
On the Self-Supervised Learning of Perceptual Concepts via Interpreting

Omid Madani

OMIDMADANI@YAHOO.COM

Abstract

We advance Prediction Games (PGs) as a framework for learning structured perceptual concepts in a self-supervised cumulative fashion, under the unifying objective of improving one's predictions. PGs enjoy a symmetric property: *(perceptual) concepts function as both the predictors and the predictands* in the system, in addition to serving as the building blocks for making higher-level concepts. The learner begins with primitive concepts, corresponding to a finite set $\mathcal{A}$ of symbols, and learns by updating prediction connections among its concepts as well as expanding the concept vocabulary over time, via combining existing concepts (generating new concepts corresponding to n-grams in $\mathcal{A}^+$), via the *repeated practice of interpreting*. Interpretation is finding a good structuring of one's input, in an episode, in terms of one's current network of concepts. We develop and simplify our existing information-theoretic techniques, and break the perception problem into two subproblems of estimation (enriching and expanding the network of concepts) and decision making (how best to interpret an episode using the network). We present a formalization, and establish several simplifying properties of the optimal solution for the case of unigrams. The problem formulation and our analyses informs solution strategies, and we report on our preliminary experiments showing promising results on finding good n-grams and splits in challenging scenarios.

"... any visual input is consistent with an unlimited number of interpretations. The challenging question is how the perceptual system functions such that we normally are unaware of any ambiguity... our conceptual systems comes prepared with expectations.. These sorts of expectations or "constraints" occur in all facets of cognition." (preface to 'Cognitive Psychology')(Medin et al., 2000)

1. Introduction

The intelligence of humans appears to rest on entities, or representations, called **concepts** (Murphy, 2002; Lakoff & Johnson, 1999; Hofstadter & Sander, 2013; Gobet et al., 2001; Langley, 2022), and in particular *many concepts*, such as 'mother', 'water', 'table', 'democracy', 'life', 'mind', 'mad', $\dots$, that work well enough together, for instance to achieve daily common sensical behavior. Our research begins with the conjecture that concepts are themselves structured, and people use (internal) concepts, to structure and thus make sense of their world. Our work, which we refer to as **Prediction Games** (PGs) (Madani, 2025, 2007), explores the challenges of how one acquires this complexity by tackling the learning and use of more concrete structures, which may be referred to as **perceptual concepts**: such concepts correspond to patterns, for instance some *sufficiently expressive class of probabilistic automata*, that recur in one's sensory stream. We expect that most

such concepts are acquired and explicitly maintained and updated over long periods. In particular, we imagine that, in a perceptual task, for instance when one views an image, or hears an audio stream (in some time interval), a relatively small number of the concepts in one’s mind, from the many available, are activated (a ‘perceptual framing’). Two main problems arise and in PGs we address them together:

1. (perceptual interpretation: a snapshot-in-time task) Given sensory input, for instance when looking at a picture or a scene at time t , how does one, quickly and adequately, figure out (mostly unconsciously) which of one’s concepts (patterns), from a large set $\mathcal{V}$, are useful (*i.e.* adequately meet one’s needs at time t)?
2. (historical/developmental question) Where do these many concepts (in the thousands, millions, ...), richly interdependent, come from, in the first place?

Our overarching motivation, for joining the two problems, can be summarized as:

*“If you are going to repeatedly structure your world (input stream), all your life, then why not **practice doing it**.. all your life!”*

We hypothesize that interpreting, if formulated well, can be an effective guide for acquiring and using concepts (providing appropriate constraints for learning). Our research explores what the process of structuring, or interpretation, can mean, and how “practicing” can be computationally realized, *i.e.* how to update and learn from such structuring attempts, repeatedly over a lifetime. We seek unsupervised continual learning of many concepts by a system that is embedded in its environment. Many challenges remain, for instance developing an understanding of the conditions under which a system’s early errors do not impede its overall progress, in discovering concepts and in detecting good concept boundaries (splits) when segmenting one’s input. Note: although our investigation is currently focused on one dimensional sequences (*e.g.* text), our intention is to extend PGs to diverse modalities. We have made progress on the task of acquiring and using n-grams, but our goal is to expand the learnable class, for instance to include disjunctions in pattern structure. Such interpretable structures should ultimately be useful for down-stream agential tasks such as for learning one’s action effects (Madani et al., 2026), the determination of the variables useful in a reasoning episode, and for language development. For further context and motivation, and conceptual comparisons (*e.g.* to existing connectionist techniques), please see Madani (2025).

We begin with an overview of PGs (concepts, edges, interpretation) in the next section. We have simplified the information-theoretic score, CORE, that guides the interpretation process (Madani, 2023), and we motivate and present a formalization of the problem of computing edge weights and interpreting with them, which we refer to as **Stop-N-Go** (cost minimization). To obtain insights into the implications of the Stop-N-Go formulation, we establish several of its properties in Sect. 3. Throughout, we present examples and motivate our choices. The subsequent three sections report briefly on our ongoing experiments, discuss related work, and conclude.

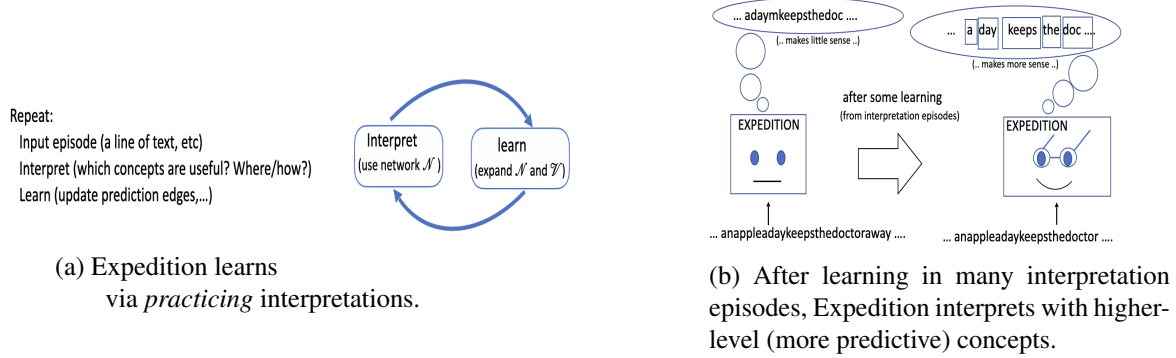


Figure 1: (a) The basic loop of Expedition: repeatedly input episode $\rightarrow$ interpret $\rightarrow$ learn (from the interpretation) (“practice interpretation”). Learning currently consists of updating prediction edges and, from time to time, adding new concepts (compositions). (b) After many interpreting+learning episodes, Expedition interprets (predicts, etc) with higher-level concepts (with increased information-utility): the same input can evoke different, highest level, concepts in the future.

2. An Overview of Prediction Games (PGs)

Our system, which we have called Expedition (Madani, 2021), works on text and its concepts correspond to n-grams of characters currently (Fig. 1). At any time point t , the system has a vocabulary of concepts $\mathcal{V}^{(t)}$ residing in a dynamic hierarchical network: $\mathcal{N}^{(t)}$: $\mathcal{V}$ grows, and edges are added and dropped from $\mathcal{N}$ over time. Initially, $\mathcal{V}$ is limited to the set of *primitive* concepts, corresponding to an alphabet (a discrete finite set),¹ $\mathcal{A}$, and initially $\mathcal{N}$ can be empty (or $\mathcal{V}^{(0)} \approx \mathcal{A}$ and $\mathcal{N}^{(0)} = \{\}$ at time $t = 0$). We assume there are computational pipelines making adequate distinctions and groupings (eg vector quantization), that convert the raw sensory impressions into this alphabet (such as to a few colors, or black and white pixels, for visual data). In text,² $\mathcal{A}$ can be the set of ASCII characters, up to 100 unique characters in our experiments.³

Expedition learns from repeatedly interpreting: inputting a text string $s \in \mathcal{A}^*$, such as a line (the contents of its input buffer), randomly picked from a text source (e.g. a large text file), interpreting it, and learning from this process (Fig. 1): $\mathcal{N}$ and $\mathcal{V}$ grow over time. For example, in our experiments, over say 100s of thousands of episodes, $\mathcal{V}$ grows from ≈ 100 to 10s of thousands of concepts.

Fig. 2 shows the composition relation between primitives, concepts, and interpretations. Composing, over the short and long term, is inline with views of human learning, thinking, and brain structure (Fosnot, 2005; Langley, 2022; Kruger et al., 2013). We next give an overview of the

1. A string is a finite, possibly empty, sequence of symbols from alphabet $\mathcal{A}$, and $\mathcal{A}^*$ denotes the Kleene closure of $\mathcal{A}$ (the infinite set of all strings on $\mathcal{A}$), e.g. with $\mathcal{A} = \{\mathbf{a}, \mathbf{b}\}$, $\mathcal{A}^* = \{\epsilon, \mathbf{a}, \mathbf{b}, \mathbf{ab}, \mathbf{bb}, \mathbf{ba}, \dots\}$ and $\mathcal{A}^+ = \mathcal{A}^* - \{\epsilon\} = \{\mathbf{a}, \mathbf{b}, \mathbf{ab}, \dots\}$ (Hopcroft & Ullman, 1979).
2. Our examples in this section assume natural language English text.
3. We also experiment with lower level primitives, to test the accuracy of the boundaries (splits) of detected concepts in episodes (Sect. 4). We seek techniques that are robust to the choice of initial alphabet $\mathcal{A}$ (given that the higher-level patterns are expressible in $\mathcal{A}$ and adequate learning time is provided).

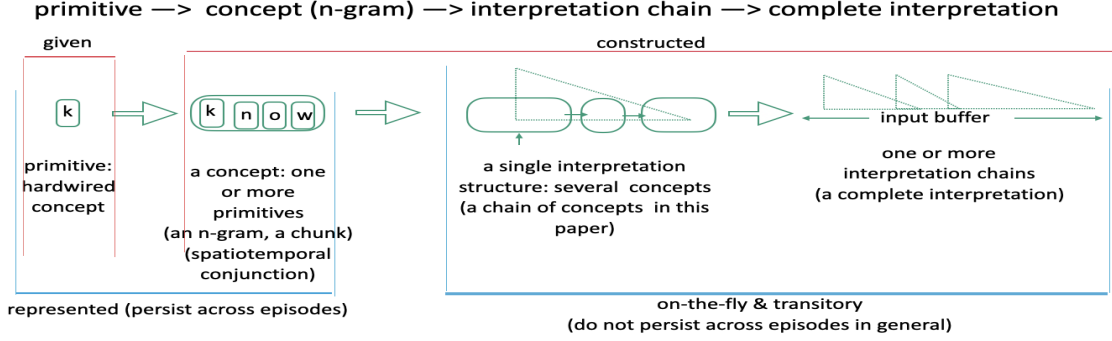


Figure 2: From primitives to (complete) interpretations (in terms of composition, persistence, etc). An episode, a string, may require more than one interpretation chain to be covered, *i.e.* to yield a complete interpretation. For instance, this is the case initially, when primitives have no edges.

main parts (concepts, edges, and interpretation), while Sect. 3 develops a concrete version of the interpretation problem. Table 1 in Appendix A provides a summary of terminology and notation.

2.1 Concepts (Nodes)

A concept in our current system corresponds to an n-gram,⁴ which is its *canonical* (or exact-match) form. For example, the concept (unigram) **n** corresponds to 'n', meaning that it best matches the letter 'n' appearing in the input buffer. The concept (the n-gram) **cat** corresponds to **c** (immediately) followed by **a** followed by **t** in a left-to-right (**L2R**) order. However, with approximate match, a concept can match (equivalently generate) any string in $\mathcal{A}^*$, and one can view a concept as a probabilistic finite-state machine (Dupont et al., 2005): it generates its canonical form with highest probability, but it can also generate (or approximately match) any other string with some (much smaller positive) probability (Madani, 2023). Therefore, each concept is an infinite probability distribution, over $\mathcal{A}^*$ (under approximate match). Matching is 'top-down' probabilistic prediction. Appendix B.1 explains our choice of the term 'concept' to refer to such n-grams (as opposed to alternatives such as feature or chunk).

