SYMSEQBENCH: a unified framework for the generation and analysis of rule-based symbolic sequences and datasets

Zajzon, Barna[*1], **Bouhadjar, Younes**[2], **Fabre, Maxime**[2, 3], **Schmidt, Felix**[2, 4],
Ostendorf, Noah[1, 4], **Neftci, Emre**[2, 6], **Morrison, Abigail**[1, 5], and **Duarte, Renato**[7, 8]

[1]Institute for Advanced Simulation (IAS-6), Jülich Research Centre, Germany
[2]Peter-Grunberg Institute, Neuromorphic Software Ecosystems (PGI-15), Jülich Research Centre, Germany
[3]Groningen Cognitive Systems and Materials Center (CogniGron), University of Groningen
[4]RWTH Aachen University, Aachen, Germany
[5]Department of Computer Science 3 - Software Engineering, RWTH Aachen University, Aachen, Germany
[6]Department of Electrical Engineering and Information Technology, RWTH Aachen University, Aachen, Germany
[7]Center for Neuroscience and Cell Biology (CNC-UC), University of Coimbra, Portugal
[8]Center for Innovative Biomedicine and Biotechnology (CIBB-UC), University of Coimbra, Portugal

January 1, 2026

Abstract

Sequential structure is a key feature of multiple domains of natural cognition and behavior, such as language, movement and decision-making. Likewise, it is also a central property of tasks to which we would like to apply artificial intelligence. It is therefore of great importance to develop frameworks that allow us to evaluate sequence learning and processing in a domain agnostic fashion, whilst simultaneously providing a link to formal theories of computation and computability. To address this need, we introduce two complementary software tools: `SymSeq`, designed to rigorously generate and analyze structured symbolic sequences, and `SeqBench`, a comprehensive benchmark suite of rule-based sequence processing tasks to evaluate the performance of artificial learning systems in cognitively relevant domains. In combination, `SymSeqBench` offers versatility in investigating sequential structure across diverse knowledge domains, including experimental psycholinguistics, cognitive psychology, behavioral analysis, neuromorphic computing and artificial intelligence. Due to its basis in Formal Language Theory (FLT), `SymSeqBench` provides researchers in multiple domains with a convenient and practical way to apply the concepts of FLT to conceptualize and standardize their experiments, thus advancing our understanding of cognition and behavior through shared computational frameworks and formalisms. The tool is modular, openly available and accessible to the research community.

1 Introduction

Symbols and symbol systems are central constructs in both cognitive sciences and theoretical computer science, providing a unique formal link between cognition and computation [64]. This relation, manifested as a branch of the mathematical theory of computation known as Formal Language Theory (FLT), grounds the investigation of complex cognitive, psychological and behavioral processes onto a systematic terminology and set of conventions for describing generative rule systems and the structures they generate [74, 50]. From simple pattern perception to language and sequential decision-making, such formal systems allow the algorithmic specification of complex cognition and can be used to understand, model and analyze cognition and behavior [114, 119, 120].

Evaluating sequence learning and processing has long been central to advancing our understanding of both artificial and natural intelligence, from Lashley's seminal work on serial order and the syntax of action [85] to contemporary large-scale neural sequence models. Accounting for the full scope of sequence processing is nevertheless challenging: system-specific constraints and the many dimensions along which task characteristics and model demands vary mean that most tasks probe only a limited portion of the problem space. Canonical experimental paradigms and principled benchmarks make it possible to address this challenge by providing shared tasks and evaluation protocols that allow systematic comparison and reveal key limitations in learning and generalization [70, 134, 55, 124]. The rise of linguistically-competent and computationally proficient artificial intelligence [144] has led to renewed interest in tasks and metrics that can evaluate systems' capacity to learn and generalize complex structured sequences [146, 52, 138]. Ideally, and given that human intelligence

---

[*]b.zajzon@fz-juelich.de

remains the ultimate reference point, such tasks should be grounded in cognitive theory and capture relevant aspects of human reasoning, behavior and cognition [84, 66, 83, 113]. We argue that moving beyond monolithic approaches or discrete language-theoretic classes toward a more fluid understanding of temporal processing requires tools that capture its multifaceted nature and complexity at multiple scales, enabling the principled design of tasks with controllable sequence complexity. These allow researchers to scrutinize the properties of sequential structure, generative capacity and the requirements it imposes on the computational machinery.

Strongly inspired by decades of human psycholinguistic research, the tools we present here address these needs by providing a comprehensive framework for the specification, generation, manipulation, and analysis of symbolic sequences and corresponding embeddings, as well as a benchmark suite of datasets and tasks that can be used to evaluate the performance of artificial cognitive systems. With applications spanning experimental psychology, behavioral analysis, neuromorphic computing, and artificial intelligence, among others, our framework aims to facilitate interdisciplinary research and the adoption of formal constructs to multiple scientific domains that investigate structured sequential data [140, 112]. We illustrate the scope and applicability with concrete use-cases in areas such as dataset generation for artificial grammar learning experiments, the analysis and comparison of behavioral ethograms in different experimental conditions, the evaluation of neuromorphic, artificial neural network architectures, and the mechanistic dissection of neural sequence models.

While other tools and approaches have been proposed over the years that address specific aspects of symbolic sequence processing, cognitive modeling and computational benchmarking, they largely do so with a narrow focus. For example, specialized software exists to automatically generate and select string sets for artificial grammar learning (AGL) experiments [5], to evaluate large language models' meta-learning and generalization from minimal exposure [1], to systematically deploy standardized neuroscience experiments and tasks [105] or to investigate neuromorphic embodied agents' performance in standardized environments and tasks [165, 76], among others. An integrated, broadly applicable tool requires a higher level of abstraction and generalization. The adoption of FLT concepts, as we propose here, has proven insightful for predicting the generalization limits of artificial neural networks [40], proposing architectural augmentations that could increase their expressiveness [40, 176] or to expose computational similarities and differences between biological and artificial intelligence [49].

These studies provide powerful tools and valuable theoretical insights into the algorithmic and computational properties and limitations of neural architectures designed to cope with sequential structural regularities. They demonstrate the importance of appropriate computational formalisms and benchmark tasks, but are not easily generalizable across research domains. The ubiquity of temporal structure in animal cognition and behavior, as well as the power of adopting formal, systematic constraints and notations across scientific domains, urges a more substantive integration and the development of tools that can homogenize experimental paradigms and metrics, providing a shared conceptual framework to interpret and analyze different aspects of biological and artificial intelligence.

Unlike existing specialized software, `SymSeqBench` offers a comprehensive, multi-disciplinary approach to analyze human, animal, and artificial intelligence, from designing psycholinguistic experiments and evaluating neuromorphic architectures to dissecting the syntactic structure of behavioral and biological sequences. Below, we explain the theoretical bases and formalisms linking the approaches proposed throughout the paper, we proceed to explain how the tool is organized and implemented, followed by a set of concrete use-cases demonstrating its scope and applicability across a wide range of problem domains.

1.1 Theoretical Foundations

We consider sequences as elements of (potentially infinite) formal languages, defined over a finite alphabet and generated by a formal grammar. A formal **language** $\mathcal{L}$ comprises the set of all **words** or **strings** $\mathcal{S}$ that can be formed by following a generative **grammar** $\mathcal{G}$, i.e. a set of production rules. Each string S_i comprises a finite sequence of **symbols** σ_i, drawn from a finite **alphabet** $\mathcal{A}$. The generative grammar $\mathcal{G}$ defines the **syntax** of the language, specifying how symbols can be combined to form valid strings and string-sets, and can be described using various formalisms, such as regular expressions, context-free grammars, or more expressive representations like context-sensitive or unrestricted grammars.

The notion of syntax is the defining anchor, as it captures, quantifies and formalizes all aspects pertaining to temporal structure: the systematic patterns governing how elements are ordered and combined over time. By formalizing syntax through generative grammars, we provide a principled framework for specifying, manipulating, and analyzing the temporal dependencies that characterize complex sequences.

Due to the deep relationship between formal languages and abstract automata [25, 24], a grammar can be represented by a corresponding automaton $\mathcal{A}$, such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{A})$, where $\mathcal{L}$ denotes the language generated or recognized. For every grammar, there exists a corresponding automaton that recognizes or generates the same language. As such, to simplify the terminology, we often describe generative grammars using the tuple notation of their corresponding automata. We will primarily be dealing with regular grammars, which can be represented as $\mathcal{G} = <\mathcal{Q}, \mathcal{A}, \mathcal{T}, q_0, \mathcal{F}>$, where $\mathcal{Q}$ is a finite set of states, $\mathcal{A}$ is the alphabet, $\mathcal{T}$ is the state

transition table, q_0 and $\mathcal{F} \subseteq \mathcal{Q}$ are the subsets of start and terminal states, respectively. Additionally, for convenience and without loss of generality, we typically consider grammatical structures as directed graphs, where nodes represent states, edges represent transitions between states, and edge labels represent symbols from the alphabet. We also use this notation interchangeably with a Markov chain formulation where symbols label states rather than transitions; the correspondence between these two equivalent representations is detailed in Section 2.1.1.

The universality of symbolic sequences and the underlying theoretical notions extends across diverse scientific domains, reflecting the fundamental role of structured temporal patterns in both natural and engineered systems. In molecular biology, stochastic context-free grammars serve as standard tools for RNA secondary structure prediction and analysis [174]. In neuroscience, sequential patterns of neural activity or behavioral states can be characterized using formal language frameworks [50, 22, 17]. In cognitive psychology, hierarchical phrase structure grammars capture the compositional nature of language and action planning [99, 26, 73]. This domain-general applicability stems from the abstract nature of formal grammars: by imposing no constraints on the semantic interpretation of symbols, which can represent any abstract state or event, the framework applies to any sequentially structured phenomenon.

We can thus define different levels of generation and analysis where we can either quantify or manipulate the qualities and characteristics of syntactic structure: (i) individual symbols or tokens $\sigma_i \in \mathcal{A}$; (ii) individual sub-sequences or strings $S_i \in \mathcal{S}$; (iii) collections of valid strings or string-sets $[S_i, ..., S_T] \in \mathcal{S}$; (iv) generative grammars $\mathcal{G}_x$; and complete (potentially infinite) languages $\mathcal{L}$. The implementations we propose provide the ability to manipulate and/or analyse the properties at each of these levels independently and, by imposing no constraints or specific meanings to the nature of the symbols (which can represent any abstract state), are applicable to a very broad range of sequentially structured data.

It is important to distinguish our use of certain formal language theoretic concepts from their traditional interpretations, particularly the notion of ambiguity. In formal language theory, ambiguity typically refers to the existence of multiple parse trees for a single string under a given context-free grammar and thus refer to a structural property of the grammar itself. In our framework, we use the term more loosely to capture uncertainty or variability at multiple levels: symbol-level ambiguity (e.g., noisy or partially observable tokens), string-level ambiguity (e.g., multiple valid interpretations of the same surface form), and grammar-level ambiguity (e.g., underspecification in the generative rules). This generalized notion of ambiguity is essential for modeling realistic sequence processing scenarios where uncertainty pervades observation, representation, and inference.

In the following sections, we describe the implementation of this multi-level framework and demonstrate how it enables systematic exploration of sequence complexity across scales, providing both theoretical insights and practical tools for designing cognitively grounded experimental paradigms.

2 Architecture, design and implementation

`SymSeqBench` is an open-source, modular Python framework for symbolic sequence processing, integrating grammar-based sequence generation with synthetic and real-world datasets, along with a wide range of structural and statistical analysis within a unified system. The framework divides responsibilities across two specialized components: `SymSeq`[1] for defining, generating and analyzing symbolic sequences and tasks, encapsulated into a standardized interface, and `SeqBench`[2] for transforming and embedding (grounding) symbolic representations and assigning a functional meaning to symbols. This layered organization supports both synthetic and user-provided data, enabling analyses that range from abstract structural properties to distributed vector representations compatible with machine learning and biological models of computation. The advantage of this separation is that researchers interested primarily in symbolic generation and analysis can use the lightweight `SymSeq` component without the additional overhead and dependencies of `SeqBench`. At the same time, the flexible and user-friendly architecture emphasizes reproducibility and customization, while optimizations and a backend-agnostic design ensure scalability across different experimental and computational settings.

2.1 `SymSeq`: Symbolic sequence generation and analysis

`SymSeq` provides the foundation for defining, generating, and analyzing symbolic sequences, structuring them into relevant computational tasks and specifying all the relevant mappings. It contains core data structures for grammar and sequence generation, a diverse library of artificial language generators, interfaces for parsing user-supplied data, and a comprehensive suite of analysis metrics spanning multiple structural and linguistic scales. In addition, `SymSeq` formalizes task targets in terms of language recognition and language transduction, thereby supporting both *abstract rule generalization* and *structure-sensitive transformations*. Together, these
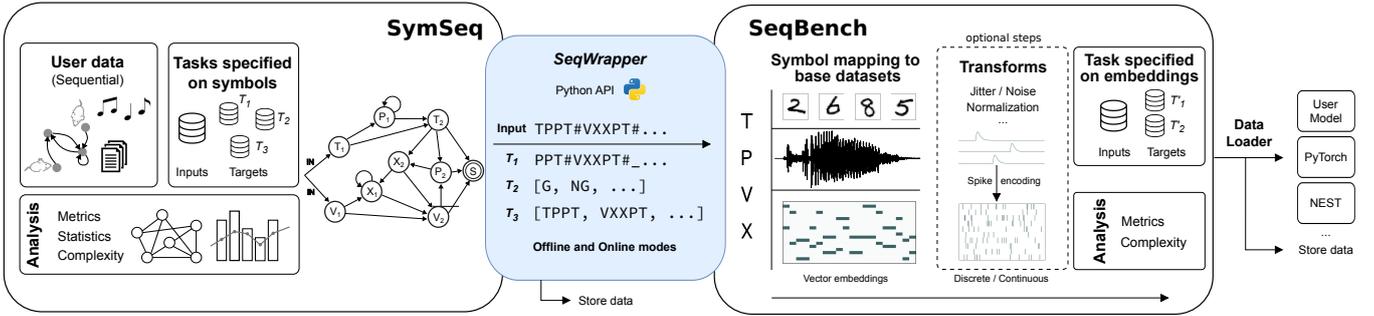
---

[1] https://github.com/symseqbench/symseq
[2] https://github.com/symseqbench/seqbench

Figure 1: Conceptual overview of `SymSeqBench`, the individual components and their interactions. `SymSeq` focuses on handling symbolic sequences, specifying processing tasks and enabling the structural analysis of both synthetic and user-provided sequences. SeqWrapper conveniently bundles `SymSeq` outputs — such as input/output sequences and generator objects — and represents the primary interface for downstream processing. From such wrapper objects, `SeqBench` provides functionality for grounding symbolic representations, assigning each symbol to a corresponding discrete- or continuous-time distributed representation, along with additional tasks and analysis that are based on embedded representations. By combining a tool-agnostic design with multiple backends, `SeqBench` enables fast and efficient integration with simulation frameworks such as PyTorch and NEST.

elements establish a flexible and extensible framework for modeling and evaluating symbolic sequence processing across domains.

Leveraging a modular design, `SymSeq` allows users to use individual components and functions in isolation or instantiate complete input/output datasets from configuration files, thereby significantly reducing time-to-data in most practical scenarios. These include small, carefully hand-crafted datasets commonly used in AGL experiments (Section 3.1), as well as large-scale datasets for neuromorphic and AI applications (Section 3.2), which can be pre-generated efficiently through parallelized routines or prepared for online generation at a later stage. In both cases, the data is exposed through a simple wrapper interface that enables flexible (custom, user-defined) downstream processing and seamless integration with `SeqBench`.
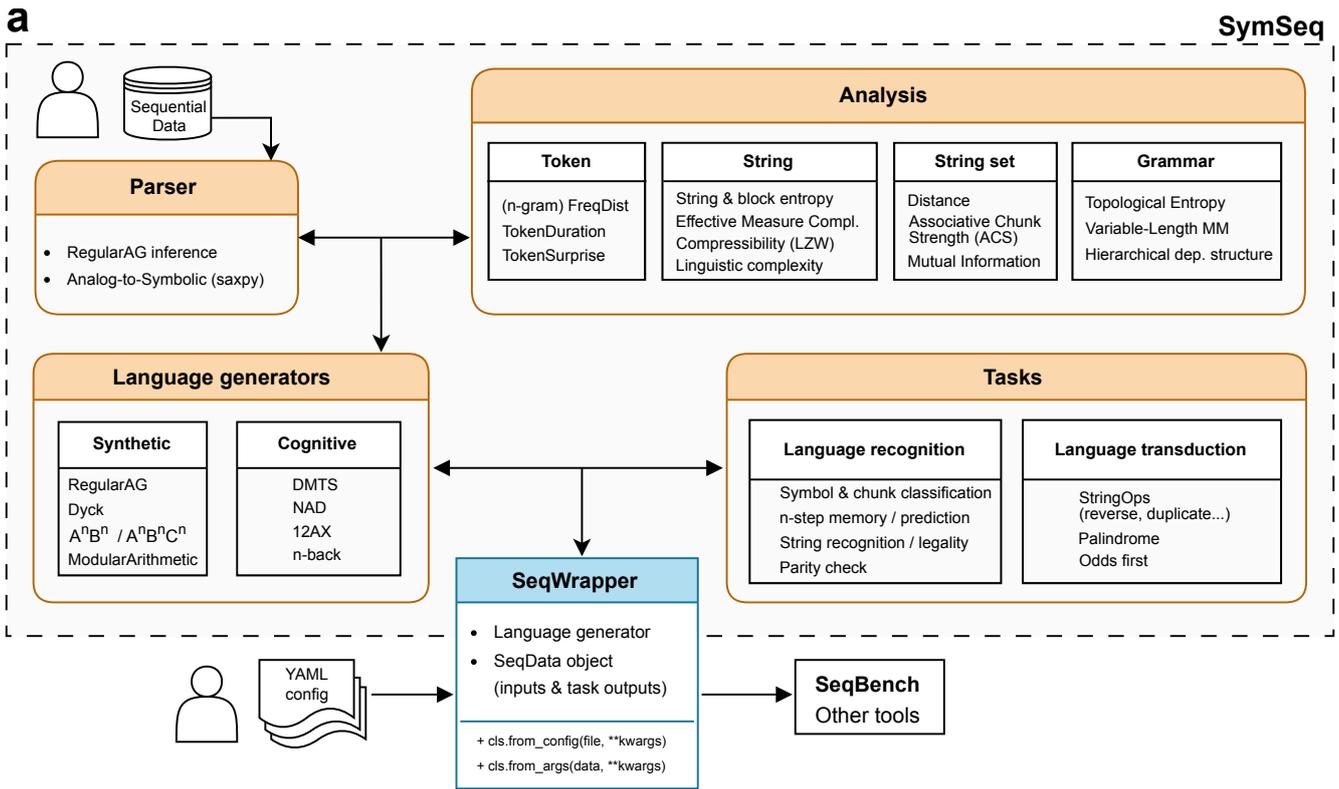
Although certain features of `SymSeq` are shared with other libraries and codebases accompanying related publications, these alternatives are typically not actively maintained, depend on proprietary software, or are too narrowly tailored to specific contexts to be broadly deployed. For instance, the Matlab library AGL StimSelect [5] for the generation of controlled AGL experimental datasets is not actively developed while the more recent AGSuite [32] is entirely browser-based, limited to small datasets and does not include grammar-generation functionality. On the other hand, works on formal languages in artificial neural networks usually provide custom implementation of their tasks based on the study's specific requirements [40] and are not easily generalizable beyond their intended application domain.

While tools like NeuroGym [105] provide flexible and extensible reinforcement learning environments for cognitive neuroscience with predefined decision-making tasks, they lack the flexible symbolic framework and controllable complexity generation that `SymSeq` offers for sequence processing. Similarly, experimental software such as PsychoPy [117] and PsyToolkit [149, 150] excel at stimulus presentation and behavioral data collection in human experiments but do not provide grammar-based task generation, sequence complexity control, or the multi-scale analysis metrics central to `SymSeq`'s design. To the best of our knowledge there is no related software that matches the scope that abstract symbolic representations and FLT endow or the wide range of analysis tools included in `SymSeq`. In the following, we describe the main components of the tool.
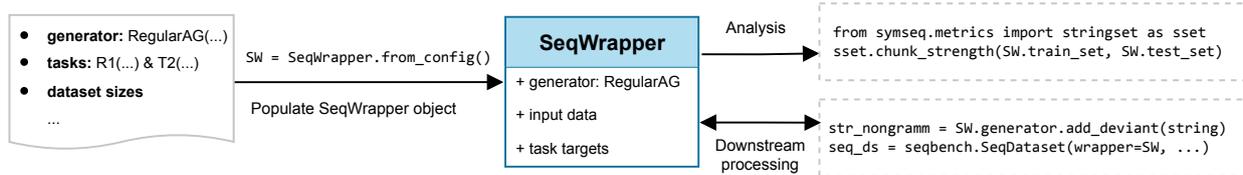
2.1.1 Main components

Parser Reads user-provided sequential data either in a symbolic or continuous time-series form. Symbolic data can be directly analyzed or used to infer regular grammars and instantiate a corresponding language generator object (see also Section 3.3). Continuous data is first converted to symbolic representation using the saxpy library [135], which is based on the SAX algorithm [93] to discretize the signal and transform it to a sequence of strings based on a predefined alphabet. The resulting data can be further processed similarly to any internal symbolic sequence. After inferring the underlying structure and mapping it to a language generator (e.g., regular grammar), newly generated strings can be converted back to analog signals using `SeqBench`. Thus, `SymSeqBench` provides a full `continuous → symbolic → grammar → synthetic continuous` pipeline for generating synthetic time series with prescribed and controllable complexity. Note that grammar induction at this stage is limited to regular, finite-state structures (see Section 4 for a more extended explanation).

