Google

Two Ideas For Structured Data:

- Reward augmented maximum likelihood
- Order matters

Samy Bengio, and the Brain team

Google

Reward augmented maximum likelihood for neural structured prediction

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly,

Mike Schuster, Yonghui Wu, Dale Schuurmans

[NIPS 2016]

Structured prediction

Prediction of complex outputs:

- Image captioning

$\mathbf{x}$ $\mathbf{y}^*$

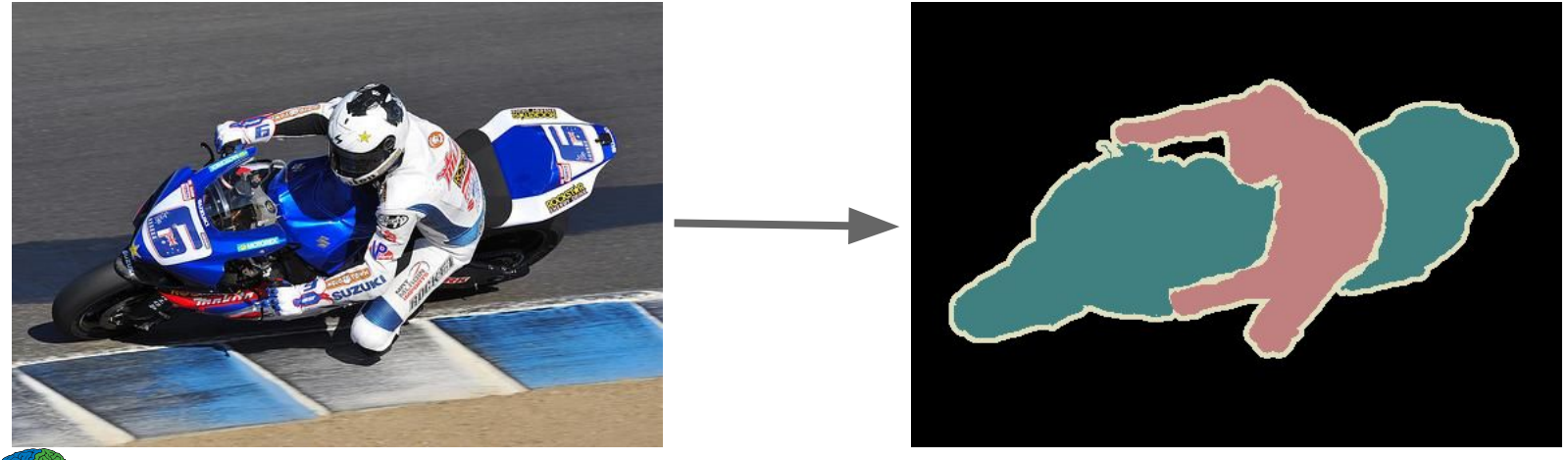
A dog and a cat lying in bed next to each other.

Structured prediction

Prediction of complex outputs:

- Image captioning
- Semantic segmentation

$\mathbf{x}$ $\mathbf{y}^*$



Google

Structured prediction

Prediction of complex outputs:

- Image captioning
- Semantic segmentation
- Speech recognition
- Machine translation

multivariate, correlated, constrained, discrete

$\mathbf{x}$

As diets change, people get bigger but plane seating has not radically changed.

$\mathbf{y}^*$

Comme les habitudes alimentaires changent, les gens grossissent, mais les sièges dans les avions n'ont pas radicalement changé.

Google

Reward function

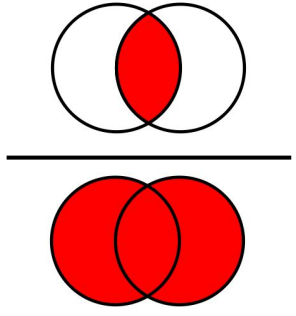
Reward is negative loss

In classification, we use 0/1 reward,

$$r(\mathbf{y}, \mathbf{y}^*) = \mathbb{1}[\mathbf{y} = \mathbf{y}^*]$$

In segmentation, we use intersection over union,

$$r(\mathbf{y}, \mathbf{y}^*) = \cap(\mathbf{y}, \mathbf{y}^*) / \cup(\mathbf{y}, \mathbf{y}^*)$$



In speech recognition, we use edit distance or WER

In machine translation, we use BLEU score

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

Structured prediction problem

Given a dataset of input output pairs $\mathcal{D} \equiv \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)*})\}_{i=1}^{N}$

learn a conditional distribution $p_\theta(\mathbf{y} \mid \mathbf{x})$

Approximate inference using *beam search*

such that model's predictions, $\widehat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$

achieve a large empirical reward: $\sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} r(\widehat{\mathbf{y}}(\mathbf{x}), \mathbf{y}^*)$

Performance measure

Probabilistic structured prediction

Chain rule to build a locally-normalized model:

