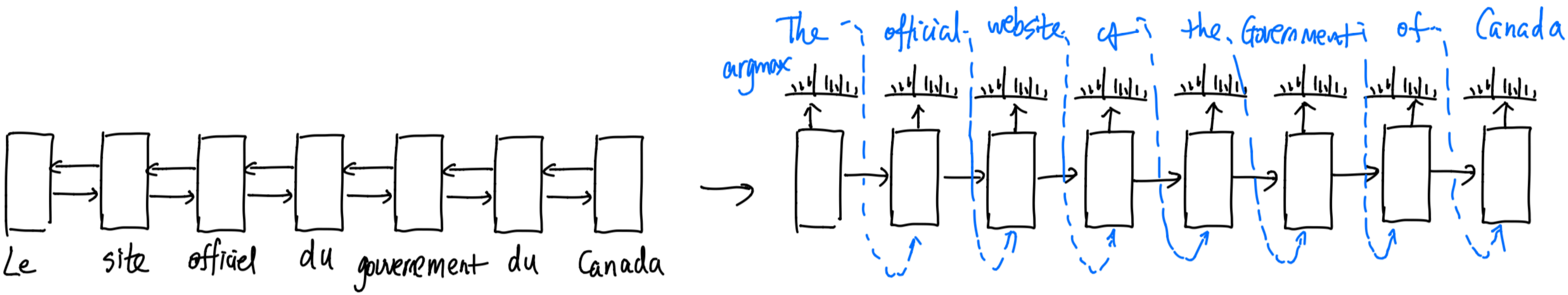
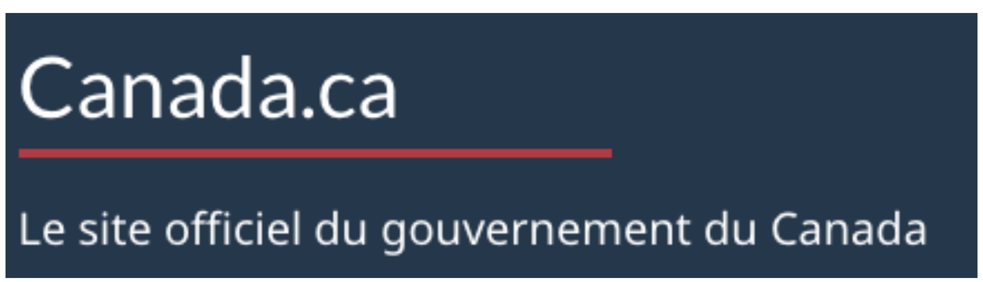
Seq2Seq Models
&
Attention Mechanism

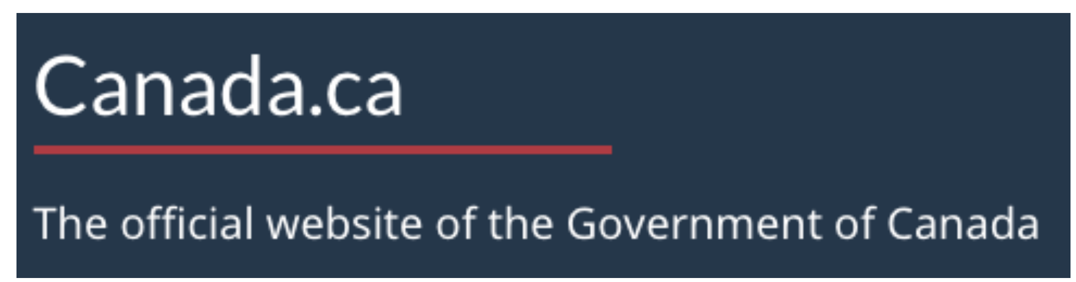
Lili Mou
lmou@ualberta.ca
lili-mou.github.io

UNIVERSITY OF ALBERTA

Seq2Seq







[Source: canda.ca]

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." In *NIPS*, 2014.

UNIVERSITY OF ALBERTA

Seq2Seq

- **Question:**

  Why do we feed back the generated words?

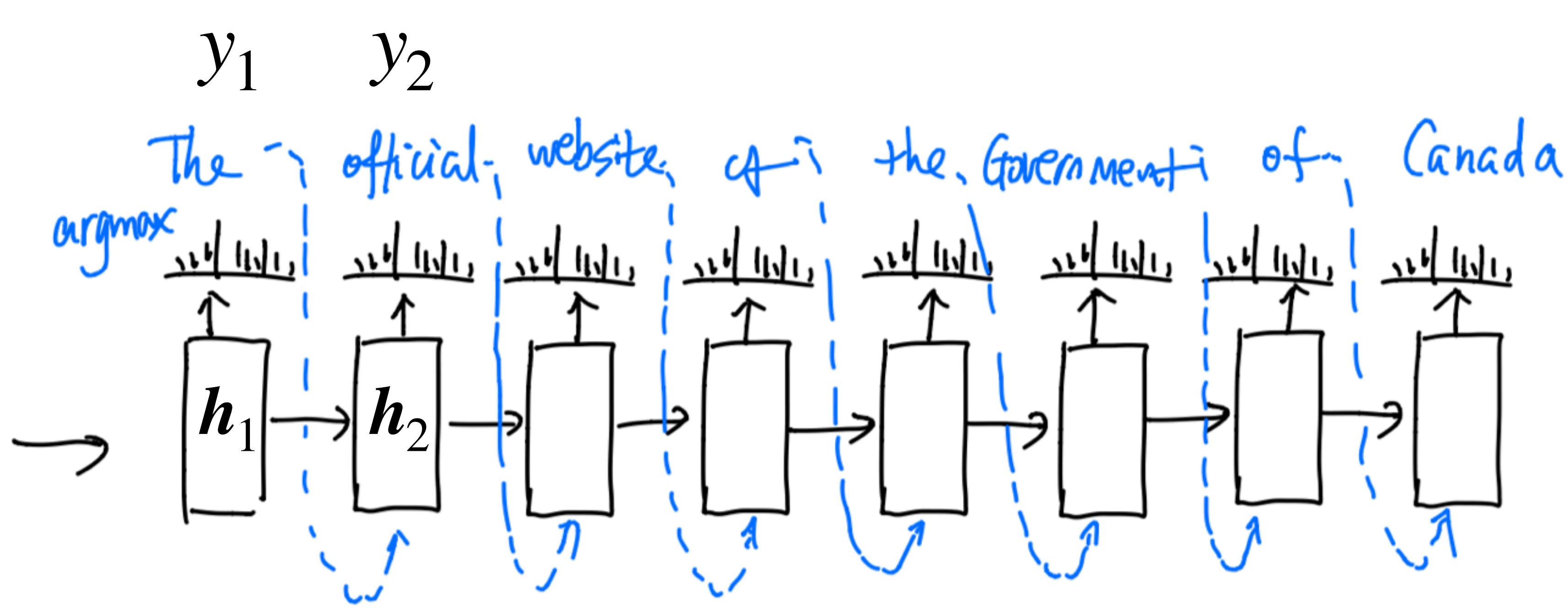
  How can we train Seq2Seq models?

  How do we do inference?

Why do we feed back the generated words?

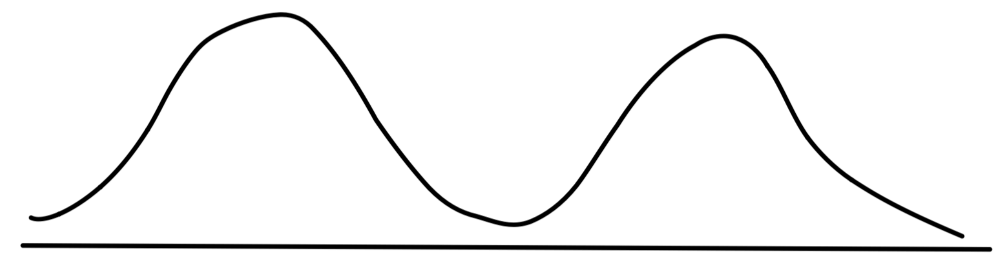
- **First thought:** Feeding back is unnecessary
  - y_2 is predicted from $\boldsymbol{h}_2$
  - $\boldsymbol{h}_2$ depends on $\boldsymbol{h}_1$
  - y_1 depends on $\boldsymbol{h}_1$

$$\left.\begin{array}{l} \boldsymbol{h}_2 \text{ depends on } \boldsymbol{h}_1 \\ y_1 \text{ depends on } \boldsymbol{h}_1 \end{array}\right\} \Rightarrow \quad y_1 \text{ brings no information}$$

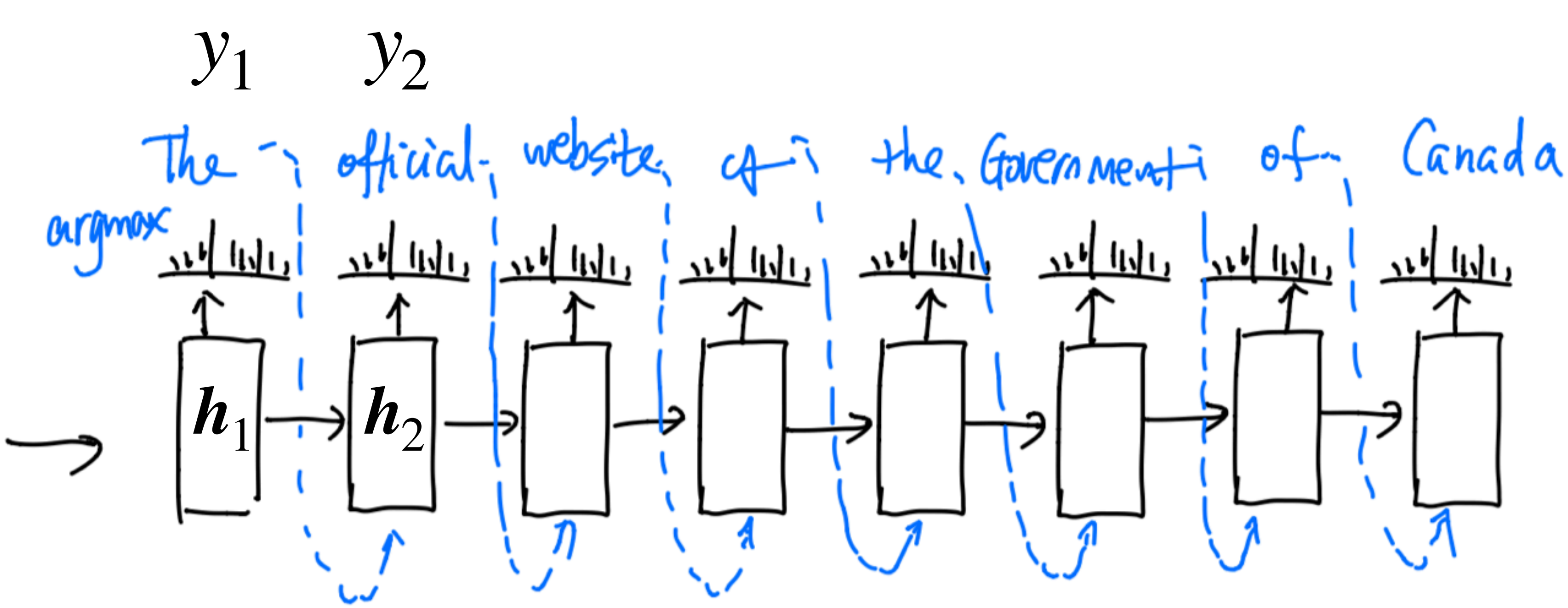


Multi-Modal Distribution

- **Continuous distribution**

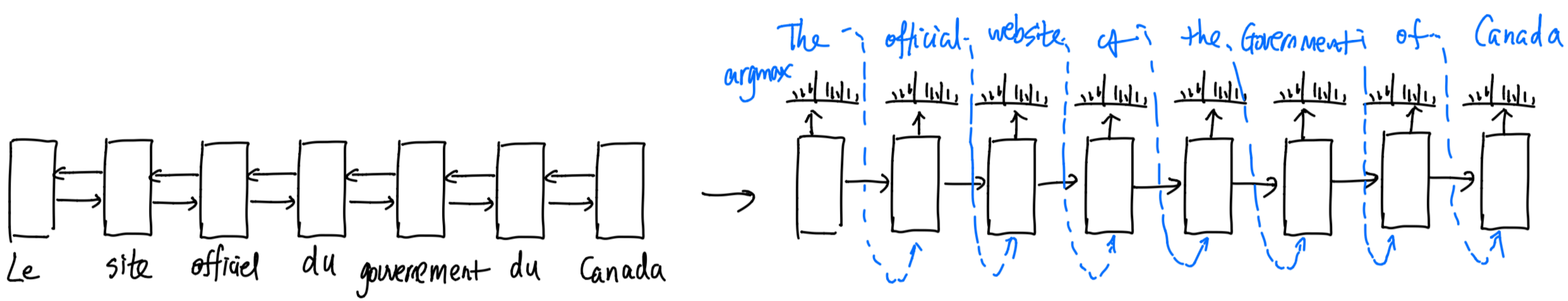


- **Discrete distribution**
  - Image in some embedding space, sentences are multi-modal distributed
  - `“A beats B”` vs. `“B is beaten by A”`