Language generators This central component generates complete sets of structured symbolic sequences in the form of artificial languages (see Section 1.1), which constitute the inputs to various tasks. Supported languages can be grouped as *synthetic*, which includes random or established artificial (regular) grammars, Dyck
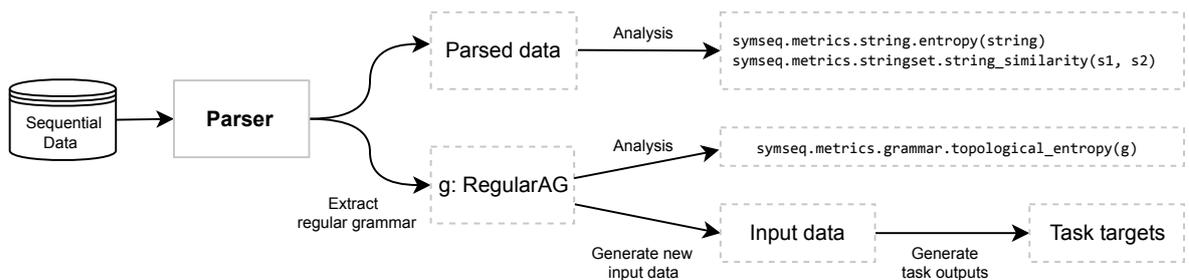
Figure 2: Schematic overview of `SymSeq` and typical use-case workflows. a) The library consists of four conceptual components that interact through standard Python data structures representing symbolic sequences: The *Parser* accepts sequential data from the user, either as continuous time-series from which symbolic sequences are built, or as discrete symbolic sequences (string-sets). From first-order transition probabilities, the parser extracts a regular (finite-state) grammar and, resorting to the large battery of *analysis* metrics, provides extensive and detailed information about the complexity of the sequences and the underlying generative grammar. *Language generators* comprise various classes for creating task-specific sequences, grouped into synthetic and cognitive/behavioral categories. Both the generated strings and their underlying structure (e.g., transition table) can be directly used by functions in *Analysis* which provide complexity metrics at multiple scales. The *Tasks* module is a collection of functions to create target labels for the provided input sequences, organized into recognition and transduction categories; Finally, *SeqWrapper* serves as the unified input-output interface of `SymSeq`, coordinating flow logic and instantiating a complete and ready-to-use experimental setup and dataset. **b)** SeqWrapper objects can be instantiated from YAML configuration files, allowing subsequent analysis or manipulation of datasets, as well as downstream processing using `SeqBench`. **c)** User-provided sequential data in symbolic form can be analyzed directly or used to infer a regular grammar and create input/output datasets from the corresponding language generator.

5

languages or modular arithmetic, to *cognitive*, which includes delayed match-to-sample, conditional one-back 12AX or various non-adjacent dependency (NAD) tasks.

The tool allows for flexible customization of the specific details and parameters of many of these languages. For example, languages with NADs may contain simple long-distance (e.g., in counting tasks), cross-serial or center embedded dependencies, with many variations regarding the nature and position of the filler/distractor elements (see Section 3.1). In addition, many language generators include functionality for producing sequences with diverse forms of rule violations (deviants).

Regular grammars, a central focus of `SymSeq`, are internally represented using the indexed, Markov-style notation [164] with symbols in the states and probabilities on the transitions. The index, which is a subscript added to the state label but is not part of the output symbol (e.g., $A_2 \rightarrow A$), is a method to unambiguously distinguish between states with the same symbol occurring in different contexts. This representation also enables tapping into a wide array of Markov analysis tools for studying artificial grammars. Grammar objects can be instantiated in four different ways: inferred from user-provided sequences, loaded from a set of preset grammars, manually specified using a parameter dictionary, and randomly generated based on certain constraints. This latter method is a novel feature of `SymSeq`, enabling users to generate constrained regular grammars with a prescribed level of complexity (via the procedure detailed in the next section), which makes it suitable to systematically evaluate the computational constraints imposed by sequences of varying complexity on biological and artificial models (Section 3.2).

Analysis `SymSeq` includes a comprehensive suite of tools for analyzing sequential data at multiple scales of structural organization. Rather than categorizing metrics by methodological approach (distance-based, information-theoretic, statistical), we organize them hierarchically according to the level of linguistic structure they target – from individual symbols to complete generative grammars. This taxonomy reflects the nested organization inherent to sequential structure and enables systematic investigation of complexity at each level independently. We briefly outline the four hierarchical levels below, with formal definitions, references and computational details provided in Section 5.2, practical applications illustrated across Section 3, and an integrative discussion in Section 4.

Token-level metrics focus on individual symbols and their local context, capturing statistical properties such as frequency distributions, n-gram patterns, and temporal persistence. These measures characterize the basic building blocks of sequences but remain agnostic to global structure.

String-level metrics evaluate the internal organization, information content, and compressibility of individual sequences. This includes entropy-based measures of unpredictability (Shannon entropy, permutation entropy), algorithmic complexity metrics reflecting compressibility (Lempel-Ziv, effective measure complexity), and linguistic diversity measures. Most operate on isolated strings without requiring corpus-level statistics.

String-set level metrics characterize relationships and variability across collections of sequences, enabling corpus-level and comparative analyses. Examples include pairwise distance measures (edit distance, normalized compression distance), information-theoretic quantities (mutual information, string-set entropy), and psycholinguistically motivated metrics such as associative chunk strength that assess sequential predictability based on training exposure.

Grammar-level metrics overcome the limitations of finite string samples by characterizing the underlying generative mechanisms themselves. These measures capture the computational capacity and structural principles of the grammar, including topological entropy (quantifying exponential growth in generative capacity), Markov order estimation, hierarchical dependency structure, and production rule complexity. By operating at the grammar level, these metrics provide language-theoretic characterization independent of sampling biases.

This hierarchical organization enables both fine-grained analysis at specific structural scales and systematic comparison across grammars and datasets. The above selection represents a foundational subset drawn from cognitive science, information theory, and formal language theory, designed for extensibility. At the implementation level, analysis functions are modular and can be used independently of the broader toolkit, accepting standard Python data structures (nested lists, NumPy arrays) as input. Where multiple definitions exist in the literature (e.g., variants of associative chunk strength in Knowlton and Squire [82] and Bailey and Pothos [5]), `SymSeq` includes all relevant versions with precise documentation of sources and differences.

Task targets `SymSeq` distinguishes between two primary types of symbolic sequence processing tasks: *language recognition* and *language transduction*.

Language recognition involves determining whether a given input string conforms to a specific grammar. This is commonly framed as a classification problem – deciding whether a sequence like "abba" belongs to a language defined, for example, by palindromic or nesting rules. However, such tasks can be problematic in practical machine learning contexts because generating negative examples (i.e., strings not in the language) in a principled and exhaustive way is often intractable. As an alternative, recognition is frequently approximated via proxy tasks like next-symbol prediction (e.g., predicting the next character in a well-formed nested sequence like

"a(b(c))") which are easier to implement but introduce issues when sequence length distributions are unknown or variable. Implemented tasks include n-step memorization and prediction at the level of single symbols or longer chunks, along with grammar-specific string legality tests.

Language transduction, on the other hand, shifts focus from classification to functional mapping: given an input sequence from a language $\mathcal{L}_{\text{inp}}$, produce a corresponding output from a possibly different language $\mathcal{L}_{\text{out}}$ according to a well-defined transformation. Examples in `SymSeq` include copying a string in reverse (e.g., "abc" → "cba"), generating closing brackets for a partial open sequence (e.g., "[((" → "))]") or the odds-first task which requires processing cross-serial dependencies (for a string $s_1s_2s_3s_4s_5s_6$ it should output the symbols with odd indices first $s_1s_3s_5s_2s_4s_6$, e.g., "aababb" → "abbaab"). These tasks are particularly useful for evaluating a model's ability to learn deterministic, structure-sensitive mappings between pairs of sequences.

`SymSeq` supports both recognition and transduction tasks, with the former targeting the ability to abstract and generalize grammatical rules, while transduction tasks assess whether a system can implement deterministic, rule-governed transformations. This separation aligns with theoretical distinctions in FLT, while also supporting diverse evaluation strategies across both cognitive modeling and machine learning. Both types include tasks that place varying levels of processing demands on the underlying system.

Despite this conceptual distinction, `SymSeq` follows a "one-input – multiple targets" principle and leverages a modular structure to enable users the specification of multiple tasks and corresponding target outputs for the same language generator. This can be particularly useful in the context of Reservoir Computing [96], where it is common to evaluate the same network on many tasks simultaneously.

SeqWrapper Most components of `SymSeq` can be used individually by accessing the relevant functions and classes through standard Python calls. To standardize the entry point and interface to `SeqBench` and other postprocessing pipelines, we implement a lightweight wrapper class `SeqWrapper`, which provides convenient loading routines to instantiate complete datasets and experimental setups based on YAML or Python dictionary parameter configurations. It constructs and exposes language generator objects and creates train/test datasets, which can be pre-generated (offline mode) or created on-the-fly (online mode) by accessing the relevant objects.

2.1.2 Complexity-Guided Grammar Synthesis

A core feature of `SymSeq` is the generation of structurally parameterized regular grammars with controllable complexity. Users can instantiate fully-specified grammar objects or sample a random grammar based on constraints from a wide range of properties, including but not limited to: alphabet size, level of ambiguity, number of initial and terminal states, transition density. As a primary measure for grammar complexity, we use the information-theoretic Topological Entropy (TE) introduced by Robinson [128] and applied to AGs by Bollt and Jones [14] (see also Section 5.2). In these works, an AG is treated as a dynamical system that can generate an infinite number of sequences from a finite set of symbols. Quantifying the intuitive premise that the complexity of a grammar increases with the number of unique strings it can produce, TE is defined as the asymptotic exponential growth rate of the number of grammatical strings as a function of string length.

In practice, the TE of a grammar $\mathcal{G}$ can be obtained by computing the largest real eigenvalue of its topological (boolean) transition matrix (see Methods for details). This *"direct"* approach enables a fast and precise computation of the metric, in contrast to the slower, cumbersome and error-prone *"lifting technique"* proposed initially by Bollt and Jones [14]. For comparison purposes, `SymSeq` provides efficient implementations of both methods.

For grammars satisfying certain mild constraints (see Section 5.2), understanding how its properties affect the complexity reduces to analyzing how the structure of a binary matrix influences its spectral radius. By the Perron-Frobenius theorem [68], the spectral radius is bounded from above by the maximum outdegree, although this limit is rarely reached for sparse matrices typically used in AGL studies. In practice, transition density and grammar size are among the most significant determinants of TE (see Figure 3a,b), consistent with the intuition that a larger set of symbols and more frequent transitions between them yield more complex sequences. To a lesser extent, TE is also impacted by finer structural elements: hub states (here, nodes with high outdegree) can reduce complexity and increase variability across specific instances, particularly for small and low density grammars (see Figure 3c). This is in contrast with the general expectation that high-degree nodes dominate the principal eigenvector and therefore translate to a larger spectral radius and TE. Other structural features, such as clustering, can also modulate complexity: stronger clustering tends to elevate TE (see Figure 3d), although the magnitude depends on the number and size of the clusters.

To generate grammars with specific target complexity and priors on various properties (e.g., density, see Figure 3e), `SymSeq` uses an iterative sampling algorithm that leverages Glauber dynamics on exponential random graph models (ERGMs) to construct a directed graph that satisfies all constraints. Starting from an initially random graph of specified size, the algorithm iteratively updates one edge at a time and converges to the stationary distribution encoded by the energy function (Hamiltonian). This scoring function, which incorporates the (weighted) target values of the specified properties, evaluates the fit of the current graph. The approach is
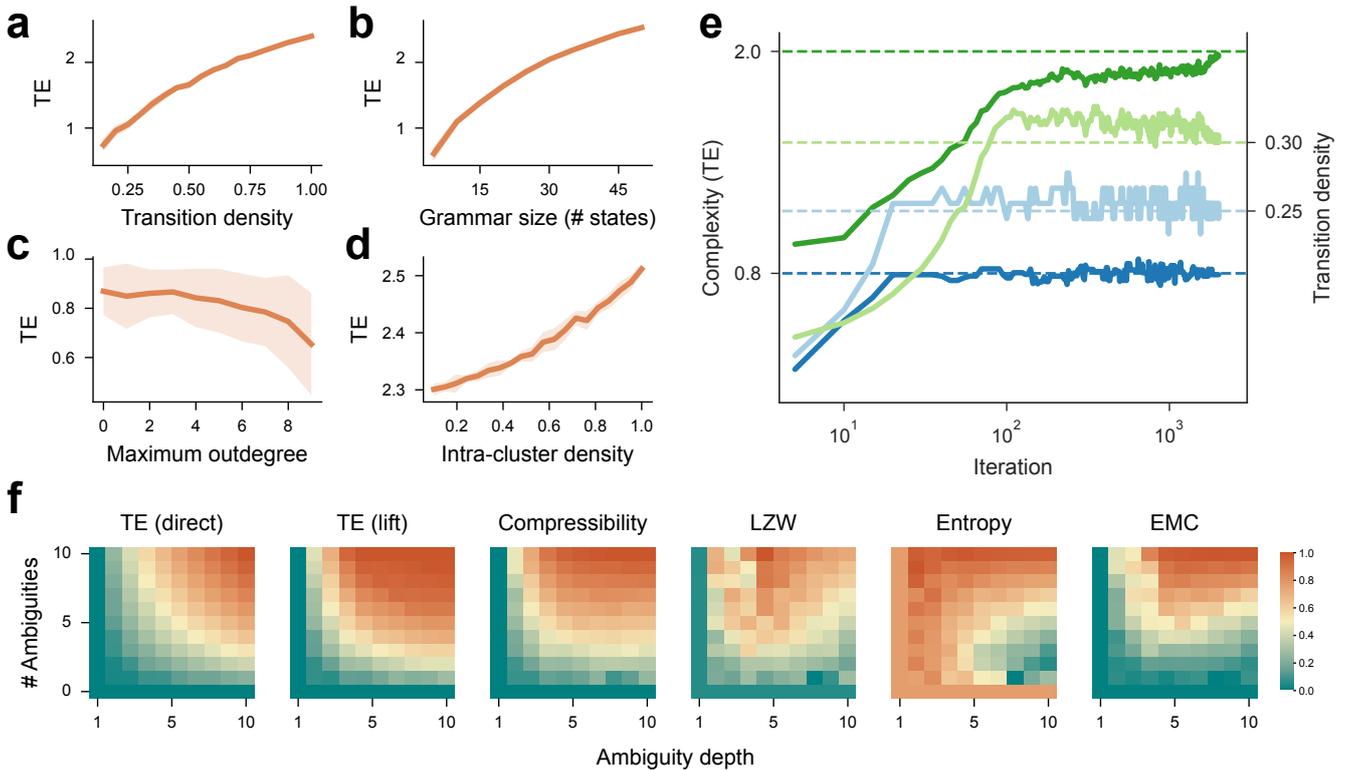
Figure 3: Topological entropy as a measure of grammar complexity. a-d) TE as a function of various grammar properties: **a)** mean transition density ($|\mathcal{Q}| = 11$); **b)** grammar size (total number of states in $\mathcal{Q}$, density fixed at 0.25); **c)** random grammar with fixed maximum outdegree of one state ($|\mathcal{Q}| = 10$, mean density 0.25); **d)** clustering strength ($|\mathcal{Q}| = 100$, 4 clusters composed of 10 states, density 0.1). All results are averaged across 100 trials. **e)** Convergence of grammar generation with target TE (0.8 blue and 2.0 green, dark shades) and mean transition density (0.25 blue and 0.3 green, light shades). **f)** Comparison of TE and other standard complexity measures, as a function of the number and depth of ambiguities (transition density 0.25). Non-structural metrics – Entropy, Compressibility, Lempel-Ziv-Welch (LZW) and Effective Measure Complexity (EMC) – were computed on 20000 strings drawn randomly from the respective grammars. Results are averaged across 10 different grammars and normalized individually for each metric. Note that the depth parameter is ignored for no ambiguities.

efficient for the small and intermediate sized grammars typically used in sequence learning studies, and can be easily extended to account for additional features (e.g., cycles) as long as they can be quantified in a reasonable time.

Compared to other complexity measures, the (direct) TE method with subscripts used here is the most sensitive and consistent with the grammar complexity as a function of ambiguity (Figure 3f). Importantly, TE is *structural but blind to bias*: although it does capture combinatorial growth of valid strings, it is insensitive to the actual symbol identities (reflected by the symmetry in Figure 3f) as well as any structure engraved in transition probabilities. Compression-based metrics exhibit a similar overall pattern, but compressibility (compression ratio) emphasizes the number of ambiguous states more than their repetitions, whereas LZW [167] does not accurately reflect the increase in complexity with ambiguity depth. In contrast, information-theoretic measures of unpredictability, such as Shannon entropy, display an opposite and more nonlinear behavior: for a small number of ambiguities entropy actually decreases with repeated occurrences, but otherwise it shows minimal variation with the number of ambiguities. Effective Measure Complexity [EMC, 58], which quantifies the balance between predictability and surprise in a sequence, exhibits a pattern similar to LZW but with a more graded response. For a given number of ambiguities, EMC shows a bell-shaped dependence on repetition depth – low for small depth, peaking at intermediate levels, and declining again for many repetitions. Thus, such measures have difficulty capturing subtle contextual variations in sequences and are not well-suited for systematic control.

Using the direct TE method and iteratively sampling the grammar space allows `SymSeq` to generate regular grammars of prescribed complexity, thus enabling systematic navigation of the complexity landscape to design controlled experimental paradigms, construct graded benchmark suites, and investigate structure-learning relationships across architectures.

2.2 `SeqBench`: Token embeddings and datasets

The `SeqBench` package offers a versatile pipeline for transforming abstract symbolic sequences into concrete, task-ready datasets, with fine-grained control over both the structural complexity of sequences (inherited from

`SymSeq`), input representations, and symbol semantics (grounding) (Figure 5). Its core functionality lies in the flexible mapping between symbolic sequences and embedded representations. Users can define custom symbol-to-stimulus mappings via base datasets (e.g., mapping symbols to images, audio samples, or arbitrary objects), specify vector embeddings of controlled dimensionality and geometric structure, or combine both approaches. This enables seamless adaptation from simple one-hot encodings for theoretical analyses to naturalistic, multi-modal stimuli for cognitive experiments or neural network training.

Beyond basic encoding, `SeqBench` supports a wide range of custom transformations of the embeddings, including domain-specific operations for audio (spectral processing, noise injection) and vision (geometric distortions, color manipulations), and allows the controlled induction of temporal perturbations, including config-urable gaps between sequence items, temporal jitter, and presentation timing manipulations. The framework also includes tools to quantify embedding complexity through representational metrics (dimensionality, effective rank) and geometric measures (pairwise distances, manifold structure).

This modular design enables straightforward data augmentation, ablation studies across representational modalities, and task-specific adaptations while maintaining independent control over rule-based symbolic structure and input characteristics. To streamline usage, `SeqBench` can generate raw sequences on-the-fly via call-backs to `SymSeq`, load stored precomputed sequences, or import external datasets, with complete experimental pipelines specified through human-readable YAML configuration files accessible via the `SeqWrapper` interface.

2.2.1 Main components

SeqDataset inherits from `torch.utils.data.Dataset` and is the central dataset interface of `SeqBench`. Depending on the configuration, sequences are either read from pre-generated files or produced on-the-fly via the `SymSeq` sequence generator (see DatasetGenerator). Raw sequences are buffered in CPU memory. Sequence items are then mapped to dataset samples, or the specified embedding (see Symbol mapping). After these mappings, they are assembled into complete training examples that contain both input and target sets. We provide dedicated modules to define target sets, for example, predicting the set of possible next items in the sequence. The SeqDataset also supports configurable gaps between sequence items and other perturbations that are controlled by the configuration dictionary, which specifies probabilities, durations, gap lengths, and noise magnitudes. These operations are applied within the `__getitem__` method, so the same base data can be reused across experiments.