2.2 Edges

A concept is also a node in a network $\mathcal{N}$ of concepts: it has out edges to concepts (directed outgoing prediction edges), as well as in-edges (incoming edges) from concepts (nodes), with self-arcs possible. The edges are used during interpretation (and also, and ultimately, for plain prediction). There are two main edge types, horizontal and vertical. A horizontal edge can be either L2R or R2L (right-to-left). A vertical edge connects a concept to a higher-level concept that includes it (*e.g.* **c** can connect to **ca**), supporting bottom-up activations (predictions). Every edge has (match) offset

4. More generally, a spatiotemporal conjunction. We also hope to extend the techniques for incorporating disjunctions into concept structure (Madani, 2025).

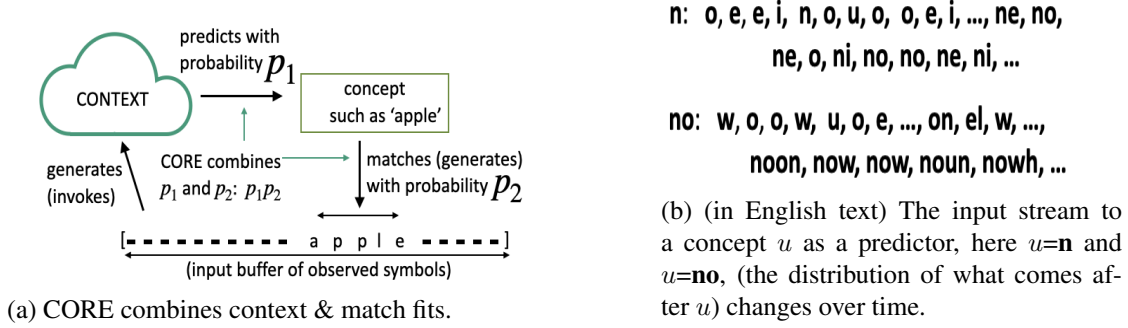


Figure 3: (a) The CORE score is a way of combining fit to context, or COherence, with fit to REality (matching the observations in the input buffer). It is a gain in information, and, in general, it promotes learning larger n-grams. (b) A concept is a predictor. Its input stream of what comes next to it (or what concept to predict) changes over time (internal and external non-stationarity). For the unigram **n**, at first (for 100s of episodes in which it occurs), only unigram concepts (mostly vowels) are observed. Later, as some horizontal edges become vertical, its bigrams are observed.

information encoded in it: once **c** is matched (to 'c'), and the vertical edge to **ca** is activated, **ca** need only match starting at its offset 1 (**c** at offset 0 is already matched, and **a** needs to be matched).⁵

Associated with each edge e is also a weight p , or $p(e)$, which can be roughly understood to be a (conditional) probability. Edge weights are updated in a manner that supports change, *i.e.* non-stationarity in the probabilities, and open-endedness in particular, *i.e.* connecting to new concepts (Madani, 2026). See Fig. 3(b). For a horizontal L2R edge $e : u \rightarrow v$, $p(e)$ is roughly the probability that concept v immediately occurs on the right of u in a final selected interpretation (see Sect. 3 for how the edge weights are determined according to an objective). Similarly, for a vertical edge. We think of outgoing edges from concept u as edges of u . In our online learning work, a concept at first does not have any edge, and edges are added and updated over time via learning. Initially, *e.g.* for a new concept, horizontal edges are acquired, and, later, some of these horizontal edges may be replaced with (converted to) vertical edges, to support using higher level concepts.

2.2.1 Why Generate New Concepts (Longer N-Grams)?

The concept **new** (with canonical form “new”) in a corpus of English text is a more powerful predictor of what occurs around it, at the word and phrasal level, than any of its individual parts (concepts **n** or **e** or **ne**, etc). Simply adding (linear summation of) the individual parts’ predictions does not achieve the accuracy of using the whole (or the popular adage: “The whole is more than the sum of its parts.”). By combining lower-level concepts to create new concepts and by learning the predictions of higher level ones, the system achieves non-linearity, which often leads to more accurate predictions. The CORE score, Fig. 3(a), in general improves by combining two part concepts that frequently co-occur (Appendix B.2).

5. For horizontal edges, the offset is often 0, unless if overlaps are allowed when interpreting.

2.2.2 Why Concepts as both Predictands and Predictors?

In PGs concepts also function as predictands (targets of prediction): concepts have in-edges. Why should concepts also be predictands in addition to predictors? Once the concept **new** is created, we want to know when it (likely) occurs in the input episodes, in order to use its predictions. However, plain matching of characters is insufficient in general. Context matters. For instance, in an episode the n-gram “new” could be part of the larger word “knew”: in this case, the “new”, appearing in the input buffer, (likely) does not have its meaning of newness (and concept **new**, if activated, should use/activate its vertical edge to **knew**). Conversely, there can be noise and typos (missing or mistyped letters), such as “nev”, and here we want **new** to be activated. Thus, concepts need to be predicted as well, by the context. CORE combines both how well a concept matches its region in the buffer (match to ‘reality’) and how well it *coheres* with (is well-predicted by) other activated concepts (Madani, 2023). But this situation is a bit of a chicken-and-egg problem, as concepts of an episode *together* comprise the (full) context. We describe the interpretation process, which determines the context, next.

2.3 Interpreting an Episode (a String)

An episode is an input string s of symbols (an element of $\mathcal{A}^*$). The act of interpreting is structuring one’s input, a string s in the input buffer, in terms of ones internal structures (existing concepts). The interpreter, given the network, of concepts and prediction weights, uses some technique (such as search and/or dynamic programming) to find a good structuring of all of s . Expedition scores candidate interpretation structures using the information-theoretic CORE score (Madani, 2023).

The system can go through many candidate interpretations in its search, but it finally selects a single complete interpretation. However, a final (complete) interpretation could itself be composed of multiple interpretation substructures: These substructures will be **chains** of concepts⁶ in this paper (Fig. 2 and 4(b)). Each (interpretation) chain covers a substring of the episode. Previously CORE was a reward-based information-gain score (the system sought to maximize it). In this paper, we simplify it to an information-theoretic *cost*: we no longer require computing a reward per (primitive) character (which was based on a prior kept updated from the input stream). Instead, we require the system to come up with an interpretation, consisting of one or more chains, that **matches** (or **covers** or explains) *all* of (the symbols of) s (*i.e.* a **complete interpretation**). The interpreter, in effect, selects which nodes and edges of the network to activate and use in order to cover s . However, each edge is associated with a probability, or equivalently, incurs a cost, rendering interpretation a dynamic decision making problem (Sect. 2.3.2) (Kull, 2018), with the objective of lowering the cost of covering the entire buffer.

2.3.1 The Structure of the Interpretation of an Episode

Let $|s|$ denote the length of string s . The first (left-most) character of s is at index 0, and its last (right-most) character is at index $|s| - 1$ (*e.g.*, if $s = \text{'cat'}$, then ‘c’, ‘a’, and ‘t’ are respectively at indices 0, 1, and 2). Each chain is a sequence of interpretation cells (or interpretants), each cell

6. This is another point of simplification in our current work. Other structures such as trees maybe possible, as long as learning and inference techniques for such exist.

containing a single concept and specifying which portion of s (consecutive substring, start index i_1 and end index i_2) is covered by that concept. There is a directed edge from one cell (containing concept c_1) to next (with c_2), the edge from c_1 to c_2 .⁷ A chain w is L2R if all its edges, $E(w)$, are L2R (in one direction, the focus of Sect. 3). A chain w with n nodes, has $n - 1$ edges: $|E(w)| = n - 1$. Simple chain examples are shown in Fig. 4(b).

2.3.2 The Cost of a Chain, and an Interpretation

The cost of a chain w is simply the sum of the costs of its edges (assuming exact match; see Sect. 2.5), where the cost of an edge e , with probability $p(e)$, is its surprisal, *i.e.* $-\log p(e)$.⁸ The cost of an interpretation is simply the sum over the costs of its chains (equations 1 to 4 in Fig. 5).

Equivalently, we can talk about the probability of a chain as products over edge probabilities: minimizing (sum of) costs is equivalent to maximizing (product of) probabilities.

2.4 A High Level Flow of How Expedition Works

We present a quick picture of how the current Expedition system works, given our latest design and algorithms, and given the description of the parts above. The learning is designed to be online, and resilient to non-stationarities, thus we are exploring inserting such a (perception) subsystem within autonomous agents.

The system begins with primitive concepts (corresponding to symbols in $\mathcal{A}$), and these concepts have no (out) edges initially (bare concepts, or *tabula rasa*). We assume the detection of such primitives is hardwired (a given), though errors, up to some level, can be tolerated (*e.g.* mixing ‘c’ for ‘e’ at times).

Horizontal edges are learned first for a concept (before vertical edges), via *sparse moving averages* (SMA) for learning the prediction probabilities (Madani, 2026). SMA learning (prediction and update techniques) is local (per node) and designed for non-stationarity and open-endedness (in particular, a dedicated *learning-rate per prediction edge* or per predictand). Even if there is no external non-stationarity (episodes are samples from a fixed corpus), the generation and use of new concepts results in significant internal non-stationarity (Fig. 3(b)).

Interpretation, in each episode, uses the edge probabilities to come up with a winning (low cost or high probability) interpretation. An interpretation itself is composed of multiple interpretation chains. For instance, initially, with no edges, an episode (string) of 50 symbols (unigrams or characters) yields a final interpretation composed of 50 interpretation chains. Adjacent (and highest level) concepts between two adjacent interpretation chains updates their horizontal edges to one another, leading to formation of new horizontal edges. Edges within a chain are also updated. Gradually, as horizontal edges are learned and used, the chains grow in length, and the number of chains per episode decreases. We have explored a number of criteria for generating and incorporating new concepts. Currently, converting a horizontal edge to vertical, after it has appeared in a final interpretation of some k episodes appears adequate (*e.g.* $k = 10$), but more investigation is required. Thus, at some timepoint, the horizontal edge $\mathbf{n} \rightarrow \mathbf{o}$ can be converted to the vertical $\mathbf{n} \uparrow \mathbf{no}$ (*i.e.* the concept

7. The first and last cell of chains will correspond to special (non-concept) nodes. See Sect. 3.

8. This term is common in information theory and is attributed to Tribus (Tribus, 1961; Hale, 2016).

no is created, and over time, incoming and outgoing edges are learned for it). All operations are local, to activated concepts (no global optimization), and because the out edges are upper bounded, the computational cost is curbed (Appendix B.3).

2.5 Discussion: Every Decision has a Cost (when Interpreting)

When using CORE, every decision made by the interpreter, as it grows an interpretation chain, *i.e.* whether to use a horizontal edge and, depending on the edge type, match to the right or to the left of the currently matched region, or follow a vertical edge (bottom-up: invoke a higher-level concept), and match accordingly, has associated with it a probability, and therefore a cost. The matching of a concept has an associated probability too, specially if the match is imperfect (due to typos or noise). In this paper we assume exact matches only, thus the cost of matching is 0 here.⁹

By incurring a cost for every interpretation action (selection of an edge, whether vertical or horizontal), *i.e.* unifying different types of edges, we hope that the search for a good (low cost) interpretation is better guided. Previously, we treated horizontal and vertical edges differently (Madani, 2023). Here, to keep the treatment uniform, even stopping an interpretation chain is a decision, and should incur a cost. In addition to a unified treatment on all decisions, this motivates learning calibrated (or proper) probabilities (Brier, 1950; Madani, 2026), as the following example illustrates:

Example 1. Assume a concept **a** sees concept **b** (to its right), say 0.8 of the time, and otherwise, a randomly picked concept (noise, or probability near 0) from a large set of concepts, 0.2 of the time. If we don't incur a cost for the stop decision, concept **a** has the incentive to put all its probability mass, 1.0, on the edge to **b**, which is not truthful.