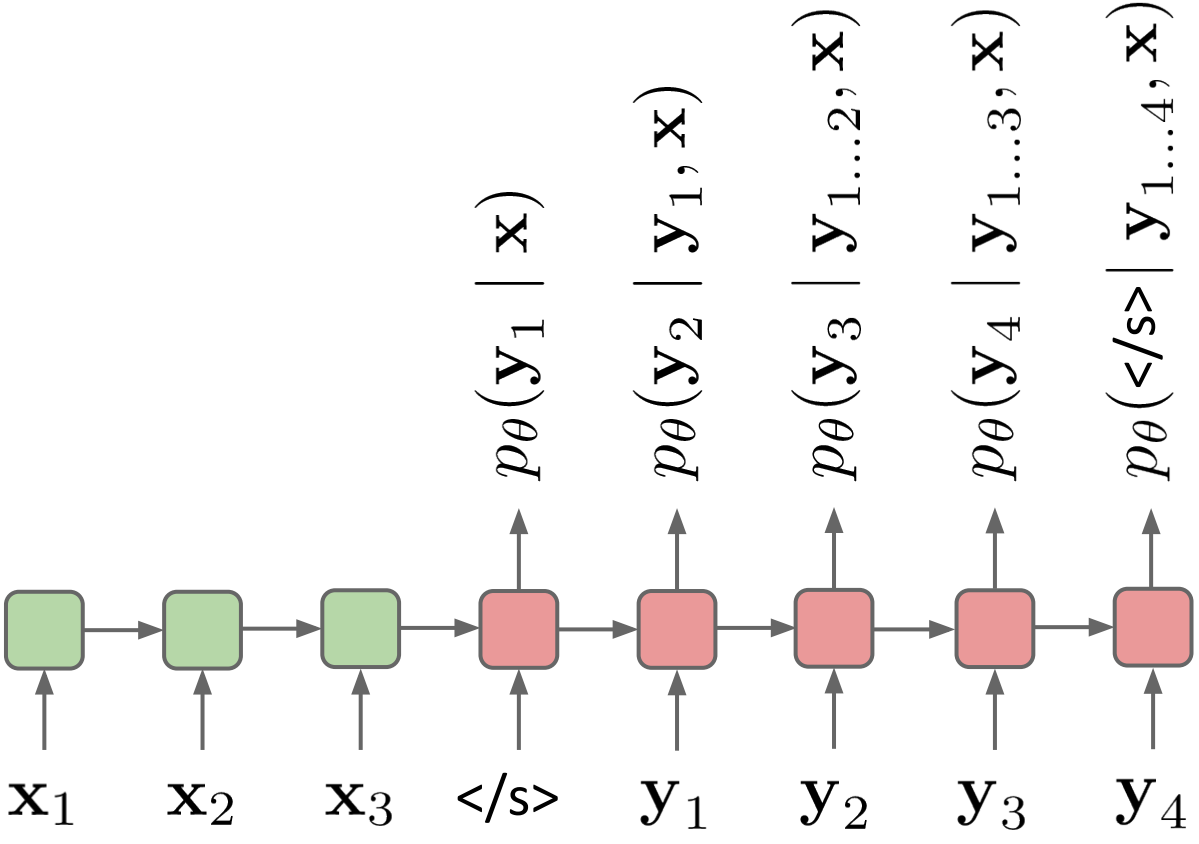
$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_i p_\theta(\mathbf{y}_i \mid \mathbf{y}_{1\ldots(i-1)}, \mathbf{x})$$

Globally normalized models...

Neural sequence models

[Sutskever, Vinyals, Le, 2014]

[Bahdanau, Cho, Bengio, 2014]

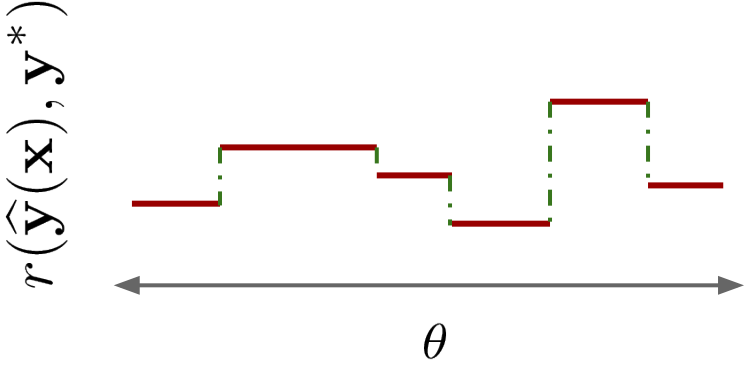


Google

Empirical reward is *discontinuous* and *piecewise constant*

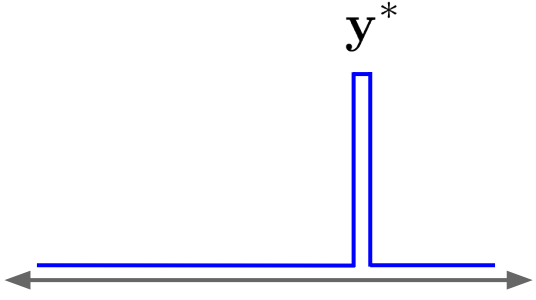
$$\sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} r(\widehat{\mathbf{y}}(\mathbf{x}), \mathbf{y}^*)$$

