# Global Autoregressive Models for Data-Efficient Sequence Learning

Tetiana Parshakova, Jean-Marc Andreoli, Marc Dymetman
tetianap@stanford.edu; {jean-marc.andreoli,marc.dymetman}@naverlabs.com

## Problem

- Standard seq2seq autoregressive models [2, 3] are trained by through **local supervision** (NLL loss relative to next observed symbol) and are **myopic** to global properties of the generated sequences.
  - Can be lost in test conditions not seen during training (**observation bias**).
  - Can produce situations not encountered in training data: repetition of words, hallucinations, non-existent words, syntactic/semantic inconsistencies, etc.
- The problem is more severe under limited training data conditions and can lead to bad test performance → poor sample-efficiency .
- **Can we exploit global properties observed on training data to improve sample efficiency ?**

## Contributions

- **Definition of GAMs.** We formulate seq2seq training based on an *Energy-Based Model (EBM)* [1] of a specific kind, a GAM: a combination of a standard Autoregressive component $r$ and of a Log-Linear component over a set of features. The combination is trained by log-likelihood over the training data $D$.
- **Adressing training challenges:**
  1. Training-1: **Training the EBM by log-likelihood** over $D$. We use either Rejection Sampling (rs) or Self-Normalized Importance Sampling (snis). At the end of this phase, we obtain an *unnormalized* EBM, which better represents the data than the initial autoregressive component, but which can not be used directly for inference.
  2. Training-2: **Distillation of the EBM** obtained in Training-1, using RS, to obtain an "augmented" training set $D'$, with $|D'| \gg |D|$, from which we train a final autoregressive model $\pi_\theta$, which can be used for inference or for computing sequence probabilities.
- **Sample Efficiency** Experiments on synthetic data showing better test performance of $\pi_\theta$ over $r$.

## GAMs: Global Autoregressive Models

A GAM is an unnormalized potential (aka EBM) $P_\eta(x|C)$ over $x$, parametrized by a vector $\eta = \eta_1 \oplus \eta_2$:

$$P_\eta(x|C) = r_{\eta_1}(x|C) \cdot e^{\langle \lambda_{\eta_2}(C),\ \phi(x;C) \rangle}. \tag{1}$$

The factor $r_{\eta_1}(x|C)$ is an autoregressive model for generating $x$ in the context $C$, parametrized by $\eta_1$. The factor $e^{\langle \lambda_{\eta_2}(C),\ \phi(x;C) \rangle}$ is a *log-linear* potential, where $\phi(x;C)$ is a vector of predefined features and $\lambda_{\eta_2}(C)$ a vector of reals, computed by a network parametrized by $\eta_2$. The normalized distribution associated with the GAM is $p_\eta(x|C) = \frac{P_\eta(x|C)}{Z_\eta(C)}$, where $Z_\eta(C) = \sum_x P_\eta(x|C)$.
The features $\phi(x;C)$ provide **prior knowledge** to the model by drawing its attention to potentially useful global sequence properties that may be difficult for the AM component to discover on its own.
In our experiments, we focus on a simple unconditional (language modelling) version of GAMs, of the form:

$$P_\lambda(x) \doteq r(x) \cdot e^{\langle \lambda,\ \phi(x) \rangle} \tag{2}$$

where the autoregressive factor $r = r_{\eta_1}$ is first learnt on the training dataset of sequences $D$ and then kept fixed, and where the parameter vector $\lambda$ is then trained on top of $r$, also on $D$. We denote by $p_\lambda(x)$ the normalized distribution associated with $P_\lambda(x)$.

## References

[1] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A Tutorial on Energy-Based Learning. 2006.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 1997.

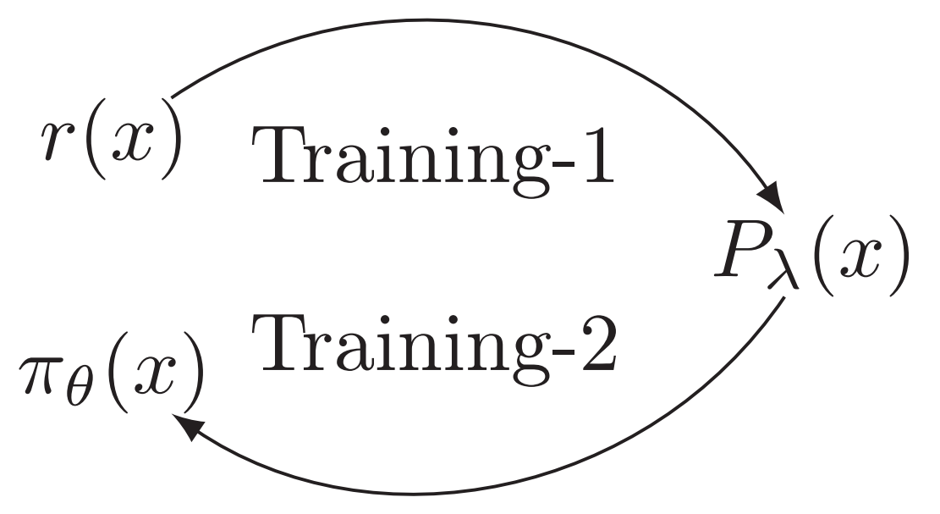
[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[4] Tony Jebara. Log-Linear Models, Logistic Regression and Conditional Random Fields, 2013.