DatasetGenerator manages how symbolic sequences are created, serialized, and restored. It serves as the interface between the `SymSeq`-based sequence generator and the storage format used in `SeqBench`. The generation can be performed either in a single process or in parallel across multiple processes to accelerate dataset creation, with each sample written directly to file in a consistent textual format. Each sample contains the class sequence, the corresponding state sequence, and its length. The generator also records the transition matrix of the underlying Markov chain and writes it to a dedicated file, ensuring that the probabilistic dynamics can be replayed or inspected after generation. A YAML configuration file is written alongside the data, capturing the parameters used in the run and allowing reproducibility.

Figure 4: Example sequence encodings produced within `SeqBench`. (a) Discrete token embedding using one-hot vectors. (b) Embedded sequence filtered with an exponential kernel and extended across input channels. (c) Rate-based Poisson spike trains derived from the embedded representation. Encoded continuous signals **(b)** are considered the mean density (instantaneous firing rate) of a series of independent Poisson processes to translate the signals into spike trains. (d) Google Speech Commands (GSC) representation, where each token is mapped onto one individual spoken command instance. (e) Rate-based spike encoding of GSC. (f) Spiking Speech Commands dataset representation, where each token is mapped to an instance of a spike-coded speech command.

Symbol mappings specify how each sequence token is embedded and how abstract symbolic sequences are mapped onto dynamic, numerical data structures that have a defined meaning.

The **BaseDataset** class defines the core interface for mapping abstract symbolic tokens to concrete samples. Any classification dataset – whether natively supported by SeqBench, loaded through the Tonic backend [89], or provided by user-integrated backends – can serve as a source for embedding symbolic sequences. To enable this mapping, BaseDataset maintains an internal index structure that links each class label to all samples belonging to that class. When `SeqBench` encounters a symbolic token, it resolves the token to its corresponding class and draws a specific sample from the appropriate index set. This mapping is constructed once during SeqDataset initialization by scanning the dataset and is subsequently stored for efficient reuse across sequence generation.

Beyond dataset-based mappings, `SeqBench` supports **vector embeddings** that provide static representations of symbolic tokens through methods such as one-hot encodings, binary codewords, random vectors, or custom embedding functions. These lightweight alternatives enable rapid prototyping and theoretical investigations without requiring external datasets.

Transformations provide a flexible interface to modify or extend embeddings produced by either datasets or embedding modules. Users can compose arbitrary transformation functions to adapt data representations to specific tasks and modalities without altering the core dataset logic. This interface supports operations at multiple stages, including raw samples, embedded representations, or temporally structured data, making it straightforward to implement custom preprocessing pipelines or alternative coding schemes for downstream experiments. Available transformations span multiple modalities: audio operations (mel features, MFCCs, power law compression, adaptive normalization), vision preprocessing (standard image transforms), generic tensor manipulations (convolutions, dimension operations), and spiking neural network encodings (temporal binning, Poisson spike generation, rate coding).
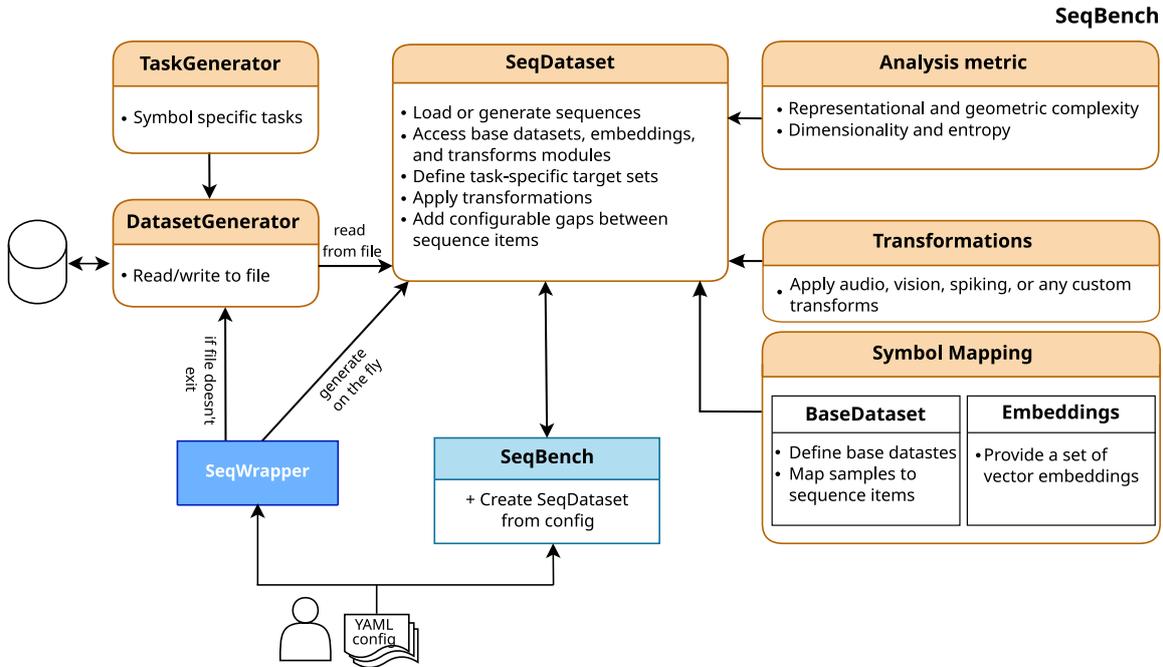

Figure 5: Schematic overview of `SeqBench`. `SeqBench` provides a modular pipeline for generating, transforming, and analyzing symbolic sequence datasets. A user-defined configuration is passed to the SeqWrapper, which serves as the interface to the `SymSeq` sequence generator. The DatasetGenerator either loads existing symbolic data from disk or invokes the SeqWrapper to create new sequences. The SeqDataset forms the core of the system, combining symbolic sequences with base datasets or embeddings, applying perturbations such as gaps and noise, and supporting optional transformation modules for audio, vision, generic tensor operations, or spiking representations. Symbol Mapping defines how tokens correspond to dataset samples or vector embeddings, while the Analysis module computes complexity metrics, including dimensionality and entropy.

Together, these components provide a modular framework that decouples symbolic structure from concrete instantiation, enabling systematic control over sequence complexity, input representation, and stimulus grounding as independent experimental dimensions. `SeqBench` thus bridges formal sequence generation via `SymSeq` with flexible dataset construction, supporting applications ranging from controlled cognitive experiments with naturalistic stimuli to systematic benchmark suites for evaluating sequential processing in neural networks.

3 Applications and use-cases

3.1 Psycholinguistics and cognitive psychology: designing experimental paradigms and datasets

Experimental paradigms in psychology and cognitive neuroscience often appear distinct, designed to probe particular aspects of cognition such as memory, attention or language. Yet beneath this apparent diversity lies a common structure that can be described using the same symbolic framework. Tasks like delayed match-to-sample, 12AX, odd-ball paradigms, serial reaction times or sequence recall all define structured relations between temporally or contextually separated symbols, often captured by simple production rules. Despite targeting different cognitive domains, they depend on similar processes of maintaining, updating, and binding representations across intervening material. However, experimental protocols and benchmark datasets in behavioral research and computational modeling are often tailored to a single task or domain, making it difficult to compare models or results across paradigms. `SymSeqBench` leverages the shared symbolic structure to address this gap, by using a declarative, grammar-based format that lets researchers generate, analyze and compare datasets across different domains using the same representational framework. This unified approach makes it easier to systematically explore how task complexity affects performance, how learning transfers between tasks, and how models generalize across paradigms, offering a principled way to identify computational mechanisms that connect cognitive phenomena that might otherwise seem unrelated. Whether the objective is to exploit these commonalities or to generate constrained experimental protocols and datasets, `SymSeqBench` provides the necessary functionality to cover a wide range of cognitive tasks.

3.1.1 Non-adjacent dependencies and temporal binding

One powerful conceptual bridge across perceptual learning, psycholinguistics, working memory, and executive control are non-adjacent dependencies (NADs), which involve maintaining and integrating information that is separated in time, space, or through noise. As illustrated by the stereotypical A-X-B structure, a NAD occurs when two (or more, dependent) elements (A and B) must be related or integrated while divided by intervening, potentially distracting elements (X, fillers).

The specific NAD structure form can range from simple, linear dependencies to more complex, hierarchical ones (Figure 6a). Linear statistical NADs involve surface co-occurrence learning based on probabilistic associations between distant elements [63], while rule-based NADs [160, 80] go beyond this by encoding abstract category-to-category mappings that generalize to novel items, requiring symbolic binding and working-memory maintenance. Multiple or alternating NADs [169, 41, 163] interleave several dependency types within a sequence, increasing interference and taxing parallel rule tracking. Contextual NADs [160, 100] (e.g., 12AX), add another layer of complexity by making the relevant dependency conditional on a preceding cue, thus demanding dynamic context updating and executive control. Finally, hierarchical or center-embedded NADs [38] introduce genuine structural recursion, where dependencies are nested within one another, imposing heavy syntactic and memory loads. Across this continuum, complexity grows from purely statistical tracking to context-dependent rule selection and ultimately to hierarchically embedded representations that require compositional structure and recursive processing.

While these structures can, in principle, be expressed through formal grammars (Figure 6b), `SymSeq` implements most of the NAD generators and related tasks as customizable modules (instead of explicit grammars) for several reasons: dedicated modules allow more intuitive control of desired statistics and experimental design; specifying complex grammars can be impractical; and `SymSeq` currently provides explicit support only for regular grammars. The *n-back* task, for instance, can benefit from a low match probability [e.g., 132] to avoid developing a biased "match" response, the introduction of lures (near misses) and repetitions for fine-grained complexity control [80], and a homogeneous distribution of matches to counter primacy/recency and serial position effects [77].

An agent's ability to handle such structures can be tested in different ways (Figure 6c), most directly by assessing whether it can recognize string legality from statistical, rule-based or contextual dependencies. Other approaches include evaluating next-item(s) prediction, syntactic generalization to novel sequence lengths or structure-preserving compositions, and perceptual generalization across variations that preserve the underlying dependencies.

Crucially, many well-known experimental paradigms can be reinterpreted or reformulated as variants of NAD tasks and are therefore easily implementable in `SymSeqBench`, as illustrated schematically in Figure 6d. In delayed response protocols [13, 31, 54], the sample (A_i) and match (B_i) items serve as dependent elements, while other stimuli possibly presented during fixation or delay periods act as fillers (X) [103]. Through flexible symbol mappings in `SeqBench` – each symbol can be independently mapped to multiple samples from distinct datasets –, users can generate both arbitrary experimental protocols (e.g., delayed match-to-category) and scenarios involving multimodal datasets.
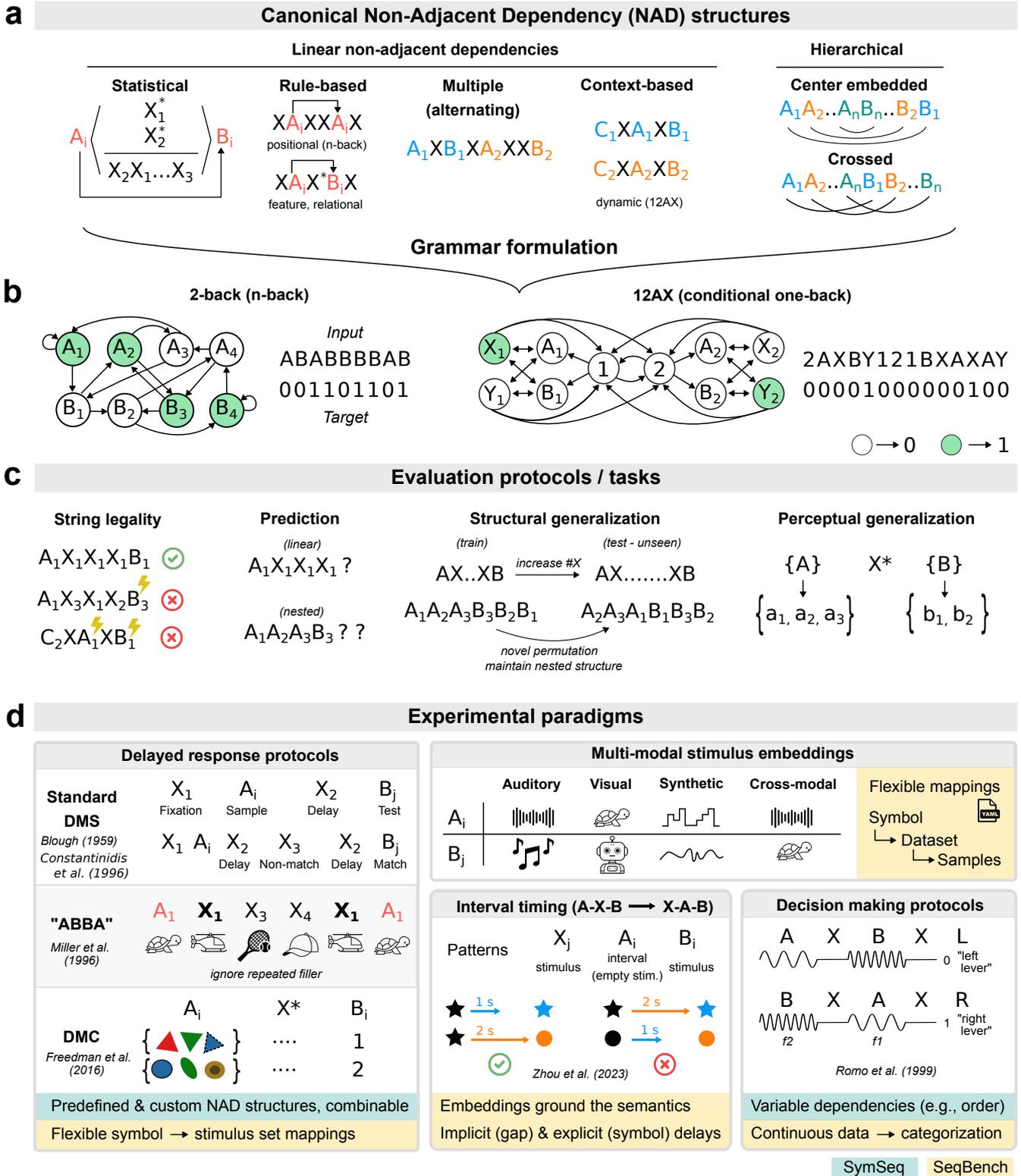
a   Canonical Non-Adjacent Dependency (NAD) structures

Linear non-adjacent dependencies   Hierarchical

Statistical

$A_i \left\langle \begin{array}{c} X_1^* \\ X_2^* \\ \hline X_2X_1...X_3 \end{array} \right\rangle B_i$

Rule-based

XA_iXXA_iX
positional (n-back)

$XA_iX^*B_iX$
feature, relational

Multiple (alternating)

$A_1XB_1XA_2XXB_2$

Context-based

$C_1XA_1XB_1$

$C_2XA_2XB_2$
dynamic (12AX)

Center embedded

$A_1A_2..A_nB_n..B_2B_1$

Crossed

$A_1A_2..A_nB_1B_2..B_n$

Grammar formulation

b

2-back (n-back)

Input
ABABBBBAB
001101101
Target

12AX (conditional one-back)

2AXBY121BXAXAY
00001000000100

○ → 0   ● → 1

c   Evaluation protocols / tasks

String legality

$A_1X_1X_1X_1B_1$ ✓

$A_1X_3X_1X_2B_3$ ✗

$C_2XA_1XB_1$ ✗

Prediction

(linear)
$A_1X_1X_1X_1$?

(nested)
$A_1A_2A_3B_3$? ?

Structural generalization

(train)   increase #X   *(test - unseen)*
$AX..XB \longrightarrow AX.......XB$

$A_1A_2A_3B_3B_2B_1 \quad A_2A_3A_1B_1B_3B_2$
novel permutation
maintain nested structure

Perceptual generalization

$\{A\} \quad X^* \quad \{B\}$
$\downarrow \qquad\qquad \downarrow$
$\{a_1, a_2, a_3\} \quad \{b_1, b_2\}$

d   Experimental paradigms

Delayed response protocols

Standard DMS
Blough (1959)
Constantinidis et al. (1996)

X_1 (Fixation) A_i (Sample) X_2 (Delay) B_j (Test)

X_1 A_i X_2 (Delay) X_3 (Non-match) X_2 (Delay) B_j (Match)

"ABBA"
Miller et al. (1996)

A_1 $\mathbf{X_1}$ X_3 X_4 $\mathbf{X_1}$ A_1
ignore repeated filler

DMC
Freedman et al. (2016)

A_i X^* B_i
{ ▲ ▼ ◢ }   ….   1
{ ● ● ● }   ….   2

Predefined & custom NAD structures, combinable
Flexible symbol → stimulus set mappings

Multi-modal stimulus embeddings

	Auditory	Visual	Synthetic	Cross-modal
A_i				
B_j				

Flexible mappings
Symbol
→ Dataset
→ Samples

Interval timing (A-X-B → X-A-B)

Patterns: X_j (stimulus) A_i (interval, empty stim.) B_i (stimulus)

★ —1 s→ ★   ★ —2 s→ ★
★ —2 s→ ●   ● —1 s→ ●
✓   *Zhou et al. (2023)*   ✗

Decision making protocols

A X B X L → 0 "left lever"
B X A X R → 1 "right lever"
f2   f1
Romo et al. (1999)

Embeddings ground the semantics
Implicit (gap) & explicit (symbol) delays

Variable dependencies (e.g., order)
Continuous data → categorization

SymSeq   SeqBench

Figure 6: Non-Adjacent Dependencies (NADs) as a unifying template across cognitive domains. Many seemingly distinct cognitive tasks can be reduced to a single underlying principle: maintaining and binding non-adjacent elements in a sequence. `SymSeqBench` enables seamless implementation of these paradigms by parametrizing the sequence, dependency rule, and context, as well as the symbol mappings and embeddings. **(a)** Canonical NAD structures may include linear (e.g., statistical) and hierarchical dependencies (e.g., crossed). **(b)** Although these typically require careful experimental design for better control, many of them can be formulated and represented as regular grammars in conjunction with appropriate target labels. For simplicity, some formal notations (e.g., start and end states) are not depicted. **(c)** Computationally, NAD structures can be evaluated in a variety of ways, including string legality, prediction, or structural and perceptual generalization. **(d)** Examples of psychological paradigms expressible as NAD structures, which can be explored using our framework through flexible configurations of `SymSeq` (teal boxes), `SeqBench` (yellow boxes), or a combination of both. These include but are not limited to Delayed Match-to-Sample (DMS) and Delayed Match-to-Category (DMC) protocols, multi- and cross-modal experiments with different symbol - dataset - sample mappings, learning of interval timings through symbol - silent stimulus mappings, and (perceptual) decision making tasks requiring a rule-based stimulus comparison and action selection.

For example, interval timing experiments [175] can be modeled as an X-A-B structure, where the cue A_i is the delay/interval itself and determines the associated item B_i. In `SeqBench`, delays can be expressed implicitly (dedicated inter-stimulus-gap parameter) or explicitly (mapping symbols to empty/silent stimuli). This flexibility in the NAD structure also supports decision-making protocols [129], for instance, using synthetic datasets that combine continuous-time and discrete inputs at the embedding level.

This section presented only a high-level overview NADs, intended to illustrate the links between computational and experimental paradigms. While all of these approaches are supported in `SymSeqBench`, for clarity, we omitted specific implementation details here and refer the user to the extensive documentation accompanying the tool.

3.1.2 AGL dataset generation

NADs have classically been studied in the context of working memory and attention experiments, because detecting and maintaining non-local relations in a sequence places demands on these cognitive systems. By contrast, artificial grammar learning (AGL) represents a broader experimental framework for investigating how humans (as well as computational models and other mammals) learn sequential dependencies of various kinds – adjacent or non-adjacent, linear or hierarchical – through mere exposure to structured inputs. A typical AGL study involves exposing subjects to *grammatical* (G) strings obeying certain rules that are to be acquired (often) implicitly, and later require them to distinguish novel G strings from non-grammatical (NG) ones. Beyond the complexity of the rule-encoding grammar itself [133], the structural properties of the stimulus (string) sets can be tailored to control a range of psychologically relevant factors in order to manipulate task difficulty and to probe specific learning theories [158, 82, 102].

