit that the percolation threshold corresponds to the point at which our model generalizes from the sparse concept density matrix to a more complete representation of the concept class. When the number of edges surpasses the threshold, the model can infer novel compositions, even for entity-feature pairs that were not explicitly present in the training data. This suggests that the emergence of compositional abilities in our framework can be understood as a percolation phase transition on a bipartite graph.

7. Discussion and conclusion

In this work we proposed a data generation and learning task framework and demonstrated that many of the "emergent" properties discussed in larger model and data experiments are essentially captured. We have proposed a specific scenario regarding the mechanism of emergence, which is the phase transition behavior of percolation in the bipartite graph.

A key next step will be to quantitatively test the percolation scenario within this synthetic data approach as well as in larger model experiments. Candidates for this test include the data size dependence of the onset of emergence, which should match with the prediction of the percolation threshold. For sufficiently complicated tasks of language learning that involve syntax as well as semantics as in our proposed setup, it will be also of significant interest to seek which specific properties fit and do not fit with the emergence and phase transition scenario.

References

Shun-Ichi Amari. A universal theorem on learning curves. *Neural networks*, 6(2):161–166, 1993.

Philip W Anderson. More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047):393–396, 1972.

Andrej Karpathy. nanoGPT, 2023. https://github.com/karpathy/nanoGPT,.

Anthropic. Introducing Claude, 2023. https://www.anthropic.com/index/introducing-claude. Accessed on: June 21, 2023.

Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana,

Erik Jenner, Stephen Casper, Oliver Sourbut, et al. Foundational challenges in assuring alignment and safety of large language models. *arXiv preprint arXiv:2404.09932*, 2024.

Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*, 2023.

Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

Olivier Bousquet, Steve Hanneke, Shay Moran, Ramon Van Handel, and Amir Yehudayoff. A theory of universal learning. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 532–541, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Marco Cascella, Federico Semeraro, Jonathan Montomoli, Valentina Bellini, Ornella Piazza, and Elena Bignami. The Breakthrough of Large Language Models Release for Medical Applications: 1-Year Timeline and Perspectives. *Journal of Medical Systems*, 2024.

Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. Sudden Drops in the Loss: Syntax Acquisition, Phase Transitions, and Simplicity Bias in MLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=MO5PiKHELW.

Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3): 113–124, 1956.

Reuven Cohen, Daniel Ben-Avraham, and Shlomo Havlin. Percolation critical exponents in scale-free networks. *Physical Review E*, 66(3):036113, 2002.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in neural feature learning: Data, compute, width, and luck. *arXiv preprint arXiv:2309.03800*, 2023.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, et al. Predictability and surprise in large generative models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1747–1764, 2022.

Gemini Team. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

David Haussler, H Sebastian Seung, Michael Kearns, and Naftali Tishby. Rigorous learning curve bounds from statistical mechanics. In *Proceedings of the seventh annual conference on Computational learning theory*, pages 76–87, 1994.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Margot Kaminski. Regulating the Risks of AI. *Boston University Law Review*, 103:1347, 2023.

Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking Beyond Algorithmic Data, March 2023.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D. Manning. Grokking of Hierarchical Structure in Vanilla Transformers, May 2023.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, sep 2022. URL https://openreview.net/forum?id=9XFSbDPmdW.

M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, January 2003. ISSN 0036-1445, 1095-7200.

Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical review E*, 64(2):026118, 2001.

Maya Okawa, Ekdeep Singh Lubana, Robert P Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. *arXiv preprint arXiv:2310.09336*, 2023.

OpenAI. Gpt-4, 2023. https://openai.com/research/gpt-4. Accessed on: June 21, 2023.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of Large Language Models a mirage? *arXiv preprint arXiv:2304.15004*, 2023.

Hyunjune Sebastian Seung, Haim Sompolinsky, and Naftali Tishby. Statistical mechanics of learning from examples. *Physical review A*, 45(8):6056, 1992.

Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Jacob Steinhardt. Emergent Deception and Emergent Optimization, feb 2023. URL https://bounded-regret.ghost.io/emergent-deception-optimization/.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, ..., and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023.

Tom Viering and Marco Loog. The shape of learning curves: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Timothy LH Watkin, Albrecht Rau, and Michael Biehl. The statistical mechanics of learning a rule. *Reviews of Modern Physics*, 65(2):499, 1993.

Jason Wei. 137 emergent abilities of large language models, 2022. https://www.jasonwei.net/blog/emergence. Accessed on: October 20, 2023.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wikipedia. Context Sensitive Grammars, 2023. https://en.wikipedia.org/wiki/Context-sensitive_grammar.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the Tool Manipulation Capability of Open-source Large Language Models. *arXiv preprint arXiv:2305.16504*, 2023.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-Mix: A flexible and expandable family of evaluations for AI models. *arXiv preprint arXiv:2310.17567*, 2023.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

Bojan Žunkovič and Enej Ilievski. Grokking phase transitions in learning local rules with gradient descent, October 2022.