But how should stop probabilities be set, *e.g.* is being calibrated a goal? If set to zero or too low, interpretation chains may be rendered too long (introducing errors in discovery of good predictive concepts). When and how should chains begin and terminate? One may plausibly expect that the probability of a horizontal (say L2R) edge $e : c_1 \rightarrow c_2$ should be close to the conditional probability $P(c_2|c_1)$, of observing c_2 next given c_1 is observed (the foundation for statistical language modeling based on n-grams (Rosenfeld, 2000)). But does optimizing according to CORE lead to such probabilities? The next section, in presenting and analyzing a formalization, provides answers to these questions in a simplified setting, confirming that the intuition (of learning conditional probabilities) holds to a good extent, and sheds light on the exceptions.

3. A Simplified Stop-N-Go Cost Minimization Problem

Informally, the Stop-N-Go cost minimization problem is finding probability assignments to edges among concepts (imagine a fixed set of concepts) that minimize the interpretation cost over a corpus. Here, we look at a simple version of the problem, for the case of unigrams, in which, for optimal interpreting, the only decision is whether to extend the current chain or stop and start a new chain. We next present a formalization, and derive several properties for the case of unigrams, exact match and no overlap, and a single edge direction: horizontal L2R. Our analysis sheds insights on a few

9. In general situations, when allowing approximate matches (deviating from the canonical form of the concept), the cost of an exact match will be positive but near 0.

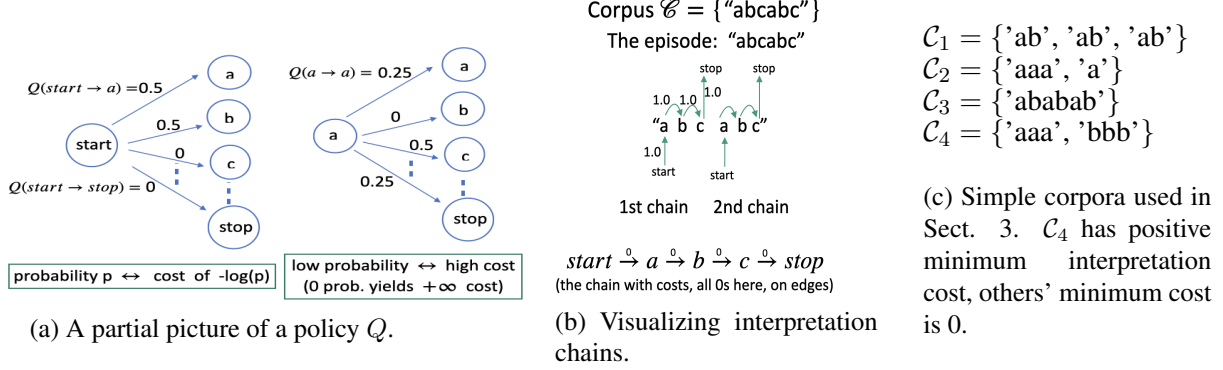


Figure 4: (a) A policy assigns a probability to every edge, forming a distribution on out edges of every node. Out edges for two nodes, with their probabilities, are shown (equivalently, edge costs could be shown). (b) A corpus $\mathcal{C}$ of one episode, and its optimal interpretation: two identical chains yielding 0 cost (per-edge and total). The picture shows two ways of visualizing interpretation chains. (c) Minimal corpora (each representing a class of examples) illustrating the power and limitations of a few candidates policies.

aspects of the problem, such as problem difficulty and the form optimal solutions take. See Sect. 3.2 for a summary.

Proof sketches and additional examples are provided in the appendix.

3.1 Problem Formulation

A **bag** is a collection, similar to a set (*i.e.* no ordering), but with repetitions of its elements allowed, such as a bag of strings or a bag of interpretation chains below. A **corpus** $\mathcal{C}$ is a non-empty finite¹⁰ bag of strings $s \in \mathcal{A}^*$. For example, $\mathcal{C} = \{'a', 'a', 'ab', 'c', 'a', 'c'\}$, and for this corpus, $\mathcal{A} = \{'a', 'b', 'c'\}$, and $|\mathcal{C}| = 6$.

Given a corpus $\mathcal{C}$ over alphabet $\mathcal{A}$, construct a corresponding directed graph $G = (V, E)$, with $|V| = |\mathcal{A}| + 2$ auxiliary nodes, as follows. V has two special nodes, *start* and *stop*, and otherwise there is a one-to-one correspondence between the remaining nodes of V , which we call **concepts**, (the unigram concepts) and elements of $\mathcal{A}$. Every node in $V - \{\text{stop}\}$ has a directed edge to every node in $V - \{\text{start}\}$ (the *stop* node has no out edges, a pure predictand, and *start* has no incoming, a pure predictor). We will abuse object types, treating concepts (nodes in the graph) and letters in $\mathcal{A}$ as the same, and write $V = \mathcal{A} \cup \{\text{start}, \text{stop}\}$ (thus concepts are both nodes in V and members of $\mathcal{A}$). Note: for us the helper nodes, *start* and *stop*, are *not* concepts (below, when we say an edge $e : u \rightarrow v$ between concepts, we mean $u, v \in \mathcal{A}$).

10. In exploring properties, and establishing theoretical results, we found it simpler to work with a finite corpus. We expect the insights to carry over when drawing strings from a statistical distribution.

A **policy** Q assigns¹¹ a probability (a value in $[0, 1]$) to every edge e of G (see Fig. 4), such that the out edges $E(v)$ of a node v form a distribution (the **distribution constraint**): $\forall v \in \mathcal{A} \cup \{\text{start}\}, \sum_{e \in E(v)} Q(e) = 1$ (with $E(v)$ denoting out edge set of node v). Equivalently, Q assigns a cost, $-\log(Q(e)) \in [0, +\infty]$, to every edge e . $\mathcal{Q}$ denotes the set of policies for $\mathcal{C}$. The *start edges* and *start probabilities* of a policy Q refers to the out edges of the start node and their probabilities, and the stop edge of a node $\mathbf{a}$ is $e : \mathbf{a} \rightarrow \text{stop}$.

A **chain** w is a finite walk over the nodes of G (a node sequence where repetitions are allowed), satisfying the conditions that the first node is the *start* node and the last node is the *stop* node, and w contains no more occurrences of *start* or *stop* (thus, for any chain w , $|w| \geq 2$). See Fig. 4(b). Given a policy Q , the cost of a chain w is the sum of its edge costs, Eq. 1 in Fig. 5 (chain costs, just like edge costs, are non-negative).

An (L2R) interpretation chain w for string s covers (matches) some non-empty substring of s as described in 2.3.1 (the indices of coverage into s are specified in the interpretation chain). We assume exact match here, thus w covers a substring of s of length $|w| - 2$. For example, if $s = \mathbf{abc}$, and w_1 is $\text{start} \rightarrow \mathbf{c} \rightarrow \text{stop}$ ($|w| = 3$, with $|E(w)| = 2$ edges), then w_1 covers the last letter of s , while with $w_2 = \text{start} \rightarrow \mathbf{a} \rightarrow \mathbf{b} \rightarrow \text{stop}$, then w_2 covers the prefix ' $\mathbf{ab}$ ' of s . The two chains w_1 and w_2 together form a (complete) interpretation $\mathcal{I}(s)$ of $s = \mathbf{abc}$ ($\mathcal{I}(s) = \{w_1, w_2\}$). For the degenerate case of the empty string $s = ''$, we take the chain $\text{start} \rightarrow \text{stop}$ to cover it. The interpretation $\mathcal{I}(s)$ corresponds to a bag¹² of one or more chains such that *all* symbols of s are covered by at least one chain (*i.e.* it provides 'complete' coverage) (exactly one chain covering when we don't allow chain overlap), with cost as given by Eq. 2 (sum of chain costs). Thus, interpreting an entire corpus $\mathcal{C}$ (interpreting each $s \in \mathcal{C}$), yields a bag of chains with size at least $|\mathcal{C}|$.

Given corpus $\mathcal{C}$ and a policy Q , an optimal interpreter for Q , for every episode $s \in \mathcal{C}$ finds a minimum cost interpretation, $\text{cost}_Q(s)$ (Eq. 3). Note that the interpreter is not allowed to change the probabilities of policy Q (only to do its best given a fixed policy). The decisions made by an optimal interpreter on one episode does not affect its decisions on other episodes: optimizing an interpretation on a string can be done independently of others (while the assignment choices in Q affects multiple episodes in general). The minimum cost is always non-negative, and it can be infinite for some policies. However, finite-cost policies always exist (examples below).

Definition 1. (*Stop-N-Go optimization*) Given a corpus $\mathcal{C}$, find a policy Q^* that yields the minimum interpretation cost, c^* , for $\mathcal{C}$ (over all policies $Q \in \mathcal{Q}$).

The following policy is akin to the situation that a new concept may be created without any connections initially. This and another example policy (Example 3) are special in that we make use of them in proposing algorithms and understanding the problem, etc.

Example 2. The *all-stops* policy, denoted Q_{as} , has every concept connecting to the stop node with probability 1.0. The start node's connections are symbol-start probabilities (see Definition 2) (other possibilities include uniform probabilities over $\mathcal{A}$).

11. We use the term 'policy' to emphasize some similarities with other decision making and control problems. Other terms could be 'transition-probabilities', 'cost-structure', or simply 'graph'.

12. The same chain, *i.e.* the same sequence of nodes, may have different *instantiations*, covering different parts of an episode and different episodes of $\mathcal{C}$, but for analysis and proofs, we treat such as the same chain (as long as the node sequence is identical).

$$cost_Q(w) = \sum_{e \in E(w)} -\log(Q(e)) \quad (\text{cost of a chain } w, \text{ given policy } Q: \text{ sum of } w\text{'s edge costs}) \quad (1)$$

$$cost_{Q,\mathcal{I}}(s) = \sum_{w \in \text{chains}(\mathcal{I}(s))} cost_Q(w) \quad (\text{cost of an interpretation } \mathcal{I} \text{ for string } s, \text{ given } Q) \quad (2)$$

$$cost_Q(s) = \sum_{w \in \text{chains}(\mathcal{I}_Q(s))} cost_Q(w) \quad (\text{cost of an optimal interpretation for } Q, \mathcal{I}_Q(s)) \quad (3)$$

$$cost_Q(\mathcal{C}) = \sum_{s \in \mathcal{C}} cost_Q(s) \quad (\text{cost of interpreting a corpus } \mathcal{C}, \text{ given } Q) \quad (4)$$

Figure 5: Interpretation costs, given a policy Q : from a single chain to an entire corpus. All are sums over constituent costs. Equivalently, the corpus cost is a product over all edge probabilities.

We can verify that the all-stops policy leads to finite cost as long as the *start* node has positive probabilities to all nodes.¹³ However, with no connections among concepts, intuitively, it is far from an optimal policy in general. For instance, consider $\mathcal{C}_1 = \{\text{'ab'}, \text{'ab'}, \dots\}$. While the cost of the all-stops policy increases as we expand $\mathcal{C}$, the chain $start \xrightarrow{0} \mathbf{a} \xrightarrow{0} \mathbf{b} \xrightarrow{0} stop$ yields 0 cost (edge cost shown above arrows), irrespective of size (in terms of optimal Q^* , $Q^*(start \rightarrow \mathbf{a}) = 1$, $Q^*(\mathbf{a} \rightarrow \mathbf{b}) = 1$, and $Q^*(\mathbf{b} \rightarrow stop) = 1$). On a single letter but non-trivial corpus $\mathcal{C}_2 = \{\text{'aa'}, \text{'a'}, \text{'aaa'}\}$, the all-stops policy yields only one chain type (but 6 identical chains), $start \xrightarrow{0} \mathbf{a} \xrightarrow{0} stop$, yielding 0 cost and is thus optimal (more generally when $\mathcal{A} = \{\mathbf{a}\}$ and $\forall s \in \mathcal{C}, s \in \mathcal{A}^+$).

Lemma 1. *In the Stop-N-Go problem, for any corpus, the minimum interpretation cost is finite and non-negative.*

The next results lead to reducing the space of policies that contains an optimal to a finite set, and making the minimization problem combinatorial.