Constructing balanced datasets that account for many factors simultaneously is a delicate business and has traditionally required hand-picking stimulus sets refined through iterative trial-and-error. However, such approaches do not scale well for complex grammars, larger datasets, or more than a handful controlled properties. Existing software typically focuses on the a posteriori analysis of such datasets [32, 9], and only a few tools – such as `RNNExploration4SymbolicTS` [19] and `AGL StimSelect` [5] – were released as packages to automate dataset generation. Of these, only `AGL StimSelect` incorporates psychologically relevant factors and yields experiment-ready datasets, but it was developed in MATLAB (a proprietary environment) and is no longer maintained. Building on its core concepts while greatly extending functionality, `SymSeq` fills a key gap by providing an actively maintained, open-source framework for generating flexible and well-controlled AGL datasets.
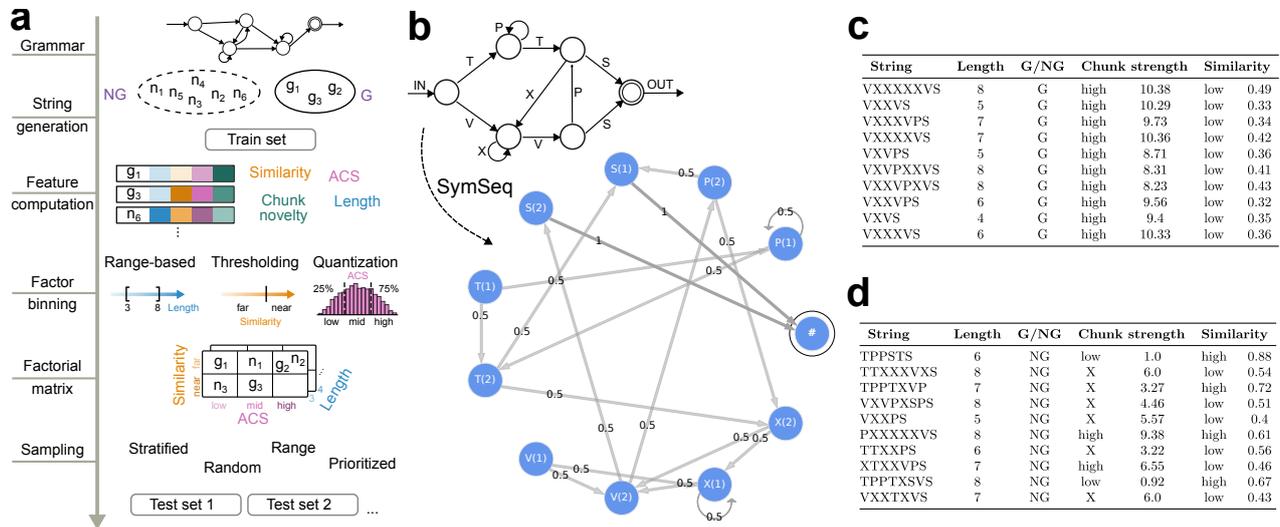

Figure 7: Generation of AGL datasets using `SymSeq`. With a few lines of code, users can generate training and test strings based on a range of constraints and complexity measures from preset or custom grammars. **a)** Schematic of the factorial sampling procedure, which begins with the generation of set of G and NG candidate strings from which a training set is selected. User-specified features are then computed for all remaining candidates, and continuous features are binned into discrete factor levels (e.g., via quantiles or thresholds). Candidate strings are subsequently filtered according to factor-level specifications, and final test items are drawn using one of several sampling procedures implemented (e.g., stratified or range-restricted). **b)** Graphical representations of the Reber grammar in AGL style (upper left) and Markov-style used in `SymSeq` (lower right). **(c-d)** Example datasets using 16 training strings, with factors including grammaticality, global ACS [82] and similarity (to training strings), and string length restricted between 3 and 8. **c)** Set of grammatical test strings with high chunk strength ($\leq$ 25th or $\geq$ 75th quantile) and low train-set similarity (< 0.5). **d)** Set of non-grammatical test strings with no chunk-strength restrictions and including both low and high similarity.

Users can specify a wide range of constraints and factors, from the number of grammatical and ungrammatical strings to the inclusion of specific deviations or positional exclusions (Figure 7a). Supported *factors* include

all quantitative metrics available within `SymSeq`'s analysis module, with *levels* individually definable for each factor or derived automatically using multi-bin quantile partitioning. Initially, a variable number of G and a larger pool of NG candidate strings is produced, after which all relevant factors are computed in a parallelized manner to ensure efficiency even for large-scale datasets. A heuristic selection stage then samples specific *factor cells* according to the user's experimental design and stratification requirements. While this step allows for fine-grained control over the factor structure of the resulting dataset, it can face limitations when strict constraints prevent sufficient sample generation.

The resulting datasets (see, for example, Figure 7b-d), including the computed factor structures, are returned as pandas dataframes for seamless downstream integration. Compared to `AGL StimSelect`, `SymSeq` adopts a more flexible and factor-driven approach. Whereas the former iteratively constructs training sets in a controlled and incremental manner, `SymSeq` emphasizes exploratory and large-scale generation, supporting a broader range of metrics and stratified sampling schemes. This design makes it particularly suited for high-throughput experimentation, model benchmarking, and systematic comparisons across AGL paradigms.

3.2 Cognitive computing benchmarks: evaluating biological, neuromorphic and machine learning architectures on psychologically meaningful tasks

Understanding how different neural systems process and learn sequences is central to both neuroscience and artificial intelligence. Biologically detailed models aim to uncover the circuit and cellular-level mechanisms that give rise to temporal and contextual processing in the brain. Neuromorphic implementations, in turn, translate these principles into efficient, real-time computation under physical and energy constraints. The following sections demonstrate how `SymSeqBench` provides a systematic benchmarking framework for both domains: first by evaluating biologically inspired models trained with local learning rules, and then by assessing neuromorphic architectures designed for low-power temporal processing.

3.2.1 Biological neural networks

Biologically-detailed models of sequence learning, such as recurrent networks of spiking neurons equipped with local or unsupervised learning rules, are typically developed to capture the transition and timing aspects [106, 30, 15, 16, 90, 44], with fewer propositions addressing chunking [53, 4], artificial grammar learning [42], symbolic sequence encoding [43], or hierarchical structure transfer [172]. Recent work has also shown that spike-rate adaptation can support working memory demands in language processing [51] and that simplified dendritic morphologies preserve essential memory-expanding transformations [123] suitable for sequential learning. While these models provide insights into the biophysical and circuit-level mechanisms supporting cognitive computations, they are often evaluated on a narrow range of handcrafted tasks with limited input variation, intended to highlight a single functionality. As a result, their computational scope and limitations remain difficult to assess methodically, and direct comparisons across models are rare. A case in point is the ability to process higher-order (non-)Markovian sequences [97, 30, 4, 15], which is typically evaluated on a few randomly and manually selected sequences with fairly static, synthetic stimulus encodings (e.g., Poisson-rate spikes), seldom considering essential factors such as sequence complexity, sensitivity to deviants or symbol repetitions, or broader variability in stimulus statistics.

`SymSeqBench` facilitates experiments beyond prototypical demonstrations by offering a unified, systematically parameterized suite of sequence-processing tasks that vary structural dependencies, timing, and stimulus complexity, enabling rigorous model-to-model and model-to-data comparisons. The tool enables automatic string generation of tunable complexity that can be further constrained by experimentally grounded metrics, allows flexible switching between synthetic and real input encodings as well as their easy manipulation, and provides fine-grained control over temporal stimulus properties such as duration, inter-stimulus intervals, and noise.

To illustrate some of these capabilities, we compared two models by Cone and Shouval [30, CS] and Asabuki et al. [4, AKF] on three example tasks assessing memory capabilities. First, `SymSeq`'s AG generator constructs grammars of varying complexity by manipulating the number and repetitions of ambiguous states. For each grammar, we generate a set of input strings using symbols from the alphabet, and define two tasks and corresponding sets of target labels: (i) n-step memory ($n \leq 10$), in which the n-th previous symbol must be recalled, and (ii) context resolution, in which not just the current symbol but rather the underlying indexed state must be identified. Identifying the exact state in the grammar is equivalent to disambiguating the relevant history and, in such regular grammars, it can be considered as a proxy for prediction (the indexed representation is Markovian, i.e., memoryless). Performance on both tasks generally declines with increasing TE complexity (see Figure 8a), but the AKF model consistently outperforms CS, possibly reflecting crucial computational benefits of its dendritic processing and nonlinear gating mechanisms.

The third task probes the system's working memory using sequences of the form $A_i X^* B_i$, a classical formulation of non-adjacent dependencies. Also known as the *counting* task [86, 42], this involves correctly predicting
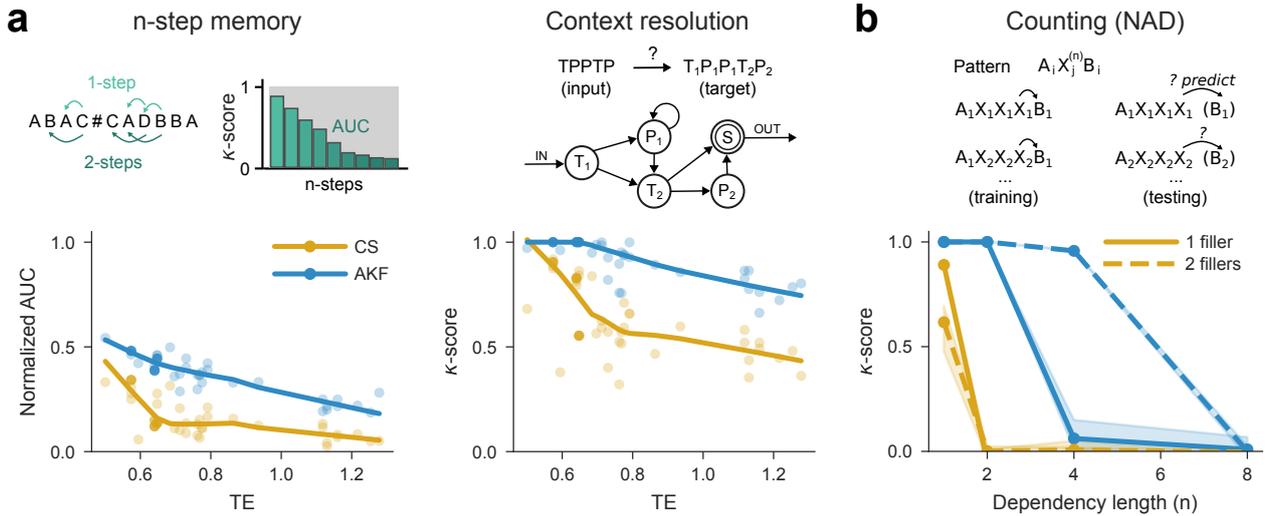
Figure 8: Performance of biologically-plausible models on cognitive sequence processing tasks. Two models featuring non-supervised learning – Cone and Shouval [30, red] and Asabuki et al. [4, purple] – were evaluated on memory-related tasks using reservoir computing (see Methods for details). **a)** Learning ambiguous adjacencies in regular grammars created using `SymSeq`, shown as a function of complexity: (left) aggregated n-step symbol (label) memorization (normalized area under the curve (AUC) of Cohen's κ across valid n-steps, here 9); (right) context resolution (classification of the indexed state, e.g., A_1 or C_3). For each property combination ($|\Sigma| = 5$, $|q_0| = 2$, $\text{len}(S_i) \in [2, 10]$, ≤ 3 ambiguities with depth ≤ 4), three random grammars were generated. Each data point represents such a grammar, with curves showing nonparametric LOWESS reggression fits. **b)** Non-adjacent dependency (counting) task with two dependent elements (A_1B_1 and A_2B_2) and one (solid curves) or two (dashed curves) distinct filler items. Performance is plotted against filler repetitions, based on the prediction accuracy of the final (dependent) element.

the last dependent element B_i after seeing the final filler item X_j. While the CS model is unable to handle more than one filler occurrences, AKF can solve the task up to a dependency length of 8. More interestingly, having two distinct filler items (X_1 and X_2) significantly improves the model's performance. This is in line with experimental findings in Humans, which demonstrate that larger filler variability can be computationally beneficial [111].

3.2.2 Artificial and neuromorphic neural networks

Neuromorphic systems are designed to learn and process data in real time while operating under strict energy constraints. The input they receive is often sequential and heterogeneous, combining spatial and temporal patterns of varying complexities and scales. Many of these systems are still under active development, exploring novel mechanisms that demand rapid iteration to uncover their strengths and limitations. Designing dedicated benchmarks is therefore becoming increasingly critical [171]. Current datasets are either too simple to capture long-term temporal dependencies, such as the Spiking Heidelberg Digits (SHD), Spiking Speech Commands (SSC) [34], or the NeuroMorse [161] datasets; others are narrowly specialized to evaluate particular capacities, such as memorization [153]; and datasets such as [79, 69] are designed for large-scale networks that are unsuitable for energy-constrained settings. Recent efforts, such as the Neuromorphic Sequential Arena (NSA), address this gap by providing a suite of real-world temporal processing tasks that evaluate spiking models across performance, efficiency, and scalability dimensions [21]. However, NSA lacks explicit control over the complexity of temporal dependencies, limiting its utility for systematically probing models. The DVS-Gesture-Chain (DVS-GC) introduced in [157] constructs a temporal recognition task by forming sequences of gestures, where each sample is built by concatenating multiple individual gestures from the DVS Gesture dataset [2]. Given that the samples are concatenated randomly, it is difficult to assess or tune the complexity of the gesture chain.

`SymSeqBench` generates sequences with controllable spatial and temporal complexity. Spatial complexity is modulated by selecting a base dataset and concatenating samples into sequences through grammar-based compositional rules (see Figure 4). These rules govern how elements are combined, thereby introducing temporal dependencies at both short and long timescales. Although synthetically generated, these tasks capture essential characteristics of real-world problems, making them both interpretable and relevant for practical applications. This controllability directly addresses the lack of adjustable temporal scales in benchmarks such as NSA, enabling systematic evaluation of how models cope with varying degrees of temporal dependencies.

To demonstrate these properties in practice, we consider representative `SymSeqBench` configurations with the AG grammar settings specified in Table 1 and Table 2 using SHD, SSC, and GSC as base datasets. SHD and SSC samples are processed using a 10 ms temporal binning resolution, with their spiking inputs spatially reduced from 700 to 140 channels. GSC inputs are binned at 10 ms and transformed using a 40-channel Mel filterbank.

The neuromorphic baselines are implemented as spiking neural networks composed of 5 fully connected layers of 256 leaky integrate-and-fire (LIF) or adaptive LIF (adLIF) neurons (for more details, see methods and the study by Bittar and Garner [12]). We first discuss these SNN baselines before turning to the ANN baselines (GRU [23], Mamba [61], Transformer [156]), which are provided as reference points to contextualize the performance of SNN models against widely used sequence learning architectures. We benchmark the models on classifying the indexed states underlying the input sequences (see Section 2.1.1). Across the sequences composed of SSC and SHD as base datasets, adLIF networks outperform their LIF counterparts (see Table 1), indicating that adaptive neuronal dynamics can be beneficial for learning complex spatio-temporal dependencies [7]. For sequences with the GSC base dataset, the adLIF exhibits reduced performance compared to LIF. Despite being more computationally expressive, the vanilla adLIF used in this study is known to suffer from training instabilities, as outlined in [7]. We leave for future work the improvement of the adLIF model used in the current study by incorporating features discussed in [7] and [47]. Overall, the performance gap to the theoretical upper bound indicates substantial room for further improvement in model architectures and training strategies. In contrast, the ANN baselines obtain higher accuracies overall (see Table 2). For sequences with lower complexity (TE = 1.71), we use 4-layer architectures with 256 units per layer. In this regime, all three architectures perform well, with the GRU achieving the highest accuracy, suggesting that moderate temporal dependencies can already be captured effectively by classical recurrent models. For sequences with higher complexity (TE = 2.61), we use larger networks with 8 layers of 1024 units. In this setting, Mamba attains the best performance, indicating that its state space formulation scales favorably as sequences become longer and more complex. Nevertheless, even the strongest ANN baselines exhibit a noticeable performance gap at higher sequence complexity, indicating the need for further development of architectures capable of handling such sequences.

In summary, `SymSeqBench` enables systematic evaluation of sequential models by providing explicit control over spatial and temporal complexity. It reveals clear benefits of adaptive spiking dynamics while exposing persistent performance gaps as sequence complexity increases, even for strong ANN baselines. This makes SymSeqBench a useful diagnostic tool for identifying limitations in current architectures and guiding the development of more effective temporal learning mechanisms.

| | Seq SHD TE = 1.35 | | Seq SSC TE = 1.35 | | Seq GSC TE = 1.35 | |
	LIF	adLIF	LIF	adLIF	LIF	adLIF
Nb. param.	327k	332k	327k	332k	295k	299k
Accuracy	56.80% ± 1.39	**65.79% ± 1.44**	49.92% ± 0.48	**57.44% ± 0.6**	**61.46% ± 0.28**	59.70% ± 4.84

Table 1: Neuromorphic network performance on the context resolution task using temporal SHD, SSC, and GSC as base datasets (abbreviated as Seq SHD, Seq SSC, and Seq GSC). A single gap element is inserted between sequence items, and accuracy is measured during this gap interval. Experimental configuration: alphabet size $|\Sigma| = 11$, 10 ambiguities with depth 10, transition density = 0.04, $|q_0| = 4$ wit a resulting sequence complexity of TE = 1.35, minimum and maximum sequence length: $\text{len}(S_i) \in [1, 30]$. Results are reported as mean ± standard deviation over 4 runs.

| | Seq GSC TE = 1.71 | | | Seq GSC TE = 2.61 | | |
	Attention	GRU	Mamba	Attention	GRU	Mamba
Nb. param.	4.276M	1.646M	2.113M	60.041M	61.632M	51.941M
Accuracy	86.62% ± 2.8	**92.93% ± 2.7**	88.78% ± 3.2	45.09% ± 0.5	46.25% ± 0.652	**60.60% ± 1.852**

Table 2: Artificial network performance on the context resolution task using the temporal GSC as a base dataset (abbreviated Seq GSC). A varying number of gaps in the range $[0, 30]$ is introduced during training, and evaluation uses a fixed gap duration of 30. Parameters for the sequences with a complexity of TE = 1.71: $|\Sigma| = 15$, 14 ambiguities with maximum depth 15, transition density = 0.02, $|q_0| = 4$, $\text{len}(S_i) \in [1, 30]$. Parameters for the sequences with a complexity of TE = 2.61: $|\Sigma| = 35$, 34 ambiguities with maximum depth 35, transition density = 0.01, $|q_0| = 4$, $\text{len}(S_i) \in [1, 60]$. Results are reported as mean ± standard deviation over 4 runs.