[5] Y. Bengio and J. S. Senecal. Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model. 2008.

[6] Tetiana Parshakova, Jean-Marc Andreoli, and Marc Dymetman. Distributional Reinforcement Learning for Energy-Based Sequential Models (to appear: OptRL WS, Neurips, Dec. 2019).

## Two-Stage Training

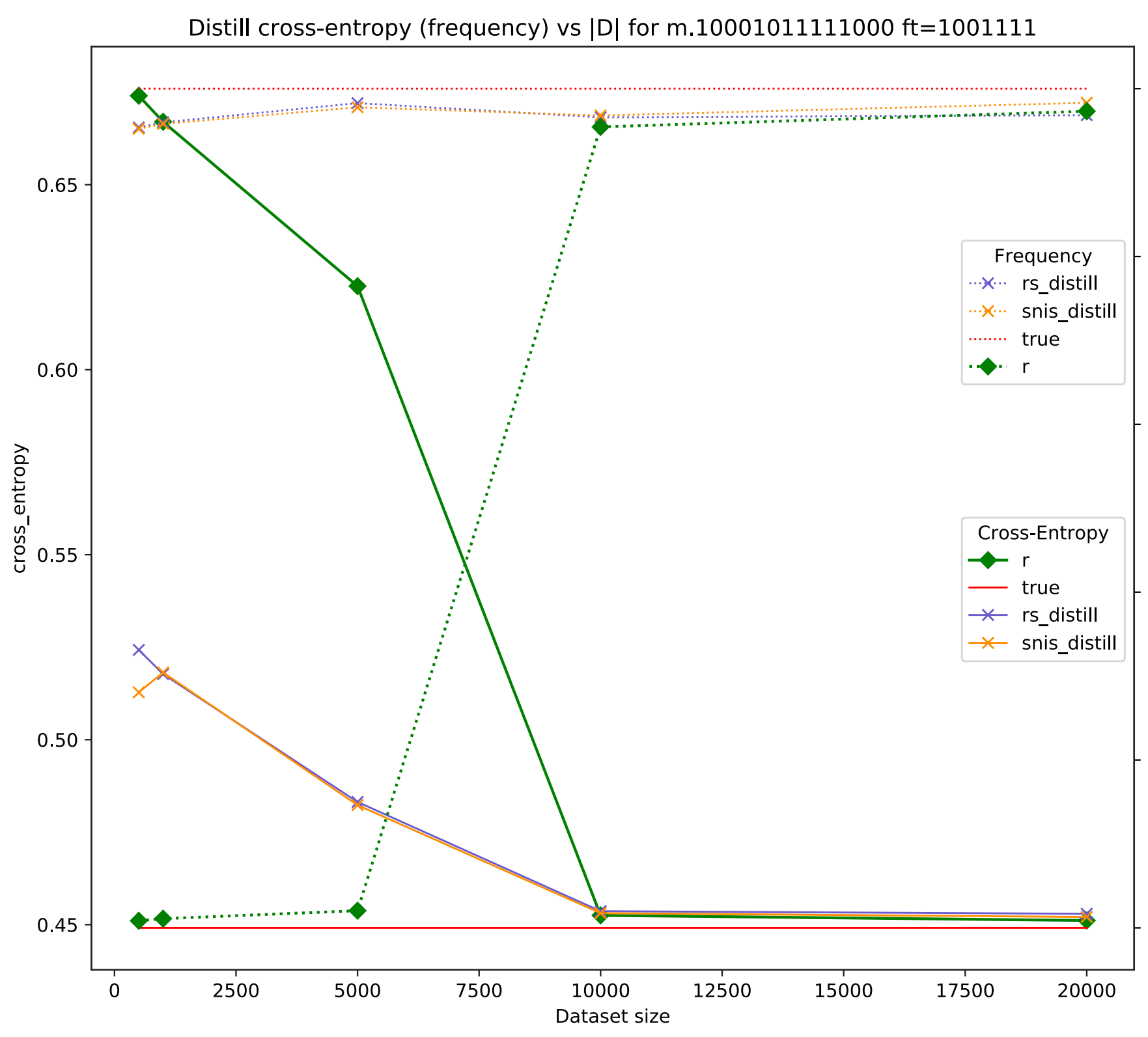


- Training-1: Exploit "moment-matching" property of Log-Linear Models (Exponential Family):

$$\nabla_\lambda \log p_\lambda(x) = \phi(x) - E_{x \sim p_\lambda(\cdot)}\ \phi(x)$$

  - Issue: we do not directly have $p_\lambda$, but only $P_\lambda$. We resort either to Rejection Sampling (rs) from $P_\lambda$, or (more general) to Self-Normalized Importance Sampling (snis) [5], using $r$ as our proposal function.
- Training-2: Obtain large $D'$ by rejection sampling from $P_\lambda$, then distill $\pi_\theta(x)$ from $D'$.