Training Seq2Seq Models

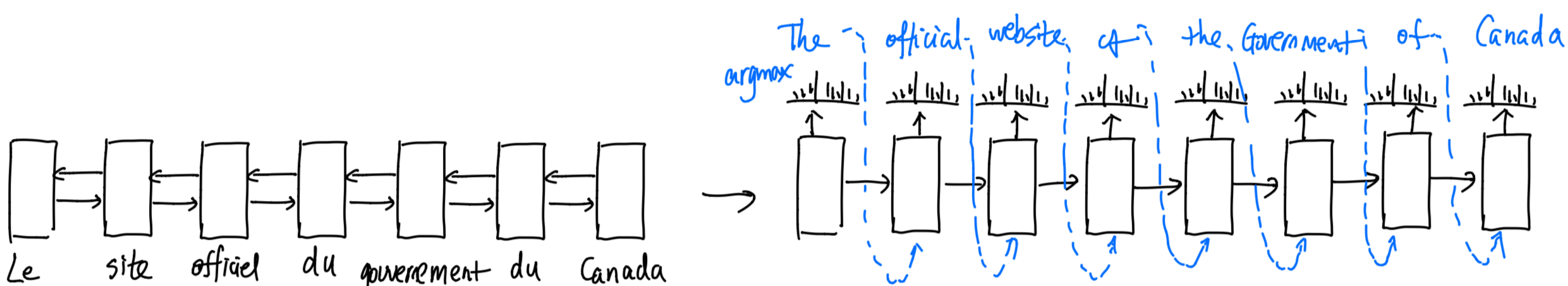
- Decoder's input layer
  - Attempt#1: Feed in the predicted words
  - Attempt#2: Feed in the groundtruth word



Training Seq2Seq Models

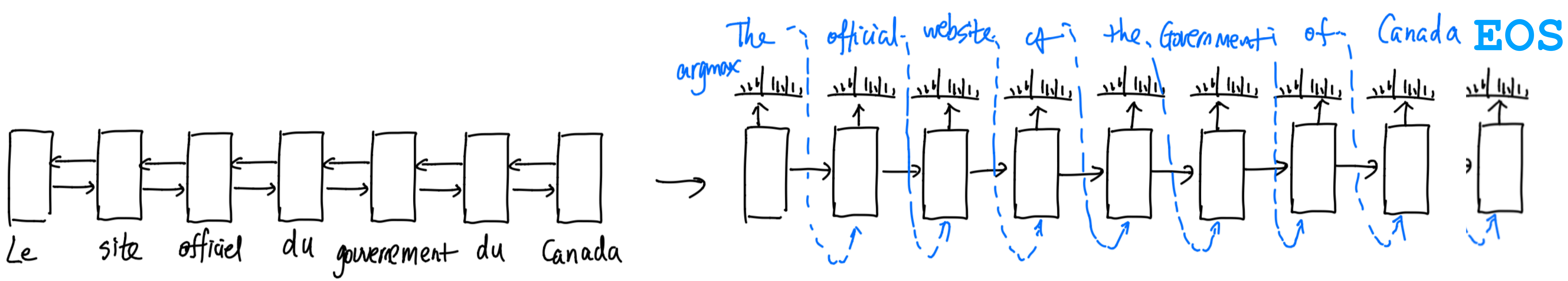
- Decoder's input layer
  - Attempt#1: Feed in the predicted words
  - Attempt#2: Feed in the groundtruth word

- Loss $J = J_1 + J_2 + \cdots + J_T$
  - Suppose we known "groundtruth" target sequence
  - Recall BP with multiple losses



Inference

- Decoder's input layer
  - Feed in the predicted words
- When do we terminate?
  - Include a special token "EOS" (end of sequence) in training
  - If "EOS" is predicted, the sentence is terminated by def



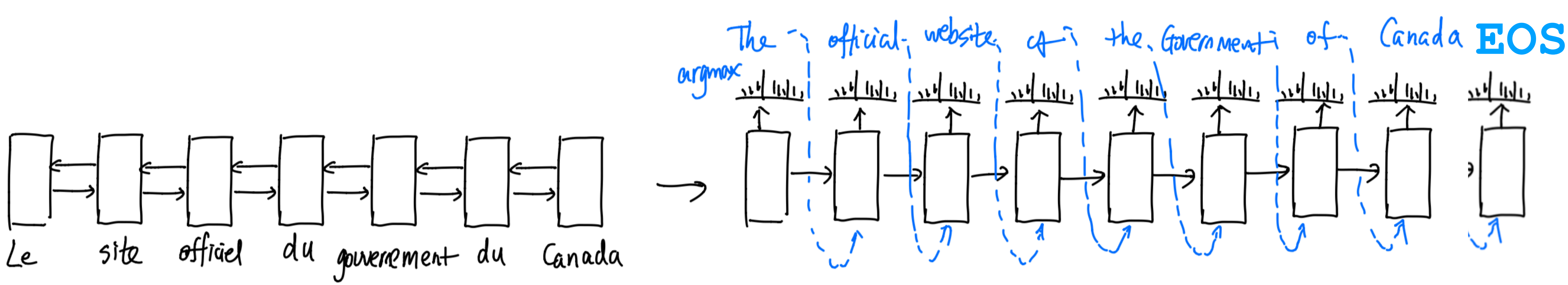
Caveat

- Batch implementation
  - Padding EOS or **0** vector=> Incorrect
  - Masking => Correct

$$\tilde{\boldsymbol{h}}_t = \mathrm{RNN}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

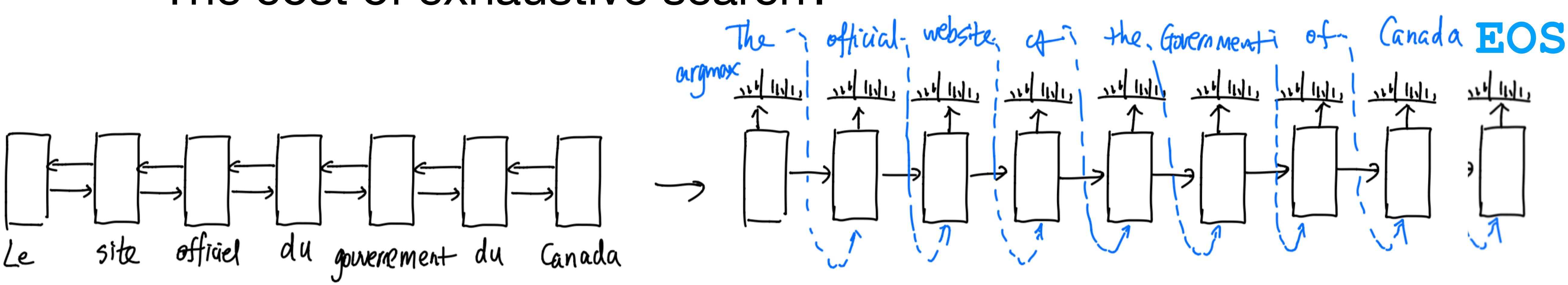
$$\boldsymbol{h}_t = (1-m)\boldsymbol{h}_{t-1} + m\tilde{\boldsymbol{h}}_t$$

- Implementation should always be equivalent to math derivations



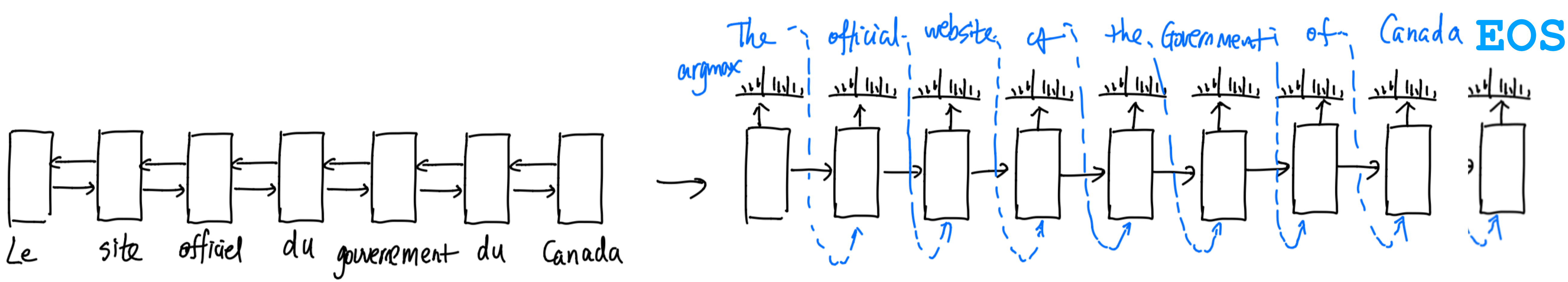
Inference Criteria

- Single-output classification
  - Max a posteriori inference ⇔ Minimal empirical loss
  - $y = \operatorname{argmax} p(y \mid x)$
- Sentence generation
  - If we want to generate the "best" sentence:
    $\mathbf{y} = \operatorname{argmax} p(\mathbf{y} \mid \mathbf{x})$
  - Is greedy correct? $y_i = \operatorname{argmax} p(y_i \mid y_{<i}, \mathbf{x})$
  - The cost of exhaustive search?



Greedy vs. Exhaustive Search

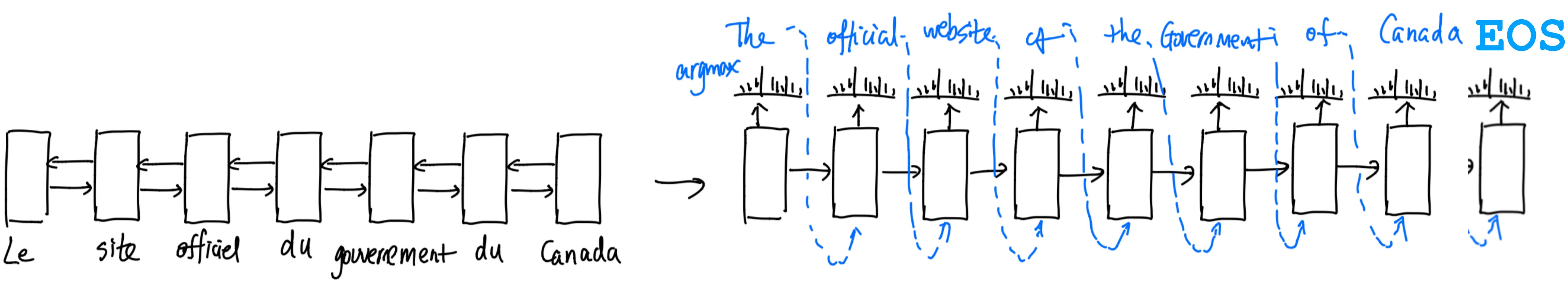
	Greedy	**Exhaustive search**
For each step	Pick the **best** word	Try **every** word
Maintain	**One** sequence	**All** possible combinations



Greedy vs. Exhaustive Search

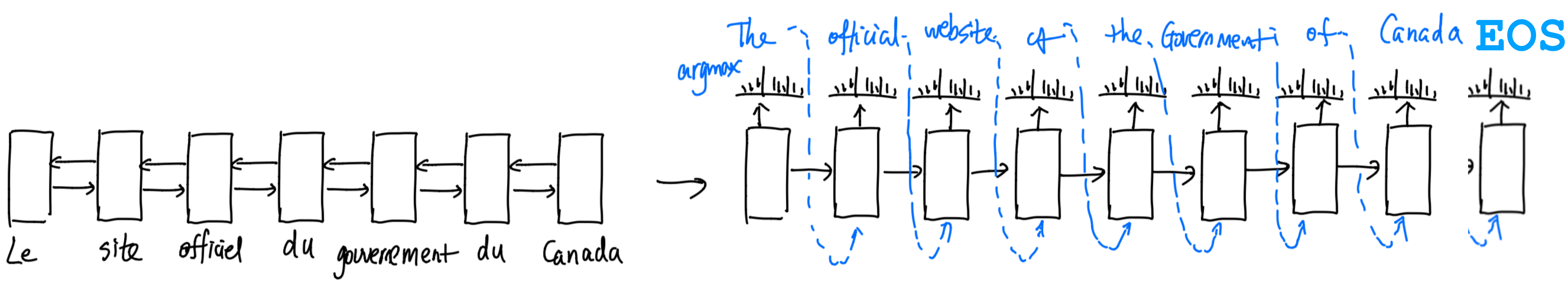
	Greedy	**Exhaustive search**
For each step	Pick the **best** word	Try **every** word
Maintain	**One** sequence	**All** possible combinations

