

Let's wrap this up! Incremental structured decoding with resource constraints

Breandan Considine

Main Idea

- Language models have trouble with single-shot constraint satisfaction
- Typically solved via rejection sampling or backtracking style decoders
- We implement an incremental structured decoder for autoregressive LLMs
- Guarantees monotonic progress and preservation of resource constraints
- Ensures all valid words are generable and all generable words are valid

Motivation

Suppose we want to force an autoregressive LLM to generate syntactically valid next tokens $P(x_n | x_1, \dots, x_{n-1})$, under certain resource constraints. Here is a concrete example: “generate a valid arithmetic expression in ten or fewer tokens”. If we sample the partial trajectory:

(a + (c * (

then we will spend quite a long time rejecting invalid completions, because this trajectory has past the point of no return. Even though (is a locally valid continuation, we need to avoid this scenario, because we would like a linear sampling delay and to guarantee this, we must avoid backtracking.

Semiring Parsing

Given a CFG $\mathcal{G} = \langle V, \Sigma, P, S \rangle$ in Chomsky Normal Form (CNF), we may construct a recognizer $R_{\mathcal{G}} : \Sigma^n \rightarrow \mathbb{B}$ for strings $\sigma : \Sigma^n$ as follows. Let 2^V be our domain, where 0 is \emptyset , \oplus is \cup , and \otimes be defined as:

$$s_1 \otimes s_2 = \{C \mid \langle A, B \rangle \in s_1 \times s_2, (C \rightarrow AB) \in P\}$$

If we define $\hat{\sigma}_r = \{w \mid (w \rightarrow \sigma_r) \in P\}$, then construct a matrix with unit nonterminals on the superdiagonal, $M_0[r+1 = c](G', \sigma) = \hat{\sigma}_r$ the fixpoint $M_{i+1} = M_i + M_i^2$ is fully determined by the first diagonal:

$$M_0 = \begin{pmatrix} \emptyset & \hat{\sigma}_1 & \emptyset & \emptyset \\ & \emptyset & \hat{\sigma}_2 & \emptyset \\ & & \emptyset & \hat{\sigma}_n \\ \emptyset & & & \emptyset \end{pmatrix} \Rightarrow \begin{pmatrix} \emptyset & \hat{\sigma}_1 & \Lambda & \emptyset \\ & \emptyset & \hat{\sigma}_2 & \Lambda \\ & & \emptyset & \hat{\sigma}_n \\ \emptyset & & & \emptyset \end{pmatrix} \Rightarrow \dots \Rightarrow M_{\infty} = \begin{pmatrix} \emptyset & \hat{\sigma}_1 & \Lambda & \Lambda_{\sigma}^* \\ & \emptyset & \hat{\sigma}_2 & \Lambda \\ & & \emptyset & \hat{\sigma}_n \\ \emptyset & & & \emptyset \end{pmatrix}$$

CFL membership is recognized by $R(G', \sigma) = [S \in \Lambda_{\sigma}^*] \Leftrightarrow [\sigma \in \mathcal{L}(G)]$.

Parsing Dynamics

Let us consider an example with two holes, $\sigma = 1 _ _$, with the grammar being $G = \{S \rightarrow NON, O \rightarrow + \mid \times, N \rightarrow 0 \mid 1\}$. This can be rewritten into CNF as $G' = \{S \rightarrow NL, N \rightarrow 0 \mid 1, O \rightarrow + \mid \times, L \rightarrow ON\}$.

	2^V	$\mathbb{B}^{ V }$	$\mathbb{B}^{ V } \rightarrow \mathbb{B}^{ V }$
M_0	$\begin{pmatrix} \{N\} \\ \{N, O\} \\ \{N, O\} \end{pmatrix}$	$\begin{pmatrix} \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \end{pmatrix}$	$\begin{pmatrix} V_{0,1} \\ V_{1,2} \\ V_{2,3} \end{pmatrix}$
M_1	$\begin{pmatrix} \{N\} & \emptyset \\ \{N, O\} & \{L\} \\ \{N, O\} & \{N, O\} \end{pmatrix}$	$\begin{pmatrix} \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \end{pmatrix}$	$\begin{pmatrix} V_{0,1} & V_{0,2} \\ V_{1,2} & V_{1,3} \\ V_{2,3} & \end{pmatrix}$
M_{∞}	$\begin{pmatrix} \{N\} & \emptyset & \{S\} \\ \{N, O\} & \{L\} \\ \{N, O\} & \{N, O\} \end{pmatrix}$	$\begin{pmatrix} \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \blacksquare \end{pmatrix}$	$\begin{pmatrix} V_{0,1} & V_{0,2} & V_{0,3} \\ V_{1,2} & V_{1,3} \\ V_{2,3} & \end{pmatrix}$

This procedure decides if $\exists \sigma' \in \mathcal{L}(G) \mid \sigma' \sqsubseteq \sigma$ but forgets provenance.