## Experiments: strings with motifs



- Here, $D$ consists of random binary strings (of length 30), filtered by the condition that they contain a fixed motif 10001011111000
- Features: $m$ (binary feature $m = 0$ means motif is present), $d_0, d_1, d_2, d_3$ are "distractor" binary features, with small correlation with presence of motif (0 means feature is present).
- rs_distill (resp. snis_distill) is the $\pi_\theta$ obtained using rs (resp. snis) in training-1, and distillation in training-2.
- We vary $D$ and observe CE (i.e. perplexity) on test data (solid lines), and also motif frequency in samples from different models (dotted lines).

## Illustration

| | |
|---|---|
| *true* | 101**10001011111000**1000001001001 |
| $r$ | 0111110$\underline{00010111111000}$1110001011 |
| $\pi_\theta$ | 111010**10001011111000**0111111100 |
| features | $[m, \_, \_, d_0, d_1, d_2, d_3]$ |
| $\lambda$'s | $[-10.1, \_, \_, -0.15, -0.06, 0.0, -0.14]$ |
| moments *true* | $[0.0, \_, \_, 0.47, 0.99, 1.0, 0.91]$ |
| moments $r$ | $[0.95, \_, \_, 0.53, 0.99, 1.0, 0.91]$ |
| moments $\pi_\theta$ | $[0.0006, \_, \_, 0.43, 0.99, 0.99, 0.91]$ |
| cross entropy (CE) | *true*: 0.45, $r$: 0.56, $\pi_\theta$: 0.47 |
| motif freqs | *true*: 1.0, $r$: 0.045, $\pi_\theta$: 0.959 |

## Results

| $\lvert D \rvert$ | m: $\frac{\mathrm{CE}(T,r)}{\mathrm{CE}(T,\pi_\theta)}$ | m: $\frac{\mathrm{CE}(T,\pi_\theta)}{H(p_{true})}$ | m: $\frac{\mathrm{mtf\_frq}(\pi_\theta)}{\mathrm{mtf\_frq}(r)}$ | mam: $\frac{\mathrm{CE}(T,r)}{\mathrm{CE}(T,\pi_\theta)}$ | mam: $\frac{\mathrm{CE}(T,\pi_\theta)}{H(p_{true})}$ | mam: $\frac{\mathrm{mtf\_frq}(\pi_\theta)}{\mathrm{mtf\_frq}(r)}$ |
|---|---|---|---|---|---|---|
| 500 | $1.24 \pm 0.07$ | $1.19 \pm 0.07$ | 32.0 | $1.23 \pm 0.03$ | $1.16 \pm 0.03$ | 59.26 |
| 1000 | $1.24 \pm 0.07$ | $1.16 \pm 0.07$ | 23.87 | $1.21 \pm 0.03$ | $1.14 \pm 0.03$ | 26.29 |
| 5000 | $1.18 \pm 0.08$ | $1.09 \pm 0.05$ | 3.59 | $1.16 \pm 0.05$ | $1.08 \pm 0.04$ | 7.32 |
| 10000 | $1.08 \pm 0.1$ | $1.04 \pm 0.02$ | 0.89 | $1.02 \pm 0.03$ | $1.04 \pm 0.03$ | 1.0 |
| 20000 | $0.99 \pm 0.01$ | $1.02 \pm 0.01$ | 0.81 | $0.99 \pm 0.0$ | $1.02 \pm 0.0$ | 0.85 |

## Discussion

- **Training-1 vs. Training-2**
  - Training-2 can be more difficult than Training-1.
  - In some extreme cases, the EBM obtained at the end of Training-1 is (i) a perfect representation of the true process, but (ii) cannot be approximated by an autoregressive model.
- **Connections with Reinforcement Learning** (developped in [6])
  - The unnormalized EBM $P_\lambda$ can be seen as a form of **reward**, and Training-2 as a form of "Distributional" Reinforcement Learning.
  - Training-1 can then be seen as a form of **Inverse RL**, but where the reward is obtained through max-likelihood, rather than being externally imposed. We still have prior knowledge, but *only* in terms of the features that we suggest the model to observe.