Definition 2. *With $u, v \in \mathcal{A}$ (note that $u = v$ is possible) (concrete examples given next):*

- *(corpus conditionals) Let $P_{\mathcal{C}}(u \rightarrow v)$ denote the (corpus) conditional probability (given $u = \mathbf{a}$ is observed) that $v = \mathbf{b}$ occurs next to $\mathbf{a}$ to its right (L2R), i.e. the substring 'ab' , in $\mathcal{C}$. Similarly, we can define $P_{\mathcal{C}}(u \rightarrow stop)$ to be the conditional probability that u ends an episode s , and $P_{\mathcal{C}}(start \rightarrow u)$ be the probability that u begins an episode s (note: $P_{\mathcal{C}}(start \rightarrow u)$ is not conditioned on u , but on the beginning of an episode). We use $P_{\mathcal{C}}(u \rightarrow *)$ for the conditional distribution of a node u (probabilities of out edges of u).*
- *(corpus frequencies) Let $P_{\mathcal{C}}(u)$ denote the unconditional probability of concept u being seen (a random position in a string is picked).*

13. Any policy with all graph edges having positive probability yields finite interpretation cost. But also any policy with a positive edge probability from all nodes to the *stop* node, and from *start* to all nodes.

- (interpreter conditionals) For an interpreter $\mathcal{I}$, $P_{\mathcal{I}}(u \rightarrow v)$ denotes the conditional probability (given u is matched) that v is then matched in the same chain (the interpreter $\mathcal{I}$ does not stop current chain). Define $P_{\mathcal{I}}(u \rightarrow \text{stop})$ and $P_{\mathcal{I}}(\text{start} \rightarrow u)$ similarly. To compute these induced probabilities, one can use the bag of chains created by $\mathcal{I}$ in the wake of interpreting all of $\mathcal{C}$.

For example, with $\mathcal{C} = \{', 'ab', 'a'\}$, we have¹⁴ $P_{\mathcal{C}}(\text{start} \rightarrow \text{stop}) = 1/3$, $P_{\mathcal{C}}(\mathbf{a} \rightarrow \mathbf{b}) = 1/2$, $P_{\mathcal{C}}(\mathbf{a} \rightarrow \text{stop}) = 1/2$, $P_{\mathcal{C}}(\mathbf{b} \rightarrow \text{stop}) = 1$, and $P_{\mathcal{C}}(\mathbf{a}) = 2/3$. interpreter conditionals, $P_{\mathcal{I}}()$, can be similarly computed by treating the bag of chains, from applying $\mathcal{I}$ to $\mathcal{C}$, as a corpus of strings itself.

The following simplifying property is established (Appendix C) by showing that the behavior of an optimal interpreter can be consistent (on **'ab'** always pick $e : \mathbf{a} \rightarrow \mathbf{b}$, or always pick the stop-edge, $e' : \mathbf{a} \rightarrow \text{stop}$), and noting that $P_{\mathcal{I}}(\mathbf{a} \rightarrow \mathbf{b}) \leq P_{\mathcal{C}}(\mathbf{a} \rightarrow \mathbf{b})$ (or for short: $P_{\mathcal{I}}(e) \leq P_{\mathcal{C}}(e)$).

Lemma 2. *Given any corpus $\mathcal{C}$, for any policy Q , there is an optimal interpreter $\mathcal{I}$ for Q such that $\forall u, v \in \mathcal{A}$, $P_{\mathcal{I}}(u \rightarrow v) \in \{0, P_{\mathcal{C}}(u \rightarrow v)\}$.*

The following corollary, that $Q^*()$ must in turn match $P_{\mathcal{I}}()$, is established using the properties of KL-divergence minimization (Cover & Thomas, 1991):

Corollary 1. *For any corpus $\mathcal{C}$, there is an optimal policy Q^* such that $\forall u, v \in \mathcal{A}$, with $e : u \rightarrow v$, then $Q^*(e) \in \{0, P_{\mathcal{C}}(u \rightarrow v)\}$. (thus $\forall u \in \mathcal{A}$, $Q^*(u \rightarrow \text{stop}) \geq P_{\mathcal{C}}(u \rightarrow \text{stop})$).*

The corollary reduces the scope of options one needs to consider for optimal policies substantially: we need only consider keeping an edge (among concepts), with probability equal to the corpus conditional probability, or otherwise **zapping** it entirely (zeroing or pruning, moving the edge probability to the stop edge). It is a binary choice (a boolean variable) for each edge between two concepts: we need only consider the lattice of 0-1 policies (an edge is either zapped or not). We next show that some zapping, in general, is necessary to get to (near) optimal. Note: even a high probability edge can be zapped for optimality (e.g. the all-stops policy on $\mathcal{C}_1 = \{\text{'aaaa'}\}$).

Example 3. *The **plain-conditionals** policy, Q_{pc} , has every node $u \in \mathcal{A} \cup \{\text{start}\}$ connecting to another node $v \in \mathcal{A} \cup \{\text{stop}\}$ with probability $P_{\mathcal{C}}(u \rightarrow v)$.*

The plain-conditionals policy Q_{pc} and the all-stops policy sit at two extremes of the lattice of 0-1 policies: Q_{pc} results in the minimal number of chains, $|\mathcal{C}|$ many, while the other yielding the most chains. Q_{pc} is the optimal policy, yielding 0 cost, on example $\mathcal{C}_1$ of above (see Fig. 4(c))¹⁵ (where the all-stops policy failed), but in turn Q_{pc} fails on $\mathcal{C}_2$ (yielding positive cost), on which the all-stops policy is optimal, with 0 cost. Finally, on a corpus such as $\mathcal{C}_3 = \{\text{'ababab'}\}$ (i.e. $\mathcal{C}_3$ is a finite subset of $(\mathbf{ab})^+$), both policies fail to reach the minimum cost of 0. We note that Q_{pc} does very well, in our empirical experiments (Sect. 4), a foundation for n-gram language modeling (Rosenfeld, 2000). The following *shallow (zap) test*, inspired by the above properties, leads to optimality on all of $\mathcal{C}_1, \mathcal{C}_2$, and $\mathcal{C}_3$ (see Fig. 4(c)) when applied to Q_{pc} :

14. To compute all the conditionals, we can imagine, for each $s \in \mathcal{C}$, inserting *start* at the beginning and appending *stop* to the end).

15. When the corpus is repetitions of the same string, where the string has length 2 or higher, with no character appearing more than once inside it.

Definition 3. (*zap eligible*) Given policy Q , an edge $e : \mathbf{a} \rightarrow \mathbf{b}$, ($\mathbf{a}, \mathbf{b} \in \mathcal{A}$), passes the **shallow test** if $0 < Q(\mathbf{a} \rightarrow \mathbf{b}) < (Q(\mathbf{a} \rightarrow \mathbf{b}) + Q(\mathbf{a} \rightarrow \text{stop}))Q(\text{start} \rightarrow \mathbf{b})$. (*stopping and restarting is better*)

Note that the all-stops policy does not provide any (non-zero) edges between concepts, thus the shallow test cannot improve it. For policies that have concept-concept edges (such as Q_{pc}), zapping any such edge improves the policy (a sufficient condition for improving):

Lemma 3. Given policy Q_1 , let $e : \mathbf{a} \rightarrow \mathbf{b}$ pass the shallow test. Let policy Q_2 be the zapped result: $Q_2(e) = 0$ and $Q_2(\mathbf{a} \rightarrow \text{stop}) = Q_1(e) + Q_1(\mathbf{a} \rightarrow \text{stop})$, and otherwise, for any other edge e' , $Q_2(e') = Q_1(e')$. Then $\text{cost}_{Q_2}(C) < \text{cost}_{Q_1}(C)$.

The result holds because the change is a local one (only affects the next-cost of a single concept). The extension to zapping multiple edges simultaneously (passing the test) also holds for the same reason. It can be shown that the cost can further be reduced after the edge zap(s) by recomputing start probabilities $Q(\text{start} \rightarrow)$ (e.g. when only $e : \mathbf{a} \rightarrow \mathbf{b}$ is zapped, the start probability of $\mathbf{b}$ should go up (recall: for optimality, $Q()$ and $P_{\mathcal{I}}()$ should match in general)).

However, passing the shallow test on plain-conditionals policy is not necessary for improving it: the policy may be improvable even when no edge passes the shallow test, as the following illustrates:

Example 4. On $\mathcal{C}_4 = \{\text{'aaa'}, \text{'bbb'}\}$, there are two edges between concepts, $\mathbf{a} \rightarrow \mathbf{a}$ and $\mathbf{b} \rightarrow \mathbf{b}$ and neither passes the shallow test given policy Q_{pc} . However, zapping either (but not both), and recomputing start edges, improves the cost.

There are 4 possibilities to get Q^* on $\mathcal{C}_4$. The shallow test does not pass: $2/3 > (2/3 + 1/3)(1/2)$, but zapping an edge yields best cost ($4.16 < 5.8$ for $Q_{pc}()$) (C.1 provides details). Therefore, to achieve the optimal cost, one may need to sequentially (deep) test many edges, each involving the recomputation of the start edges, which in turn is a computation over all chains (an entire corpus recalculation). The ordering of the tests may matter too. Our experience suggests that zap heuristic below, based on the shallow test, may always remain close to the minimum cost (i.e. a *constant-ratio* approximation algorithm). We report on its empirical performance in Sect. 4:

Definition 4. *The repeated-zapping heuristic: Start with plain-conditionals then repeatedly zap all eligible edges (shallow test), and re-evaluate start probabilities, until no zappable edges remains.*

3.2 Summary and Discussion

Appendix C provides further details (proofs and proof sketches, and example cases). We highlight the main findings below, for the case of unigrams:

- Optimal discrete, in particular 0-1, policies exist.
- Edge direction does not add power: all edges can be unidirectional (and there is symmetry).
- Focusing on zapping edges from left-end to right-end symbols (L2R) suffices.
- (complexity) Local optima exist (greedy techniques may not be optimal).

- Empirical experiments and examples suggest simple heuristic may suffice for good approximation.

We expect good constant-ratio approximation techniques exist (*e.g.* via the shallow zap heuristic), even if optimizing were NP-hard (Papadimitriou, 1994). We hope to generalize (versions of) some of the above properties, such as edge direction, to the more general setting of n-grams. Our focus was on optimization here, to understand the form of an optimal policy (the edge weights), but pure optimization is likely unnecessary for robust learning of the higher-level concepts (and less applicable to online non-stationary settings). Furthermore, it risks overfitting even if computationally feasible.

4. Experiments

We have run experiments on multiple text files, and results exhibit the same pattern (two corpora reported here). We report **per-character-position cost** (total interpretation cost divided by corpus size in number of characters), so we can compare costs on different corpus sizes. We use (estimated) character frequencies for the all-stops policy, thus its cost is the character-level entropy of the corpus.

The plain-conditionals policy (Q_{pc}) handily beats the all-stops policy, as might be expected. For instance, on 500k lines from the NSF abstracts corpus ($|\mathcal{A}| = 96$, with $|s| \approx 55$ characters per episode) (Dua & Graff, 2017), all-stops gets 4.44 cost, while plain-conditionals gets 3.674. Repeated-zapping (Defn. 4) finishes in two rounds and gets minor improvement to 3.67, while the number of edges drops significantly from nearly 5k to nearly 4k. Note: with $|\mathcal{A}| \approx 100$ there can be at most 10k directed edges. Convergence of cost (as corpus size is increased) is very fast: On 20k lines, the cost of Q_{pc} after pruning is 3.675. The average number of chains per episode is close to 1 (*e.g.* 1.07). In an interactive setting, when the interpreter is given a string of consonants only (*e.g.* 'bsx') indeed we get several (singleton) chains, while if we insert vowels (*e.g.* 'basox'), we get a single chain for the whole string.

On the newsgroups text corpus (Lang, 1995), 300k lines, all-stops yields 5.01, plain-conditionals yields 4.12, and repeated-zapping gets 4.11 (the number of edges goes from 8.3k to 7k).