Regular Expression Propagation

Regular expressions that permit union, intersection and concatenation are called generalized regular expressions (GREs) and can be constructed as follows:

$$\begin{aligned} \mathcal{L}(\emptyset) &= \emptyset & \mathcal{L}(R^*) &= \{\varepsilon\} \cup \mathcal{L}(R \cdot R^*) \\ \mathcal{L}(\varepsilon) &= \{\varepsilon\} & \mathcal{L}(R \vee S) &= \mathcal{L}(R) \cup \mathcal{L}(S) \\ \mathcal{L}(a) &= \{a\} & \mathcal{L}(R \wedge S) &= \mathcal{L}(R) \cap \mathcal{L}(S) \\ \mathcal{L}(R \cdot S) &= \mathcal{L}(R) \times \mathcal{L}(S) & \mathcal{L}(\neg R) &= \Sigma^* \setminus \mathcal{L}(R) \end{aligned}$$

Finite slices of a CFL are finite and thus regular a fortiori. Just like sets, bitvectors and other datatypes, we can also propagate GREs through a parse chart. Here, the algebra will carry $\text{GRE}^{|V|}$, where 0 is \emptyset , and \oplus, \otimes are defined:

$$s_1 \otimes s_2 = \left[\bigvee_{(v \rightarrow AB) \in P} s_1[A] \cdot s_2[B] \right]_{v \in V} \quad s_1 \oplus s_2 = [s_1[v] \vee s_2[v]]_{v \in V}$$

Initially, we have $M_0[r+1 = c](G', \sigma) = \sigma_r$. Now after computing the fixpoint, when we unpack $\Lambda_{\sigma}^*[S]$, this has type GRE, recognizing the finite CFL slice.

Brzowski Differentiation

Janusz Brzowski (1964) introduced a derivative operator $\partial_a : \text{REG} \rightarrow \text{REG}$, which slices the prefix off a language: $\partial_a S = \{b \in \Sigma^* \mid ab \in S\}$. The Brzowski derivative over a GRE is effectively a normalizing rewrite system:

$$\begin{aligned} \partial_a \emptyset &= \emptyset & \delta(\emptyset) &= \emptyset \\ \partial_a \varepsilon &= \emptyset & \delta(\varepsilon) &= \varepsilon \\ \partial_a a &= \varepsilon & \delta(a) &= \emptyset \\ \partial_a b &= \emptyset \text{ for each } a \neq b & \delta(R^*) &= \varepsilon \\ \partial_a R^* &= (\partial_a R) \cdot R^* & \delta(\neg R) &= \varepsilon \text{ if } \delta(R) = \emptyset \\ \partial_a \neg R &= \neg \partial_a R & \delta(\neg R) &= \emptyset \text{ if } \delta(R) = \varepsilon \\ \partial_a R \cdot S &= (\partial_a R) \cdot S \vee \delta(R) \cdot \partial_a S & \delta(R \cdot S) &= \delta(R) \wedge \delta(S) \\ \partial_a R \vee S &= \partial_a R \vee \partial_a S & \delta(R \vee S) &= \delta(R) \vee \delta(S) \\ \partial_a R \wedge S &= \partial_a R \wedge \partial_a S & \delta(R \wedge S) &= \delta(R) \wedge \delta(S) \end{aligned}$$