Need something in between

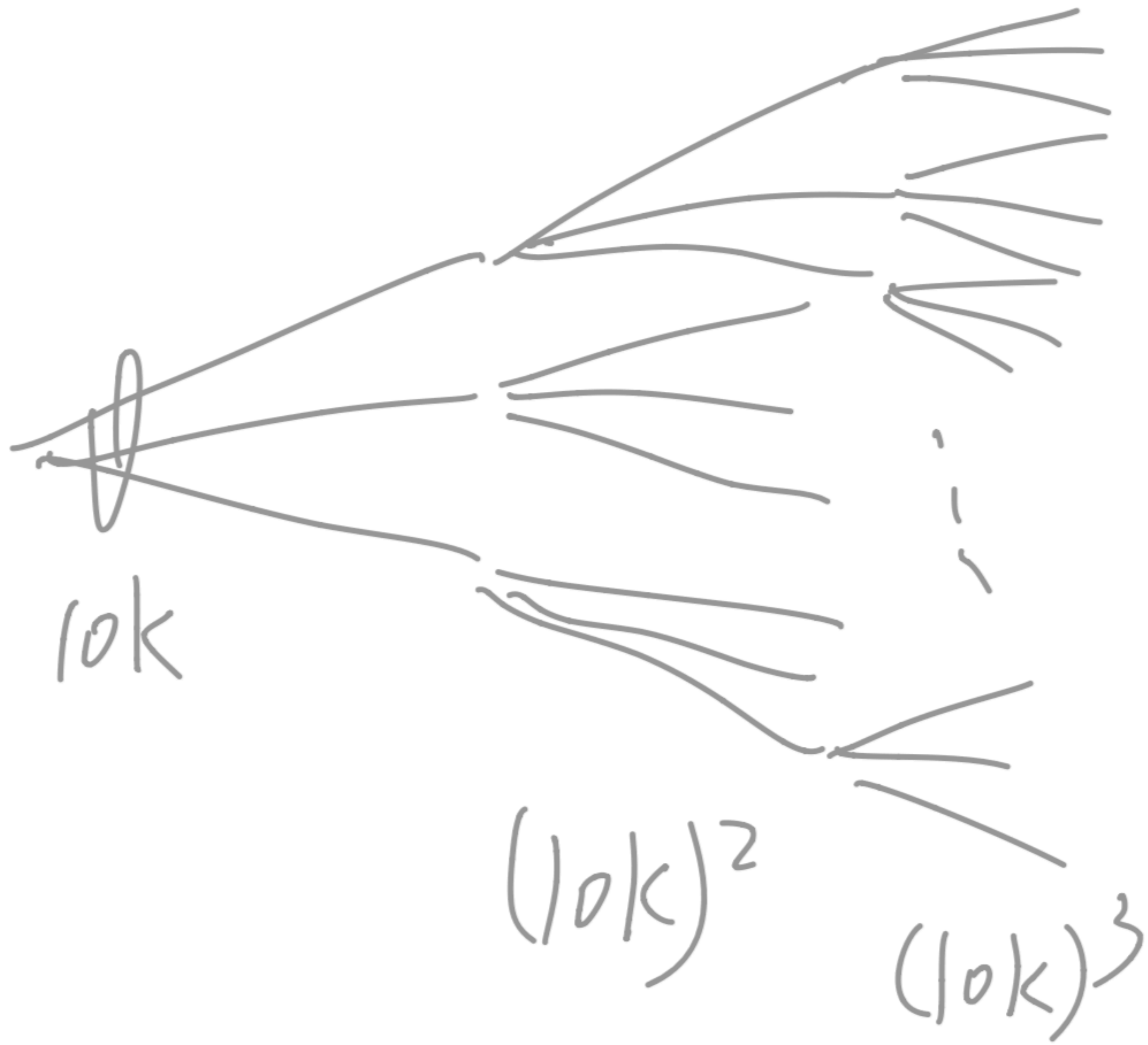


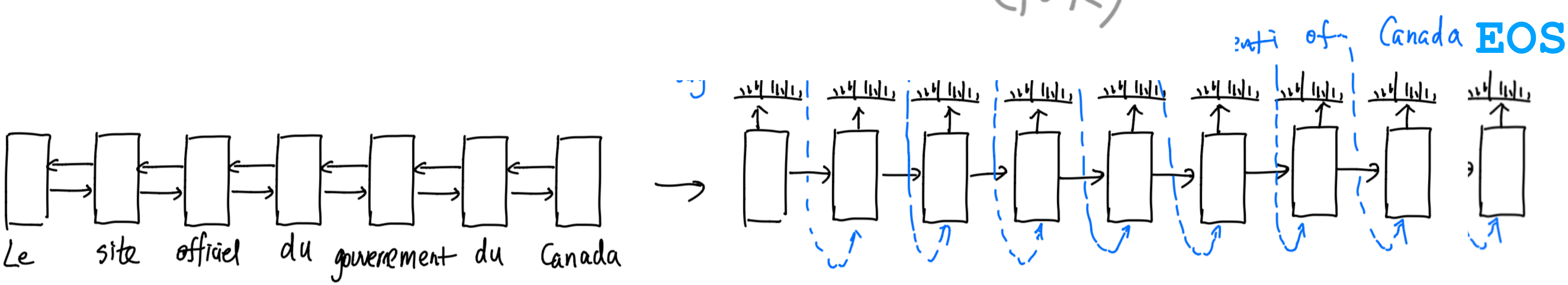
Greedy vs. Exhaustive Search

	Greedy	Beam Search	Exhaustive search
For each step	Pick the **best** word	Try **a few** best words	Try **every** word
Maintain	**One** sequence	**Several** good partial sequences	**All** possible combinations