4.1 Summary of Findings when Learning N-grams

As concepts (n-grams) are built, we want to assess their quality, for instance whether they correspond to words and phrases when Expedition is run on natural language. Furthermore, we want to assess the splits made by the selected final interpretation. We continue to explore simple techniques for generating and using new concepts. Due to space limits and paper scope we only quickly and qualitatively summarize our preliminary findings: when learning higher n-grams (bigrams, $\dots$), indeed the cost of covering episodes goes down substantially compared to unigrams only (*e.g.* from 3.7, with unigrams, to below 2 per character cost on NSF). Furthermore, the quality of splits also show much promise (*e.g.* increasingly aligning with white space and punctuation).

In one class of experiments we call *deciphering*, we encode the characters in $\mathcal{A}$ into a smaller (lower-level) alphabet, such as a new alphabet $\mathcal{A}' = \{0, 1, 2, \dots, 9\}$ ($|\mathcal{A}'|=10$), using codes of fixed size 2 for each original character (which suffices when original $|\mathcal{A}| < 100$). An extreme version of this is encoding into binary ($\mathcal{A}' = \{0, 1\}$). The question is whether the system, by repeated

composing, can in effect recover the original alphabet and reach the character level cost performance (and beyond). Or otherwise, do the composing and detection errors compound? Earliest versions of Expedition had shown some promise (Madani, 2021). We are obtaining further improved results (*e.g.* the ratio of good-splits to all splits reaching near 100% with good sample efficiency), by using the new interpretation cost formulations and appropriate search techniques.

5. Related Work

This work has several roots and makes connections with a number ideas and research topics, including the importance of concepts to human intelligence (Murphy, 2002; Lakoff & Johnson, 1999; Hofstadter & Sander, 2013; Medin et al., 2000), constructivism (in education, epistemology, and AI), composition and hierarchy (Fosnot, 2005; Thorisson, 2012; Langley, 2022; Kruger et al., 2013), centrality of prediction (or expectations, anticipations, ...) for the workings of our minds (Medin et al., 2000; Ballard, 2000; Hawkins & Blakeslee, 2004; Clark, 2013), more programmable and capable units of computation as models for neurons (Valiant, 1994), and interpretation as a central component of the semiotics approach to cognition and meaning making (Sowa, 2006; Kull, 2018; Konderak, 2018; Raczaszek-Leonardi & Deacon, 2018).

Research shows that lexical probabilistic information, in particular surprisals, do very well as predictors of behavior and neural activity in language comprehension (Slaats & Martin, 2025; Hale, 2016). The free energy principle (FEP), also claims that, broadly speaking, organisms act to minimize surprisals (Friston, 2010): our work suggests that even within perception (and not just when performing overt actions), lowering surprisals is a useful goal.¹⁶

We use techniques that are closely related to language modeling and related fields, in particular via the n-gram counting methods (Manning & Schutze, 1999; Rosenfeld, 2000), but the vocabulary there is often fixed and given (see also Appendix B.4 for further discussion). Other related work includes tokenization (often fixed vocabulary) and feature induction (often for supervised tasks). Learning structure and in particular finite state machines can be prohibitive computationally and in terms of sample complexity, but there is progress and positive results empirically and in special cases (Verwer et al., 2013; Castro & Gavalda, 2016; Ron et al., 1998) though the focus has traditionally been on learning a single machine vs. many in a self-supervised fashion. In n-gram learning research (Pfleger, 2004; Curtin et al., 2026), the discovery task is typically separated from prediction, and n-grams are mined before hand for subsequent uses. In our approach, the system needs to learn and determine which n-grams are present in an episode, in the presence of noise in general (approximate match), and overlaps among activated n-grams in episodes will be limited due to use of CORE, which substantially reduces the number of redundant structures kept (see also Appendix B.4). We also aim to incorporate learning of disjunctions into the patterns (*i.e.* to go beyond pure conjunctions). There is also much work on grammar learning, in natural language and beyond (Nevill-Manning & Witten (1997); Stolcke & Omohundro (1994); Langley & Stromsten (2000); Wolff (1982, 2013)). The works of Wolff (Wolff, 2013) share a number of goals and a bit of similarity in techniques: ours has had a more limited scope, focusing on subsets of perceptual

16. We do not assume hidden variables in an external system to be modeled, unless the sensory stream of symbols is taken as such an external system, and do not in general take an optimization and static view of organisms.

learning. That work utilizes compression, via the minimum description principle (MDL). We hope to use ideas such as state merging, MDL, and factoring, when incorporating disjunctions into concept structure, and include more of grammar (type) learning (in an unsupervised online fashion). We conjecture that the inferences performed in each interpretation episode will be an important part of guiding the learner over a lifetime.

The literature on segmentation is also vast (in computer vision and audio processing), but often the problem is (partly) supervised (such as in semantic segmentation), and/or the (concept) vocabulary fixed (Thoma, 2016; Wang et al., 2017). Several research threads in computer vision and audio has focused on learning part (or compositional) hierarchies, and acquiring such with increasingly less supervision (Zhu et al., 2009; Tu et al., 2013; Si & Zhu, 2013; George et al., 2017; Jansen et al., 2020). Our self-supervised approach in treating patterns as both predictors and predictands, as well as our current formulation involving decision-making, *within perception*, is unique. Some *problem solving capabilities*, using an inventory of hierarchical patterns, has been ascribed to perceptual and post-perceptual inference stages in human vision (Pylyshyn, 1999; Schyns et al., 1998).

The psychological notion of a chunk (and chunking) (Miller, 1956; Chase & Simon, 1973; Gobet et al., 2001; Rakison & Oakes, 2003) is close to our n-gram modeling. Gobet *et al.* classify chunks as either perceptual or goal-oriented (Gobet et al., 2001), and ask whether chunking mechanisms can lay any claims towards being a universal theory of cognition. Our emphasis has been that such (perceptual) chunks, or representations underlying them, are not transitory: they are explicitly represented, in primarily a localist manner (Gross, 2002), with rich inter-connections dynamically maintained over a lifetime, and our goal is to provide computational accounts based on (unsupervised) learning and inference. See also work on engrams and cell assemblies (Josselyn et al., 2017).

Well-studied (dynamic) decision making problems such as MDPs (Dean & Wellman, 1991), including ones with learning, such as RL (Sutton & Barto, 1998), are often more general in terms of constraints on transitions and reward structure: here, the rewards (costs) and the ('transition') probabilities are intimately tied to one another. This is a much more constrained problem, which led to simplifying properties on the structure of optimal policies and optimization solutions.

The work on cognitive architectures explores how candidate components such as for perception, memory, and control can be combined and integrated, inspired by research in human and animal intelligence (Kotseruba & Tsotsos, 2025; Franklin et al., 2016; Langley et al., 2009). While we have focused on the narrower task of learning for *open-ended* perception, we have also assumed that multiple processes (tightly integrated) are likely needed, even for (a subset of) perception (Madani, 2007, 2025). The perceptual (framing) problem of *which* of one's concepts are relevant or useful at a given time point, has overlaps with what has been called 'relevance realization' as well as the (epistemological) frame problem (Jaeger et al., 2024; Käufer & Chemero, 2015; Shanahan, 2016).

The work on PGs has gone through a few revisions as problem formulations and solution techniques evolve and improve (Madani, 2007, 2021, 2023, 2026), such as: one-directional predictions appears adequate (no need for leave-one-out bidirectional scoring, nor classifiers). This paper simplifies and advances the scoring mechanism (CORE), in working towards unifying horizontal and vertical edges. We are also exploring applications to efficient online learning in agents, with actions and perception loops (Madani et al., 2026).

6. Summary

We began with the assumption that the human mind breaks its perceptual input into explicit structures (perceptual concepts). These concepts are also combined, over time, to create new concepts. In this way, the low level and vague senses can lead to higher level concepts that yield better predictions. The Prediction Games (PGs) framework tackles together the two problems of interpretation (analysis and synthesis in a time-snapshot), and learning new concepts (synthesis over a long period), and posits that learning from interpreting can be powerful and effective in acquiring perceptual concepts. In PGs, concepts function as both predictors and predictands, and a good coherent interpretation rests heavily on prediction (both when invoking concepts, lateral and bottom up, as well as top down: when matching the invocations to raw observations). Importantly, this task also involves a decision making component. We proposed and explored a formalization of the decision problem in this work. Each decision that the interpreter makes, the choice of which concept to invoke or whether to stop the current interpretation, has an associated probability (abiding by the probability distribution constraint) which the system’s scoring mechanism converts to costs (surprisals), and the job of the interpreter is to find a good *i.e.* low-cost (high coherence) interpretation. We presented a formalization of the interpretation+learning problem, taking interpretation structures to be simple chains, and made progress in analyzing the problem for the case of unigram concepts, with implications for more general cases. We hope to extend the work both empirically and theoretically towards more powerful concept representations with robust self-supervised techniques.

Acknowledgements

Thanks to the reviewer who pointed us to valuable prior work: we hope the presentation has improved based on the feedback. Thanks to my colleagues for their continued encouragement throughout the many stages of this work.

References

- Ballard, D. H. (2000). *An introduction to natural computation*. The MIT Press.
- Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*.
- Castro, J., & Gavalda, R. (2016). Learning prob. distributions generated by finite-state machines.
- Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, 4, 55–81.
- Clark, A. (2013). Whatever next? predictive brains, situated agents, and the future of cognitive science. *The Behavioral and brain sciences*.
- Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. John Wiley & Sons.
- Curtin, R. R., Lu, F., Raff, E., & Ranade, P. (2026). Intermediate n-gramming: Deterministic and fast n-grams for large n and large datasets. *AAAI*.
- Dean, T. L., & Wellman, M. P. (1991). *Planning and control*. Morgan Kaufmann.
- Dua, D., & Graff, C. (2017). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. From <http://archive.ics.uci.edu/>

- m1. <https://archive.ics.uci.edu/dataset/134/nsf+research+award+abstracts+1990+2003>.
- Dupont, P., Denis, F., & Esposito, Y. (2005). Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*.
- Fosnot, C. T. (Ed.). (2005). *Constructivism: Theory, perspectives, and practice*. Teachers College Press; 2nd edition.
- Franklin, S., Madl, T., Strain, S., Faghihi, U., Dong, D., Kugele, S., Snaider, J., Agrawal, P., & Chen, S. (2016). A LIDA cognitive model tutorial. *Biologically Inspired Cognitive Architectures*.
- Friston, K. J. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*.
- George, D., et al. (2017). A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*.
- Gobet, F., Lane, P., Croker, S., Cheng, P., Jones, G., Oliver, I., & Pine, J. (2001). Chunking mechanisms in human learning. *TRENDS in Cognitive Sciences*.
- Gross, C. G. (2002). Genealogy of the grandmother cell. *The Neuroscientist*.
- Hale, J. (2016). Information-theoretical complexity metrics. *Lang. Linguistics Compass*.
- Hawkins, J., & Blakeslee, S. (2004). *On intelligence: How a new understanding of the brain will lead to truly intelligent machines*. Owl Books.
- Hofstadter, D. R., & Sander, E. (2013). *Surfaces and essences: Analogy as the fuel and fire of thinking*. Basic Books.
- Hopcroft, J. E., & Ullman, J. D. (1979). *Intro. to automata theory, languages and computation*.
- Jaeger, J., Riedl, A., Djedovic, A., Vervaeke, J., & Walsh, D. (2024). Naturalizing relevance realization: why agency and cognition are fundamentally not computational. *Frontiers in Psychology*.
- Jansen, A., Ellis, D. P. W., Hershey, S., Moore, R. C., Plakal, M., Popat, A., & Saurous, R. A. (2020). Coincidence, categorization, and consolidation: Learning to recognize sounds with minimal supervision. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Josselyn, S. A., Köhler, S., & Frankland, P. W. (2017). Heroes of the engram. *The Journal of Neuroscience*.
- Käufer, S., & Chemero, A. (2015). *Phenomenology: An introduction*.
- Konderak, P. (2018). *Mind, cognition, semiosis: Ways to cognitive semiotics*. Lublin, Polska: Maria Curie-Sklodowska University Press.
- Kotseruba, I., & Tsotsos, J. (2025). *The computational evolution of cognitive architectures*.
- Kruger, N., Janssen, P., Kalkan, S., Lappe, M., Leonardis, A., Piater, J. H., Rodriguez-Sanchez, A. J., & Wiskott, L. (2013). Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Kull, K. (2018). Choosing and learning: Semiosis means choice. *Sign Systems Studies*.