A. Percolation threshold in the bipartite graph setup

For general bipartite graphs that are uncorrelated, meaning that they are completely described by the degree distributions $P_1(k)$ and $P_2(k)$ for the objects and features, respectively, the percolation threshold is

$$p_c = \sqrt{\frac{\langle k \rangle_1 \langle k \rangle_2}{\langle k(k-1) \rangle_1 \langle k(k-1) \rangle_2}}. \tag{2}$$

Here, $\langle \cdot \rangle_i$ denotes the expected value with respect to $P_i(k)$, and we require $N\langle k \rangle_1 = M\langle k \rangle_2$ for consistency. The case of randomly selecting connecting edges as demonstrated in the main text will correspond to starting from a complete bipartite graph, in which case $P_1(k) = \delta_{k,M}$ and $P_2(k) = \delta_{k,N}$, leading to $p_c = \sqrt{1/(N-1)(M-1)} \simeq \sqrt{1/NM}$.

To derive Eq. (2) we use the generating function as explained in (Newman et al., 2001). Firstly, we introduce the generating function for the degree distribution of two concepts, $i = 1, 2$:

$$G_i^0(x) \quad := \quad \sum_{k=0}^{\infty} P_i(k) x^k \tag{3}$$

The generating function can be used to calculate moments of the probability distribution, such as the mean and variance, by taking derivatives:

$$\left.\frac{dG_i^0(x)}{dx}\right|_{x=1} \quad = \quad \left.\sum_{k=0}^{\infty} k P_i(k) x^{k-1}\right|_{x=1} = \sum_{k=0}^{\infty} k P_i(k) = \langle k \rangle_i \tag{4}$$

$$\left.\frac{d^2 G_i^0(x)}{dx^2}\right|_{x=1} \quad = \quad \left.\sum_{k=0}^{\infty} k(k-1) P_i(k) x^{k-2}\right|_{x=1} = \sum_{k=0}^{\infty} k(k-1) P_i(k) = \langle k(k-1) \rangle_i. \tag{5}$$

Here we denoted the average over the degree distribution of concept i as $\langle \cdot \rangle_i$.

Another useful property of generating functions is that the generating function of the sum of the degrees can be described by the power of generating functions. For example, the distribution of the sum of degrees from two randomly selected nodes from sets i and j, denoted by $\tilde{P}_{ij}(k)$, will satisfy

$$\sum_{k=0}^{\infty} \tilde{P}_{ij}(k) x^k = G_i^0(x) G_j^0(x). \tag{6}$$

With these properties in mind, we further introduce the generating function for the distribution of outgoing edges from a node that we arrive at by following a randomly chosen edge:

$$G_i^1(x) \quad := \quad \frac{\sum_{k=0}^{\infty} k P_i(k) x^{k-1}}{\sum_{k=0}^{\infty} k P_i(k)} = \frac{G_i^{0\prime}(x)}{\langle k \rangle_i}. \tag{7}$$

which can be obtained by noticing that the probability of the degree of a node arrived at from a randomly chosen edge is proportional to $kP_i(k)$. The decreased power of x by one in the numerator is to exclude the originally chosen edge.

We further introduce the generating function for the distribution of the number of concepts in i that can be reached from a node in the concept $j(\neq i)$ that is connected to a randomly chosen edge as $\tilde{G}_i^1(x)$, and the same when randomly choosing a node in concept j, $\tilde{G}_i^0(x)$. These functions satisfy

$$\tilde{G}_i^0(x) \quad = \quad G_i^0(G_j^1(x)) \tag{8}$$

$$\tilde{G}_i^1(x) \quad = \quad G_i^1(G_j^1(x)). \tag{9}$$

We also introduce the generating function for the distribution of the sizes of components in concept i that are reached by choosing an edge, $H_i^1(x)$, and the same when choosing a node in concept i, $H_i^0(x)$. These satisfy

$$H_i^0(x) \quad = \quad x\tilde{G}_i^0(H_j^1(x)) \tag{10}$$

$$H_i^1(x) \quad = \quad x\tilde{G}_i^1(H_j^1(x)). \tag{11}$$

Here, the key assumption is that there is no closed loop of edges in the network, which holds if the fraction of connection is low and there is no cluster (i.e., sub-critical regime).