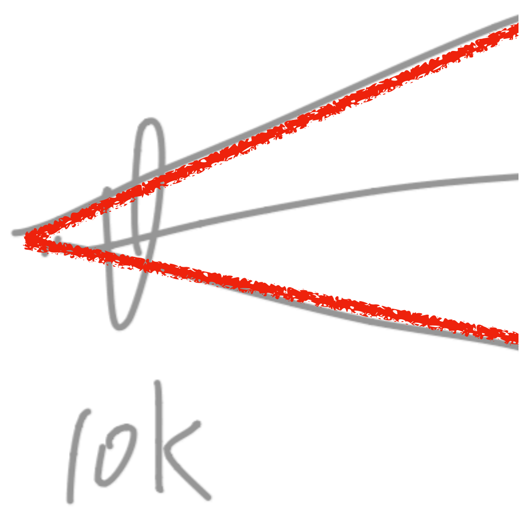
Beam Search

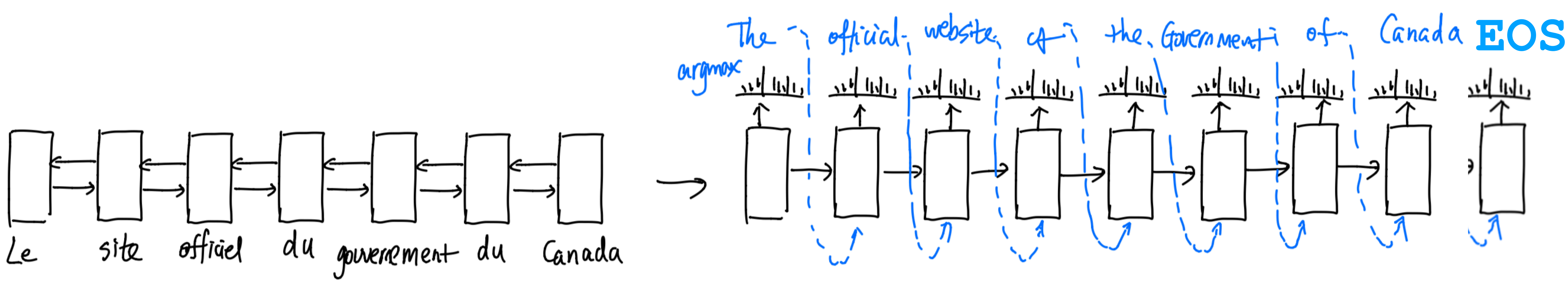

10k
(10k)^2
(10k)^3


Le site officiel du gouvernement du Canada
of Canada EOS

Beam Search

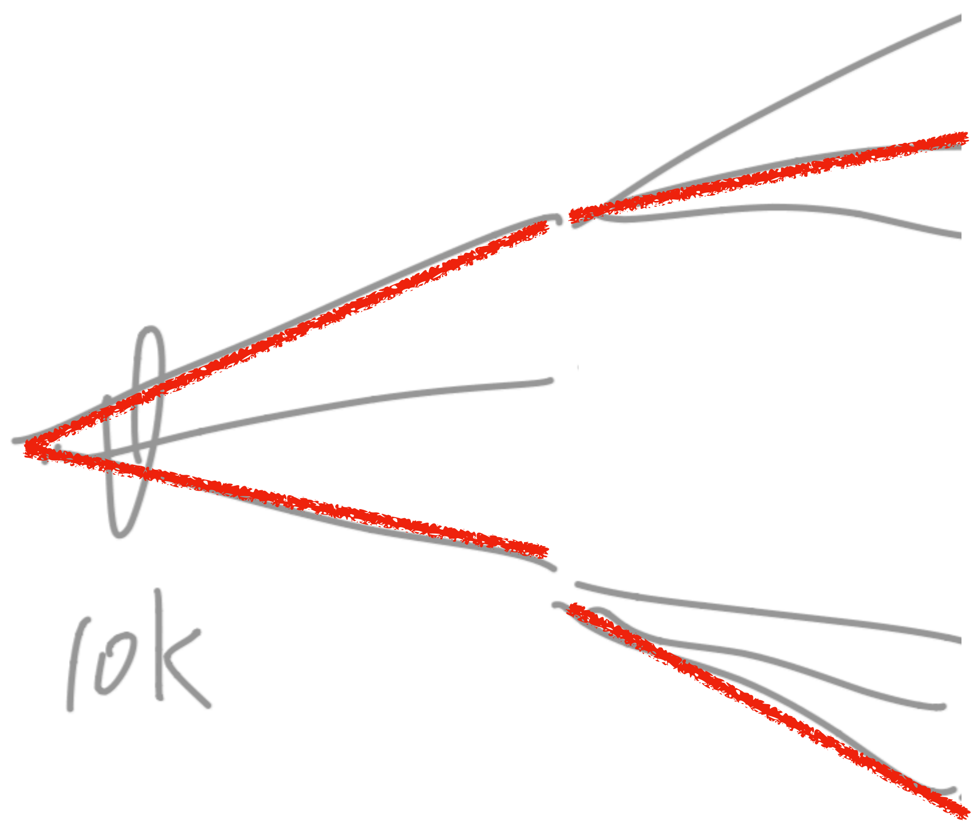
B=2

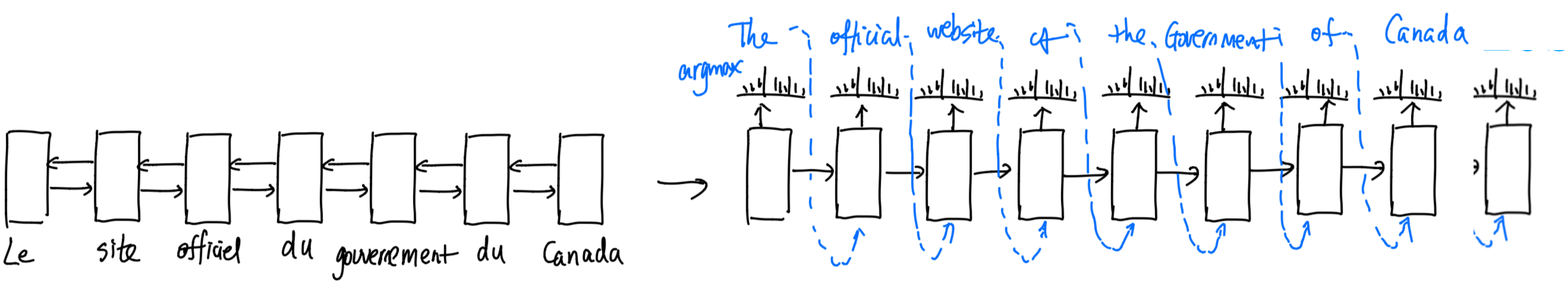




Beam Search

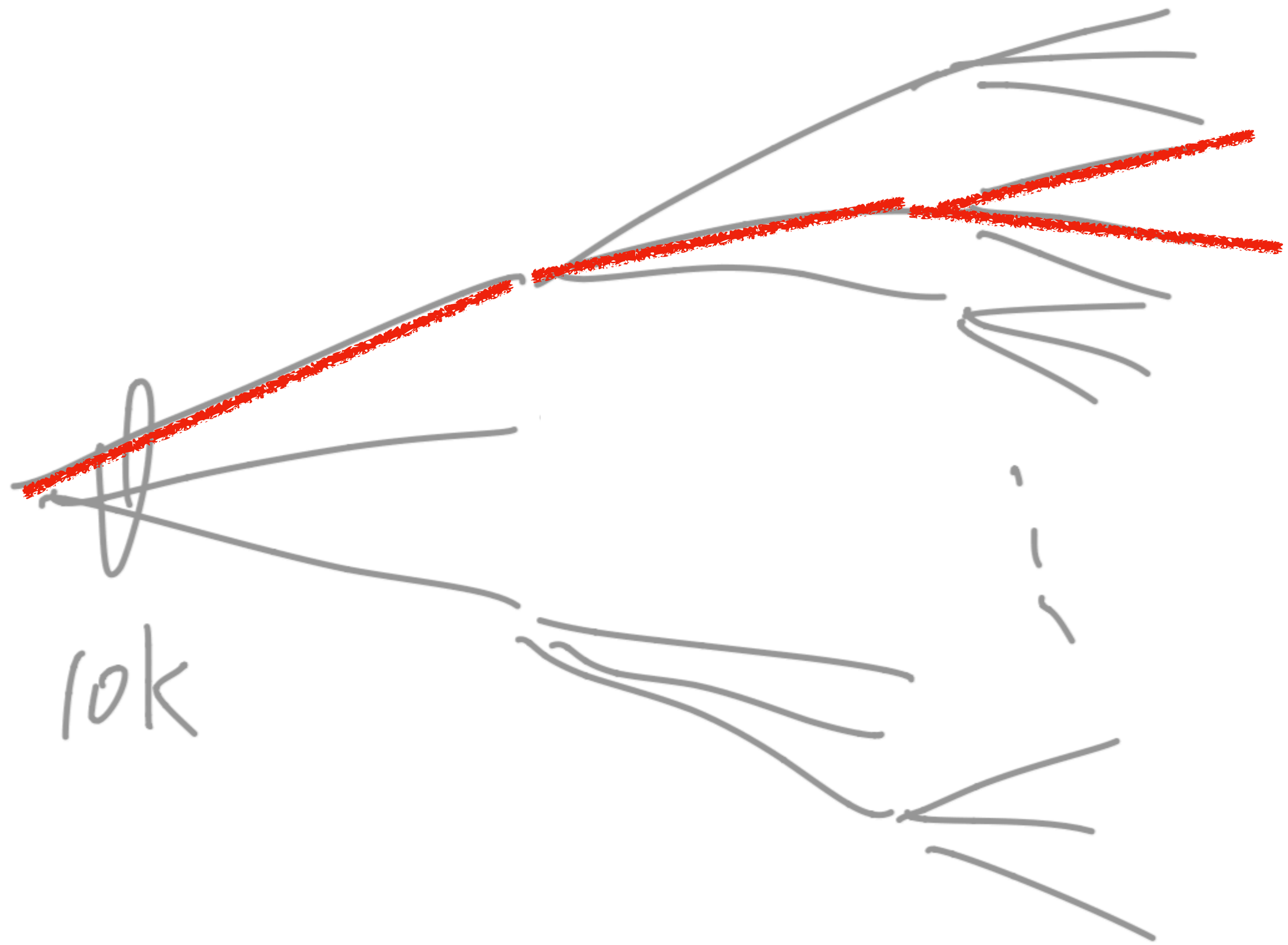
B=2

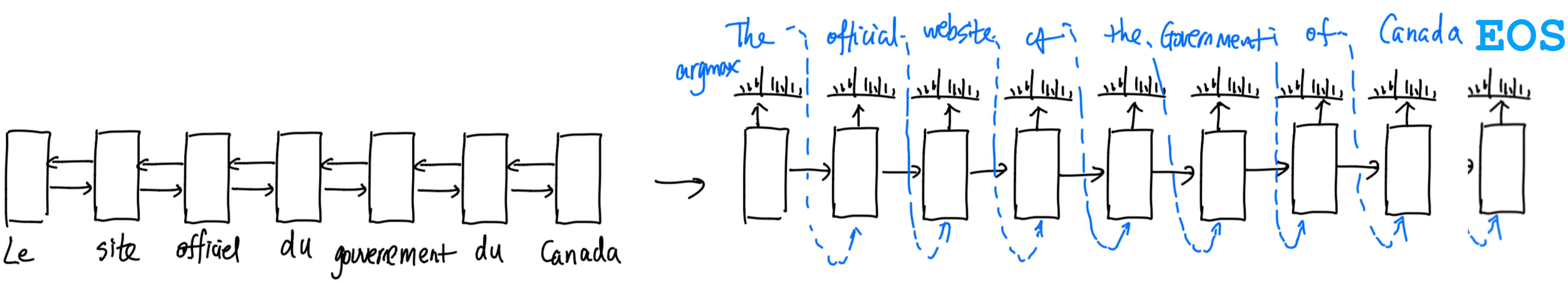




Beam Search

B=2

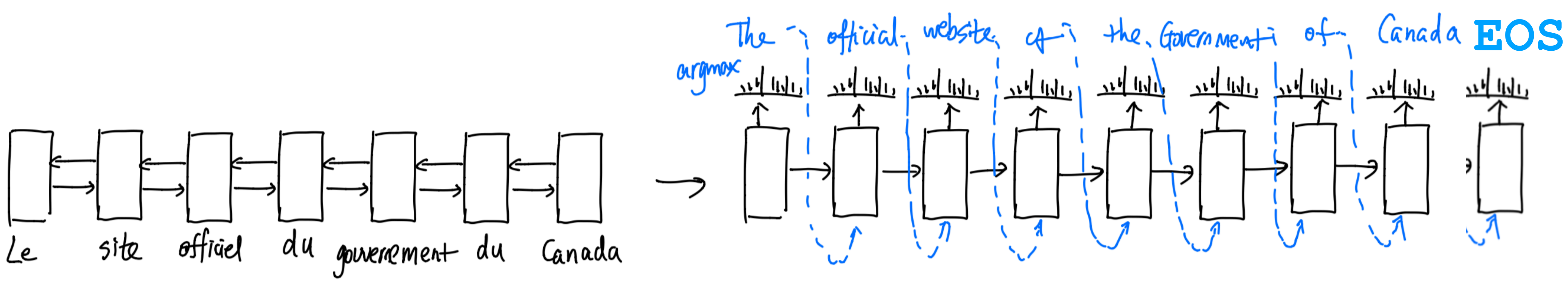




Beam Search

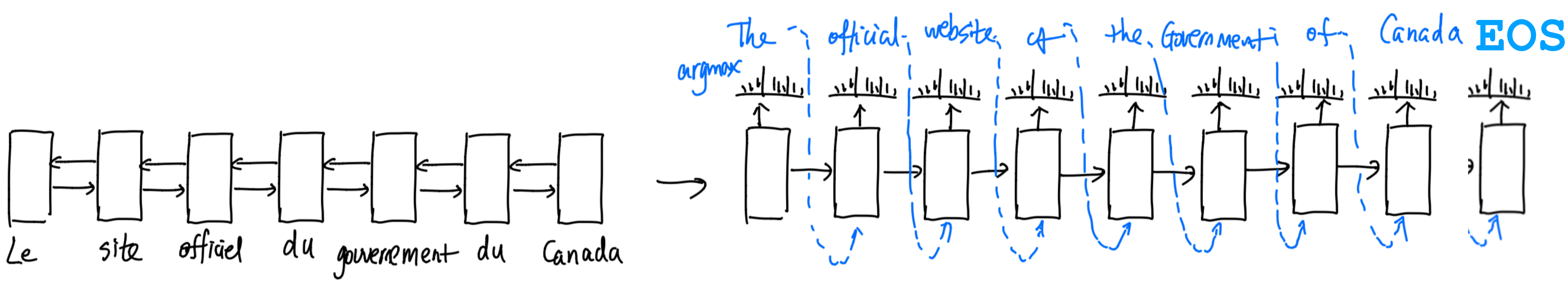
- A list of best partial sequences S [ ]
- For every decoding step t
  - For every partial seq $s \in S$ and every word $w \in \mathscr{V}$
    - Expand s as (s, w)
  - $S =$ top-B expanded subsequences among all (s, w)
- Return the most probable sequence in the beam

(existing a terminated sequence better than all $s \in S$)

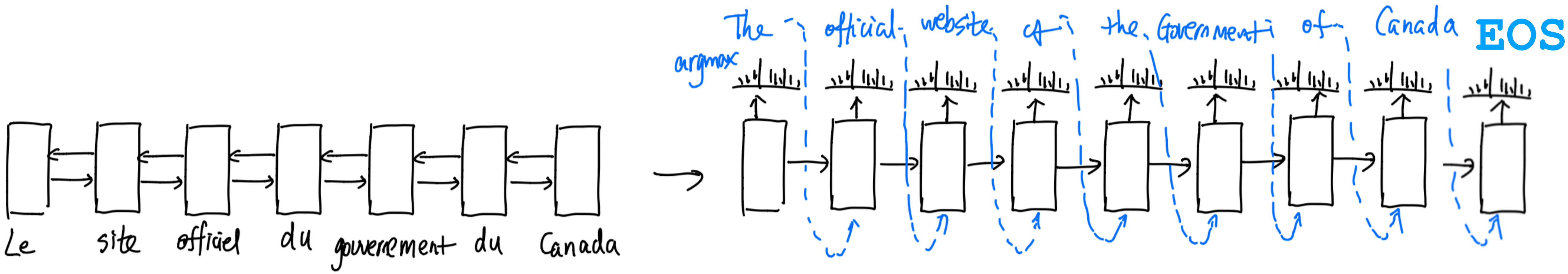


Issues with Autoregressiveness

- Error accumulation
  - 1st word good, 2nd worse, 3rd even worse, etc.
- Label bias
  - Not “label imbalance” problem
  - BS bias towards high probable words at the beginning
  - Locally normalized models prefer high probable (but possibly unimportant) words



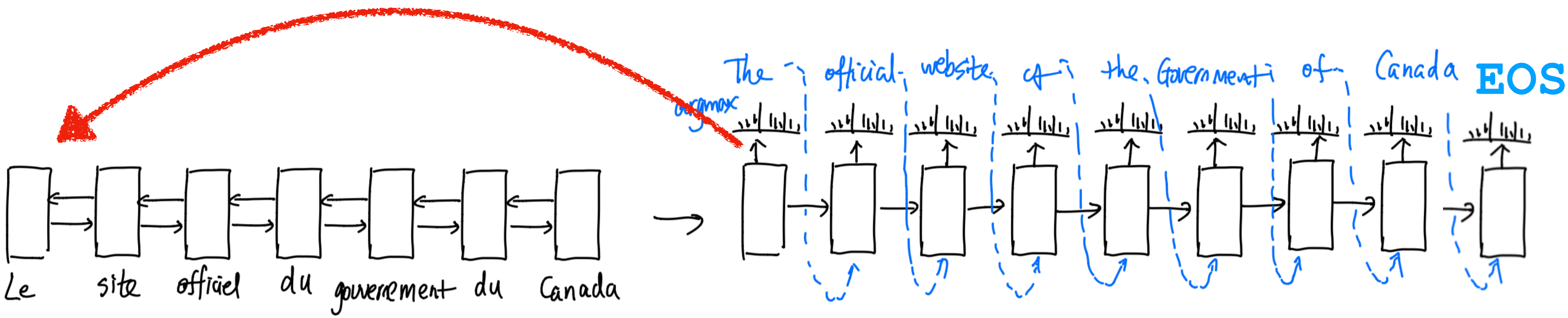
Information Bottleneck



- Last hidden state
  - Has to capture all source information
- Average/Mean pooling
  - Still loses information
  - Not directly related to the current decoded word

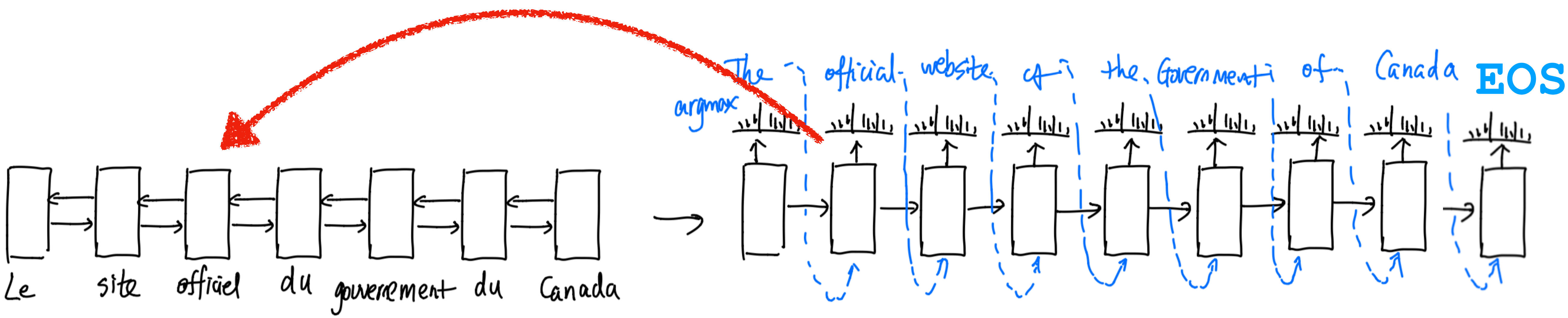
UNIVERSITY OF ALBERTA

Attention Mechanism



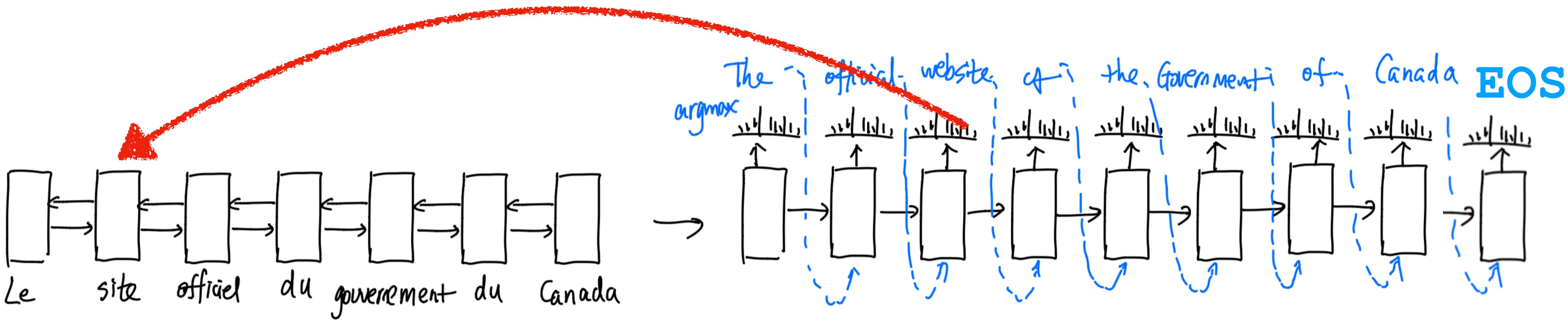
- Dynamically aligning words

Attention Mechanism



- Dynamically aligning words

Attention Mechanism



- Dynamically aligning words
  - **Average pooling => weighted pooling**
  - Alignment dependents on the current word to be generated
  - Alignment to a particular source word obviously also dependents on that source word itself

