Limitations of Autoregressive Models and Their Alternatives

Chu-Cheng Lin*[♯], Aaron Jaech[♭], Xin Li[♯], Matt Gormley[♮], Jason Eisner[♯]

*kitsing@cs.jhu.edu

[♯]Johns Hopkins University
[♭]Facebook AI
[♮]Carnegie Mellon University

Commonly held beliefs:

“RNN language models are Turing-complete. So they can model any computable language!”
“RNNs can fit any finite language. If they do not fit, just add more parameters!”

This work:

Not really! Even with unlimited compute/annotation during training, there is a distribution over strings, that cannot be fit by any autoregressive model (e.g., RNN/Transformer), even if you allow longer strings to use larger models (with polynomial growth).
But this language can be easily “fit” by a short hand-written Python program!

P:

a decision problem class. It is the set of all languages that can be decided in polynomial time.

Efficiently Computable (EC):

an abstraction of **Energy-Based Models (EBMs).**
A normalizable efficiently computable weighted language defines $p(\mathbf{x}) \propto \tilde{p}(\mathbf{x})$ where $\tilde{p}(\mathbf{x})$ can be computed in $O(\text{poly}(|\mathbf{x}|))$.
Their support can be (and can only be) anything in P.

Efficiently Locally Normalized (ELN):

an abstraction of **Autoregressive Models (including ordinary RNNs/ LSTMs/Transformers/…)** They parametrize probability of string **x** as $p(\mathbf{x}) = \prod_t p(x_t \mid \mathbf{x}_{<t})$ with a fixed size parameter vector. Computing $p(x_t \mid \mathbf{x}_{<t})$ takes $O(\text{poly}(t))$.

Efficiently Locally Normalizable with Compact Parameters (ELNCP):

a generalization of ELNs. An ELNCP model has infinitely many parameter vectors. When $|\mathbf{x}| = n$, an ELCNP model uses parameters θ_n to compute $p(\mathbf{x}) = \prod_t p(x_t \mid \mathbf{x}_{<t})$. They provide a conceptual upper bound to the just-train-a-slightly-larger-model paradigm for autoregressive models.
ELNCP weighted languages can have support outside of P because of the precomputed parameters.

Why is it bad that ELNCP models can't decide all languages in P?

Because then they can't choose among continuations of a prompt. That is, there's no way to ensure that p(**x#y**) > 0 iff **y** is a valid continuation of prompt **x**, even if that property can be checked in polytime.

P/poly:

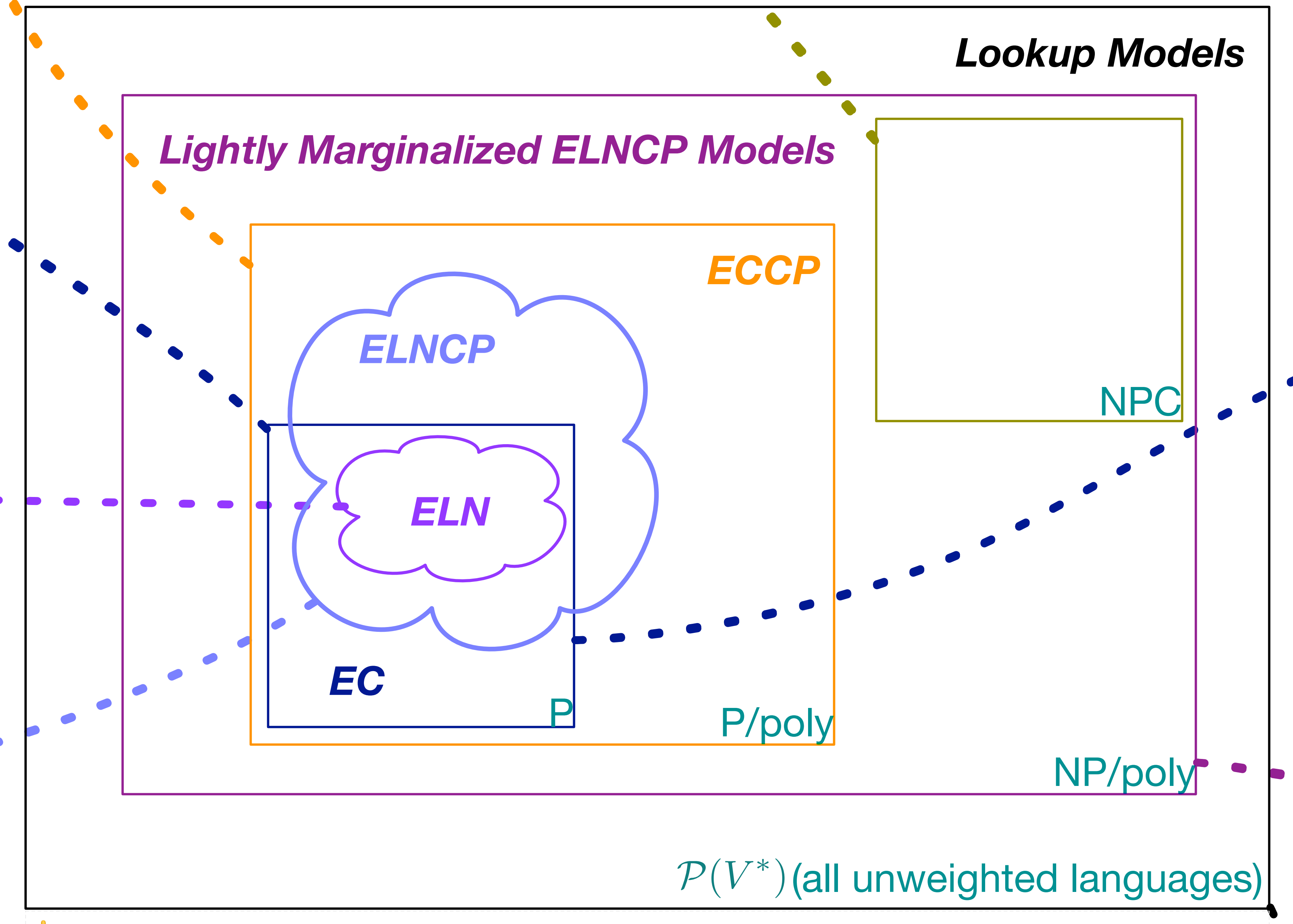
P with the help of **poly-sized advice strings** that can come from an oracle. P/poly is therefore more powerful than P — they can model undecidable problems due to the oracle access!