- Lakoff, G., & Johnson, M. (1999). *Philosophy in the flesh: the embodied mind and its challenge to western thought*. Basic Books.
- Lang, K. (1995). Newsweeder: Learning to filter netnews. *ICML*.
- Langley, P. (2022). The computational gauntlet of human-like learning. *AAAI*.
- Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*.
- Langley, P., & Stromsten, S. (2000). Learning context-free grammars with a simplicity bias. *ECML 2000*.
- Madani, O. (2007). Prediction games in infinitely rich worlds. *AAAI Fall Symp.* (Yahoo! Research).
- Madani, O. (2021). Expedition: A system for the unsup. learning of a hierarchy of concepts. *ArXiv*.
- Madani, O. (2023). An information theoretic score for learning hierarchical concepts. *Frontiers in Computational Neuroscience*, 17.
- Madani, O. (2024). Tracking changing probabilities via dynamic learners. *arXiv*. Code at github.com/omadaniTet/sparse-moving-averages.
- Madani, O. (2025). Towards understanding and developing open-ended intelligences for infinite worlds. *Bio. Inspired Cognitive Architectures (BICA)*.
- Madani, O. (2026). Lifelong open-ended probability predictors. *TMLR*.
- Madani, O., Burns, B., Kugele, S., Georgiev, G., Eghbali, R., & Dean, T. (2026). Learning for survival. *BICA*.
- Manning, C. D., & Schutze, H. (1999). *Foundations of statistical natural language processing*. Cambridge University Press.
- Medin, D., Ross, B., & Markman, A. (2000). *Cognitive psychology*. Wiley.
- Miller, G. A. (1956). The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological review*.
- Murphy, G. L. (2002). *The big book of concepts*. MIT Press.
- Nevill-Manning, C. G., & Witten, I. H. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. *JAIR*.
- Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley.
- Pfleger, K. (2002). *On-line learning of predictive compositional hierarchies*. Doctoral dissertation, Stanford University.
- Pfleger, K. (2004). On-line cumulative learning of hierarchical sparse n-grams. *International Conference on Digital Libraries*.
- Pylyshyn, Z. W. (1999). Is vision continuous with cognition?: The case for cognitive impenetrability of visual perception. *Behavioral and Brain Sciences*, 22, 341 – 365.
- Raczaszek-Leonardi, J., & Deacon, T. W. (2018). Ungrounding symbols in language development: implications for modeling emergent symbolic communication in artificial systems. *Joint IEEE 8th ICDL-EpiRob*.

- Rakison, D. H., & Oakes, L. M. (Eds.). (2003). *Early category and concept development: Making sense of the blooming, buzzing confusion*. Oxford University Press.
- Ron, D., Singer, Y., & Tishby, N. (1998). On the learnability and usage of acyclic probabilistic finite automata. *J. Comput. Syst. Sci.*, 56.
- Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here? *IEEE*, 88.
- Schyns, P. G., Goldstone, R. L., & Thibaut, J. (1998). The development of features in object concepts. *Behavioral and Brain Sciences*, 21, 1 – 17.
- Shanahan, M. (2016). The Frame Problem. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy*. Metaphysics Research Lab, Stanford University, Spring 2016 edition.
- Si, Z., & Zhu, S. (2013). Learning AND-OR templates for object recognition and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Slaats, S., & Martin, A. E. (2025). What’s surprising about surprisal. *Computational Brain & Behavior*.
- Sowa, J. F. (2006). Peirce’s contributions to the 21st century. *Intrn. Conf. on Conceptual Structures*.
- Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by bayesian model merging. *Grammatical Inference and Applications*. Springer Berlin Heidelberg.
- Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. MIT Press.
- Thoma, M. (2016). A survey of semantic segmentation. *ArXiv*, abs/1602.06541.
- Thorisson, K. R. (2012). A new constructivist AI: from manual methods to self-constructive systems. In *Theoretical foundations of artificial general intelligence*. Springer.
- Tribus, M. (1961). *Thermostatistics and thermodynamics*.
- Tu, K., Pavlovskaja, M., & Zhu, S.-C. (2013). Unsupervised structure learning of stochastic and-or grammars. *NIPS*.
- Valiant, L. G. (1994). *Circuits of the mind*. New York: Oxford University Press.
- Verwer, S., E., R., & Higuera, C. (2013). Pautomac: a probabilistic automata and hidden markov models learning competition. *Machine Learning*, 96.
- Wang, C., Wang, Y., Huang, P.-S., Mohamed, A.-R., Zhou, D., & Deng, L. (2017). Sequence modeling via segmentations. *International Conference on Machine Learning*.
- Wolff, J. G. (1982). Language acquisition, data compression and generalization. *Language & Communication*.
- Wolff, J. G. (2013). The SP theory of intelligence: an overview. *Information*.
- Zhu, L., Chen, Y., & Yuille, A. L. (2009). Unsupervised learning of probabilistic grammar-markov models for object categories. *IEEE Trans. on Pattern Analysis and Machine Intelligence*.

alphabet $\mathcal{A}$: A finite non-empty set of symbols, such as $\mathcal{A} = \{0, 1\}$ or $\mathcal{A} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{n}\}$
concept ($u, v, \mathbf{a}$, etc): A concept is both an n-gram in $\mathcal{A}^+$ and a node in a directed (prediction) graph
primitive ($\mathbf{a}, \mathbf{b}$, etc): a unigram, and a basic (given/hardwired) concept corresponding to an element in $\mathcal{A}$.
episode s : a string in $\mathcal{A}^*$ (e.g. a line of text)
corpus $\mathcal{C}$: a bag of episodes (strings in $\mathcal{A}^*$), e.g. $\mathcal{C} = \{\mathbf{'a'}, 'a', 'ab', 'ab', 'a', 'ccc'\}$.
<i>start</i> and <i>stop</i> nodes: two helper (i.e. non-concept) nodes for starting and ending interpretation chains.
interpretation: a mapping from <i>all</i> positions in an episode to interpretants (concepts). An interpretation is composed of one or more interpretation chains. (Sect. 2.3.1 and 3)
interpretation chain: a finite sequence of nodes, beginning with <i>start</i> and ending in the <i>stop</i> node
interpretant: a cell in an interpretation chain specifying which substring of an episode is covered/matched.
policy $Q \in \mathcal{Q}$: an assignment of probabilities to a node's out edges, forming a distribution (Sect. 3) c^* denotes the minimum achievable cost on corpus $\mathcal{C}$ and Q^* is an optimal policy
cost of an edge e (under policy Q): the surprisal $-\log(Q(e))$ (where $Q(e)$ is the edge probability)
cost of a chain interpretation corpus: sum over the cost of its parts (e.g. for a chain, sum over its edge costs)
optimal Q^* : an optimal policy leading to the minimum interpretation cost on corpus $\mathcal{C}$
zapping (dropping) edge e : Given Q , changing it to Q' via setting $Q'(e) = 0$, possibly improving corpus cost.

Table 1: The main notation and terminology, and abbreviations.

Appendix A. Summary Terminology and Notation

Table 1 contains a summary of the main terms and notations used.

Appendix B. More on PGs (the Expedition System)

B.1 Terminology: On the Choice of the Word 'concept'

The patterns the system learns are relatively low level n-grams. However, as described in Sections 2.2 and 2.4, the learned representations have connections that are dynamically maintained, and such patterns are repeatedly reused, in making sense of the input, during the lifetime of the system. In particular, the acquired patterns are targets of prediction, and the system needs to learn when to predict a newly created concept (a process often called concept learning in supervised learning). Finally, we intend for the investigation of such processes to pave the way for the learning and use of more complex or abstract concepts and categories (general types, etc, such as chair, parents, democracy, unicorn, ...). Therefore, referring to them as (perceptual) concepts should be overall helpful: we prefer to use the more holistic (perceptual) 'concept' term, to other candidates, including: proto concepts, percepts, chunks, features, (represented) patterns, and so on (see also Sect. 5).¹⁷

17. These n-grams are 'represented' (or 'explicitly' stored) in the system (in the sense of being maintained as nodes with dynamic connections), but a question could be: what do these concepts or entities '*represent*', or correspond to, in the external world? The position we currently take is that such learned patterns are useful for better predicting the contents of the input buffer, and future contents of one's senses.

B.2 Further on CORE

When does the CORE score improve by generating a new concept uv , given part concepts u and v ? Let p_u and p_v be the probabilities that the concepts u and v get on average by the context (the preceding concept in our chain set up), and similarly let p_{uv} be the average that the concept uv would get if generated. In general $p_{uv} \leq \min(p_u, p_v)$, but as long as p_{uv} is greater than the product $p_u p_v$, CORE improves by creating uv . More generally, CORE rewards a system that can increase the prediction probability beyond the baseline of making the independence assumption and simply multiplying the unigram priors (Madani, 2023). But note that within an online setting, the system cannot predict the utility of a concept, beyond its parts, prior to its creation, especially when multiple concepts are being generated basically simultaneously. In this unsupervised setting, a challenge is making sure the system makes continued progress, as it generates new concepts, or otherwise remains robust in dynamically maintaining useful concepts and connections.

B.3 Computational Cost of Operating Expedition

The most expensive operation within an episode is interpretation, which is a bottom up process initiated by the activated primitives, and the number of out edges of each activated concept (and buffer size). The number of out edges, degree d , is a function of the minimum probability that is supported. For instance, if probabilities above $p_{min} = 0.01$ are to be kept track of, d is a few hundred edges, $O(\frac{1}{p_{min}})$ (Madani, 2026). The search for a highest scoring interpretation begins with activating a primitive, and exploring its edges and matches to grow the interpretation chain. We are looking at dynamic programming as well. The number of concepts in a chain is $O(|s|)$ (the length of the episode). Note: addition of each interpretation cell covers at least one more position (except first and last nodes). With average degree d , the cost of growing a single chain is $O(d|s|)$. Many chains are considered during the search, but the cost is roughly a multiple of $O(d|s|)$. Updating of edges, given a final selected interpretation, has similar $O(d|s|)$ cost (Madani, 2026).

Importantly, we do not require optimization or inference that would involve examining the entire concept network. Periodically, functions such as garbage collection may involve the whole graph (outside the loop of episode processing).

B.4 Relations to N-Gram Language Modeling and Related Techniques

We briefly describe the main technique behind statistical language modeling (SLM) (Manning & Schutze, 1999; Rosenfeld, 2000), and compare to our approach. Given a sentence such as “He goes to school.”, the tokens are “He”, “goes”, “to”, and so on, and in typical n-gram based SLM techniques, histories of several tokens are kept track of for predicting the next token. In this example, the n-grams would be “He”, “He goes”, “He goes to”, and so on, each serving as a predictor for the next token. SLM does not typically concern itself with identifying the tokens (the units) of meaning. In general, this leads to a large number of n-grams with high overlap (redundancy) for n-gram based SLM techniques. Prediction accuracy is the main objective. We note that there are other NLP tasks for discovering meaningful phrases, though they are not concerned with segmentation (Rosenfeld, 2000).

Pfleger explores a number of approaches to discovering n-grams (connectionist and otherwise) on unsegmented data (Pfleger, 2002, 2004). In (Pfleger, 2004), sparsity (non-redundancy) is a main objective, and sparsity accuracy tradeoffs are investigated. However, the approach remains primarily a data mining one (frequent association rule mining), and accurate interpretation (segmentation) is not utilized.

Appendix C. More on the Properties of Stop-N-Go Problem

We derive several simplifying properties of the unigrams exact-match L2R setting (such as symmetry in edge direction), but also provide examples pointing to hardness. The examples readily apply to more general cases (e.g. when the vocabulary has n-grams). Given a substring 'ab', there are two options for an interpreter given that it has matched **a**: either stop or otherwise follow **a** $\rightarrow$ **b** (continue the chain).