Convex vs Linear Weighting

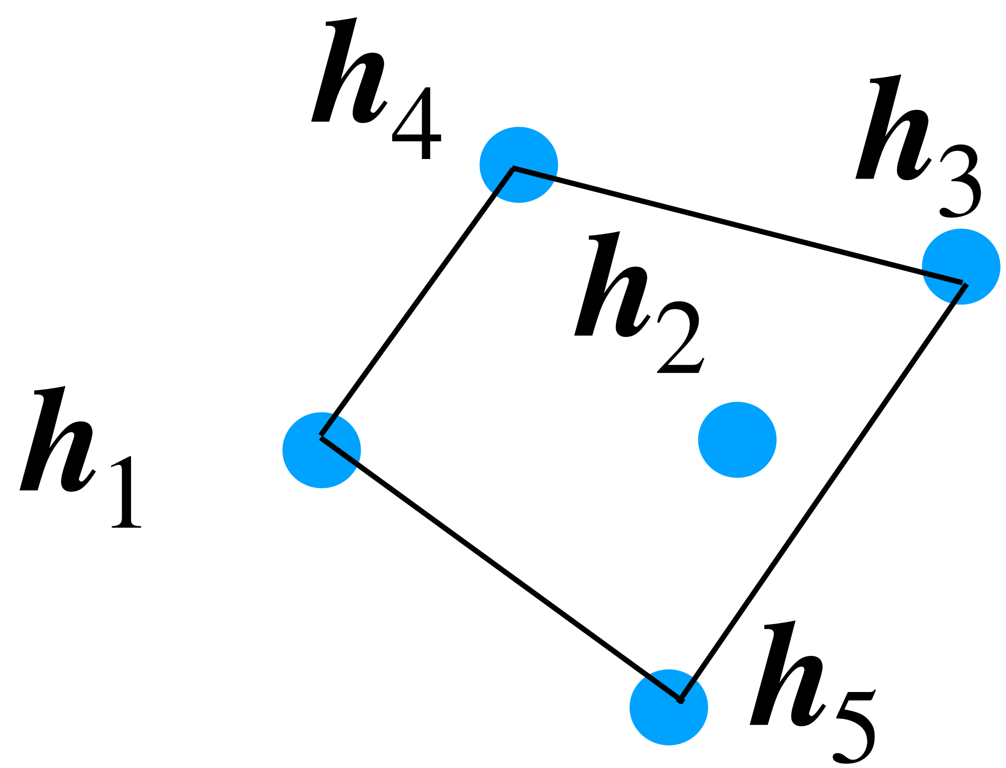
- Average pooling

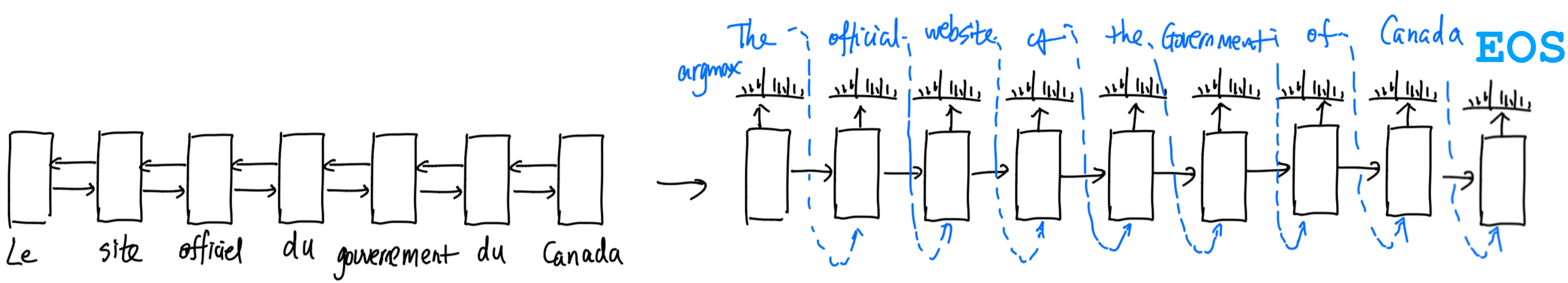
$$c = \frac{1}{N} h_1 + \cdots + \frac{1}{N} h_N$$

- Weighted pooling

$$c = \alpha_1 h_1 + \cdots + \alpha_N h_N$$

What are $\alpha_1, \cdots, \alpha_N$?





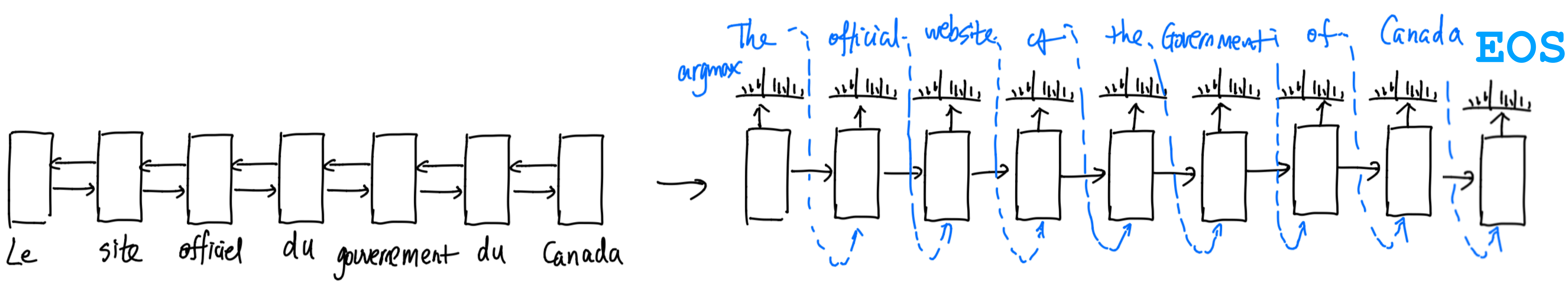
Computing Attention

Score (-Energy) $s_j^{(t)} = s(\boldsymbol{h}_j^{(\text{tar})}, \boldsymbol{h}_i^{(\text{src})})$

Unnormalized measure $\widetilde{\alpha}_j^{(t)} = \exp(s_j^{(t)})$

Probability $\alpha_j^{(t)} = \dfrac{\exp(s_j^{(t)})}{\sum_{j'} \exp(s_{j'}^{(t)})}$

Denominator: Partition function



Computing Attention

Score (-Energy) $s_j^{(t)} = s(\boldsymbol{h}_j^{(\text{tar})}, \boldsymbol{h}_i^{(\text{src})})$

Unnormalized measure $\widetilde{\alpha}_j^{(t)} = \exp(s_j^{(t)})$

Probability $\alpha_j^{(t)} = \dfrac{\exp(s_j^{(t)})}{\sum_{j'} \exp(s_{j'}^{(t)})}$

Denominator: Partition function

Inner-product

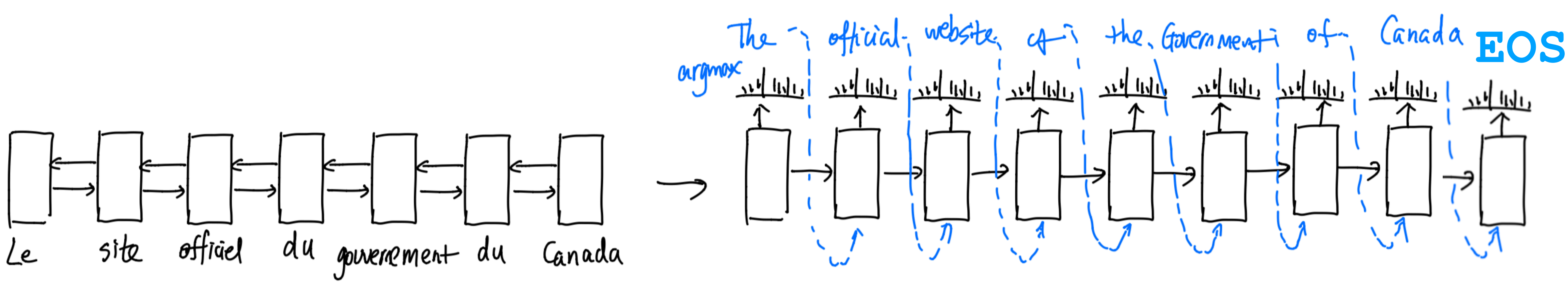
$$s_j^{(t)} = \left(\boldsymbol{h}_j^{(\text{tar})}\right)^\top \boldsymbol{h}_i^{(\text{src})}$$

Metric learning

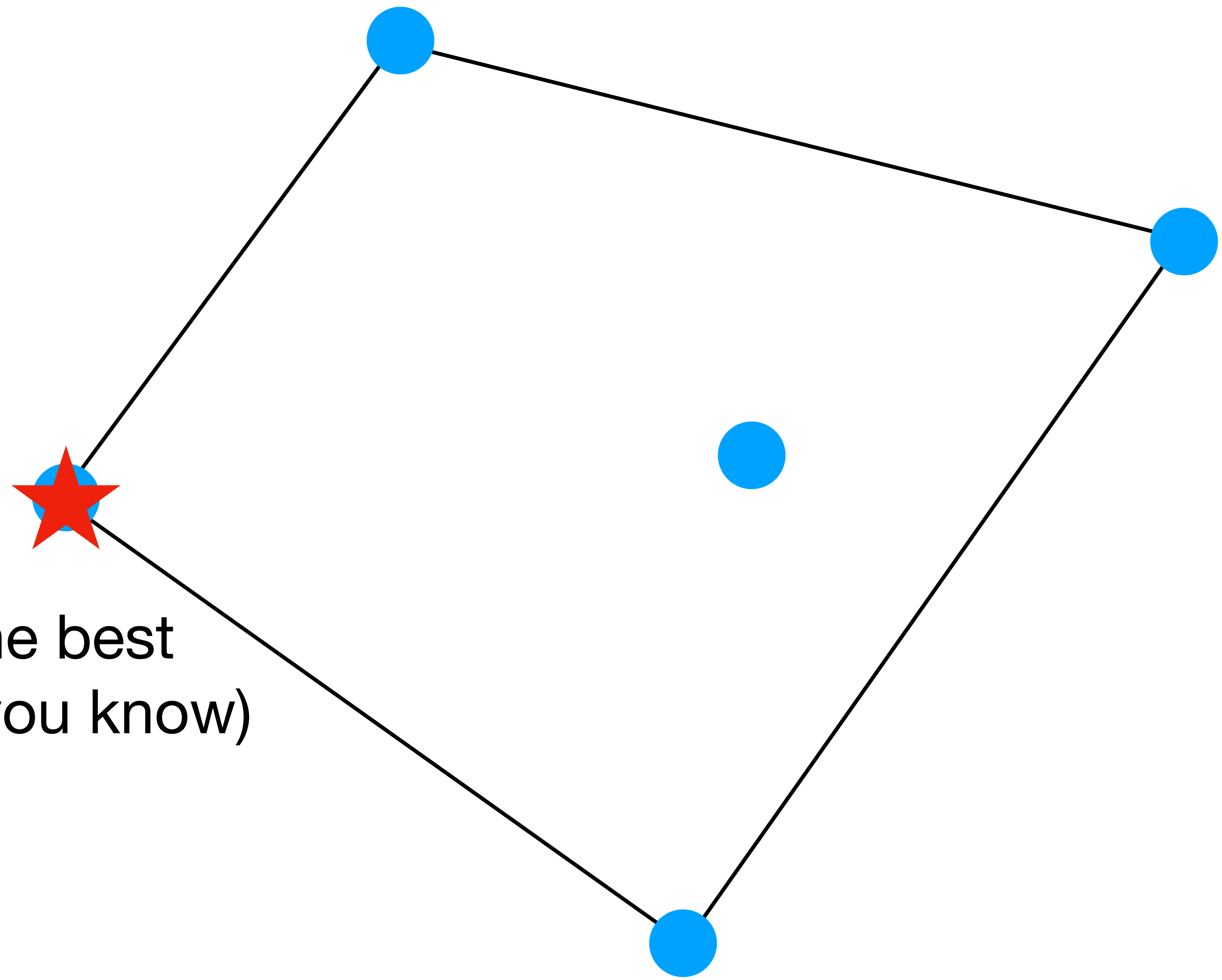
$$s_j^{(t)} = \left(\boldsymbol{h}_j^{(\text{tar})}\right)^\top W \boldsymbol{h}_i^{(\text{src})}$$