Efficiently Computable with Compact Parameters (ECCP):

is a generalization of ECs. Similar to ELNCPs, ECCPs is a conceptual upper bound to the just-train-a-slightly-larger-model paradigm of EBMs.

NP-complete (NPC):

is a set of languages that are widely believed to be outside P/poly (and therefore cannot be support of ECCP languages)


Lookup Models
Lightly Marginalized ELNCP Models
ECCP
ELNCP
ELN
EC
P
P/poly
NPC
NP/poly
$\mathcal{P}(V^*)$ (all unweighted languages)

👆The space of unweighted languages. Each rectangular outline corresponds to a complexity class and encloses the languages whose decision problems fall into that class. Each shape (whose name is colored to match the shape outline) corresponds to a model family and encloses the languages that can be expressed as the support of some *weighted* language in that family.

Future work:

Average-case analysis?
Are there model families that have all the good stuff but none of the bad stuff?

The sequence order problem:

consider the distribution
$p(\underbrace{\mathbf{x}}_{\text{problem encoding}} \# \underbrace{\mathbf{y}}_{\text{solution candidate}})$
where $p(\mathbf{x}\#\mathbf{y}) \neq 0$ iff **y** is a solution to **x**.

The **prefix probability** $p(\mathbf{x}\#) > 0$ if and only if **x** has a solution.
In other words, autoregressive models that factor $p(\mathbf{x}\#\mathbf{y}) = p(\mathbf{x}\#) \cdot p(\mathbf{y} \mid \mathbf{x}\#)$ **must** have the capacity to decide whether **x** has a solution, to ensure the joint distribution is accurate. If **x** is *hard enough* (e.g. NP-hard), no autoregressive models can even get the support right, as long as they use polytime/polysize (i.e. ELN/ELNCPs)!
The other sequence order does fine under autoregressive models (if **x** is in NP):
$p(\underbrace{\mathbf{y}}_{\text{solution candidate}} \# \underbrace{\mathbf{x}}_{\text{problem encoding}}) = p(\mathbf{y}\#) \cdot p(\mathbf{x} \mid \mathbf{y}\#)$
But we don’t always get to decide the sequence order 😢

Fix #1: use EBMs

EBMs do not suffer the sequence order problem because they don't even try to compute the possibly expensive factors $p(x_t \mid \mathbf{x}_{<t})$!
Downside: it is not easy to sample from EBMs. Training them requires estimating the partition function.

Fix #2: marginalize

A Lightly marginalized ELNCP model marginalizes over an ELNCP language (*lightly so* because it des not have too many latent variables). The sequence of latent and observed symbols can be sampled from the ELNCP model.
Intuitively, they avoid the *sequence order problem* with latent variables:
$p(\underbrace{\mathbf{x}}_{\text{problem encoding}} \# \underbrace{\mathbf{y}}_{\text{solution candidate}}) = p(\underbrace{\mathbf{y}}_{\text{latent variable}} \#) \cdot p(\underbrace{\mathbf{x}}_{\text{problem encoding}} \mid \underbrace{\mathbf{y}}_{\text{latent variable}} \#)$
They can have *any* language in NP/Poly as support!
$p(\underbrace{\mathbf{x}}_{\text{problem encoding}}) = \sum_{\mathbf{y} \in V^*} p(\underbrace{\mathbf{y}}_{\text{latent variable}} \#) \cdot p(\underbrace{\mathbf{x}}_{\text{problem encoding}} \mid \underbrace{\mathbf{y}}_{\text{latent variable}} \#)$
Downside: marginalization is required even at test time.

Fix #3: memorize anything we need

We can model anything if we have a big big database!
Examples: kNNLM, adaptive semi parametric language models, …
Downside: Need a vast database of observed or precomputed answers.

Model family	Compact parameters?	Efficient scoring?	Efficient sampling and normalization?	Support can be . . .
ELN/ELNCP: Autoregressive models (§3.1)	✓	✓	✓	*some but not all L* ∈ P
EC/ECCP: Energy-based models (§4.1)	✓	✓	✗	*all L* ∈ P but *no L* ∈ NPC
Lightly marginalized ELNCP: Latent-variable autoregressive models (§4.2)	✓	✗	✓	*all L* ∈ NP
Lookup models (§4.3)	✗	✓	✓	*anything*