Given a fixed policy Q , informally a consistent (stable) interpreter always makes the same consistent decision in the same context (either always stop and restart, or continue extending current chain).

Definition 5. *Given is a corpus $\mathcal{C}$ and policy Q . We call an interpreter for Q **consistent** if it is **uv-consistent** $\forall u, v \in \mathcal{A}$. Meaning of **uv-consistency**: assume substring 'uv' occurs in some episode s , and we see the same chain covering both u and v , i.e. the edge $u \rightarrow v$ is used. Then for any 'uv' occurrence in $\mathcal{C}$, we should see one chain (always $u \rightarrow v$ used).*

Note that because there are up to two options for the interpreter, it follows that if it is consistent then, if on any occurrence of 'uv' in $\mathcal{C}$ it makes two chains (stops, then starts), then it should do the same on all 'uv' occurrences.

Lemma 4. *For any corpus $\mathcal{C}$ and policy Q , there is an optimal interpreter for Q that is consistent.*

Proof. Take any 'ab' where the behavior is different, and we conclude the two alternatives must incur the same cost (otherwise, we contradict the optimality of the interpreter wrt Q). We can pick the same alternative for all occurrences (make the interpreter consistent) without changing the cost. $\square$

Lemma 5. *For any corpus $\mathcal{C}$, and policy Q . Take any optimal consistent interpreter $\mathcal{I}$ for Q , and assume on 'uv' it takes the edge $e : u \rightarrow v$ (does not stop the chain after matching u). Then the fraction of time that it takes $e : u \rightarrow v$, or $P_{\mathcal{I}}(e)$, is equal to $P_{\mathcal{C}}(e)$.*

Proof. Consider a consistent interpreter optimal for Q . Given u is observed, v is observed $P_{\mathcal{C}}(u \rightarrow v)$ of the time, and if v is observed, the consistent interpreter takes it, thus $P_{\mathcal{I}}(u \rightarrow v) = P_{\mathcal{C}}(u \rightarrow v)$. $\square$

Corollary 1 follows (and using the characteristic property for minimization of KL divergence).

Corollary. *For any corpus $\mathcal{C}$, there is an optimal policy Q^* such that for every edge between two concepts $e : u \rightarrow v$, $Q^*(e) \in \{0, P_{\mathcal{C}}(u \rightarrow v)\}$. Furthermore letting $\mathcal{I}^* := \mathcal{I}_{Q^*}$, then $Q^*(\cdot) = P_{\mathcal{I}^*}(\cdot)$.*

Proof. Take an optimal consistent interpreter $\mathcal{I}$ for Q^* . Take any concept u , and let $V' \subseteq \mathcal{A}$ denote the set of concepts on which $\mathcal{I}$ doesn't stop (extends chain after matching u). If $v \in V'$ then $P_{\mathcal{I}}(u \rightarrow v) = P_{\mathcal{C}}(u \rightarrow v)$ from previous lemma. Let $p_{stop} := p_{\mathcal{I}}(u \rightarrow stop)$ and we must have $p_{stop} = 1 - \sum_{v \in V'} P_{\mathcal{C}}(u \rightarrow v)$.

The expected cost of next action when u is matched is: $-(\sum_{v \in V'} P_{\mathcal{C}}(u \rightarrow v) Q^*(u \rightarrow v) + p_{stop} \log(Q^*(u \rightarrow stop)))$. Recall that $P_{\mathcal{I}}(u \rightarrow *)$ is a distribution: $\sum_{x \in V} P_{\mathcal{I}}(u \rightarrow x) = 1$, or $\sum_{v \in V'} P_{\mathcal{C}}(u \rightarrow v) + p_{stop} = 1$. So the expected cost if the KL-divergence $KL(Q^*|P_{\mathcal{I}})$ (the divergence of Q^* from $P_{\mathcal{I}}$), and is minimized when $\forall v \in V', Q^*(u \rightarrow v) = P_{\mathcal{C}}(u \rightarrow v)$. $\square$

Note that there can be other non-optimal Q , such that the pair $Q()$ and the conditionals $p_{\mathcal{I}_Q}()$ equal one another. For example, the all-stops policy when we set the start probabilities to corpus frequencies: $\forall u \in \mathcal{A}, Q_{as}(start \rightarrow u) = P_{\mathcal{C}}(u)$.

Lemma 6. *Given any corpus $\mathcal{C}$, and any corpus (bag) of chains on it (some episodes broken into more than one chain). The chains corpus defines a $P_{\mathcal{I}}()$. The probability assignment $Q() = P_{\mathcal{I}}()$ is optimal for the chains corpus.*

C.1 The Shallow Test is not Necessary for Policy Improvement

A corpus such as $\mathcal{C}_4 = \{'aaa', 'bbb'\}$ (Example 4), shows that the shallow test is not necessary for improving a policy, *i.e.* sometimes a 'deep' test is required: first zap, then compute start probabilities too. This tends to be more expensive than a shallow test, requiring up to corpus wide computation. There are two concept-concept edges ($\mathbf{a} \rightarrow \mathbf{a}$ and $\mathbf{b} \rightarrow \mathbf{b}$) for $\mathcal{C}_4$, or two boolean variables, so 4 possibilities to get an optimal policy, with two policies being symmetric (zap exactly one edge). We have $Q_{pc}(\mathbf{a} \rightarrow \mathbf{a}) = 2/3$, $Q_{pc}(\mathbf{a} \rightarrow stop) = 1/3$ and $Q_{pc}(start \rightarrow \mathbf{a}) = 1/2$, and identically for edges involving $\mathbf{b}$. The shallow test does not pass: $2/3 > (2/3 + 1/3)(1/2)$. The cost of plain-conditionals $Q_{pc}()$ is $2(1 - 2\log(2/3) - \log(1/3)) \approx 5.8$, while if we zap say $\mathbf{a} \rightarrow \mathbf{a}$ and recompute start probabilities, we get $Q(start \rightarrow \mathbf{a}) = 3/4$, $Q(start \rightarrow \mathbf{b}) = 1/4$, and thus (3 chains of $start \rightarrow \mathbf{a} \rightarrow stop$, and one chain $start \rightarrow \mathbf{b} \dots$), and thus $-3\log(3/4) - \log(1/4) - 2\log(2/3) - \log(1/3) \approx 4.16$. The reduction in uncertainty of both starting with $\mathbf{a}$ and stopping (several times), outweighs the increase in uncertainty over $start \rightarrow \mathbf{b}$. Zapping both (*i.e.* the all-stops policy) performs the worst, yielding 6 (6 chains of cost 1 each).

Example $\mathcal{C}_4$ also shows that *not all* right-end to left-end nodes (see next section) should be zapped for optimality.

C.2 Other Examples Showing Complexity

Variations of $\mathcal{C}_4$ such as $\mathcal{C}_5 = \{'aaaaa', 'aaaaa', 'bbbbbb'\}$ suggests that zapping to a symbol that starts episodes more often can be better. This example also shows the possibility of 'local optima': The optimal is zapping $\mathbf{a} \rightarrow \mathbf{a}$, yielding a minimum cost of 8.44, and zapping the wrong edge (here, $\mathbf{b} \rightarrow \mathbf{b}$) may yield an improvement in cost (13.26) over our starting point of no zapping Q_{pc} (13.8 for Q_{pc}), but any further zap or unzip move increases cost (to 13.6 for Q_{pc} and 13.8 for Q_{as}). So this is a local optimum.

C.2.1 An Example showing Greedy Moves are Suboptimal (leading to local optima)

This example shows greedy zapping moves, even starting at Q_{pc} , can lead to a local optimum. Let $\mathcal{C} = \{ 'ababab', 'acac', 'ejejeje', 'djdjdjdj' \}$. The best (single) move is zapping $b \rightarrow a$ to give cost of 26.8548. Other single moves are either zapping $j \rightarrow e$ or $j \rightarrow d$ (symmetric), yielding 26.8966 (finally, zapping only $c \rightarrow a$ yields 26.9546) (Note: no zapping, Q_{pc} , has cost 28.1).

Now, the optimal policy is zapping the edges of j , *i.e.* two edges zapped only ($j \rightarrow e$ & $j \rightarrow d$), yields 24.8289. All other policies have higher costs, starting with 24.9085 (zap 3 edges, j 's and c 's) (next best is zap 4 edges, j 's, c ', and b 's, costing 25.35).

Now, I then ran it for one next move again, *i.e.* after zapping $b \rightarrow a$ (*i.e.* $\mathcal{C}' = \{ 'ab', 'ab', 'ab', 'acac', 'ejejeje', 'djdjdjdj' \}$), and got zapping $c \rightarrow a$ to yield best cost of 25.3868. If we don't take this move, the next best single move is zapping nothing! (to get 26.8548). With this (zapping $b \rightarrow a$ then $c \rightarrow a$), we reach a local optimum: every other single zap (or unzip), yields an inferior (higher cost) result.

Note that the costs are very similar: while optimization could be NP-hard, guaranteed constant-ratio approximation may be feasible (polynomial time).

C.2.2 An Example Showing Q_{pc} Can Itself be a Local Optimum

The following example shows that all single zapping moves (with a deep test), and even starting with Q_{pc} , can increase cost, and yet Q_{pc} may not be the optimal (multiple simultaneous moves required to get to optimal). In this example, $\mathcal{C} = \{ 'abab', 'acac', 'd':8 \}$, *i.e.* $\mathcal{C}$ contains 8 episodes of 'd'. For $\mathcal{C}$ (with only two zappable edges), if both **b** and **a** zap their edges, we get to the minimum with a cost of 15.02, while Q_{pc} has cost 15.22 (no edge zapped), and zapping a single edge yields 15.30. If we reduce the number of 'd' episodes in the corpus (*e.g.* from 8 to 7), then zapping a single edge becomes profitable, while if we add more 'd' episodes, Q_{pc} becomes the optimal (variations such as using a mix of letters or empty strings instead of 'd', also work to show this issue).

C.3 Additional Simplifying Properties (Symmetry, ...)

We start with the idea of the chain corpus and sketch the proofs for edge symmetry, and other simplifying results (for the case of unigrams).

C.3.1 The Chain Corpus

We have viewed the problem mostly as first defining or fixing a policy Q , then an interpreter abiding by Q to get an interpretation of a whole corpus. Another way is to just imagine a *chain corpus*, with consistent decisions, and defining a Q using it. A chain corpus, from applying some consistent interpretation $\mathcal{I}$ onto $\mathcal{C}$, is denoted $\mathcal{C}_{\mathcal{I}}$. Given a chain corpus, its optimal 'derived' $Q_{\mathcal{I}}$ can be computed, and is simply $Q_{pc}(\mathcal{C}_{\mathcal{I}})$ treating the chain corpus as a normal (string) corpus (with *start* and *stop* as extra symbols). The set of chain corpora includes at least one chain corpus that is optimum, yielding the minimum corpus cost.

Given a chain corpus, zapping an edge $u \rightarrow v$ means removing all $u \rightarrow v$ edges, and replacing each by the two edge $u \rightarrow stop$ and $start \rightarrow v$ (to obtain another chain corpus). Unzapping means doing the reverse.

C.3.2 (Symmetry) Every chain corpus $\mathcal{C}_{\mathcal{I}}$ is Symmetric (Reversible)

Reversing a chain means reversing all its edges (e.g. from L2R to R2L). For example reversing $w : start \rightarrow \mathbf{a} \rightarrow \mathbf{b} \rightarrow stop$, yields $w' : start \rightarrow \mathbf{b} \rightarrow \mathbf{a} \rightarrow stop$ (or $stop \leftarrow \mathbf{a} \leftarrow \mathbf{b} \leftarrow start$).