Neural layer

$$s_j^{(t)} = \boldsymbol{u}^\top f\left(\boldsymbol{W}[\boldsymbol{h}_j^{(\text{tar})}; \boldsymbol{h}_i^{(\text{src})}]\right)$$



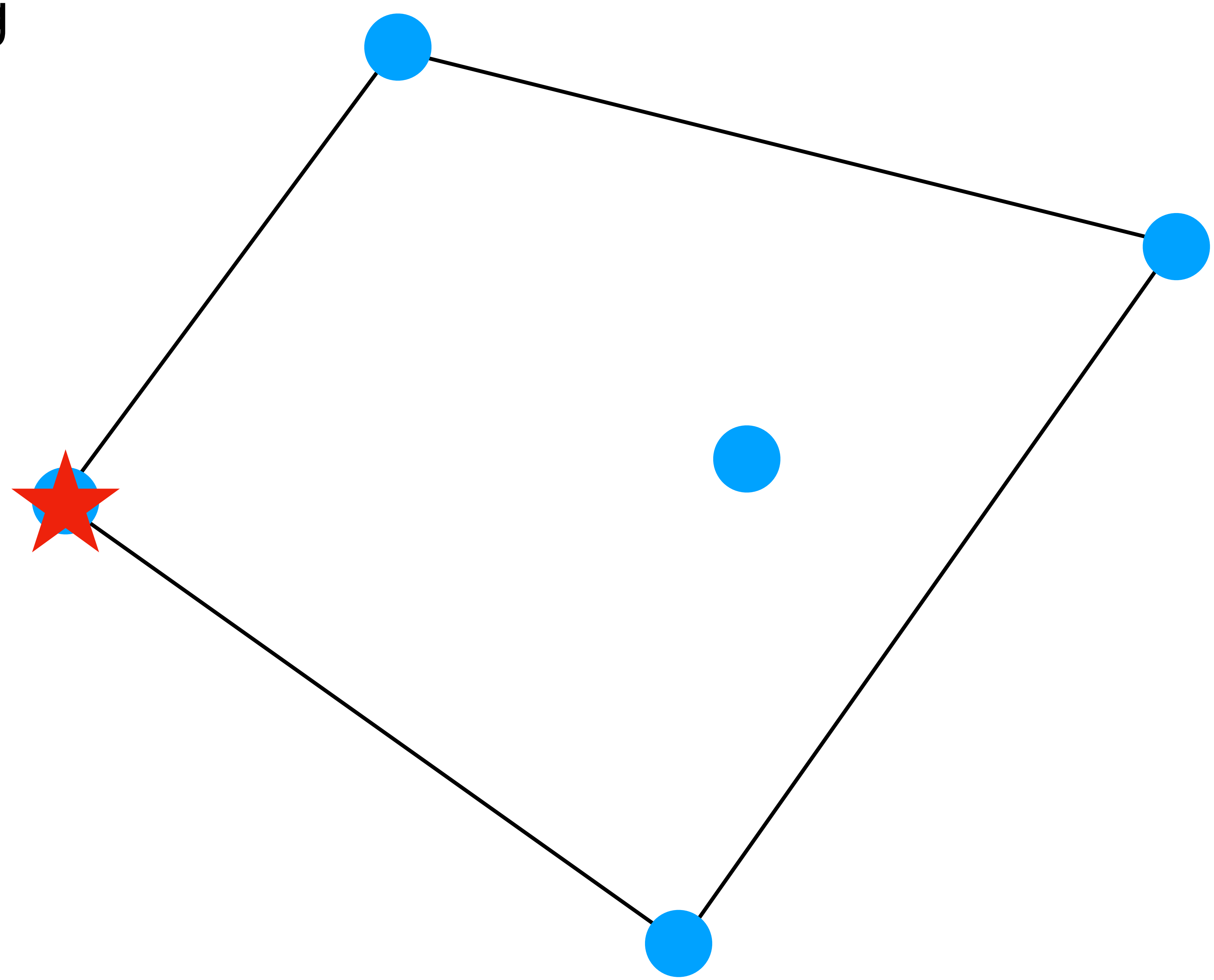
Where are we?



Picking the best
(how do you know)

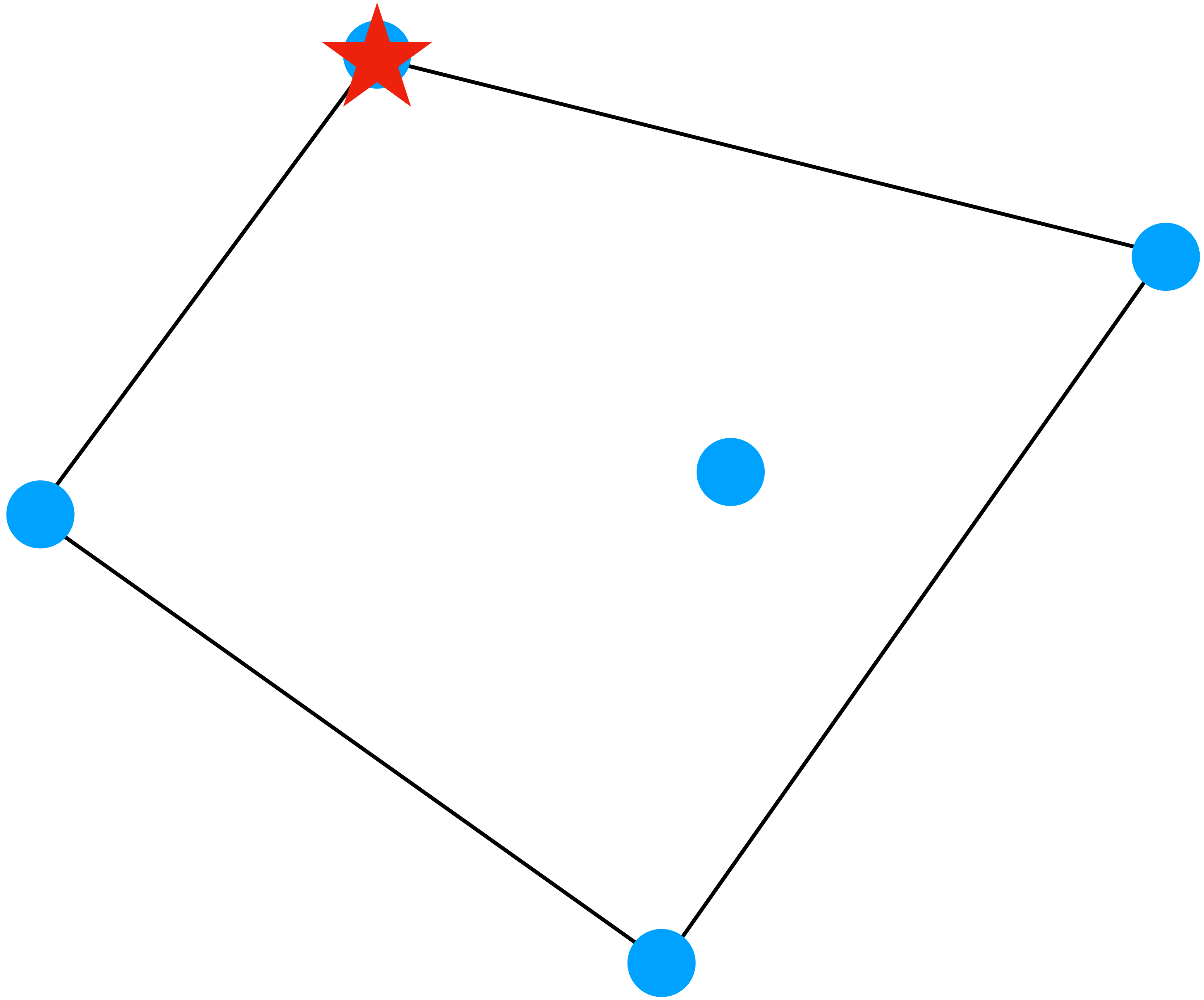
Where are we?

Sampling



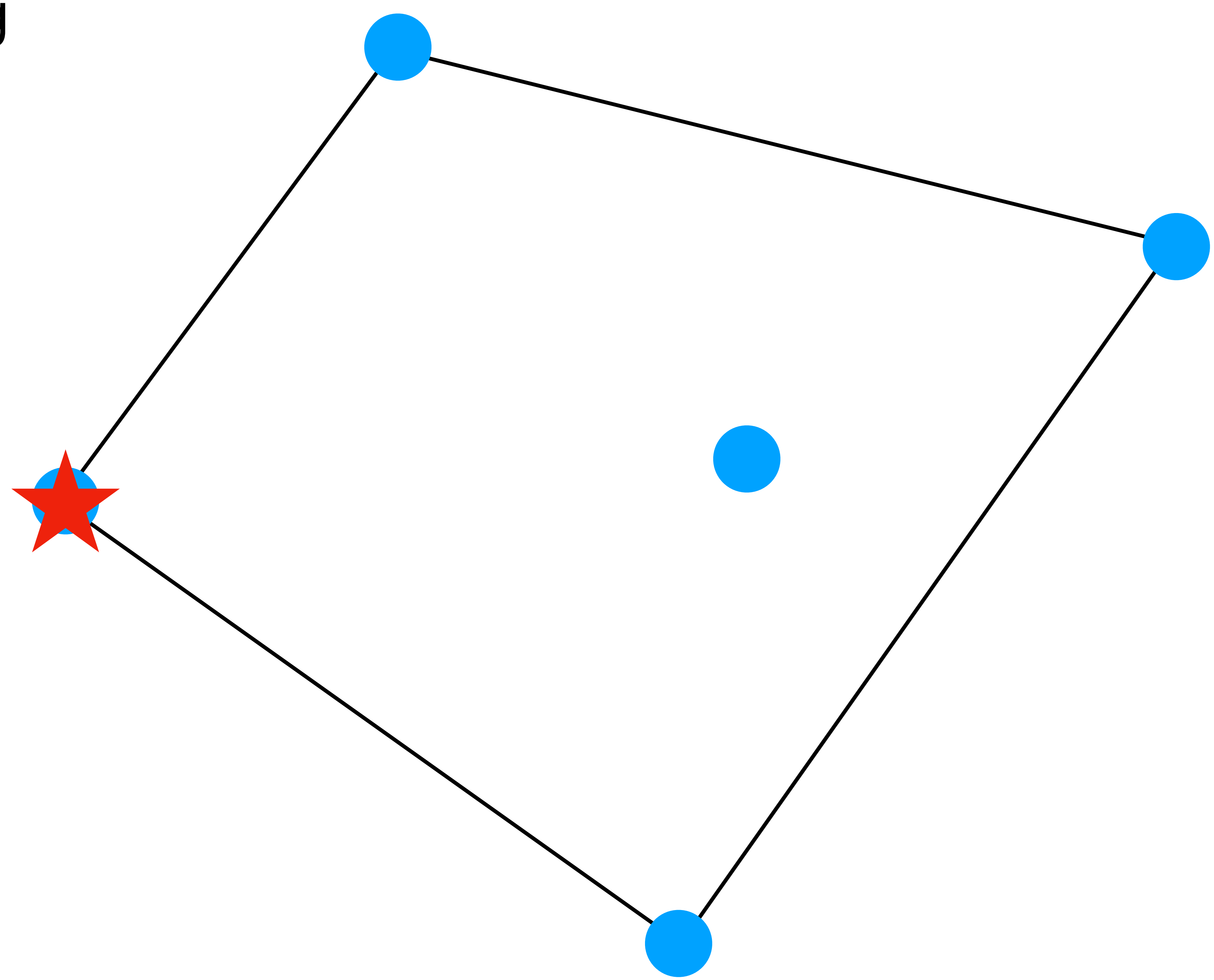
Where are we?