$$\widehat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x})$$



Maximum-likelihood objective

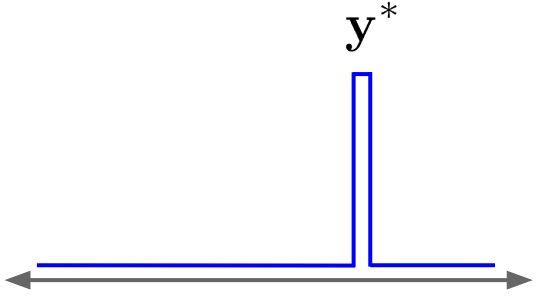
$$\mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}) = \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} -\log p_\theta(\mathbf{y}^* \mid \mathbf{x})$$



Key problems:
- There is no notion of reward
- Does not capture the inherent ambiguity of the problem

Expected reward (RL) [Ranzato et al, 2015]

$$\mathcal{L}_{\mathrm{RL}}(\boldsymbol{\theta}) = \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} -\sum_{\mathbf{y}\in\mathcal{Y}} p_\theta(\mathbf{y} \mid x)\; r(\mathbf{y},\mathbf{y}^*)$$



+ There is a notion of reward
- Hard to train because most samples yield low rewards
- Still, does not capture the inherent ambiguity of the problem

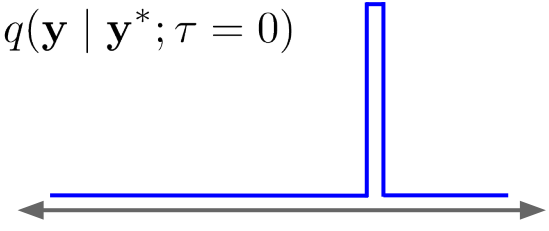
Reward augmented maximum likelihood (RML)

$$\mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta}; \tau) = \sum_{(\mathbf{x}, \mathbf{y}^*) \in \mathcal{D}} \left\{ - \sum_{\mathbf{y} \in \mathcal{Y}} q(\mathbf{y} \mid \mathbf{y}^*; \tau) \log p_\theta(\mathbf{y} \mid \mathbf{x}) \right\}$$

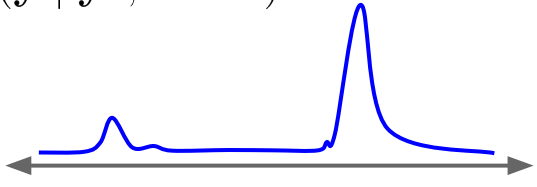
$$q(\mathbf{y} \mid \mathbf{y}^*; \tau) = \frac{1}{Z(\mathbf{y}^*, \tau)} \exp\{r(\mathbf{y}, \mathbf{y}^*)/\tau\}$$

Temperature hyperparameter τ:

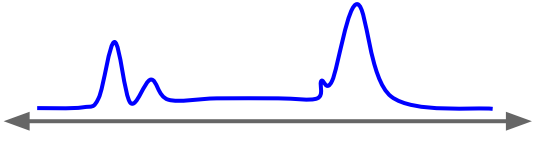
$q(\mathbf{y} \mid \mathbf{y}^*; \tau = 0)$



$q(\mathbf{y} \mid \mathbf{y}^*; \tau = .1)$



$q(\mathbf{y} \mid \mathbf{y}^*; \tau = .2)$



Google

Reward augmented maximum likelihood (RML)

$$\mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta};\tau) = \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} \left\{ -\sum_{\mathbf{y}\in\mathcal{Y}} q(\mathbf{y} \mid \mathbf{y}^*;\tau) \log p_\theta(\mathbf{y} \mid \mathbf{x}) \right\}$$

$$q(\mathbf{y} \mid \mathbf{y}^*;\tau) = \frac{1}{Z(\mathbf{y}^*,\tau)} \exp\{r(\mathbf{y},\mathbf{y}^*)/\tau\}$$

+ There is a notion of reward and ambiguity
+ Supervised labels are fully exploited
+ Simpler optimization requiring stationary samples from *q*

Google

Reward augmented maximum likelihood (RML)

$$\mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta}; \tau) = \sum_{(\mathbf{x}, \mathbf{y}^*) \in \mathcal{D}} \left\{ - \sum_{\mathbf{y} \in \mathcal{Y}} q(\mathbf{y} \mid \mathbf{y}^*; \tau) \log p_\theta(\mathbf{y} \mid \mathbf{x}) \right\}$$

$$q(\mathbf{y} \mid \mathbf{y}^*; \tau) = \frac{1}{Z(\mathbf{y}^*, \tau)} \exp\{r(\mathbf{y}, \mathbf{y}^*)/\tau\}$$

SGD update for RML?