3.3 Syntactic structure of animal behavior: analyzing latent dynamics from the perspective of generative linguistics

Beyond the generation of computational tasks and datasets of controllable complexity, `SymSeqBench` provides a comprehensive set of analysis tools and metrics to quantify the complexity of symbolic sequences and infer the properties of the underlying generative processes. Given the scope and breadth of these metrics (see Section 5.2), their application to empirical datasets can provide a valuable complement to understand the syntactic structure of different types of sequences across different scientific domains. In this section, we exemplify these applications and demonstrate the types of insights they can provide, using openly available datasets comprising behavioral sequences across different species and different behavioral categories.
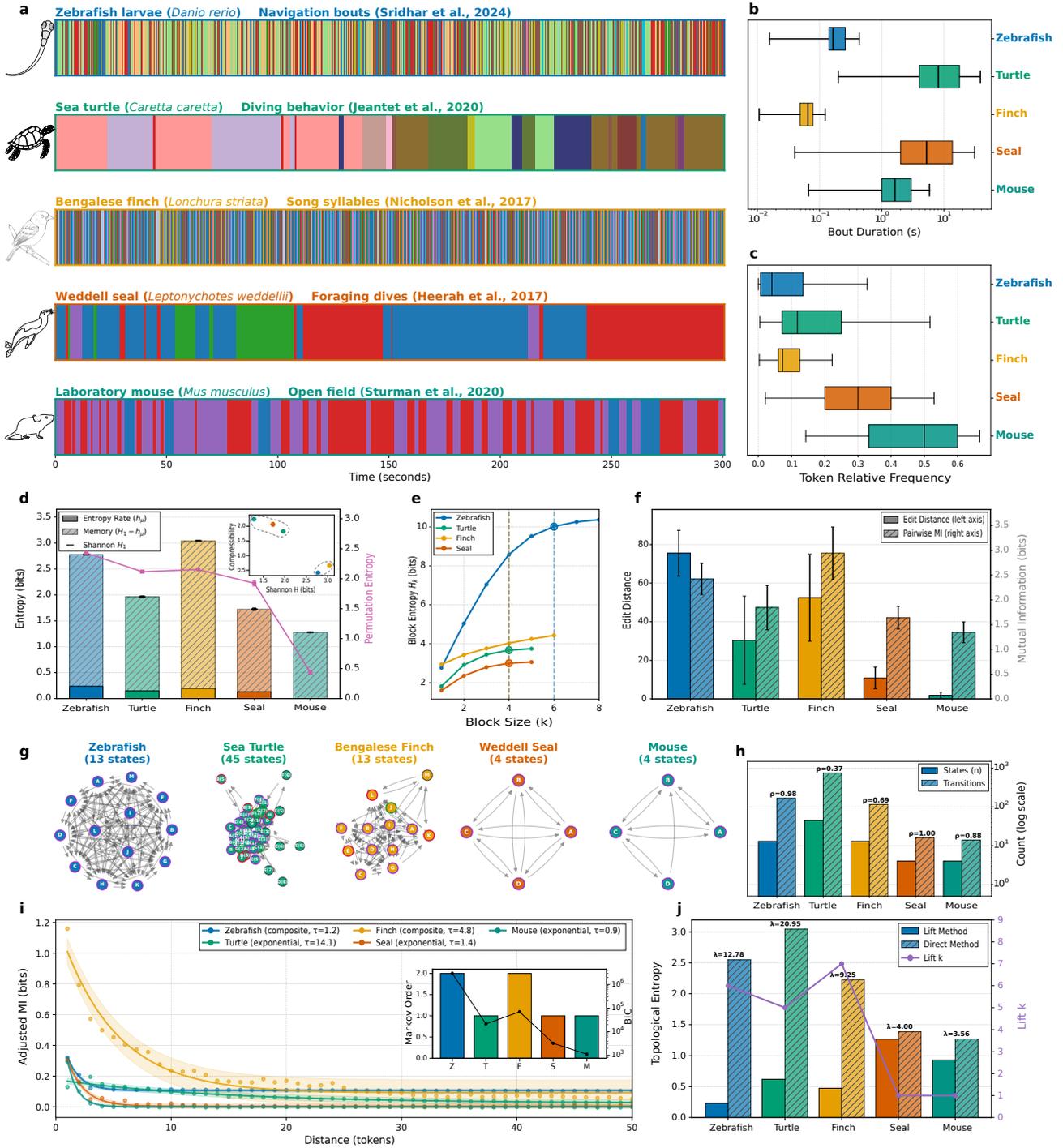
Figure 9: Multi-scale analysis of behavioral sequence complexity across animal species. Behavioral sequences from five species (zebrafish larvae, sea turtles, Bengalese finches, Weddell seals, laboratory mice) were analyzed at four hierarchical levels. **a)** Ethograms showing representative sequences with color-coded behavioral states over 300-second windows. **b-c) Token-level analysis:** bout duration distributions and relative frequencies reveal species-specific temporal patterns. **d-e) String-level analysis:** the entropy rate (h_μ) captures stochastic complexity, while the memory component ($H_1 - h_\mu$) quantifies temporal dependencies, and Shannon entropy (H_1) represents total uncertainty (d, left axis). Permutation entropy (d, right axis) provides a complementary, model-free, complexity measure and string compressibility correlates strongly with Shannon entropy (d, inset). The growth of block entropy (H_k) as a function of block size (e) quantifies long-range dependencies with the marked saturation points revealing memory depth. **f) String-set-level** edit distance and pairwise mutual information quantify stereotypy. **g-j) Grammar-level analysis:** transition graphs (g) and their properties (h) can be estimated from first-order transition probabilities, while the existence and characteristics of non-adjacent and hierarchical relations can be inferred from information-based metrics (i), Markov order fitting quality (i, inset) or topological entropy measures (j). *Data sources:* [145] (zebrafish), [75] (turtle), [109] (finch), [65] (seal), [151] (mouse). *Animal illustrations:* [33] (zebrafish), [155] (mouse) from SciDraw under CC-BY 4.0; turtle, finch, and seal from Vecteezy.com.

17

Bolstered by advanced machine learning tools that automate tracking and positional estimation from recorded behavioral observations (e.g. DeepLabCut [101, 170], DeepOF [104] or SLEAP [118]), combined with the streamlined identification, annotation and labeling of discrete behavioral bouts and sequences [166, 95], the emerging new scientific discipline of computational neuroethology [36, 159] can revolutionize how we understand naturalistic animal behavior and the characteristics of the underlying generative processes, providing a new, integrative view on brain, cognition and behavior.

By focusing on abstract and generic symbolic sequences, while retaining close inspiration from psychology and cognitive sciences, the tools and metrics provided by `SymSeqBench` can be a valuable complement to behavioral analysis pipelines (as well as other classes of empirical data that can be expressed as symbolic sequences). To illustrate the scope of analysis metrics and tools, we analyzed behavioral sequences from five different species (obtained from openly available datasets) at four hierarchical levels (see Figure 9 and the formal description of all metrics in Section 5.2): token, string, string-set, and grammar. While token-level analysis primarily emphasize differences in distributional statistics and temporal properties of individual behaviors (with Zebrafish swimming and Finch's songs showing shorter and less frequent syllables, Figure 9 a-b), string- and string-set-level analyses reveal differences in contextual depth and complexity. Decomposing the entropy of individual strings into a stochastic component (entropy rate, h_μ) and a memory component (residual entropy, $H_1 - h_\mu$) allows us to quantify both the total uncertainty / randomness in the observed behavioral sequences (Figure 9 d, left axis) and how much of that uncertainty is due to higher-order temporal dependencies (Figure 9 d, stacked bars). The permutation entropy then provides a complimentary, model-free complexity measure ((Figure 9 d, right axis) re-inforcing the overall pattern: mouse open field behavior is the simplest, seal and turtle show stereotyped and low-entropy behavioral sequences, whereas zebrafish and finch tend to exhibit higher complexity. These metrics also align with sequence compressibility (Figure 9 d, inset), yielding a clear grouping between high-entropy, low-compressibility (complex) sequences for zebrafish and finch versus low-entropy, high-compressibility (simple) sequences for the other datasets. If we assess how the sequence entropy changes if we chunk the sequences into increasingly larger groups (Figure 9 e), the same pattern is evident with zebrafish behaviors showing longer contextual dependencies (saturating at $k = 6$), turtles and seals showing the shortest ($k = 4$). It is worth noting that this analysis also emphasizes limitations of these metrics that would require longer individual strings to yield meaningful conclusions for the two remaining datasets. The complementary string-set complexity metrics (Figure 9 f) then quantify how stereotypical different behavioral epochs / strings are compared to the others in the string set, demonstrating the same pattern: simple, stereotypical behaviors for mouse, seal and turtle; complex, flexible behaviors for zebrafish and finch.

Beyond statistical quantification and structural analyses, the set of metrics we implement allow us to identify contextual dependencies and infer the properties of the generative grammars underlying observed behaviors. First-order transition probability graphs and corresponding properties (state and transition densities, Figure 9 g-h) vary according to the complexity of the observed behavioral repertoire but prove insufficient to capture the full complexity of generative grammars (Figure 9 i, j). These analyses directly confront a fundamental challenge in formal language theory: the *grammatical inference problem*, which asks whether we can unambiguously determine the generative grammar—and specifically, its position within the Chomsky hierarchy (regular, context-free, context-sensitive, or recursively enumerable)—from observed sequences alone. Seminal theoretical work has established fundamental limitations on this endeavor. Gold's theorem [57] demonstrated that no superfinite class of formal languages is identifiable in the limit from positive examples alone (i.e., without access to negative examples indicating which strings are *not* part of the language). This negative result extends even to the class of regular languages, the simplest level of the Chomsky hierarchy [3]. Furthermore, even when the target grammar class is known a priori and restricted to regular languages, finding the minimal (smallest) consistent automaton or grammar is NP-hard [121], and the approximation within constant factors remains computationally intractable [20, 87]. For context-free and more expressive grammars, complexity increases dramatically: exact inference is undecidable in the general case, and even for learnable subclasses such as substitutable context-free languages [28], polynomial-time algorithms require strong structural constraints and access to characteristic samples that may not be available in naturalistic behavioral data.

These theoretical barriers force us to adopt a pragmatic approach to analyzing sequences: rather than attempting to extract a precise minimal grammar (which is computationally infeasible), we employ a battery of complementary statistical metrics to *characterize* sequence complexity, long-range dependencies, and structural properties that collectively provide evidence for or against membership in specific grammar classes. This methodology is flawed, trading rigour for feasibility, but, by employing multiple complementary and independent metrics it can provide convergent evidence to make informed inferences about the underlying generative processes.

Sequential, adjusted mutual information (MI) decay (adapted from [92, 130, 131] Figure 9 i) quantifies long-range statistical dependencies; rapid exponential decay in seals and mice indicates short-term correlations characteristic of low-order Markov processes, while extended, composite decay in finches and zebrafish reveals longer-range, hierarchical structure consistent with supra-regular (beyond finite-state) grammars. This interpretation is validated by Markov order estimation (inset), which demonstrates that sequences observed for

turtles, seals, and mice can be adequately modeled as first-order Markov processes (low BIC scores), whereas zebrafish and finch datasets require longer contextual dependencies and are not well approximated by low-order Markov models, suggesting non-Markovianity and potentially hierarchical structure. These metrics are then complemented by topological entropy (TE) analysis (Figure 9 j) which provides a canonical, independent, model-free measure of generative capacity [128, 14]. Seal and mouse data exhibit convergence with the lift method at depth 1 (consistent with first-order Markov processes and regular grammars), whereas zebrafish, turtle, and finch datasets require larger contextual windows and show systematic discrepancies between direct computation and lift-based approximation. These divergences suggest that simple finite-state models cannot adequately capture the generative complexity, pointing toward context-free or higher-order dependencies. Importantly, the convergence values (λ) provide a quantitative complexity ordering: zebrafish exhibit the highest TE ($\lambda = 20.95$), reflecting rich navigational repertoires with extensive state spaces, while seals and mice show the lowest ($\lambda = 4.00$ and 3.56), consistent with stereotyped, low-entropy foraging routines.

Collectively, these grammar-level analyses do not definitively resolve the class membership of observed behavioral sequences within the Chomsky hierarchy, an objective that formal language theory has proven to be fundamentally unattainable without additional constraints. However, the convergent patterns across multiple independent metrics (MI decay timescales, Markov order estimates, topological entropy convergence properties, and transition graph structures) provide principled, probabilistic evidence that zebrafish navigation and finch song likely involve supra-regular generative processes (consistent with what was proposed in the corresponding original publications), while seal, turtle, and mouse behaviors appear consistent with regular or low-order Markovian grammars. This inferential framework, grounded in rigorous computational constraints, represents a pragmatic and methodologically sound approach to characterizing the syntactic complexity of naturalistic behavioral sequences.

4 Discussion and Outlook

Sequences act as the shared currency across cognitive processes, behavioral repertoires, and modern AI architectures, offering unified means to examine the structural dependencies that shape natural and artificial information processing. Leveraging this common structure demands benchmarks that are cognitively inspired, theoretically principled and methodologically flexible, capable of probing sequence complexity in a controlled yet comparable way. However, current resources remain fragmented and lack a shared representational foundation: large-scale language benchmarks are costly and opaque with respect to the specific abilities they test, while lightweight synthetic tasks rarely capture the spatio-temporal diversity and compositional structure central to human and animal cognition. `SymSeqBench` aims to overcome these challenges through an open-source initiative that offers a principled framework for generating, transforming, and analyzing symbolic sequences with controllable complexity. By unifying dataset construction, task specification, and multi-scale complexity analysis through low-barrier and user-friendly interfaces, we expect the tool to engage researchers from a variety of disciplines interested in systematic, cross-domain analysis and benchmarking of cognitive and computational models.

`SymSeqBench` taps into computability and learnability theories and uses the less-known but powerful TE metric to efficiently sample user-constrained formal grammars of desired complexity, filling a key gap in synthetic data generation tools and providing new benchmarks for biologically-detailed, artificial and neuromorphic systems. Complementing but not replacing full-fledged parsers [11, 139] and grammar induction libraries [142, 107] that typically deal with large text corpora or genome data, for example, it focuses on simple and interpretable structures and allows their manipulation in behaviorally and computationally relevant ways (e.g., by introducing deviants). Together with many pre-built cognitively inspired paradigms (e.g., NAD learning or AGL) and flexible symbolic- and embedding-level task specifications, `SymSeqBench` enables methodical investigation of statistical and rule-based learning with direct links to a rich experimental, behavioral and theoretical literature. Conversely, linguists and cognitive scientists can benefit from the automatic generation of experiment-ready, balanced AGL datasets from arbitrary grammars at an unprecedented level of fine-grained control and scale. To the best of our knowledge, no other actively maintained software offers comparable functionality: AGL Stimselect is more limited and unsupported, and AGSuite provides only a web-based feature-analysis interface.

We nevertheless expect `SymSeqbench`'s primary use-case to be the evaluation of neural network models across architectures. To this end, it includes efficient routines for mapping symbols onto either synthetically generated embeddings or established datasets while exposing flexible transformations and property-manipulations during runtime. Our tool thus addresses a critical gap in evaluating sequential processing capabilities: while large language models are typically assessed on complex, naturalistic benchmarks [147, 162, 91], these evaluations make it difficult to isolate specific computational abilities or systematically control task complexity [124, 126]. Conversely, targeted synthetic tasks designed for recurrent neural networks [59, 168] or biologically-detailed spiking neural networks [46, 34] often lack the scope and standardization needed for cross-architecture comparison. `SymSeqBench` bridges this divide by enabling controlled evaluation across the full spectrum of neural architectures – from feedforward networks and recurrent architectures (LSTMs, GRUs) [23] to attention-based

transformers and large language models [40, 156] – using tasks with systematic complexity control, explicit theoretical grounding and clear relations to Human cognitive performance.

This positions our tool uniquely between libraries focusing on dataset preprocessing [e.g., tonic 89], reinforcement-learning (RL) environments like NeuroGym [105], full-stack SNN frameworks (Norse [116], SpikingJelly [48], SNNTorch [46], Jaxley [39]) or neural simulators (NEST [56], Brian2 [148]), or applications designed for human experiments (PsychoPy [117]). Although `SymSeqBench` shares certain capabilities with these tools, it remains distinct in scope and design by providing a unified framework for systematic sequence complexity control applicable across all neural modeling paradigms.

Whenever possible, it leverages and builds on high-quality software to benefit from advances in the deep learning community, e.g., for dataset handling (tonic [89]) and transformations (torchvision [98] and torchaudio [71]). Beyond its central emphasis on temporal processing, the framework smoothly integrates and transitions between discrete vector embeddings, continuous-time signals and spiking encodings, enabling efficient integration with Torch-based models while also providing generic output formats compatible with any tool. Unlike NeuroGym or PsychRNN [45], which mostly focus on decision-making or delayed-response paradigms with pre-defined input structure and little control over task difficulty, `SymSeqBench`'s symbolic framework allows defining more general sequence-processing tasks while retaining maximum flexibility on the input embeddings. In contrast to some approaches (conn2res [152], PsychRNN), it deliberately avoids tight integration with specific simulation engines, which leads to some user overhead but also guarantees complete freedom of backend choice and shields the tool from premature obsolescence [e.g., Theano-based PyCog 143].

Decoupling the task formulation from the symbol embeddings is a novel approach that has important conceptual and practical implications: it allows the same task to be evaluated using embeddings/datasets (explicitly specified by users) of various complexities, which is known to impact generalization capacity [122, 78]; provides direct support for multi- and cross-modal experiments, a key line of research for studying multisensory integration and cross-modal generalization underlying abstract representation learning in biological [110, 136, 62] and artificial [108, 125, 72] systems; and allows `SymSeq` and `SeqBench` to be used separately, e.g., for generating only AGL datasets versus attributing a meaning to the symbols and evaluating computational models. The modular architecture naturally blends high customizability with fast prototyping, allowing the deployment of complete data generation pipelines using simple configuration files (suitable for non-coders), thereby letting researchers focus more on the models, shortening experimental round-time and ensuring fully reproducible [115, 10] datasets.

Beyond synthetic task generation, `SymSeqBench` provides a comprehensive suite of analysis tools applicable to both generated and empirical sequences. The multi-scale metric framework – spanning token, string, string-set, and grammar levels – enables the systematic characterization of sequence complexity across diverse domains, from animal behavioral to neural activity patterns and linguistic corpora. Any source of empirical data that can be cast as a set of symbolic sequences is amenable to analysis within this framework.

At the grammar level, topological entropy (TE) serves as the primary structural complexity measure, quantifying the asymptotic exponential growth rate of unique grammatical strings as a function of length [128, 14]. Unlike compression-based metrics (Lempel-Ziv, Kolmogorov complexity) that emphasize statistical regularities or string entropy metrics that emphasize local variability, TE captures the combinatorial richness of the grammar by analyzing its adjacency structure providing a fast, precise, and theoretically grounded measure that can be used both for generation (creating regular grammars of controlled complexity) and analysis (determining the complexity of generative rules). Our comparative analyses demonstrate that TE exhibits superior sensitivity to structural variations (e.g., ambiguity, clustering, transition density) compared to compression ratios or perplexity-based measures, making it especially suited for systematic control and manipulation of dataset complexity.

However, TE has inherent limitations that must be acknowledged. First, it applies exclusively to *regular grammars* (finite-state automata); context-free and more expressive grammars require alternative complexity measures or approximations (e.g., probabilistic variants or upper bounds). Second, TE quantifies only the *potential* for generating diverse strings and does not account for statistical biases in actual string distributions – two grammars with identical TE may produce vastly different empirical sequence statistics if certain paths are favoured. Third, the direct eigenvalue method assumes access to the complete transition structure; for inferred or partially observed grammars, estimation errors propagate into TE calculations, potentially yielding misleading complexity assessments.

The string-level and corpus-level metrics complement TE by capturing statistical, distributional, and information-theoretic properties that reflect how complexity manifests in observed sequences. Entropy decomposition (entropy rate, memory component, Shannon entropy) quantifies stochastic complexity and temporal dependencies; edit distance and mutual information assess inter-sequence similarity and stereotypy; perplexity and divergence measures evaluate model fit and distributional shifts. This multi-metric approach provides convergent evidence for characterizing sequence structure, mitigating the limitations of any single measure.

Crucially, these analysis tools are entirely decoupled from the generation pipeline, enabling their application to arbitrary symbolic time-series data. Users can import empirical sequences - from animal behavioral ethograms, neural population activity latent states, or time-series data converted to symbolic form via SAX

(Symbolic Aggregate approXimation) [94] - and apply the full metric suite to quantify complexity, infer approximate grammar classes, or compare statistical properties across conditions. This positions `SymSeqBench` as a versatile analytical toolkit for computational neuroethology, cognitive neuroscience, machine learning and artificial intelligence, as well as any other domain where sequential structure plays a central role.

4.1 Grammar inference and the decidability constraint

A natural application domain of `SymSeqBench`'s analysis capabilities is the inference of generative grammars from observed sequences. This task has profound implications for understanding the computational principles underlying sequential structure, inferring the characteristics of generative mechanisms and deploying rigorous computational benchmarks. However, the objective confronts fundamental theoretical limits established by formal language theory and computational learning theory. Gold's theorem [57] proved that no superfinite class of formal languages is identifiable in the limit from positive examples alone, even for regular languages [3]. Moreover, finding the minimal consistent automaton for a given set of strings is NP-hard [121], and approximation within constant factors remains intractable [20, 87]. For context-free and more expressive grammars, exact inference is undecidable in the general case, and even learnable sub-classes require strong structural constraints (e.g., substitutability) and characteristic samples that may not be available in empirical data [28, 37].

This impossibility does not invalidate grammar inference as a research goal; rather, it clarifies the boundaries of what is computationally achievable and motivate pragmatic methodologies. Instead of attempting to extract the true minimal grammar, `SymSeqBench`'s multi-scale analysis framework provides *probabilistic evidence* for grammar class membership and structural properties through consensus patterns across independent metrics. For example, rapid mutual information decay, low Markov order, and TE convergence at depth 1 collectively suggest a regular (finite-state) grammar, while extended MI decay, high block entropy growth, and TE divergence between direct and lift methods indicate supra-regular structure (context-free or beyond). This approach trades exactness for feasibility, leveraging computational tractability to make informed, evidence-based inferences about generative processes.

Currently, `SymSeqBench` provides basic grammar inference capabilities for regular languages through automaton induction from positive examples, suitable for exploratory analysis and hypothesis generation. Future development will incorporate more sophisticated inference algorithms (e.g., state-merging methods, probabilistic grammar induction, Bayesian approaches) and extend support to restricted context-free sub-classes, enabling richer structural modeling while respecting computational constraints. Users should interpret inferred grammars as *hypotheses* rather than ground truth, validated through cross-metric consistency checks and generalization performance on held-out data.

4.2 Future directions

While `SymSeqBench` already provides a comprehensive foundation for symbolic sequence generation and analysis and a solid set of controllable tasks to evaluate and compare cognitively-inspired computing, there are several extensions under active development. Although some tasks already employ hierarchical and recursive structures, central to natural language syntax, our ability to analyze, induce and infer supra-regular grammars is currently limited in scope. Pushing towards that direction will enable the development of more complex compositional reasoning tasks and, consequently, the evaluation of more naturalistic cognitive computations. This includes both CFG-based sequence generation with controllable complexity (via probabilistic CFGs and entropy-based constraints) and inference of CFG sub-classes from empirical data, building on recent advances in distributional learning [28] and Bayesian grammar induction.