The average cluster size of concept i, i.e., the number of nodes in i that are connected with each other, is then $\langle S_i \rangle = H_i^{0'}(1)$, which is

$$\langle S_i \rangle = 1 + \frac{\tilde{G}_i^{0'}(1)}{1 - \tilde{G}_i^{1'}(1)}, \tag{12}$$

using the derivatives of Eqs. (10),11). The percolation threshold is when the denominator in the second term of Eq. (12) becomes zero, so

$$\tilde{G}_i^{1'}(1) = G_i^{1'}(1) G_j^{1'}(1) = \frac{G_1^{0''}(1) G_2^{0''}(1)}{\langle k \rangle_1 \langle k \rangle_2} = \frac{\langle k(k-1) \rangle_1 \langle k(k-1) \rangle_2}{\langle k \rangle_1 \langle k \rangle_2} = 1 \tag{13}$$

Now, when the connection of each a probability of connection p associated with each bond on top of the original graph, the generating function of the degrees will become

$$\begin{aligned} G_i^0(x;p) &= \sum_{k=0}^{\infty} \sum_{n=k}^{\infty} P_i(n) \binom{n}{k} p^k (1-p)^{n-k} x^k & (14) \\ &= \sum_{n=0}^{\infty} P_i(n)(px + 1 - p)^n = G_i^0(1 + (x-1)p). & (15) \end{aligned}$$

From the first line to the second line, we used $\sum_{k=0}^{\infty} \sum_{n=k}^{\infty} = \sum_{n=0}^{\infty} \sum_{k=0}^{n}$. We can then rewrite Eq. (13) as

$$\frac{\langle k(k-1) \rangle_1^p \langle k(k-1) \rangle_2^p}{\langle k \rangle_1^p \langle k \rangle_2^p} = p^2 \frac{\langle k(k-1) \rangle_1 \langle k(k-1) \rangle_2}{\langle k \rangle_1 \langle k \rangle_2} = 1, \tag{16}$$

from which we obtain Eq. (2). Here we used $\langle k(k-1) \rangle_i^p = G_i^{0''}(1;p) = p^2 G_i^{0''}(1) = p^2 \langle k(k-1) \rangle_i$ and $\langle k \rangle_i^p = G_i^{0'}(1;p) = p G_i^{0'}(1) = p \langle k \rangle_i$.

A.1. Exponent in the cluster size

The critical exponent associated with the number of nodes in the cluster for $p > p_c$, $S \sim (p - p_c)^{\beta}$, is determined to be $\beta = 1$ when there is no specific structure in the graph. To see this, let us consider that $u = H_i^0(1)$ is the probability that a node in i is included in a finite size cluster (i.e., not the large connected cluster). Recall that $H_i^0(1)$ was the generating function of the number of nodes in concept i included in the cluster in the subcritical regime ($p < p_c$); we are here assuming that the statistics will not change even in the supercritical regime ($p > p_c$) when neglecting the large cluster. Then, from Eqs. (10,11), we have

$$\begin{aligned} H_i^0(1) = u &= \tilde{G}_i^0(H_j^1(1)) = \tilde{G}_i^0(\tilde{G}_j^1(u)) & (17) \\ &= G_i^0(G_j^1(G_j^0(G_i^1(u)))) =: f(u) & (18) \end{aligned}$$

which is a self-consistent equation.

By writing $u = 1 - \epsilon$, we have $f(1 - \epsilon, p) = 1 - \epsilon f'(1,p) + \epsilon^2 f''(1,p)/2...$, where the derivative is taken for u. Noticing that $f'(1, p_c) = 1$, we obtain the relation

$$\begin{aligned} \epsilon &= (p - p_c) \frac{\partial^2}{\partial u \partial p} f(u,p) \Big|_{u=1, p=p_c} \left[\frac{1}{2} \frac{\partial^3}{\partial u^2 \partial p} f(u,p) \Big|_{u=1, p=p_c} \right] + o(p - p_c) + o(\epsilon) & (19) \\ &\sim (p - p_c), & (20) \end{aligned}$$

indicating $\beta = 1$

As an interesting generalization, a classic result (Cohen et al., 2002) shows that even for the situation where $p_c > 0$, the power β can deviate from one. This corresponds to when the differential coefficients in Eq. (19) diverge, corresponding to cases where the second or third moment being ill-defined. For the case of $P_i(k) \sim k^{-\gamma}$ with $3 < \gamma < 4$, we can show that $\beta = 1/(\gamma - 3)$.

A.2. Transition Behavior for Finite Inference Steps

The mapping of the inference scheme to the percolation problem becomes precise only in the context of infinite inference steps. For a finite number of steps, denoted as n, the pertinent question is the number of node pairs across the sets connected within $2n + 1$ edges. Using the average degrees $\langle k \rangle_1$ and $\langle k \rangle_2$ respectively, a node in the first set can reach approximately $\langle k \rangle_1^{n+1} \langle k \rangle_2^n$ nodes after $2n + 1$ steps. Hence, the approximate fraction of connected edges within $2n + 1$ steps is $N \langle k \rangle_1^{n+1} \langle k \rangle_2^n$.