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{RML}}(\boldsymbol{\theta}; \tau) \;=\; \mathbb{E}_{q(\mathbf{y} \mid \mathbf{y}^{(i)*}; \tau)} \left[- \nabla_{\boldsymbol{\theta}} \log p_\theta(\mathbf{y} \mid \mathbf{x}^{(i)}) \right]$$

Sampling from exponentiated payoff distribution

Stratified sampling from Hamming reward:

Hamm. distance	count	exponentiated payoff
0	1	$\exp\{0/\tau\}$
1	$\binom{n}{1}(v-1)$	$\exp\{-1/\tau\}$
$\vdots$	$\vdots$	$\vdots$
k	$\binom{n}{k}(v-1)^k$	$\exp\{-k/\tau\}$
$\vdots$	$\vdots$	$\vdots$
n	$(v-1)^n$	$\exp\{-n/\tau\}$

Sampling from Edit Distance is a bit more involving (variable size) but feasible.

Sampling from BLEU: first sample from Hamming or edit distance, then apply an importance correction (i.e. **importance sampling**)

TIMIT experiments

Standard benchmark for clean phone recognition

630 speakers, each speaking 10 phonetically-rich sentences

Training from scratch either using ML or RML. Attention-based sequence to sequence model with 3 encoder layers and 1 decoder layer with 256 LSTM cells

Edit distance sampling in the phone space - 60 phones

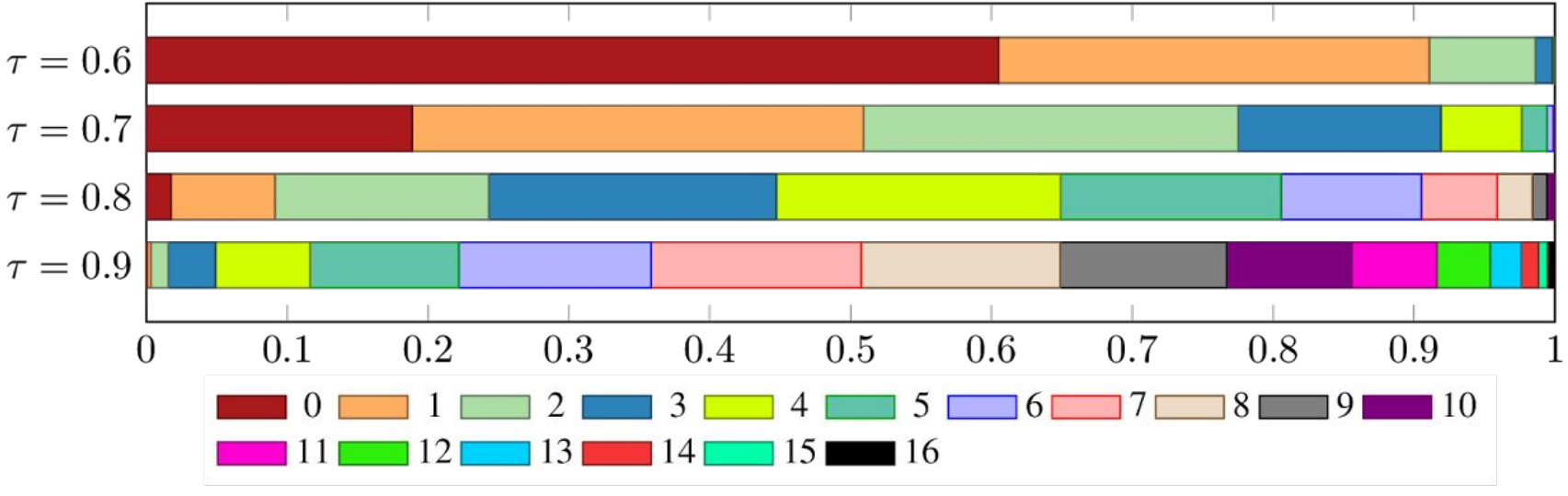
Reporting average of 4 independent runs (train / dev/ test sets)

Timit results (phone error rates, lower is better)