Regarding the symbolic embeddings implemented in `SeqBench`, there is ample space for integration with other established datasets such as TIMIT (phoneme recognition), other spoken digit corpora, and other psycholinguistic paradigms (e.g., overlap, cohort, and TISK tasks probing lexical access and temporal integration). The software is modular and developed in a highly extensible manner that will facilitate the expansion and integration of these and future datasets. Additionally, current embedding decoupling already enables cross-modal experiments (e.g., visual-auditory sequence learning), but future versions will provide tighter integration with multimodal datasets and transformations, explicit support for alignment/synchronization of multi-stream sequences, and tools for quantifying cross-modal statistical dependencies and generalization.

Regarding metrics, ongoing work includes developing robust TE approximation methods for non-regular grammars (e.g., via probabilistic bounds or hierarchical decomposition), uncertainty quantification for TE estimates from inferred grammars, and alternative complexity measures tailored to specific grammar classes (e.g., nested depth for Dyck languages, crossing dependencies for mildly context-sensitive grammars). We also aim to expand upon the categorization of generative grammars into the Chomsky hierarchy, providing a detailed scheme to determine the likelihood of an observed (or generated) sequence being caused by grammars at different complexity levels.

To improve accessibility for non-programmers and facilitate exploratory analysis, we are developing web-based interfaces for grammar visualization (state transition diagrams, derivation trees), real-time metric computation and comparative visualization, and interactive parameter tuning for grammar generation and task configuration. As an open-source project, `SymSeqBench` thrives on community contributions. We actively encourage users to share custom grammars, task definitions, analysis metrics, and empirical datasets through a centralized repository, fostering reproducibility and accelerating methodological innovation across disciplines. Ongoing feedback from cognitive scientists, neuroscientists, and machine learning researchers will guide prioritization of feature development and ensure the tool remains responsive to evolving research needs.

In summary, `SymSeqBench` provides a set of standardized, cognitively grounded benchmarks for evaluating temporal processing in both biological and artificial neural networks, the tools to generate stimulus sets for experimental psychology, as well as metrics to evaluate structural complexity of any sequentially structured dataset. Bridging the gap between highly controlled synthetic tasks, naturalistic language processing and rigorous defining characteristics of human linguistic capacities, this tool aims to approximate artificial and natural intelligence. By addressing these issues while maintaining modularity and ease of use, `SymSeqBench` aims to establish itself as a standard resource for sequence processing research, bridging theoretical linguistics, cognitive neuroscience, and artificial intelligence through a shared computational and experimental framework.

5 Methods

5.1 Network architectures

Biological neural networks. We consider two biologically-plausible spiking network models of sequence learning proposed by Cone and Shouval [30] and Asabuki et al. [4], abbreviated as CS and AKF, respectively. Since the models are described in full detail in the original publications, here we only provide a brief description and focus more on modified parameters and task evaluation.

The CS model relies on a columnar architecture composed of Timer and Messenger cells (modelled as LIF neurons) to learn both transitions between sequence items and the duration of the individual elements, using a reward-modulated learning rule. The spiking version of the CS model was designed for deterministic sequences obeying the Markov property, while handling contextual dependencies requires memory-augmented architectures, which were demonstrated in continuous-rate networks but which are difficult to achieve in spiking networks. Here we used the NEST-based implementation [141] from Zajzon et al. [173] – which corrects an error in the original implementation –, retaining all parameter values with the exception of the inter-stimulus interval (ISI), which was set to 250 ms in all tasks.

The AKF model consists of a fully connected, non-Dalean (i.e., without separate excitatory and inhibitory units) network of stochastic spiking neurons, each composed of a dendritic and a somatic compartment. Through a combination of recurrent gating and unsupervised synaptic plasticity, this model can learn context-dependent segmentation of spike pattern sequences. For the implementation, we relied on the Python source code provided by the authors and kept all parameter values.

Model evaluation is based on the reservoir computing approach, sampling the population responses at the stimulus offset (not including the ISI) and training linear readouts with ridge regression on task-specific, one-hot encoded target labels (tokens). The state variable for both models is the instantaneous firing rate, and for CS we used only the rates of the Timer cells across all columns. As the imbalance in the token frequency can bias the accuracy and make it difficult to estimate a random baseline, we instead opted for the Cohen's kappa statistic [29] as a performance measure. This metric takes values between -1 and 1, 0 being chance level and values towards 1 representing good performance.

In the classification tasks in Figure 8a,b, the input strings are composed of the alphabet's (non-indexed) symbols (see main text for parameter values). For the n-step memory task, the target labels correspond to the identity of n-th preceding (non-indexed) symbol, with valid targets restricted to lie within each individual string (i.e., no targets referring to previous strings). In contrast, the context resolution task uses the indexed states as target labels. The effective training and testing data sizes are model-specific and chosen to yield the best possible performance. Specifically, each epoch consists of a set of randomly drawn grammatical strings such that the total number of tokens (approximately) corresponds to the model size M (number of neurons used for readout training), with $M_{\mathrm{CS}} = 500$ and $M_{\mathrm{AKF}} = 1200$. This ensures that one epoch contains sufficient data points to avoid overfitting of the readout. We set the total number of training epochs $\mathcal{E}_{\mathrm{train}} = 10$ and the test size to $\mathcal{E}_{\mathrm{test}} = 0.2\mathcal{E}_{\mathrm{train}}$.

For the counting task in Figure 8c, the indexed symbols (e.g., A_1 or X_3) denote distinct stimuli. For each string, there is a single target defined for the last filler element X_j and corresponding to the correct dependent item B_i. Here an epoch consists of single presentations of all unique strings allowed by the task parameters, and we set $\mathcal{E}_{\mathrm{train}}^{\mathrm{CS}} = 250$ and $\mathcal{E}_{\mathrm{train}}^{\mathrm{AKF}} = 500$, as well as $\mathcal{E}_{\mathrm{test}} = 100$ for both models.

Artificial and neuromorphic neural networks. We first consider two standard spiking neuron models for sequence processing, the leaky integrate-and-fire (LIF) model and its adaptive extension (adLIF), as defined in [12]. At timestep t, I_t denotes the input signal, s_t the binary spike, and θ the firing threshold. The discretized membrane potential U_t of the LIF neuron evolves as:

$$U_t = \alpha(U_{t-1} - \theta s_t) + (1 - \alpha)I_t, \tag{1}$$

$$s_t = \mathbf{1}(U_t \geq \theta), \tag{2}$$

where $\alpha < 1$ is the decay rate. The adLIF model includes an intrinsic adaptation variable w_t that influences the membrane potential by capturing recent neuronal activity, incorporating both spike-triggered and sub-threshold adaptation. Its discrete dynamics are given by:

$$U_t = \alpha(U_{t-1} - \theta s_{t-1}) + (1 - \alpha)(I_t - w_{t-1}), \tag{3}$$

$$w_t = \beta w_{t-1} + a\, U_{t-1} + b\, s_{t-1}, \tag{4}$$

$$s_t = \mathbf{1}(U_t \geq \theta), \tag{5}$$

where α and β are rate constants. The parameters β, α, a, and b are learnable and optimized during training. Neurons are stacked in N_{layers} feed-forward layers, each comprising N_{hidden} neurons. An output layer of leaky integrators with a learnable decay rate α produces the final outputs. In all experiments, we perform a grid search over the following hyperparameters: $N_{\text{layers}} \in \{2, 3, 4, 5\}$, $N_{\text{hidden}} \in \{128, 256, 512\}$, and $\theta \in [0.0, 1.5]$ with step 0.01. Based on the hyperparameter search, the largest number of layers consistently performed best; therefore, all reported experiments use an architecture with 5 layers and a hidden size of 256.

In addition, we assess three representative sequence processing artificial neural networks on more challenging sequences: gated recurrent units (GRU) [23], the transformer decoder [156], and the selective state space model Mamba [60]. GRUs process sequences by recurrently updating a hidden state as each new token is processed, utilizing input and state-dependent gates to control how much past information is retained or overwritten. Despite their efficiency and performance on small sequences, GRUs have shown limited capacity for learning long contexts and are relatively slow to train. The attention mechanism, at the core of the Transformer [156], improves long-sequence processing by replacing recurrence with a global memory of past token representations, which are retrieved through query, key, and value interactions. This enables direct modeling of long-range dependencies, but incurs quadratic time and memory complexity in sequence length. State space models such as Mamba [60] revisit recurrent modeling through a state space formulation that employs linear recurrences for efficient training. Careful initialization and state expansion increase effective memory capacity, resulting in strong long-sequence performance with linear-time complexity. The three different architectures are configured as follows: the Mamba model utilizes a state space expansion factor of 64, a local convolution width of 4, and a block expansion factor of 2. The GRU architecture is configured as a single-layer unidirectional recurrent network with hidden size matching the model dimension. The Transformer decoder employs 4 attention heads with rotary positional embeddings (RoPE base frequency of 10000), RMSNorm for layer normalization, and a SwiGLU feedforward network. All models use a dropout rate of 0.2 and LayerNorm for normalization between layers. The input sequences are first projected to the model through a linear encoder, and the final outputs are produced via a linear decoder layer.

Both the ANNs and neuromorphic models are benchmarked on classifying the indexed states underlying the input sequences (which corresponds to a next-token prediction task for ambiguous sequences, see main text for more details). For the ANN experiments, we use 200000 training samples and 10000 test samples, and train the models for 50 epochs. For the SNN experiments, we also use 200000 training samples but only 1000 test samples, and train for a single epoch due to the computational overhead of SNN training. During training, a cross-entropy loss is computed at every time step of each sample, including the gap duration, by predicting the indexed state associated with the current sequence position. Gradients are propagated through the full sequence using backpropagation through time. Model performance is evaluated by measuring classification accuracy exclusively during the gap period. For the optimization of the ANNs, we used the Adam optimizer, with an initial learning rate of 0.001, and the learning rate following a cosine annealing schedule with a warmup period corresponding to 5% of the total training steps. For the SNN experiments, we also used the Adam optimizer, with a fixed learning rate of 0.005 and no learning rate scheduler.

5.2 Metrics

To enable a comprehensive characterization of sequential data, either internally generated or externally acquired, `SymSeqBench` includes a comprehensive, hierarchical suite of complexity metrics organized across four levels of structural granularity: token-level, string-level, string-set level, and grammar-level metrics. This taxonomy provides a systematic framework for quantifying different aspects of sequence complexity, from local statistical properties to global generative capacity.

Token-level metrics characterize the statistical and distributional properties of individual symbols within sequences. Basic measures include **token frequency** ($f(\sigma_i, S) = \#\{\sigma_i \in S\}/|S|$), which quantifies symbol distribution uniformity, and **token duration statistics**, capturing temporal persistence through mean duration and variance across symbol occurrences. For sequences with embedded token representations, we compute **embedding complexity** metrics including vector norms, spectral radius of the embedding matrix, and pairwise embedding distances, which characterize the geometric structure of the symbolic input space.

String-level metrics quantify complexity properties of individual sequences or concatenated string-sets, capturing information content, compressibility, and temporal dependencies. **Shannon entropy** ($H(S) = -\sum P(\sigma_i) \log_2 P(\sigma_i)$) provides a foundational measure of uncertainty in symbol distributions [137]. We extend this by decomposing entropy into **block entropy** ($H_L(S)$), computed over L-grams, and the asymptotic **entropy rate** ($h_\mu(S) \approx H_L(S) - H_{L-1}(S)$), which captures the information content per symbol for long sequences. The **Effective Measure Complexity** (EMC; [58]) quantifies deviation from maximal randomness through $\text{EMC}(S) = \sum[H_L(S) - L \times h_\mu(S)]$, measuring structural regularity beyond simple entropy.

Algorithmic complexity is assessed through compression-based metrics. **Compressibility** using standard deflate/gzip compression ($C_{\text{gzip}}(S) = |C(S)|/|S|$) and **Lempel-Ziv-Welch complexity** [88, 167] quantify sequence redundancy and repetition structure. **Linguistic complexity** captures vocabulary richness through the product of normalized unique n-gram counts across scales: $C_{\text{ling}}(S) = \prod U_i(S)$, where $U_i(S) = |V_i(S)|/\min(|A|^i, |S| - i + 1)$ [154]. **Permutation entropy** [6] further characterizes ordinal pattern distributions, providing a non-parametric measure of temporal structure invariant to monotonic transformations.

String-set level metrics characterize relationships and variability across collections of sequences. Pairwise sequence distances include the **Hamming distance** ($d_H(S_i, S_j) = \sum \mathbb{1}[S_i[k] \neq S_j[k]]$) for equal-length strings and **edit distance** (Levenshtein distance), computed via dynamic programming as the minimum number of insertions, deletions, and substitutions required for transformation. The **Normalized Compression Distance** (NCD; [27]) provides a similarity metric based on algorithmic information theory: $\text{NCD}(S_1, S_2) = (C(S_1 \circ S_2) - \min(C(S_1), C(S_2)))/\max(C(S_1), C(S_2))$. More specifically tailored to psycholinguistic studies, **Associative chunk strength** (ACS; [82, 5]) quantifies the familiarity of n-gram subsequences based on their frequency in training data, with global and anchor-based variants capturing different aspects of sequential predictability. **Mutual information** between string pairs ($I(S_i; S_j) = H(S_i) + H(S_j) - H(S_i, S_j)$) and **string-set entropy** ($H(\mathcal{S}) = -\sum P(S) \log_2 P(S)$) characterize information-theoretic dependencies and diversity at the string-set level, with conditional variants capturing prefix-based structure.

Grammar-level metrics characterize the underlying generative mechanisms and computational capacity of sequence-producing systems. **Markov order estimation** employs information-theoretic model selection criteria (Akaike Information Criterion, Bayesian Information Criterion) to determine optimal dependency range, assuming Markovianity: $\hat{k} = \arg\min_k \text{BIC}_k$, where $\text{BIC}_k = -2\log L_k + p_k \log N$ [35, 81]. For sequences exhibiting variable-length dependencies, we employ **Variable-Length Markov Models** (VLMC; [18]) to adaptively prune context trees using statistical significance criteria.

Hierarchical dependency structure is revealed through mutual information decay profiles across element distances, distinguishing exponential ($\text{MI}(d) \sim a \cdot e^{-bd}$) from power-law ($\text{MI}(d) \sim a \cdot d^{-b}$) decay signatures indicative of different organizational principles [92, 130, 131].

Topological entropy (TE) quantifies the exponential growth rate of distinct sequences as a function of length, providing a canonical measure of generative capacity [128, 14]. For a grammar $\mathcal{G}$ with N unique states, let M denote the $N \times N$ transition matrix satisfying the Markov property. TE is given by $h(G) = \log_e(\lambda_1)$, where λ_1 is the spectral radius (largest real eigenvalue) of M. This *direct* eigenvalue approach enables fast, precise computation and relies on the indexed state representation to resolve symbol ambiguities [164]. Alternative *lifting techniques* [14] construct $N^l \times N^l$ matrices from l-grams but prove computationally impractical for complex grammars [164]. Accurate TE calculation requires the transition matrix to be irreducible (strongly connected), aperiodic, and free of absorbing states - conditions necessary for eigenvalue-based methods to yield global, non-trivial entropy values [164]. While typical artificial grammar specifications naturally satisfy irreducibility and aperiodicity, `SymSeq` ensures the absence of absorbing states by adding a transition loop from the special end-of-sequence terminal symbol (`#`) to all initial states, guaranteeing indefinite generation cycles. By the Perron-Frobenius theorem, the spectral radius is bounded from above by the maximum outdegree: $\text{TE}_{\max}(A_G) \leq \log(\max_{1 \leq i \leq n} \sum |A_{G_{ij}}|)$.

Additional grammar complexity measures include **production rule complexity** ($C_{\text{rules}}(G) = |R| + \sum |r|$, normalized by alphabet and non-terminal vocabulary sizes), **state complexity** (minimal deterministic finite automaton state count), and derivation tree statistics capturing average structural depth and branching [67]. For probabilistic grammars, **language entropy** ($H(L) = \lim_{n \to \infty}(1/n)H_n$, where $H_n = -\sum P(S) \log_2 P(S)$ over length-n strings) quantifies uncertainty in the generative distribution. **Minimum Description Length**

(MDL; [127, 8]) balances grammar compactness against data encoding efficiency: $MDL(G) = L(G) + L(D|G)$, providing a principled criterion for grammar induction and model selection.

This comprehensive metric suite enables fine-grained characterization of sequence complexity across multiple organizational scales, supporting rigorous comparison of datasets and systematic investigation of structure-learning relationships in cognitive and computational models.

Acknowledgements

Funding

References

[1] Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Arhitectures and algorithms. *arXiv (Cornell University)*, 01 2024. doi: 10.48550/arxiv.2401.12973. URL `https://arxiv.org/abs/2401.12973`.

[2] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[3] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2): 117–135, 1980. doi: 10.1016/S0019-9958(80)90285-5. URL `https://doi.org/10.1016/S0019-9958(80)90285-5`.

[4] Toshitake Asabuki, Prajakta Kokate, and Tomoki Fukai. Neural circuit mechanisms of hierarchical sequence learning tested on large-scale recording data. *PLOS Computational Biology*, 18(6):e1010214, June 2022. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010214. URL `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010214`. Publisher: Public Library of Science.

[5] Todd M. Bailey and Emmanuel M. Pothos. AGL StimSelect: Software for automated selection of stimuli for artificial grammar learning. *Behavior Research Methods*, 40(1):164–176, 2008. doi: 10.3758/BRM.40.1.164.

[6] Christoph Bandt and Bernd Pompe. Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, 88(17):174102, 2002. doi: 10.1103/PhysRevLett.88.174102.

[7] Maximilian Baronig, Romain Ferrand, Silvester Sabathiel, and Robert Legenstein. Advancing spatio-temporal processing through adaptation in spiking neural networks. *Nature Communications*, 16 (1), 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-60878-z. URL `http://dx.doi.org/10.1038/s41467-025-60878-z`.

[8] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998. doi: 10.1109/18.720554.

[9] Gabriël Beckers. pyagl. `https://github.com/gbeckers/pyagl`, 2020.

[10] Fabien C. Y. Benureau and Nicolas P. Rougier. Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions. *Frontiers in Neuroinformatics*, 11:69, 2017. ISSN 1662-5196. doi: 10.3389/fninf.2017.00069.

[11] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Inc., 2009.

[12] Alexandre Bittar and Philip N. Garner. A surrogate gradient spiking baseline for speech command recognition. *Frontiers in Neuroscience*, 16, August 2022. ISSN 1662-453X. doi: 10.3389/fnins.2022.865897. URL http://dx.doi.org/10.3389/fnins.2022.865897.

[13] Donald S. Blough. Delayed matching in the pigeon. *Journal of the Experimental Analysis of Behavior*, 2:151–160, 1959. ISSN 1938-3711. doi: 10.1901/jeab.1959.2-151. Place: US Publisher: Journal of the Experimental Analysis of Behavior.

[14] Erik M. Bollt and Michael A. Jones. The complexity of artificial grammars. *Nonlinear Dynamics, Psychology, and Life Sciences*, 4(2):153–168, 2000. doi: 10.1023/A:1009588513971.

[15] Younes Bouhadjar, Dirk J. Wouters, Markus Diesmann, and Tom Tetzlaff. Sequence learning, prediction, and replay in networks of spiking neurons. *PLOS Computational Biology*, 18(6):e1010233, June 2022. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010233. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010233. Publisher: Public Library of Science.

[16] Younes Bouhadjar, Dirk J. Wouters, Markus Diesmann, and Tom Tetzlaff. Coherent noise enables probabilistic sequence replay in spiking neuronal networks. *PLOS Computational Biology*, 19(5):e1010989, 2023. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010989. URL http://dx.doi.org/10.1371/journal.pcbi.1010989.

[17] Braden A. W. Brinkman, Han Yan, Arianna Maffei, Il Memming Park, Alfredo Fontanini, Jin Wang, and Giancarlo La Camera. Metastable dynamics of neural circuits and networks. doi: 10.1063/5.0062603.

[18] Peter Bühlmann and Abraham J. Wyner. Variable length Markov chains. *Annals of Statistics*, 27(2): 480–513, 1999. doi: 10.1214/aos/1018031204.

[19] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. A comparison of LSTM and GRU networks for learning symbolic sequences, July 2021. URL https://arxiv.org/abs/2107.02248v3.

[20] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005. doi: 10.1109/TIT.2005.850116. URL https://doi.org/10.1109/TIT.2005.850116.

[21] Xinyi Chen, Chenxiang Ma, Yujie Wu, Kay Chen Tan, and Jibin Wu. Neuromorphic sequential arena: A benchmark for neuromorphic temporal processing. In *International Joint Conference on Artificial Intelligence (IJCAI 2025)*, 2025.