Sampling



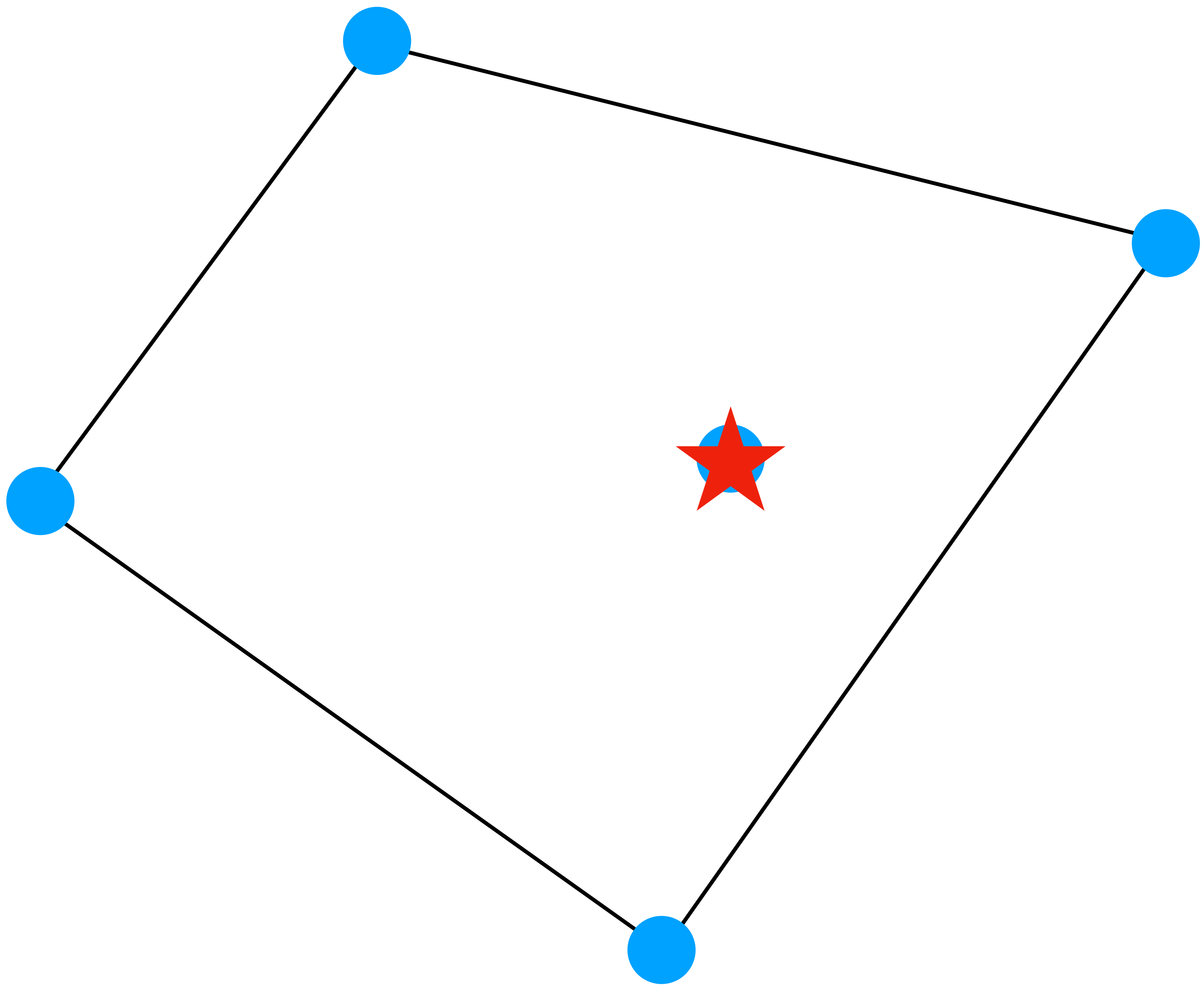
Where are we?

Sampling



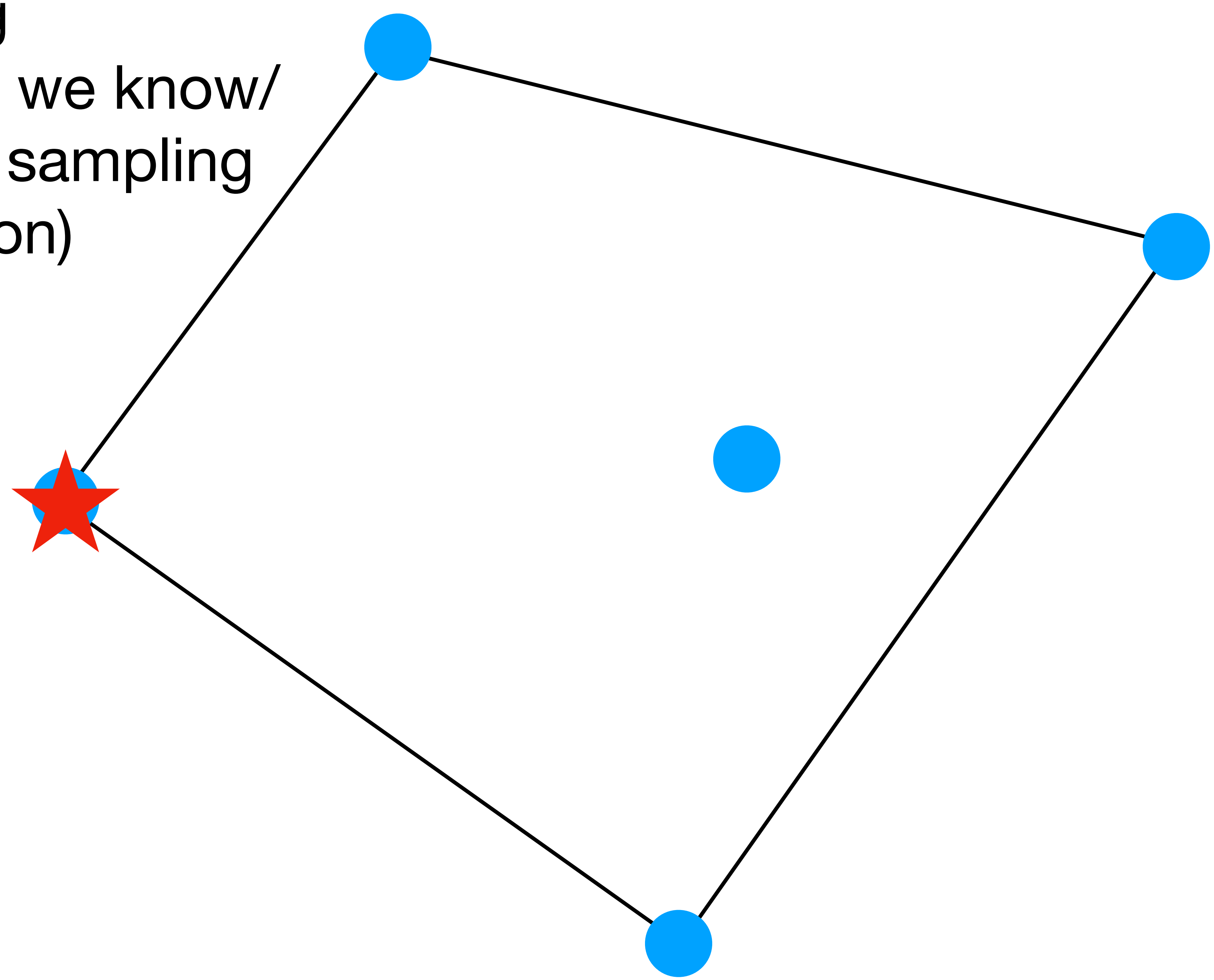
Where are we?

Sampling

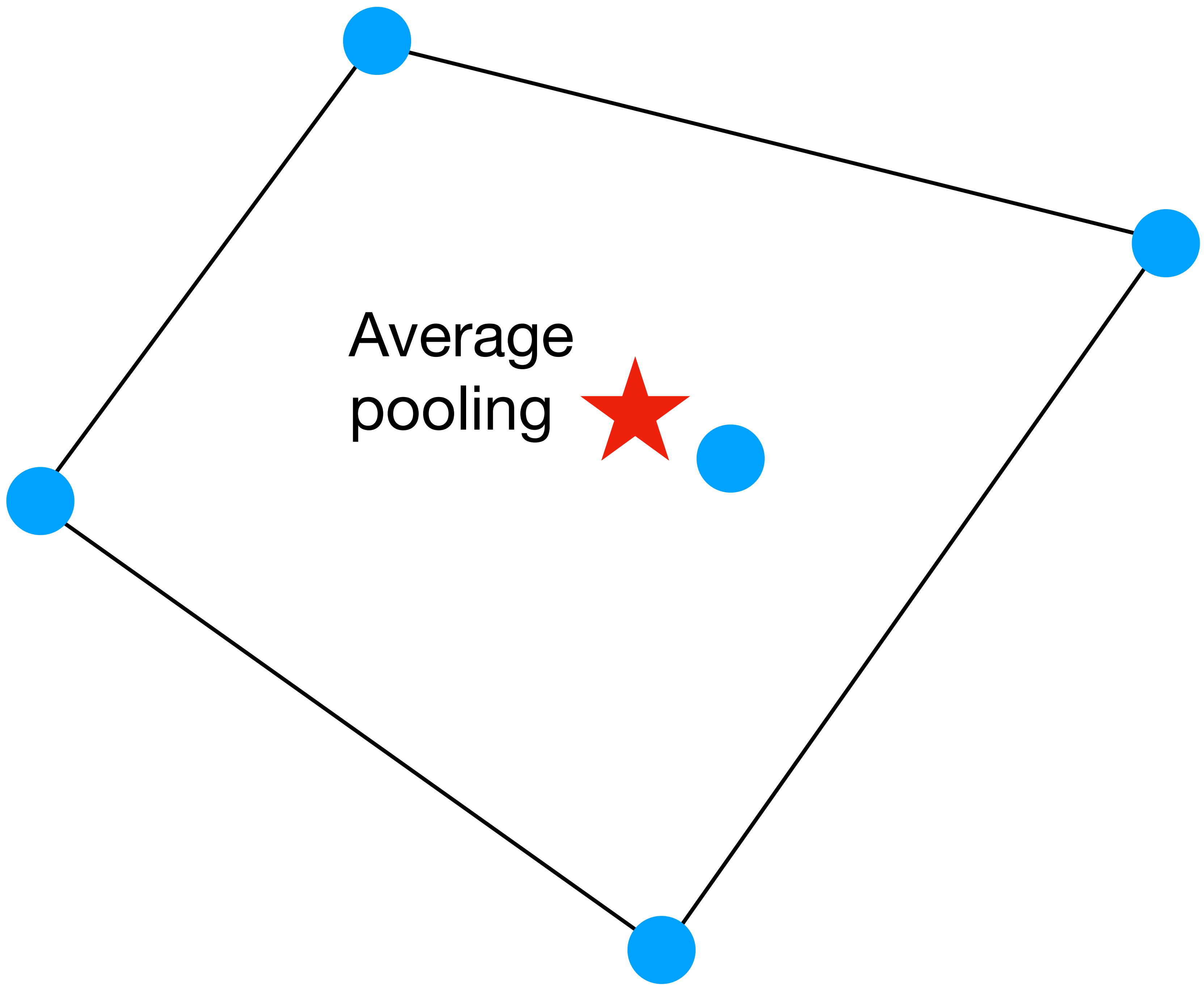


Where are we?

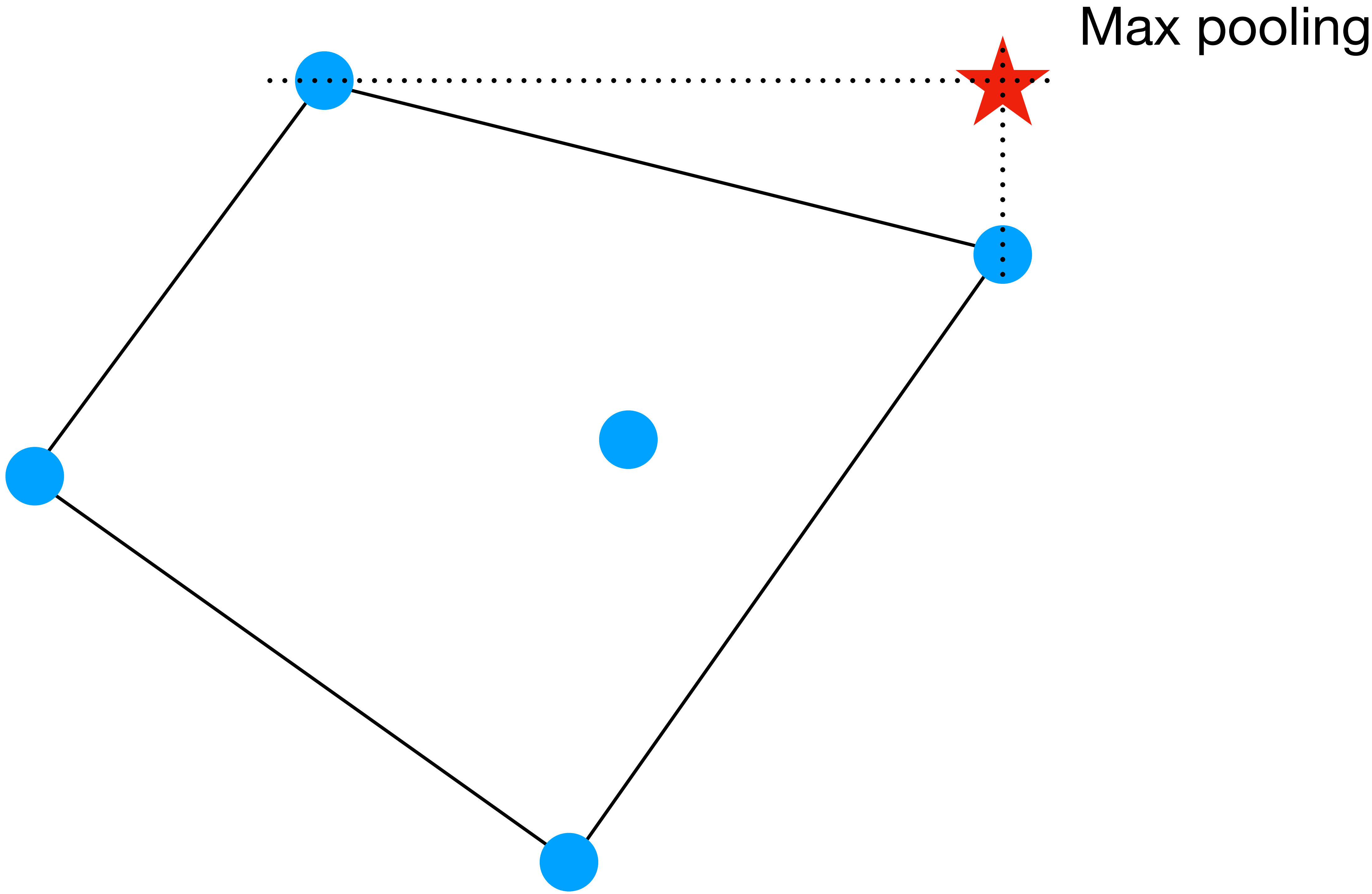
Sampling
(how can we know/
learn the sampling
distribution)