Method	Dev set	Test set
ML baseline	20.87 (−0.2, +0.3)	22.18 (−0.4, +0.2)
RML, $\tau = 0.60$	19.92 (−0.6, +0.3)	21.65 (−0.5, +0.4)
RML, $\tau = 0.65$	19.64 (−0.2, +0.5)	21.28 (−0.6, +0.4)
RML, $\tau = 0.70$	18.97 (−0.1, +0.1)	21.28 (−0.5, +0.4)
RML, $\tau = 0.75$	18.44 (−0.4, +0.4)	20.15 (−0.4, +0.4)
RML, $\tau = 0.80$	18.27 (−0.2, +0.1)	19.97 (−0.1, +0.2)
RML, $\tau = 0.85$	18.10 (−0.4, +0.3)	19.97 (−0.3, +0.2)
RML, $\tau = 0.90$	**18.00** (−0.4, +0.3)	**19.89** (−0.4, +0.7)
RML, $\tau = 0.95$	18.46 (−0.1, +0.1)	20.12 (−0.2, +0.1)
RML, $\tau = 1.00$	18.78 (−0.6, +0.8)	20.41 (−0.2, +0.5)

Timit results

Fraction of different number of edits applied to a sequence of length 20 for different τ



WMT’14 En-Fr experiments

English to French translation.

Training with 36M sentence pairs. Test with 3003 newstest-14 set.

Training from scratch either using *ML* or *RML*. Attention-based sequence to sequence model using three-layer encoder and decoder networks with layers of 1024 LSTM cells. Vocabulary of 80k words in the target and 120k in the source

Sampling based on Hamming reward

Handle rare words by copying from source according to attention

WMT'14 En-Fr results (higher is better)

Method	Average BLEU	Best BLEU
ML baseline	36.50	36.87
RML, $\tau = 0.75$	36.62	36.91
RML, $\tau = 0.80$	36.80	37.11
RML, $\tau = 0.85$	**36.91**	**37.23**
RML, $\tau = 0.90$	36.69	37.07
RML, $\tau = 0.95$	36.57	36.94

Order Matters: Sequence To Sequence For Sets

Oriol Vinyals, Samy Bengio, Manjunath Kudlur

[ICLR 2016]

Sequences in Machine Learning

- Sequences are common in many ML problems:
  - Speech recognition
  - Machine translation
  - Question answering
  - Image captioning
  - Sentence parsing
  - Time-series prediction
- Not always “aligned”:
  - Sometimes, examples are of the form

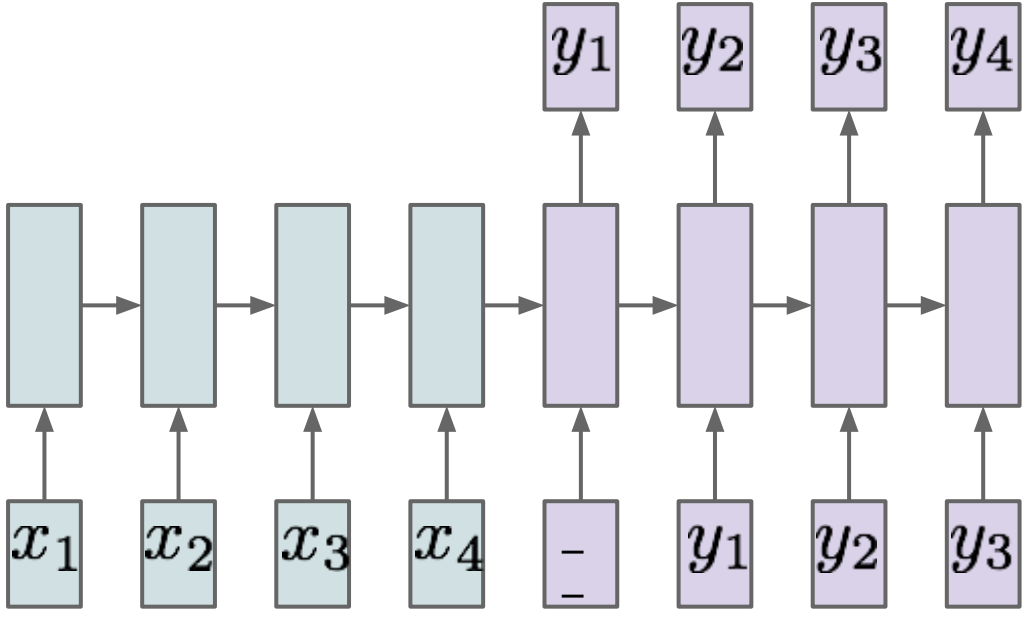
$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_T, y_T)\}$$

  - But sometimes there are of the form

$$\{x_1, x_2, \cdots, x_T, y_1, y_2, \cdots, y_{T'}\}$$

The Sequence-to-Sequence Framework [Sutskever, et al, 2014]

$$p(y_1, \cdots, y_{T'} | x_1, \cdots, x_T) = \prod_{t=1}^{T'} p(y_t | y_1, \cdots, y_{t-1}, x_1, \cdots, x_T)$$