[22] Zhiyi Chi, Wei Wu, Zach Haga, Nicholas G Hatsopoulos, and Daniel Margoliash. Template-Based Spike Pattern Identification With Linear Convolution and Dynamic Time Warping Template-Based Spike Pattern Identification With Linear Convolution and Dynamic Time Warping. 97(2):1221–1235. ISSN 0022-3077. doi: 10.1152/jn.00448.2006. URL http://www.ncbi.nlm.nih.gov/pubmed/17108096.

[23] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics, 2014. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012/.

[24] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 06 1959. ISSN 0019-9958, 1878-2981. doi: 10.1016/s0019-9958(59)90362-6. URL https://doi.org/10.1016/s0019-9958(59)90362-6.

[25] Noam Chomsky. *Language and Mind.* 01 1968. URL https://nptel.ac.in/courses/109101004/downloads/Lecture-31%20&%2032.pdf.

[26] Morten H. Christiansen and Nick Chater. The Now-or-Never bottleneck: A fundamental constraint on language. 39:1–72. ISSN 14691825. doi: 10.1017/S0140525X1500031X. URL http://journals.cambridge.org/abstract_S0140525X1500031X.

[27] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005. doi: 10.1109/TIT.2005.844059.

[28] Alexander Clark and Rémi Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007. URL http://www.jmlr.org/papers/volume8/clark07a/clark07a.pdf.

[29] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April 1960. ISSN 1552-3888. doi: 10.1177/001316446002000104. URL http://dx.doi.org/10.1177/001316446002000104.

[30] Ian Cone and Harel Z Shouval. Learning precise spatiotemporal sequences via biophysically realistic learning rules in a modular, spiking network. *eLife*, 10:e63751, March 2021. ISSN 2050-084X. doi: 10.7554/eLife.63751. URL https://doi.org/10.7554/eLife.63751. Publisher: eLife Sciences Publications, Ltd.

[31] C. Constantinidis and M. A. Steinmetz. Neuronal activity in posterior parietal area 7a during the delay periods of a spatial memory task. *Journal of Neurophysiology*, 76(2):1352–1355, August 1996. ISSN 0022-3077. doi: 10.1152/jn.1996.76.2.1352.

[32] Matthew T. Cook, Chrissy M. Chubala, and Randall K. Jamieson. AGSuite: Software to conduct feature analysis of artificial grammar learning performance. *Behavior Research Methods*, 49(5):1639–1651, 2017. ISSN 1554-3528. doi: 10.3758/s13428-017-0899-1. Place: Germany Publisher: Springer.

[33] Gil Costa. Zebrafish larvae top view. SciDraw, 2021. URL https://scidraw.io/drawing/375. Licensed under CC-BY 4.0.

[34] Benjamin Cramer et al. The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[35] Imre Csiszár and Paul C. Shields. The consistency of the BIC Markov order estimator. *Annals of Statistics*, 28(6):1601–1619, 2000. doi: 10.1214/aos/1015957472.

[36] Sandeep Robert Datta, David J. Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational Neuroethology: A Call to Action. 104(1):11–24. ISSN 08966273. doi: 10.1016/j.neuron.2019.09.038. URL https://linkinghub.elsevier.com/retrieve/pii/S0896627319308414.

[37] Colin de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997. doi: 10.1023/A:1007353007695. URL https://doi.org/10.1023/A:1007353007695.

[38] Meinou H de Vries, Karl Magnus Petersson, Sebastian Geukes, Pienie Zwitserlood, and Morten H Christiansen. Processing multiple non-adjacent dependencies: evidence from sequence learning. *Philosophical Transactions of the Royal Society B*, 367(1598):2065–2076, 2012. ISSN 1471-2970. doi: 10.1098/rstb.2011.0414. URL http://www.ncbi.nlm.nih.gov/pubmed/22688641.

[39] Michael Deistler, Kyra L. Kadhim, Matthijs Pals, Jonas Beck, Ziwei Huang, Manuel Gloeckler, Janne K. Lappalainen, Cornelius Schröder, Philipp Berens, Pedro J. Gonçalves, and Jakob H. Macke. Jaxley: differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics. *Nature Methods*, November 2025. ISSN 1548-7105. doi: 10.1038/s41592-025-02895-w. URL http://dx.doi.org/10.1038/s41592-025-02895-w.

[40] Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, et al. Neural networks and the Chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

[41] Joanne A. Deocampo, Tricia Z. King, and Christopher M. Conway. Concurrent Learning of Adjacent and Nonadjacent Dependencies in Visuo-Spatial and Visuo-Verbal Sequences. *Frontiers in Psychology*, 10, May 2019. ISSN 1664-1078. doi: 10.3389/fpsyg.2019.01107. URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2019.01107/full. Publisher: Frontiers.

[42] Renato Duarte, Peggy Seriès, and Abigail Morrison. Self-Organized Artificial Grammar Learning in Spiking Neural Networks. In P Bello, M Guarini, M McShane, and B Scassellati, editors, *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, pages 427–432. Cognitive Science Society, . URL https://escholarship.org/uc/item/75j1b2t0.

[43] Renato Duarte, Marvin Uhlmann, Dick Den Van Broek, Hartmut Fitz, Karl Magnus Petersson, and Abigail Morrison. Encoding symbolic sequences with spiking neural reservoirs. In *2018 International Joint Conference on Neural Networks (IJCNN)*, volume 2018-July, pages 1–8. IEEE, . ISBN 978-1-5090-6014-6. doi: 10.1109/IJCNN.2018.8489114. URL https://ieeexplore.ieee.org/document/8489114/.

[44] Renato C. F. Duarte and Abigail Morrison. Dynamic stability of sequential stimulus representations in adapting neuronal networks. 8:124. ISSN 1662-5188. doi: 10.3389/fncom.2014.00124. URL http://journal.frontiersin.org/article/10.3389/fncom.2014.00124/abstract.

[45] Daniel B. Ehrlich, Jasmine T. Stone, David Brandfonbrener, Alexander Atanasov, and John D. Murray. PsychRNN: An Accessible and Flexible Python Package for Training Recurrent Neural Network Models on Cognitive Tasks. *eNeuro*, 8(1), January 2021. ISSN 2373-2822. doi: 10.1523/ENEURO.0427-20.2020. URL https://www.eneuro.org/content/8/1/ENEURO.0427-20.2020. Publisher: Society for Neuroscience Section: Open Source Tools and Methods.

[46] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, September 2023. ISSN 1558-2256. doi: 10.1109/jproc.2023.3308088. URL http://dx.doi.org/10.1109/JPROC.2023.3308088.

[47] Maxime Fabre, Lyubov Dudchenko, and Emre Neftci. Structured state space model dynamics and parametrization for spiking neural networks. *arXiv preprint arXiv:2506.06374*, 2025.

[48] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40), October 2023. ISSN 2375-2548. doi: 10.1126/sciadv.adi1480. URL http://dx.doi.org/10.1126/sciadv.adi1480.

[49] W. Tecumseh Fitch. Toward a computational framework for cognitive biology: Unifying approaches from cognitive neuroscience and comparative cognition. 11(3):329–364. ISSN 18731457. doi: 10.1016/j.plrev.2014.04.005. URL http://dx.doi.org/10.1016/j.plrev.2014.04.005.

[50] W Tecumseh Fitch and Angela D Friederici. Artificial grammar learning meets formal language theory: An overview. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 367 (1598):1933–1955, 01 2012. ISSN 1471-2970. doi: 10.1098/rstb.2012.0103. URL https://doi.org/10.1098/rstb.2012.0103.

[51] Hartmut Fitz, Marvin Uhlmann, Dick Van Den Broek, Renato Duarte, Peter Hagoort, and Karl Magnus Petersson. Neuronal spike-rate adaptation supports working memory in language processing. 117(34):20881–20889. ISSN 10916490. doi: 10.1073/pnas.2000222117. URL https://www.pnas.org/content/117/34/20881.

[52] James Fodor. Line goes up? inherent limitations of benchmarks for evaluating large language models. *arXiv (Cornell University)*, 02 2025. doi: 10.48550/arxiv.2502.14318. URL http://arxiv.org/abs/2502.14318.

[53] Jordi Fonollosa, Emre Neftci, and Mikhail Rabinovich. Learning of Chunking Sequences in Cognition and Behavior. *PLOS Computational Biology*, 11(11):e1004592, November 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004592. URL https://dx.plos.org/10.1371/journal.pcbi.1004592.

[54] David J. Freedman and John A. Assad. Neuronal Mechanisms of Visual Categorization: An Abstract View on Decision Making. *Annual Review of Neuroscience*, 39(Volume 39, 2016):129–147, July 2016. ISSN 0147-006X, 1545-4126. doi: 10.1146/annurev-neuro-071714-033919. URL https://www.annualreviews.org/content/journals/10.1146/annurev-neuro-071714-033919. Publisher: Annual Reviews.

[55] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, November 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-00257-z. URL http://dx.doi.org/10.1038/s42256-020-00257-z.

[56] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007. doi: 10.4249/scholarpedia.1430.

[57] E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967. doi: 10.1016/S0019-9958(67)91165-5. URL https://doi.org/10.1016/S0019-9958(67)91165-5.

[58] Peter Grassberger. Toward a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics*, 25(9):907–938, 1986. doi: 10.1007/BF00668821.

[59] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[60] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=tEYskw1VY2`.

[61] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=tEYskw1VY2`.

[62] Maëlle Guyoton, Giulio Matteucci, Charlie G. Foucher, Matthew P. Getz, Julijana Gjorgjieva, and Sami El-Boustani. Cortical circuits for cross-modal generalization. *Nature Communications*, 16(1):4230, May 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-59342-9. URL `https://www.nature.com/articles/s41467-025-59342-9`. Publisher: Nature Publishing Group.

[63] Rebecca L. Gómez. Variability and detection of invariant structure. *Psychological Science*, 13(5):431–436, 2002. ISSN 1467-9280. doi: 10.1111/1467-9280.00476. Place: United Kingdom Publisher: Blackwell Publishing.

[64] Stephen José Hanson and Michiro Negishi. On the emergence of rules in neural networks. *Neural Computation*, 14(9):2245–2268, 09 2002. ISSN 0899-7667. doi: 10.1162/089976602320264079. URL `https://doi.org/10.1162/089976602320264079`.

[65] Karine Heerah, Sarah L Cox, Pierre Blevin, Christophe Guinet, and Jean-Benoît Charrassin. Validation of dive foraging indices using archived and transmitted acceleration data: The case of the Weddell seal. *Marine Ecology Progress Series*, 574:191–208, 2017. doi: 10.3354/meps12149. URL `https://doi.org/10.3354/meps12149`.

[66] José Hernández-Orallo. Ai evaluation: past, present and future. *arXiv (Cornell University)*, 01 2014. doi: 10.48550/arxiv.1408.6908. URL `https://arxiv.org/abs/1408.6908`.

[67] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson, 3 edition, 2006.

[68] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 2 edition, 2012.

[69] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=kIoBbc76Sy`.

[70] Mark J. Hurlstone, Graham J. Hitch, and Alan D. Baddeley. Memory for serial order across domains: An overview of the literature and directions for future research. *Psychological Bulletin*, 140(2):339–373, 2014. ISSN 1939-1455. doi: 10.1037/a0034221. Place: US Publisher: American Psychological Association.

[71] Jeff Hwang, Moto Hira, Caroline Chen, Xiaohui Zhang, Zhaoheng Ni, Guangzhi Sun, Pingchuan Ma, Ruizhe Huang, Vineel Pratap, Yuekai Zhang, Anurag Kumar, Chin-Yun Yu, Chuang Zhu, Chunxi Liu, Jacob Kahn, Mirco Ravanelli, Peng Sun, Shinji Watanabe, Yangyang Shi, Yumeng Tao, Robin Scheibler, Samuele Cornell, Sean Kim, and Stavros Petridis. Torchaudio 2.1: Advancing speech recognition, self-supervised learning, and audio processing components for pytorch, 2023.

[72] Takuya Ito, Soham Dan, Mattia Rigotti, James Kozloski, and Murray Campbell. On the generalization capacity of neural networks during generic multimodal reasoning, March 2024. URL `http://arxiv.org/abs/2401.15030`. arXiv:2401.15030 [cs].

[73] Ray Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford Univ. Press, 1. publ. in paperback, 2. impr edition. ISBN 978-0-19-926437-7.

[74] Gerhard Jäger and James Rogers. Formal language theory: Refining the chomsky hierarchy. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 367(1598):1956–1970, 01 2012. ISSN 1471-2970. doi: 10.1098/rstb.2012.0077. URL `https://doi.org/10.1098/rstb.2012.0077`.

[75] Lorène Jeantet, Fanny Dell'Amico, Marie-Anne Forin-Wiart, Matthieu Coutant, Maxime Bonola, David Etienne, Fabien Claro, Valeria Dos Reis, and Jordan Gresser. Combined use of two supervised learning algorithms to model sea turtle behaviours from tri-axial acceleration data. *Journal of Experimental Biology*, 223:jeb222513, 2020. doi: 10.1242/jeb.222513. URL `https://doi.org/10.1242/jeb.222513`.

[76] Jakob Jordan, Philipp Weidel, and Abigail Morrison. A closed-loop toolchain for neural network simulations of learning autonomous agents. *Frontiers in Computational Neuroscience*, 13, 08 2019. ISSN 1662-5188. doi: 10.3389/fncom.2019.00046. URL `https://doi.org/10.3389/fncom.2019.00046`.

[77] Ion Juvina and Niels A Taatgen. Modeling control strategies in the n-back task. In *Proceedings of the 8th international conference on cognitive modeling*, pages 73–78. Psychology Press New York, 2007.

[78] Hamidreza Kamkari, Brendan Leigh Ross, Rasa Hosseinzadeh, Jesse C. Cresswell, and Gabriel Loaiza-Ganem. A Geometric View of Data Complexity: Efficient Local Intrinsic Dimension Estimation with Diffusion Models, October 2024. URL `http://arxiv.org/abs/2406.03537`. arXiv:2406.03537 [cs].

[79] Greg Kamradt. Needle in a haystack-pressure testing llms. *URL https://github.com/gkamradt/LLMTest NeedleInAHaystack/tree/main*, page 28, 2023.

[80] Michael J. Kane, Andrew R. A. Conway, Timothy K. Miura, and Gregory J. H. Colflesh. Working memory, attention control, and the N-back task: a question of construct validity. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, 33(3):615–622, May 2007. ISSN 0278-7393. doi: 10.1037/0278-7393.33.3.615.

[81] Richard W. Katz. On some criteria for estimating the order of a Markov chain. *Technometrics*, 23(3): 243–249, 1981. doi: 10.1080/00401706.1981.10487671.

[82] Barbara J. Knowlton and Larry R. Squire. Artificial grammar learning depends on implicit acquisition of both abstract and exemplar-specific information. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(1):169–181, 1996. doi: 10.1037/0278-7393.22.1.169.

[83] Nikolaus Kriegeskorte and Pamela K. Douglas. Cognitive computational neuroscience. pages 1–31. ISSN 1097-6256. doi: 10.1038/s41593-018-0210-5. URL `http://www.nature.com/articles/s41593-018-0210-5`.

[84] Brenden M. Lake, Tomer Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *arXiv (Cornell University)*, 01 2016. doi: 10.48550/arXiv.1604.00289. URL `https://arxiv.org/abs/1604.00289`.

[85] K. S. Lashley. The problem of serial order in behavior. In *Cerebral mechanisms in behavior; the Hixon Symposium*, pages 112–146. Wiley, Oxford, England, 1951.

[86] Andreea Lazar. Sorn: a self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, 3, 2009. ISSN 1662-5188. doi: 10.3389/neuro.10.023.2009. URL `http://dx.doi.org/10.3389/neuro.10.023.2009`.

[87] Eric Lehman and Abhi Shelat. Approximation algorithms for grammar-based compression. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 205–212, 2002. URL `https://dl.acm.org/doi/10.5555/545381.545411`.

[88] Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, 22(1):75–81, 1976. doi: 10.1109/TIT.1976.1055501.

[89] Gregor Lenz, Kenneth Chaney, Sumit Bam Shrestha, Omar Oubari, Serge Picaud, and Guido Zarrella. Tonic: event-based datasets and transformations., 2021. URL `https://doi.org/10.5281/zenodo.5079802`.

[90] Johannes Leugering, Pascal Nieters, and Gordon Pipa. Dendritic plateau potentials can process spike sequences across multiple time-scales. *Frontiers in Cognition*, 2, February 2023. ISSN 2813-4532. doi: 10.3389/fcogn.2023.1044216. URL `https://www.frontiersin.org/journals/cognition/articles/10.3389/fcogn.2023.1044216/full`. Publisher: Frontiers.

[91] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models, 2023. URL `https://arxiv.org/abs/2211.09110`.

[92] Henry W. Lin and Max Tegmark. Critical behavior in physics and probabilistic formal languages. *Entropy*, 19(7):299, 2017. doi: 10.3390/e19070299.

[93] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding Motifs in Time Series. In *The 2nd Workshop on Temporal Data Mining*, .

[94] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11, 2003. doi: 10.1145/882082.882086. URL `https://doi.org/10.1145/882082.882086`.

[95] Sherry Lin, Winthrop F. Gillis, Caleb Weinreb, Ayman Zeine, Samuel C. Jones, Emma M. Robinson, Jeffrey Markowitz, and Sandeep Robert Datta. Characterizing the structure of mouse behavior using motion sequencing. 19(11):3242–3291, . ISSN 1750-2799. doi: 10.1038/s41596-024-01015-w. URL `https://www.nature.com/articles/s41596-024-01015-w`.

[96] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, November 2002. ISSN 1530-888X. doi: 10.1162/089976602760407955. URL `http://dx.doi.org/10.1162/089976602760407955`.

[97] Amadeus Maes, Mauricio Barahona, and Claudia Clopath. Learning compositional sequences with multiple time scales through a hierarchical network of spiking neurons. *PLOS Computational Biology*, 17(3):e1008866, March 2021. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1008866. URL `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008866`. Publisher: Public Library of Science.

[98] TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library, November 2016. URL `https://github.com/pytorch/vision`.

[99] Michiru Makuuchi, Jörg Bahlmann, and Angela D Friederici. An approach to separating the levels of hierarchical structure building in language and mathematics. 367(1598):2033–2045. ISSN 1471-2970. doi: 10.1098/rstb.2012.0095. URL `http://www.ncbi.nlm.nih.gov/pubmed/22688638`.

[100] Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, November 2013. ISSN 1476-4687. doi: 10.1038/nature12742. URL `http://dx.doi.org/10.1038/nature12742`.

[101] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning. 21(9):1281–1289. ISSN 1546-1726. doi: 10.1038/s41593-018-0209-y. URL `https://www.nature.com/articles/s41593-018-0209-y`.

[102] Thierry Meulemans and Martial Van der Linden. Implicit learning of complex information in amnesia. *Brain and Cognition*, 52(2):250–257, 2003. ISSN 1090-2147. doi: 10.1016/S0278-2626(03)00081-2. Place: Netherlands Publisher: Elsevier Science.

[103] E. K. Miller, C. A. Erickson, and R. Desimone. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 16(16):5154–5167, August 1996. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.16-16-05154.1996.

[104] Lucas Miranda, Joeri Bordes, Benno Pütz, Mathias V. Schmidt, and Bertram Müller-Myhsok. DeepOF: A Python package for supervised and unsupervised pattern recognition in mice motion tracking data. 8(86):5394. ISSN 2475-9066. doi: 10.21105/joss.05394. URL `https://joss.theoj.org/papers/10.21105/joss.05394`.

[105] Manuel Molano-Mazón, João Barbosa, Jordi Pastor-Ciurana, Marta Fradera, Ru-Yuan Zhang, Jérémy Forest, Jorge del Pozo Lérida, Jian Li, Christopher J. Cueva, Jaime de la Rocha, Devika Narain, and Guangyu Robert Yang. Neurogym: An open resource for developing and sharing neuroscience tasks. 02 2022. doi: 10.31234/osf.io/aqc9n. URL `https://doi.org/10.31234/osf.io/aqc9n`.

[106] James M Murray and G Sean Escola. Learning multiple variable-speed sequences in striatum via cortical tutoring. *eLife*, 6:e26084, May 2017. ISSN 2050-084X. doi: 10.7554/eLife.26084. URL `https://doi.org/10.7554/eLife.26084`. Publisher: eLife Sciences Publications, Ltd.

[107] Eric P. Nawrocki and Sean R. Eddy. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935, November 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt509. URL `https://doi.org/10.1093/bioinformatics/btt509`.

[108] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, Andrew Y Ng, et al. Multimodal deep learning. In *ICML*, volume 11, pages 689–696, 2011.