This edge symmetry property, noteworthy in itself, is used in the next section too:

Lemma 7. *Given a corpus $\mathcal{C}$, take any interpretation chain corpus $\mathcal{C}_{\mathcal{I}}$ for it. Then reversing all the edges yields the same cost ($\mathcal{C}_{\mathcal{I}}$ is symmetric).*

Proof. The cost of any chain of the corpus corresponds to a product of (edge) conditional probabilities, each can be written as a fraction as follows ($\frac{A \cap B}{A} = \frac{\text{count of A and B}}{\text{count of A}}$): The numerator is the product of the edge-seen counts. The denominator is the product of the node seen counts times number of chains in the corpus. These counts do not change by reversing the edges (only the order of the counts changes). $\square$

Corollary 2. *The optimal cost for a corpus $\mathcal{C}$ remains the same under edge reversal (same for optimal corpus chains).*

We can extend the arguments on counting to show that if a chain corpus has chains of mixed type, some in L2R and some in R2L directions, making all in one direction can only improve cost. This argument provides a basis for showing that having multiple edge directions does not add power (does not lower cost).

C.3.3 Only Zap Some of End-to-Start Edges

Given a corpus $\mathcal{C}$, let $\mathcal{A}_{left}$ (start-support of $\mathcal{C}$) be those concepts that begin some episode (or those $\mathbf{a} \in \mathcal{A}$, such that $P_{\mathcal{C}}(start \rightarrow \mathbf{a}) > 0$) (recall: $P_{\mathcal{C}}()$ is the corpus conditional probabilities), and let $\mathcal{Q}_{left}$ (start-support of Q) be those concepts $\mathbf{a}$ with $Q(start \rightarrow \mathbf{a}) > 0$. Define $\mathcal{A}_{right}$ and $\mathcal{Q}_{right}$ similarly. We can verify that for any optimal Q^* , $\mathcal{A}_{left} \subseteq \mathcal{Q}_{left}^*$. The following states an equality version.

Lemma 8. *Given a corpus $\mathcal{C}$ with $|\mathcal{A}_{left}| \geq 1$:*

1. *There exists an optimal Q^* with $\mathcal{Q}_{left}^* = \mathcal{A}_{left}$.*
2. *There exists an optimal Q^* whose only difference from Q_{pc} is that some subset of edges from nodes in $\mathcal{A}_{right}$ to nodes in $\mathcal{A}_{left}$, (if any), are zapped (if no end-to-start edges exist, Q_{pc} is optimal).*
3. *If $\mathcal{A}_{left} = \{v\}$ (i.e. $|\mathcal{A}_{left}| = 1$) and $\epsilon \notin \mathcal{C}$ (no empty string/episode), then zapping all edges to v (also more restricted: all edges from any node in $\mathcal{A}_{right}$ to v), yields an optimal solution (similarly when $|\mathcal{A}_{right}| = 1$).*

We provide sketches of the argument for the above showing that it suffices that $\mathcal{Q}_{left}^* = \mathcal{A}_{left}$ (though there can be several optimal policies with larger start support, such as $\mathcal{C} = \{\mathbf{abcnb}\}$, which has 4 optimal policies, 3 have start-support larger than $\{\mathbf{a}\}$, corresponding to the powerset of $\{\mathbf{c}, \mathbf{n}\}$) (even limiting to $\mathcal{A}_{left}$ can yield multiple optimals, as example $\mathcal{C}_4$ shows).

We next describe another way of accounting, for computing corpus cost, that allows us to make progress towards establishing the above zapping claims. Given a corpus $\mathcal{C}$, policy Q , and interpreter $\mathcal{I}$: Let $N(u)$ be the number of times $u \in \mathcal{A}$ appears in corpus $\mathcal{C}$. Let $\text{next}(u)$ be the average cost of the immediate next action or two to cover the immediate next symbol after u or stop if end of the buffer is reached (either take an edge $u \rightarrow v$, or stop and then take $\text{start} \rightarrow v$). Let $\text{start-cost}(s)$ be the cost of the first (start) edge of episode s , the average cost of starting, averaged over original episodes, being $\text{next}(\text{start})$. One can then write the corpus cost of an interpreter $\mathcal{I}$ under policy Q in terms of start-cost of original episodes ($|\mathcal{C}|$ many) ('episode start costs'), plus the next cost for each u in $\mathcal{A}$:

$$\text{cost}_Q(\mathcal{C}) = \left(\sum_{s \in \mathcal{C}} \text{start-cost}(s) \right) + \left(\sum_{u \in \mathcal{A}} N(u) \cdot \text{next}(u) \right) = |\mathcal{C}| \cdot \text{next}(\text{start}) + \sum_{u \in \mathcal{A}} N(u) \cdot \text{next}(u) \quad (5)$$

Now, when we zap say a single out edge e of node u , $e : u \rightarrow v$, the new corpus (bag) of chains has more chains than before. The start-costs of original episodes, by zapping, may also go up or remain unchanged (cannot go down, due to minimizing property of KL divergence).

Lemma 9. *The start-costs (of original episodes), $\sum_{s \in \mathcal{C}} \text{start-cost}(s)$, have their minimum with Q_{pc} (zapping any edges may only increase the original start-cost, at best, leave it unchanged).*

Lemma 10. *(on $\text{next}(u)$), given is a $Q()$ (or $Q_{\mathcal{I}}()$) derived from the chain corpus of a consistent interpreter $\mathcal{I}$.*

1. *The $\text{next}(u)$ cost of a node u where $Q(u \rightarrow \text{stop}) = P_{\mathcal{C}}(u \rightarrow \text{stop})$ (no zapping) is the entropy of $P_{\mathcal{C}}(u \rightarrow *)$.*
2. *The $\text{next}(u)$ cost of a node u where $u \notin \mathcal{A}_{\text{right}}$ (i.e. $P_{\mathcal{C}}(u \rightarrow \text{stop}) = 0$), is never below the entropy of $P_{\mathcal{C}}(u \rightarrow *)$ (and, by part 1, is achieved by the node u keeping all its edges, i.e. keeping all its positive edges in Q_{pc}).*
3. *Therefore, with $u \notin \mathcal{A}_{\text{right}}$, if we also have support of $P_{\mathcal{C}}(u \rightarrow *) \neq \mathcal{A}_{\text{left}}$ (e.g. $\exists v \in \mathcal{A}_{\text{left}}, P_{\mathcal{C}}(u \rightarrow v) = 0$), then if any edge of u has been zapped by $\mathcal{I}$ then $\text{next}(u)$ is higher than the minimum possible.*
4. *With $u \notin \mathcal{A}_{\text{right}}$, all zapped edges of u , if any (via $\mathcal{I}$), can be unzapped (change $Q(u)$ only, i.e. local change, no other change in Q), to get equal or lower corpus cost.*

Proof. (proof for part 1) With $Q(u \rightarrow \text{stop}) = P_{\mathcal{C}}(u \rightarrow \text{stop})$, then $\forall x \in \mathcal{A}, Q(u \rightarrow x) = P_{\mathcal{C}}(u \rightarrow x)$ (no zapping). The cost from using $Q()$ after matching u is: $\text{next}(u) = - \sum_{x \in \mathcal{A} \cup \{\text{stop}\}} P_{\mathcal{C}}(u \rightarrow x) \log(Q(u \rightarrow x)) = - \sum_{x \in \mathcal{A} \cup \{\text{stop}\}} P_{\mathcal{C}}(u \rightarrow x) \log(P_{\mathcal{C}}(u \rightarrow x))$.

(proof of part 2 and 3) If $Q(u \rightarrow \text{stop}) = 0$ we are done, via part 1. Assuming $Q(u \rightarrow \text{stop}) > 0$, the cost from using $Q()$ after matching u is: $\text{next}(u) = - \sum_{x \in \mathcal{A}} P_{\mathcal{C}}(u \rightarrow x) \log(\max(Q(u \rightarrow x), Q(u \rightarrow \text{stop})Q(\text{start} \rightarrow x)))$ (note the sum does not include $x = \text{stop}$, as $P_{\mathcal{C}}(u \rightarrow \text{stop}) = 0$).

Now, whether $Q(u \rightarrow x)$ is picked for some x or $Q(u \rightarrow stop)Q(start \rightarrow x)$, we first want to show that we won't violate the sub-distribution property (by using the max operator), as we explain next. For $x \in \mathcal{A}$, define $Q'_u(x) := \max(Q(u \rightarrow x), Q(u \rightarrow stop)Q(start \rightarrow x))$, and we note $\forall x, Q'_u(x) \leq 1$. We want to show $Q'_u(x)$ defines a sub-distribution on $x \in \mathcal{A}$, i.e. $\sum_{x \in \mathcal{A}} \max(Q(u \rightarrow x), Q(u \rightarrow stop)Q(start \rightarrow x)) = \sum_{x \in \mathcal{A}} Q'_u(x) \leq 1$ (note: it can be strictly below 1 as $Q(u \rightarrow stop) > 0$, while, because $u \notin \mathcal{A}_{right}$, $P_C(u \rightarrow stop) = 0$).

Let $p_x := Q(u \rightarrow x)$, and let $s := \sum_{x \in \mathcal{A}} p_x$, so $0 \leq s < 1$. Let $p'_x := Q(start \rightarrow x)$, then in $\sum_{x \in \mathcal{A}} Q'_u(x)$, some entries are p_x and some are $(1-s)p'_x$ (whichever is positive and higher). So: $\sum_{x \in \mathcal{A}} Q'_u(x) = \sum_{x \in \mathcal{A}_1} p_x + \sum_{x \in \mathcal{A} - \mathcal{A}_1} (1-s)p'_x = \sum_{x \in \mathcal{A}_1} p_x + (1-s) \sum_{x \in \mathcal{A} - \mathcal{A}_1} p'_x$ (where $\mathcal{A}_1 \subseteq \mathcal{A}$). Thus $\sum_{x \in \mathcal{A}} Q'_u(x) \leq \sum_{x \in \mathcal{A}_1} p_x + (1-s) \leq s + 1 - s = 1$. With $s < 1$, if $\mathcal{A}_1 = \mathcal{A}$, then $\sum_{x \in \mathcal{A}} Q'_u(x) = \sum_{x \in \mathcal{A}_1} p_x = s < 1$ (strict subdistribution). If $\mathcal{A}_1 = \emptyset$, then $\sum_{x \in \mathcal{A}} Q'_u(x) = (1-s) \sum_{x \in \mathcal{A}} p'_x \leq 1 - s$. In this case (all zap, complete stop), only if $Q(u \rightarrow *) = P_C(u \rightarrow *)$ (identical distributions), we get the same entropy cost. If both $\mathcal{A}_1 \neq \emptyset$ and $\mathcal{A}_1 \subset \mathcal{A}$, then, with $\sum_{x \in \mathcal{A}_1} p_x < s$, again $Q'_u(x)$ is a strict sub-distribution.

From the space of such sub-distributions $Q'_u(x)$, what minimizes $-\sum_{x \in \mathcal{A}} P_C(u \rightarrow x) \log(Q'_u(x))$ is when $Q'_u(x) = P_C(u \rightarrow x)$ (no mass should be allocated going to the stop node for u , and the other edges should also be exactly as $P_C(u \rightarrow *)$ specifies!). If $0 < Q(u \rightarrow stop) < 1$ then we get a strict sub-distribution and a strictly inferior cost (on sub-distribution divergences see (Madani, 2024, 2025)). If all edges are zapped, i.e. $Q(u \rightarrow stop) = 1$, in general we get a strictly inferior result too, unless for the very exceptional case that $Q(start \rightarrow *) = P_C(u \rightarrow *)$. Therefore, if u has no edge to some $v \in \mathcal{A}_{left}$, or vice versa, $P_C(u \rightarrow v) > 0$ but $v \notin \mathcal{A}_{left}$, then $Q(start \rightarrow *) \neq P_C(u \rightarrow *)$ (in this case, only the no zapping case yields the minimum cost).

(proof of part 4) Unzapping the zapped edges of u does not change $Q(start \rightarrow *)$, and thus does not change $\text{next}(x)$, for $x \neq u$, nor the start-cost portion of the corpus cost. From previous parts, unzapping can only improve $\text{next}(u)$. $\square$

Finally, we can use (edge) symmetry, and the above property (Lemma 10) to show that, to get to an optimal policy, nodes in $\mathcal{A}_{right}$ need not zap their edges to nodes outside of $\mathcal{A}_{left}$.