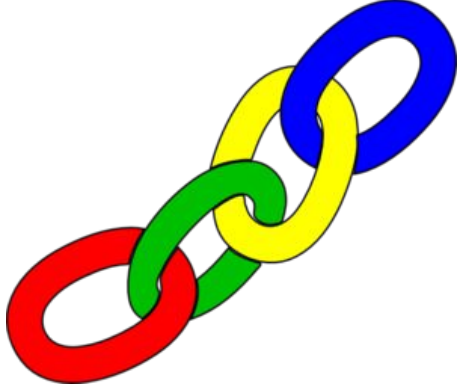


Google

Some Examples Applying Sequence-to-Sequence

- Machine Translation [Kalchbrenner et al, EMNLP 2013][Cho et al, EMLP 2014][Sutskever & Vinyals & Le, NIPS 2014][Luong et al, ACL 2015][Bahdanau et al, ICLR 2015]
- Image captions [Mao et al, ICLR 2015][Vinyals et al, CVPR 2015][Donahue et al, CVPR 2015][Xu et al, ICML 2015]
- Speech [Chorowsky et al, NIPS DL 2014][Chan et al, ICASSP 2016]
- Parsing [Vinyals & Kaiser et al, arxiv 2014]
- Dialogue [Shang et al, ACL 2015][Sordoni et al, NAACL 2015][Vinyals & Le, ICML DL 2015]
- Video Generation [Srivastava et al, ICML 2015]
- Geometry [Vinyals & Fortunato & Jaitly, NIPS 2015]
- etc...

Main Ingredient: The Chain Rule



$$p(y_1, y_2) = p(y_2|y_1)p(y_1) = p(y_1|y_2)p(y_2)$$

$$p(y_1, \ldots, y_N) = \prod_{i=1}^{N} p(y_i|y_1, \ldots, y_{i-1})$$

What About Sets?

"Unordered collection of objects"

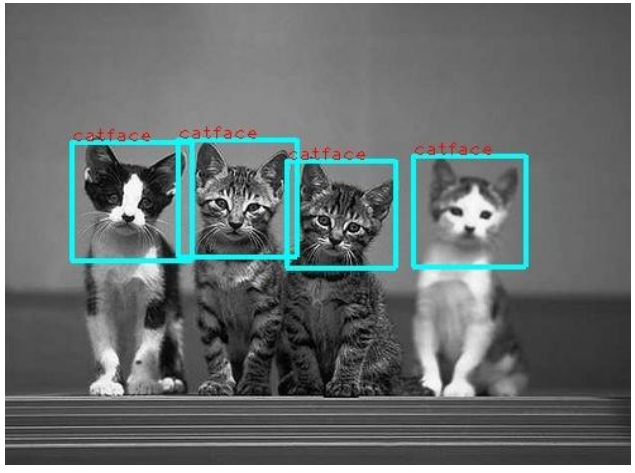
Challenge: $p(\{y_1, \ldots, y_N\}|X)$

Bad: $\prod_i p(y_i|X)$

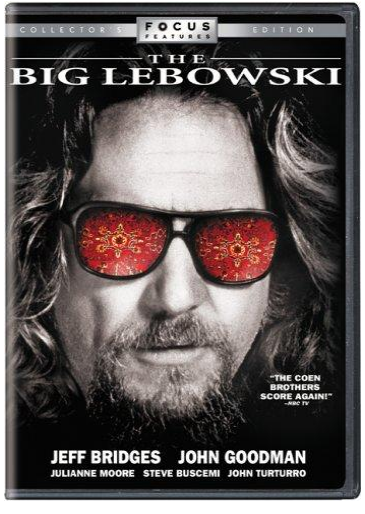
Less bad: $\prod_i p(y_i|y_1, \ldots, y_{i-1}|X)$

Examples Where Sets Appear

Image -> Set of Objects



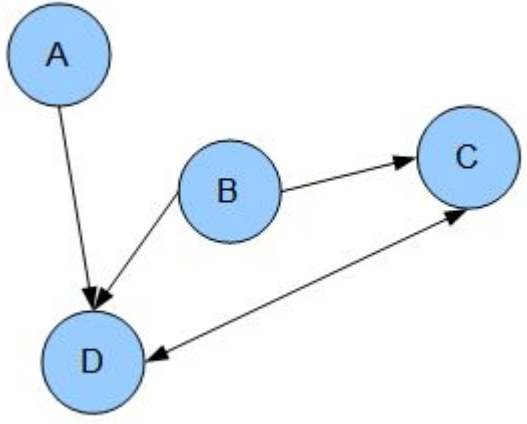
Video -> Actors



Google

More Examples of Sets

Random Variables
in a graphical model



3-SAT $(\mathbf{a} \vee \mathbf{b} \vee \neg\mathbf{c}) \wedge (\neg\mathbf{a} \vee \mathbf{c} \vee \neg\mathbf{d}) \wedge \ldots. \wedge (\neg\mathbf{b} \vee \neg\mathbf{c} \vee \mathbf{d})$

Sequences-as-Sets

The man with a hat

(a,4)
(The,1)
(hat,5)
(man,2)
(with,3)