[109] David Nicholson, Jonah E Queen, and Samuel J Sober. Bengalese finch song repository. *bioRxiv*, 2017. doi: 10.1101/157537. URL https://doi.org/10.1101/157537.

[110] Zeynep Okray, Pedro F. Jacob, Ciara Stern, Kieran Desmond, Nils Otto, Clifford B. Talbot, Paola Vargas-Gutierrez, and Scott Waddell. Multisensory learning binds neurons into a cross-modal memory engram. *Nature*, 617(7962):777–784, May 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06013-8. URL https://www.nature.com/articles/s41586-023-06013-8. Publisher: Nature Publishing Group.

[111] Luca Onnis, Morten H Christiansen, Nick Chater, and Rebecca Gomez. Reduction of uncertainty in human sequential learning: Evidence from artificial grammar learning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, (25):886–891, 2003.

[112] Gerard O'Regan. *History of Artificial Intelligence*, pages 249–273. 01 2016. doi: 10.1007/978-3-319-33138-6_19. URL https://doi.org/10.1007/978-3-319-33138-6_19.

[113] Thomas J. Palmeri, Bradley C. Love, and Brandon M. Turner. Model-based cognitive neuroscience. 76:59–64. ISSN 00222496. doi: 10.1016/j.jmp.2016.10.010. URL https://linkinghub.elsevier.com/retrieve/pii/S002224961630116X.

[114] Roma Patel, Rafael Rodríguez-Sánchez, and George Konidaris. On the relationship between structure in natural language and models of sequential decision processes. 06 2020. URL https://openreview.net/pdf?id=-KDnP4X1-0_.

[115] Robin Pauli, Philipp Weidel, Susanne Kunkel, and Abigail Morrison. Reproducing Polychronization: A Guide to Maximizing the Reproducibility of Spiking Network Models. *Frontiers in Neuroinformatics*, 12, August 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00046. URL https://www.frontiersin.org/journals/neuroinformatics/articles/10.3389/fninf.2018.00046/full. Publisher: Frontiers.

[116] Christian Pehle and Jens Egholm Pedersen. Norse - A deep learning library for spiking neural networks, October 2021. URL https://github.com/norse/norse.

[117] Jonathan Peirce, Jeremy R. Gray, Sol Simpson, Michael MacAskill, Richard Höchenberger, Hiroyuki Sogo, Erik Kastman, and Jonas Kristoffer Lindeløv. Psychopy2: Experiments in behavior made easy. *Behavior Research Methods*, 51(1):195–203, February 2019. ISSN 1554-3528. doi: 10.3758/s13428-018-01193-y. URL http://dx.doi.org/10.3758/s13428-018-01193-y.

[118] Talmo D. Pereira, Diego E. Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S.-H. Wang, Mala Murthy, and Joshua W. Shaevitz. Fast animal pose estimation using deep neural networks. 16(1):117–125. ISSN 1548-7105. doi: 10.1038/s41592-018-0234-5. URL https://www.nature.com/articles/s41592-018-0234-5.

[119] Christopher I. Petkov and Carel ten Cate. Structured sequence learning: Animal abilities, cognitive operations, and language evolution, 07 2019. ISSN 1756-8757, 1756-8765. URL https://doi.org/10.1111/tops.12444.

[120] Steven T. Piantadosi. The computational origin of representation. *Minds and Machines*, 31:1–58, 11 2020. ISSN 0924-6495, 1572-8641. doi: 10.1007/s11023-020-09540-9. URL https://doi.org/10.1007/s11023-020-09540-9.

[121] Leonard Pitt and Manfred K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 421–432, 1989. doi: 10.1145/73007.73048. URL https://doi.org/10.1145/73007.73048.

[122] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning iii: explaining the non-overfitting puzzle, 2018. URL https://arxiv.org/abs/1801.00173.

[123] Alessio Quaresima, Hartmut Fitz, Renato Duarte, van den Broek Dick, Peter Hagoort, and Karl Magnus Petersson. The Tripod neuron: A minimal structural reduction of the dendritic tree. 601(15):3265–3295. ISSN 1469-7793. doi: 10.1113/JP283399. URL https://onlinelibrary.wiley.com/doi/abs/10.1113/JP283399.

[124] Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. Ai and the everything in the whole wide world benchmark, 2021. URL https://arxiv.org/abs/2111.15366.

[125] Dhanesh Ramachandram and Graham W. Taylor. Deep Multimodal Learning: A Survey on Recent Advances and Trends. *IEEE Signal Processing Magazine*, 34(6):96–108, November 2017. ISSN 1558-0792. doi: 10.1109/MSP.2017.2738401. URL https://ieeexplore.ieee.org/document/8103116.

[126] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist, 2020. URL https://arxiv.org/abs/2005.04118.

[127] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978. doi: 10.1016/0005-1098(78)90005-5.

[128] Clark Robinson. *Dynamical Systems: Stability, Symbolic Dynamics, and Chaos.* CRC Press, Boca Raton, FL, 2 edition, 1998.

[129] Ranulfo Romo, Carlos D. Brody, Adrián Hernández, and Luis Lemus. Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, 399(6735):470–473, June 1999. ISSN 1476-4687. doi: 10.1038/20939. URL https://www.nature.com/articles/20939. Publisher: Nature Publishing Group.

[130] Tim Sainburg, Marvin Thielk, and Timothy Q. Gentner. Parallels in the sequential organization of birdsong and human speech. *Nature Communications*, 10:3636, 2019. doi: 10.1038/s41467-019-11605-y.

[131] Tim Sainburg, Anna Mai, and Timothy Q. Gentner. Long-range sequential dependencies precede complex syntactic production in language acquisition. *Proceedings of the Royal Society B: Biological Sciences*, 289: 20212657, 2022. doi: 10.1098/rspb.2021.2657.

[132] Christian Scharinger, Alexander Soutschek, Torsten Schubert, and Peter Gerjets. Comparison of the Working Memory Load in N-Back and Working Memory Span Tasks by Means of EEG Frequency Band Power and P300 Amplitude. *Frontiers in Human Neuroscience*, 11, January 2017. ISSN 1662-5161. doi: 10.3389/fnhum.2017.00006. URL https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2017.00006/full. Publisher: Frontiers.

[133] Rachel Schiff and Pesia Katan. Does complexity matter? Meta-analysis of learner performance in artificial grammar tasks. *Frontiers in Psychology*, 5, 2014. ISSN 1664-1078. doi: 10.3389/fpsyg.2014.01084. URL https://www.frontiersin.org/articles/10.3389/fpsyg.2014.01084.

[134] D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's curse? on pace, progress, and empirical rigor, 2018. URL https://openreview.net/forum?id=rJWFOFywf.

[135] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. Grammarviz 3.0: Interactive discovery of variable-length time series patterns. *ACM Trans. Knowl. Discov. Data*, 12(1):10:1–10:28, February 2018. ISSN 1556-4681. doi: 10.1145/3051126. URL http://doi.acm.org/10.1145/3051126.

[136] Daniel Senkowski and Andreas K. Engel. Multi-timescale neural dynamics for multisensory integration. *Nature Reviews Neuroscience*, 25(9):625–642, September 2024. ISSN 1471-0048. doi: 10.1038/s41583-024-00845-7. URL https://www.nature.com/articles/s41583-024-00845-7. Publisher: Nature Publishing Group.

[137] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3): 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[138] Ankur Sikarwar and Mengmi Zhang. Decoding the enigma: Benchmarking humans and ais on the many facets of working memory. *arXiv (Cornell University)*, 01 2023. doi: 10.48550/arxiv.2307.10768. URL https://arxiv.org/abs/2307.10768.

[139] Max Silberztein. *Formalizing Natural Languages: The NooJ Approach.* Wiley-ISTE, 2016. ISBN 978-1-119-26412-5.

[140] Daniel Silver and Tom M. Mitchell. The roles of symbols in neural-based ai: They are not what you think! *arXiv (Cornell University)*, 01 2023. doi: 10.48550/arxiv.2304.13626. URL https://arxiv.org/abs/2304.13626.

[141] Ankur Sinha, Robin de Schepper, Jari Pronold, Jessica Mitchell, Håkon Mørk, Pooja Nagendra Babu, Jochen Martin Eppler, Melissa Lober, Charl Linssen, Dennis Terhorst, Mohamed Ayssar Benelhedi, Abigail Morrison, Willem Wybo, Guido Trensch, Rajalekshmi Deepu, Nicolai Haug, Anno Kurth, Stine Brekke Vennemo, Steffen Graber, Sebastian Spreizer, Johannes Gille, Jan Vogelsang, Marcel Krüger, and Hans Ekkehard Plesser. Nest 3.4, 2023. URL https://zenodo.org/record/6867800.

[142] Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences*, 102(33):11629–11634, August 2005. doi: 10.1073/pnas.0409746102. URL https://www.pnas.org/doi/10.1073/pnas.0409746102. Publisher: Proceedings of the National Academy of Sciences.

[143] H. Francis Song, Guangyu R. Yang, and Xiao-Jing Wang. Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework. *PLOS Computational Biology*, 12(2): e1004792, February 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004792. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004792. Publisher: Public Library of Science.

[144] Nova Spivack, S. Douglas, Michelle Crames, and Tim Connors. Cognition is all you need – the next layer of ai above large language models. *arXiv (Cornell University)*, 03 2024. doi: 10.48550/arxiv.2403.02164. URL http://arxiv.org/abs/2403.02164.

[145] Vivek Hari Sridhar, Dominique G Roche, and Simon Gingins. Gradient sensing and decision-making in freely navigating zebrafish. *eLife*, 13:RP88289, 2024. doi: 10.7554/eLife.88289. URL https://doi.org/10.7554/eLife.88289.

[146] Aarohi Srivastava, Abhinav Rastogi, Abhishek S. Rao, Abu Awal Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew K. Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Lin, Blake Stephen Howald, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cèsar Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian C. Voigt, Christopher D. Manning, Christopher Potts, C. Héctor Ramírez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Gârbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, D. Dohan, David Drakard, and David Jurgens. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv (Cornell University)*, 01 2022. doi: 10.48550/arXiv.2206.04615. URL https://arxiv.org/abs/2206.04615.

[147] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski,

Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023. URL https://arxiv.org/abs/2206.04615.

[148] Marcel Stimberg, Romain Brette, and Dan FM Goodman. Brian 2, an intuitive and efficient neural simulator. *eLife*, 8, August 2019. ISSN 2050-084X. doi: 10.7554/elife.47314. URL http://dx.doi.org/10.7554/eLife.47314.

[149] Gijsbert Stoet. PsyToolkit: A software package for programming psychological experiments using Linux. *Behavior Research Methods*, 42(4):1096–1104, 2010. doi: 10.3758/BRM.42.4.1096. URL https://doi.org/10.3758/BRM.42.4.1096.

[150] Gijsbert Stoet. PsyToolkit: A novel web-based method for running online questionnaires and reaction-time experiments. *Teaching of Psychology*, 44(1):24–31, 2017. doi: 10.1177/0098628316677643. URL https://doi.org/10.1177/0098628316677643.

[151] Oliver Sturman, Lukas von Ziegler, Christa Schläppi, Furkan Akyol, Mattia Privitera, Daria Slominski, Christina Grimm, Laetitia Thieren, Valerio Zerbi, Benjamin Grewe, and Johannes Bohacek. Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. *Neuron*, 107(1):1–14, 2020. doi: 10.1016/j.neuron.2020.04.020. URL https://doi.org/10.1016/j.neuron.2020.04.020.

[152] Laura E. Suárez, Agoston Mihalik, Filip Milisav, Kenji Marshall, Mingze Li, Petra E. Vértes, Guillaume Lajoie, and Bratislav Misic. Connectome-based reservoir computing with the conn2res toolbox. *Nature Communications*, 15(1):656, January 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-44900-4. URL https://www.nature.com/articles/s41467-024-44900-4. Publisher: Nature Publishing Group.

[153] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

[154] Edward N. Trifonov. Making sense of the human genome. In *Structure and Methods*, pages 69–77. Adenine Press, 1990.

[155] Ethan Tyler and Lex Kravitz. Mouse drinking. SciDraw, 2019. URL https://scidraw.io/drawing/94. Licensed under CC-BY 4.0.

[156] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. URL http://arxiv.org/abs/1706.03762.

[157] Alex Vicente-Sola, Davide L Manna, Paul Kirkland, Gaetano Di Caterina, and Trevor J Bihl. Spiking neural networks for event-based action recognition: A new task to understand their advantage. *Neurocomputing*, 611:128657, 2025.

[158] John R. Vokey and Lee R. Brooks. Salience of item knowledge in learning artificial grammars. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(2):328–344, 1992. ISSN 1939-1285. doi: 10.1037/0278-7393.18.2.328. Place: US Publisher: American Psychological Association.

[159] Lukas von Ziegler, Oliver Sturman, and Johannes Bohacek. Big behavior: Challenges and opportunities in a new era of deep behavior profiling. 46(1):33–44. ISSN 1740-634X. doi: 10.1038/s41386-020-0751-7. URL https://www.nature.com/articles/s41386-020-0751-7.

[160] Jonathan D. Wallis, Kathleen C. Anderson, and Earl K. Miller. Single neurons in prefrontal cortex encode abstract rules. *Nature*, 411(6840):953–956, June 2001. ISSN 1476-4687. doi: 10.1038/35082081. URL https://www.nature.com/articles/35082081. Publisher: Nature Publishing Group.

[161] Ben Walters, Yeshwanth Bethi, Taylor Kergan, Binh Nguyen, Amirali Amirsoleimani, Jason K Eshraghian, Saeed Afshar, and Mostafa Rahimi Azghadi. Neuromorse: a temporally structured dataset for neuromorphic computing. *Neuromorphic Computing and Engineering*, 5(2):027001, 2025. ISSN 2634-4386. doi: 10.1088/2634-4386/add36c. URL http://dx.doi.org/10.1088/2634-4386/add36c.

[162] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020. URL https://arxiv.org/abs/1905.00537.

[163] Felix Hao Wang and Toben H. Mintz. Learning nonadjacent dependencies embedded in sentences of an artificial language: When learning breaks down. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, 44(4):604–614, April 2018. ISSN 1939-1285. doi: 10.1037/xlm0000483.

[164] Richard Warren and Peter J. Schroeder. Topological entropy measure of artificial grammar complexity for use in designing experiments on human performance in intelligence, surveillance, and reconnaissance (ISR) tasks. Technical Report ADA626837, Defense Technical Information Center, 2015.

[165] Philipp Weidel, Mikael Djurfeldt, Renato Duarte, and Abigail Morrison. Closed loop interactions between spiking neural network and robotic simulators based on music and ros. *Frontiers in Neuroinformatics*, 10, 08 2016. ISSN 1662-5196. doi: 10.3389/fninf.2016.00031. URL https://doi.org/10.3389/fninf.2016.00031.

[166] Caleb Weinreb, Jonah E. Pearl, Sherry Lin, Mohammed Abdal Monium Osman, Libby Zhang, Sidharth Annapragada, Eli Conlin, Red Hoffmann, Sofia Makowska, Winthrop F. Gillis, Maya Jay, Shaokai Ye, Alexander Mathis, Mackenzie W. Mathis, Talmo Pereira, Scott W. Linderman, and Sandeep Robert Datta. Keypoint-MoSeq: Parsing behavior by linking point tracking to pose dynamics. 21(7):1329–1339. ISSN 1548-7105. doi: 10.1038/s41592-024-02318-2. URL https://www.nature.com/articles/s41592-024-02318-2.

[167] Terry A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984. doi: 10.1109/MC.1984.1659158.

[168] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks, 2015. URL https://arxiv.org/abs/1410.3916.

[169] Benjamin Wilson, Michelle Spierings, Andrea Ravignani, Jutta L. Mueller, Toben H. Mintz, Frank Wijnen, Anne van der Kant, Kenny Smith, and Arnaud Rey. Non-adjacent Dependency Learning in Humans and Other Animals. *Topics in Cognitive Science*, 12(3):843–858, 2020. ISSN 1756-8765. doi: 10.1111/tops.12381. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12381. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/tops.12381.

[170] Shaokai Ye, Anastasiia Filippova, Jessy Lauer, Steffen Schneider, Maxime Vidal, Tian Qiu, Alexander Mathis, and Mackenzie Weygandt Mathis. SuperAnimal pretrained pose estimation models for behavioral analysis. 15(1):5165. ISSN 2041-1723. doi: 10.1038/s41467-024-48792-2. URL https://www.nature.com/articles/s41467-024-48792-2.

[171] Jason Yik, Korneel Van den Berghe, Douwe den Blanken, Younes Bouhadjar, Maxime Fabre, Paul Hueber, Weijie Ke, Mina A. Khoei, Denis Kleyko, Noah Pacik-Nelson, Alessandro Pierro, Philipp Stratmann, Pao-Sheng Vincent Sun, Guangzhi Tang, Shenqi Wang, Biyan Zhou, Soikat Hasan Ahmed, George Vathakkattil Joseph, Benedetto Leto, Aurora Micheli, Anurag Kumar Mishra, Gregor Lenz, Tao Sun, Zergham Ahmed, Mahmoud Akl, Brian Anderson, Andreas G. Andreou, Chiara Bartolozzi, Arindam Basu, Petrut Bogdan, Sander Bohte, Sonia Buckley, Gert Cauwenberghs, Elisabetta Chicca, Federico Corradi, Guido de Croon, Andreea Danielescu, Anurag Daram, Mike Davies, Yigit Demirag, Jason Eshraghian, Tobias Fischer, Jeremy Forest, Vittorio Fra, Steve Furber, P. Michael Furlong, William Gilpin, Aditya Gilra, Hector A. Gonzalez, Giacomo Indiveri, Siddharth Joshi, Vedant Karia, Lyes Khacef, James C. Knight, Laura Kriener, Rajkumar Kubendran, Dhireesha Kudithipudi, Shih-Chii Liu, Yao-Hong Liu, Haoyuan Ma, Rajit Manohar, Josep Maria Margarit-Taulé, Christian Mayr, Konstantinos Michmizos, Dylan R. Muir, Emre Neftci, Thomas Nowotny, Fabrizio Ottati, Ayca Ozcelikkale, Priyadarshini Panda, Jongkil Park, Melika Payvand, Christian Pehle, Mihai A. Petrovici, Christoph Posch, Alpha Renner, Yulia Sandamirskaya, Clemens J. S. Schaefer, André van Schaik, Johannes Schemmel, Samuel Schmidgall, Catherine Schuman, Jae-sun Seo, Sadique Sheik, Sumit Bam Shrestha, Manolis Sifalakis, Amos Sironi, Kenneth Stewart, Matthew Stewart, Terrence C. Stewart, Jonathan Timcheck, Nergis Tömen, Gianvito Urgese, Marian Verhelst, Craig M. Vineyard, Bernhard Vogginger, Amirreza Yousefzadeh, Fatima Tuz Zohora, Charlotte Frenkel, and Vijay Janapa Reddi. The neurobench framework for benchmarking neuromorphic computing algorithms and systems. *Nature Communications*, 16(1), February 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-56739-4. URL http://dx.doi.org/10.1038/s41467-025-56739-4.

[172] Barna Zajzon, Renato Duarte, Abigail Morrison, Renato Duartel, and Abigail Morrison. Transferring State Representations in Hierarchical Spiking Neural Networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2018-July, pages 1–9. IEEE. ISBN 978-1-5090-6014-6. doi: 10.1109/IJCNN.2018.8489135. URL https://ieeexplore.ieee.org/document/8489135/.

[173] Barna Zajzon, Renato Duarte, and Abigail Morrison. Toward reproducible models of sequence learning: replication and analysis of a modular spiking network with reward-based learning. *Frontiers in Integrative Neuroscience*, 17, June 2023. ISSN 1662-5145. doi: 10.3389/fnint.2023.935177. URL http://dx.doi.org/10.3389/fnint.2023.935177.

[174] Qi Zhao, Zheng Zhao, Xiaoya Fan, Zhengwei Yuan, Qian Mao, and Yudong Yao. Review of machine learning methods for rna secondary structure prediction. *PLOS Computational Biology*, 17(8):1–22, 08 2021. doi: 10.1371/journal.pcbi.1009291. URL https://doi.org/10.1371/journal.pcbi.1009291.

[175] Shanglin Zhou, Michael Seay, Jiannis Taxidis, Peyman Golshani, and Dean V. Buonomano. Multiplexing working memory and time in the trajectories of neural networks. *Nature Human Behaviour*, 7(7):1170–1184, July 2023. ISSN 2397-3374. doi: 10.1038/s41562-023-01592-y. URL https://www.nature.com/articles/s41562-023-01592-y. Publisher: Nature Publishing Group.

[176] Jiří Šíma. Analog neuron hierarchy. 128:199–215. ISSN 18792782. doi: 10.1016/j.neunet.2020.05.006. URL https://doi.org/10.1016/j.neunet.2020.05.006.