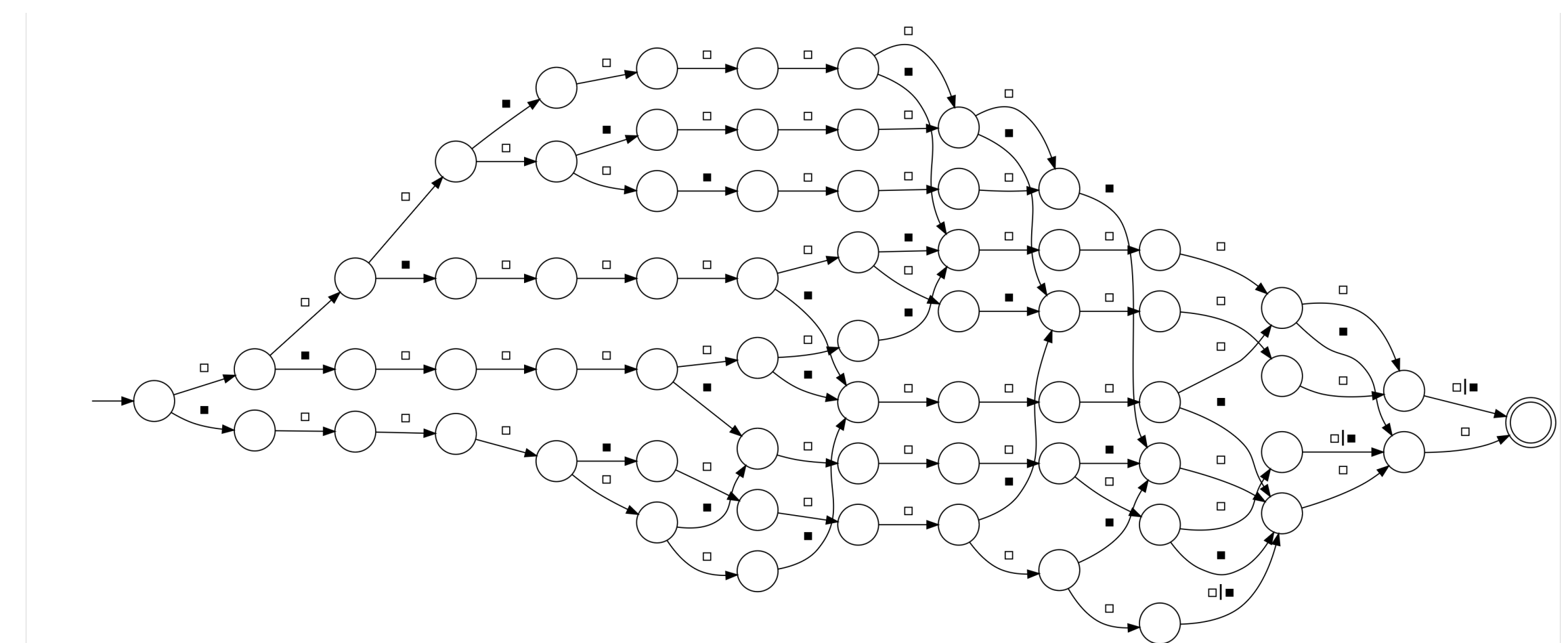
The key property we care about is, this formulation allows us to sample lazily from language intersections, without first materializing the product automaton.

Example: Determinantal Point Processes

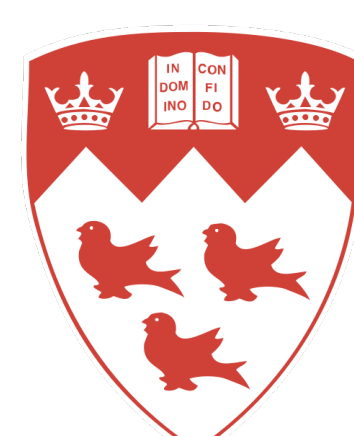
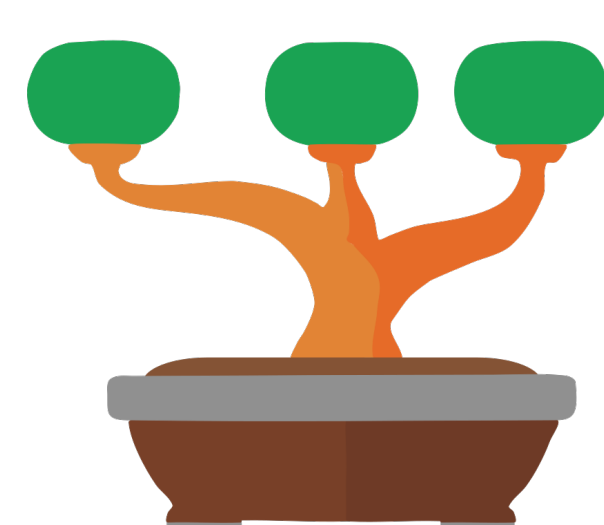
Consider a time series, A , whose points which are not too close nor far apart, and $n \leq \sum_{i=1}^{|A|} [A_i = \blacksquare]$. We want to sample the typical set using an LLM.

- The words are bitvectors of some length T , i.e., $A = \{\square, \blacksquare\}^T$
- Consecutive \blacksquare separated by $\square^{[a,b]}$, i.e., $B = \square^*(\blacksquare \square^{[a,b]})^{n, \infty} \{\blacksquare, \epsilon\} \square^*$

The DPP language is regular. Let C be an FSA such that $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$. For example, here is the minimal automaton for $T = 13, a = 3, b = 5, n = 2$.



This automaton for $\mathcal{L}(C)$ can grow very large, and we may only need to sample a small sublanguage with probabilistic support. Question: Can we incrementally sample $\mathcal{L}(C)$ while ensuring there is always a valid rightwards continuation?



McGill
UNIVERSITY