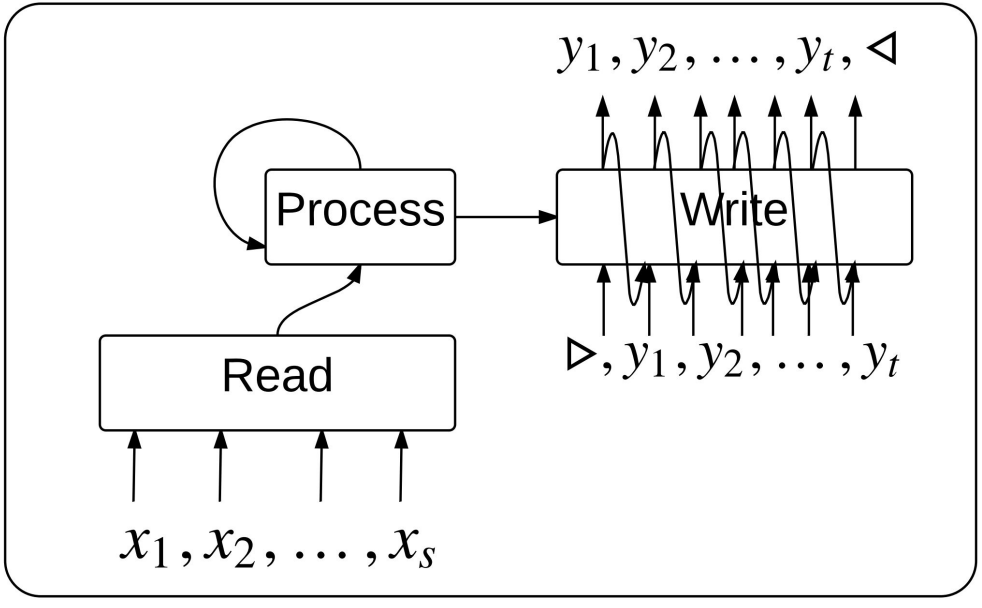
Input Order Matters - Examples

There is a lot of prior work showing that the order of input variables is important:

- Machine Translation
  - [Sutskever et al, 2014], translating from English to French
  - Reversing order of English words yielded improvement of up to 5 BLEU points
- Constituency Parsing
  - [Vinyals et al, 2015], from English sentence to flattened parse tree
  - Reversing order of English words yielded improvement of 0.5% F1 score
- Convex Hull
  - [Vinyals et al, 2015], from collection of points to its convex hull
  - Sorting points by their angle, yielded 10% improvement in most difficult cases

Read-Process-Write: Input Order Invariant Approach

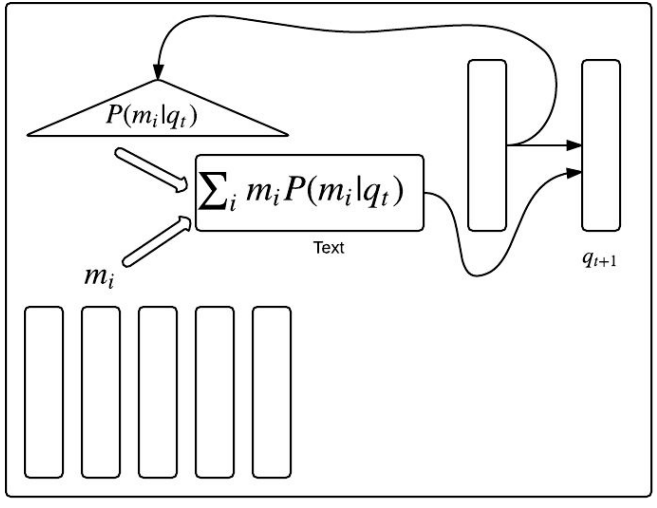
- **Reading block**:
  - Reads each input into memory, potentially in parallel
- **Process block**:
  - LSTM with no input nor output
  - Performs T steps of computation over the memory, using an attention mechanism [see next slide].
- **Writing block**:
  - LSTM (or Pointer Network)
  - Alternate between an attention step over the memory and outputting the relevant data, such as a pointer to the input memory
- **Related and recent**:
  - Adaptive Computation Time [Graves, 2016]
  - Encode, Review, Decode [Yang et al, 2016]



Attention Mechanism in the Process Block

At each step of Process, we do:

1. Get the next state of process
2. Compute a function of the state and each input memory
3. Softmax to get posteriors
4. Compute a weighted average input
5. Concatenate with the state of the process block and continue



$$q_t = \text{LSTM}(q^\star_{t-1})$$

$$e_{i,t} = f(m_i, q_t)$$

$$a_{i,t} = \frac{\exp(e_{i,t})}{\sum_j \exp(e_{j,t})}$$

$$r_t = \sum_i a_{i,t}\, m_i$$

$$q^\star_t = [q_t\ r_t]$$

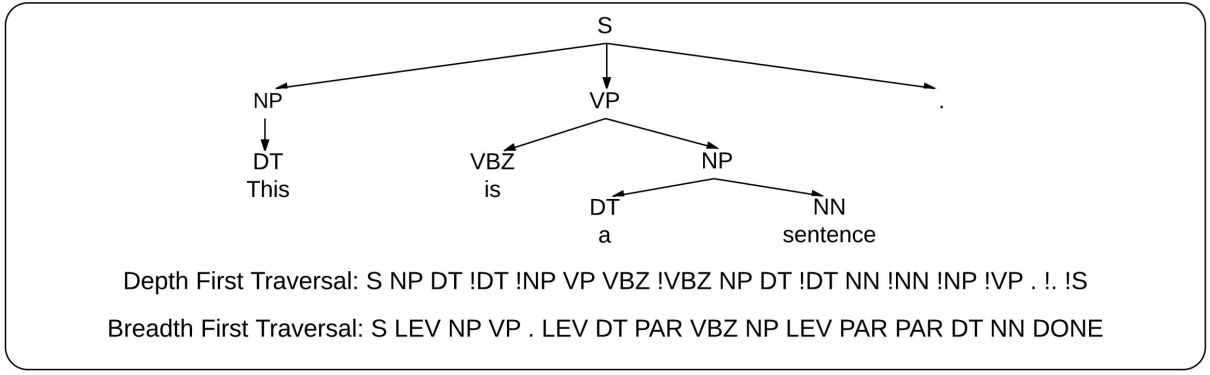
The Sorting Experiment

- Task: sort N unordered random floating point numbers (between 0 and 1)
- Compare Read-Process-Write with a vanilla Pointer Network
- Vary N the number of numbers to sort, and P, the number of process steps
- Also consider using a glimpse (attention step between each output step) or not
- 10000 training iterations
- Results: out-of-sample accuracy (either the set is fully sorted or not)

Length N	Ptr-Net		$P = 0$ step		$P = 1$ step		$P = 5$ steps		$P = 10$ steps	
glimpses	0	1	0	1	0	1	0	1	0	1
$N = 5$	81%	90%	65%	84%	84%	92%	88%	94%	90%	94%
$N = 10$	8%	28%	7%	30%	14%	44%	17%	57%	19%	50%
$N = 15$	0%	4%	1%	2%	0%	5%	2%	4%	0%	10%

Output Order Matters - Examples

- **Language Modeling**
  - Use an LSTM to maximize likelihood of sequence of words (PennTreeBank)
  - Consider these orderings and obtained perplexity on dev set:
    - Natural: “This is a sentence .” 86
    - Reverse: “. sentence a is This” 86
    - 3-word reversal: “a is This <pad> . sentence” 96
- **Constituency Parsing**
  - “Translate” between an English sentence and its flattened parse tree
  - Many ways to “flatten” a parse tree: for instance
    - depth-first obtained 89.5% F1
    - Breadth-first obtained 81.5% F1



Google

Finding Good Output Orderings While Training

- Sometimes, the optimal order of the output variables per example is unknown
- While training, we can explore all (or several) potential orderings per example
- So instead of fixing the ordering and train with:

$$\theta^\star = \arg\max_\theta \sum_i \log p(Y_i | X_i; \theta)$$

- We consider the best (or the best found) ordering:

$$\theta^\star = \arg\max_\theta \sum_i \max_{\pi(X_i)} \log p(Y_{\pi(X_i)} | X_i; \theta)$$

- Needs to pre-train the model with uniform exploration first
- After that, estimate the max by sampling from the model
- This is very similar to REINFORCE where we learn a policy over orderings
- Use the same procedure at inference.

Example with 5-gram Modeling

- Simplified task: model 5-grams with no context
- 5-gram (sequence): y1=This, y2=is, y3=a, y4=five, y5=gram
- 5-gram (set): y1=(This,1), y2=(is,2), y3=(a,3), y4=(five,4), y5=(gram,5)
- (1,2,3,4,5): train on the natural ordering
- (5,1,3,4,2): train on another ordering
- Easy: train on examples from (1, 2, 3, 4, 5) and (5, 1, 3, 4, 2), uniformly sampled.
- Hard: train on examples from the 5! possible orderings, uniformly sampled.

Task	Orders considered	Perplexity
$(1, 2, 3, 4, 5)$	1	225
$(5, 1, 3, 4, 2)$	1	280
Easy	2	225
Hard	$5!$	225

Conclusion

- The sequence-to-sequence framework is very powerful for sequences
- But what about unordered sets?
- In many cases, order matters! either for inputs or outputs sets
- For input sets, we can read them irrespective of their order and use an attention mechanism to combine them as many times as needed.
- For output sets, we can explore the space of possible ordering and favor the best ones per example, both at training and inference time.