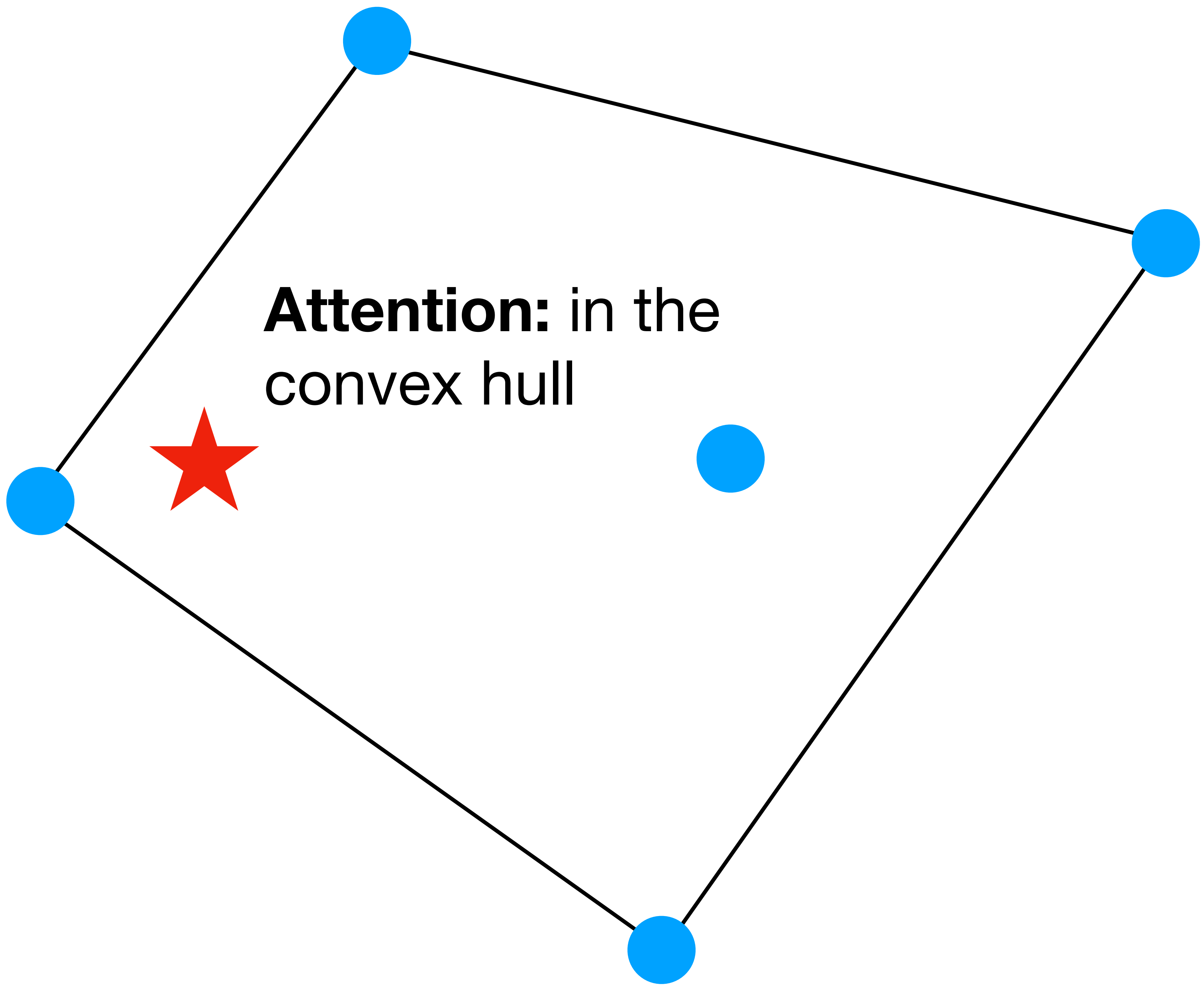
Where are we?



Where are we?

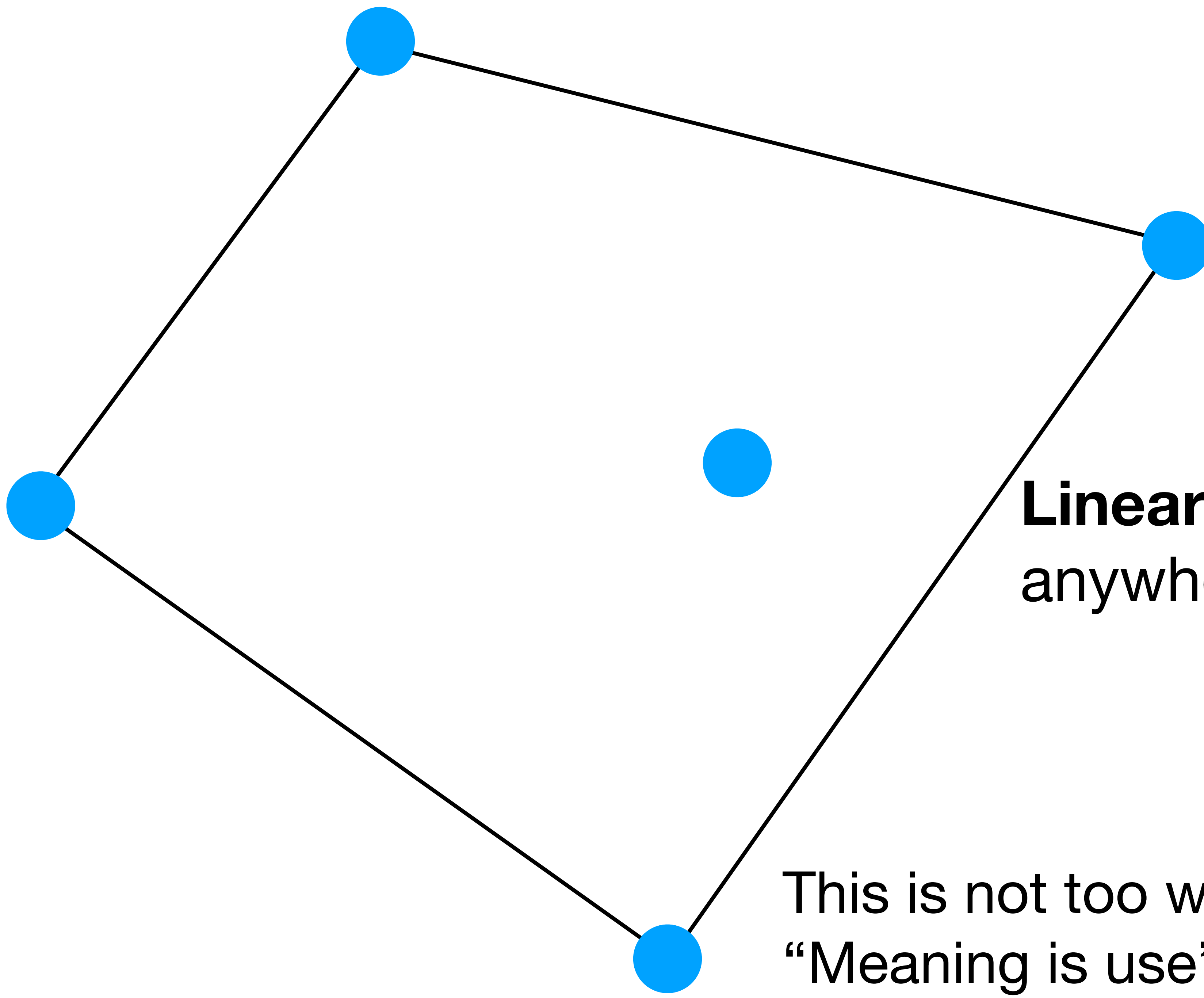


Where are we?



UNIVERSITY OF ALBERTA

Where are we?

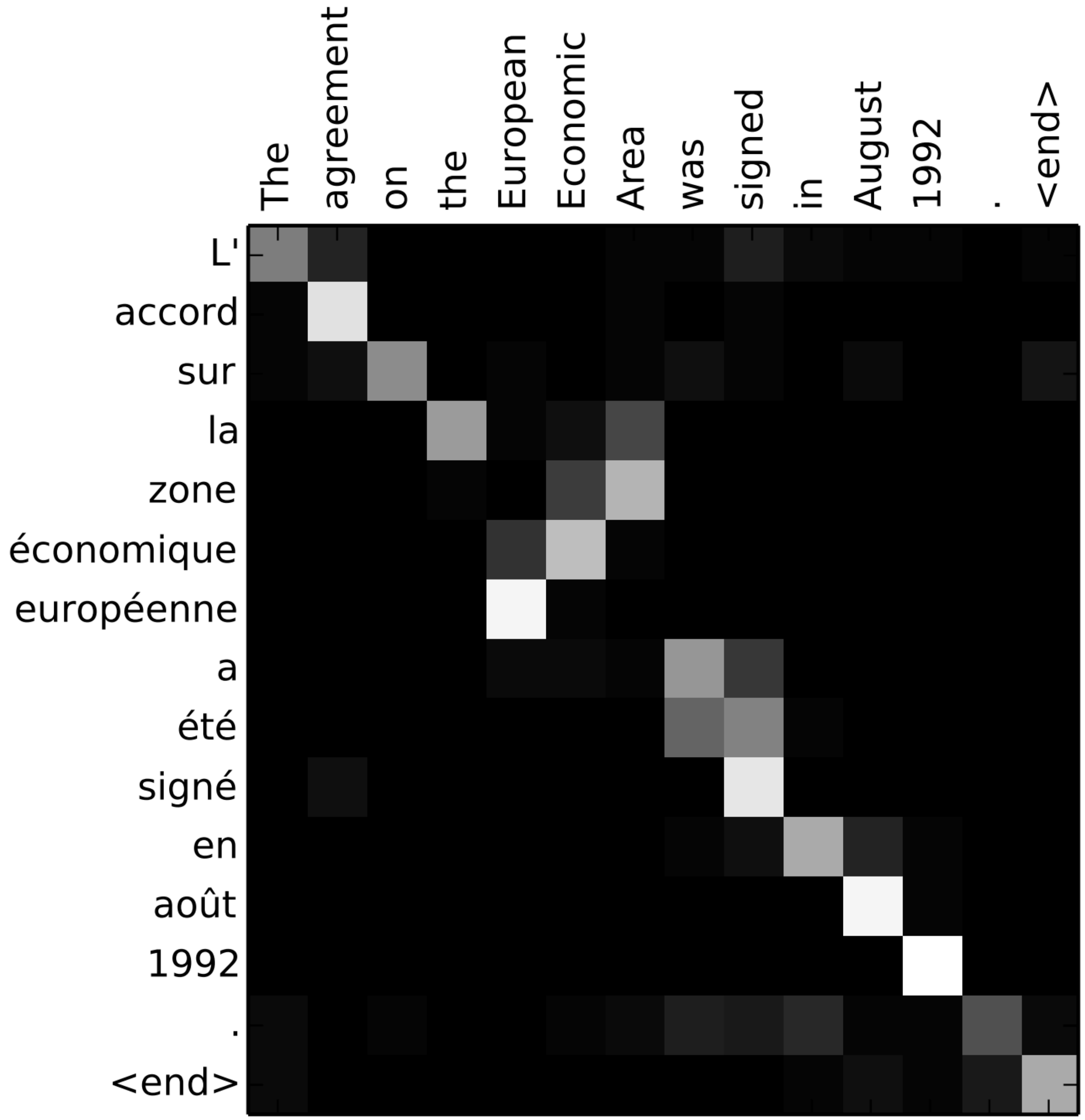


Linear combination:
anywhere in the space

This is not too wrong.
“Meaning is use” —Wittgenstein

In machine learning,
how you train is how you predict

Attention as Soft Alignment



Bahdanau, D., Cho, K. and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. ICLR, 2015.

UNIVERSITY OF ALBERTA

Traditional Phrase-Based MT

$$\Pr(\mathbf{e}|\mathbf{f}) = \frac{\Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e})}{\Pr(\mathbf{f})}$$

$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e})$$

Traditional Phrase-Based MT

$$\Pr(\mathbf{e}|\mathbf{f}) = \frac{\Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e})}{\Pr(\mathbf{f})}$$

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e})\Pr(\mathbf{f}|\mathbf{e})$$

Explicitly modeling alignment

$$\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} \Pr(\mathbf{f},\mathbf{a}|\mathbf{e})$$

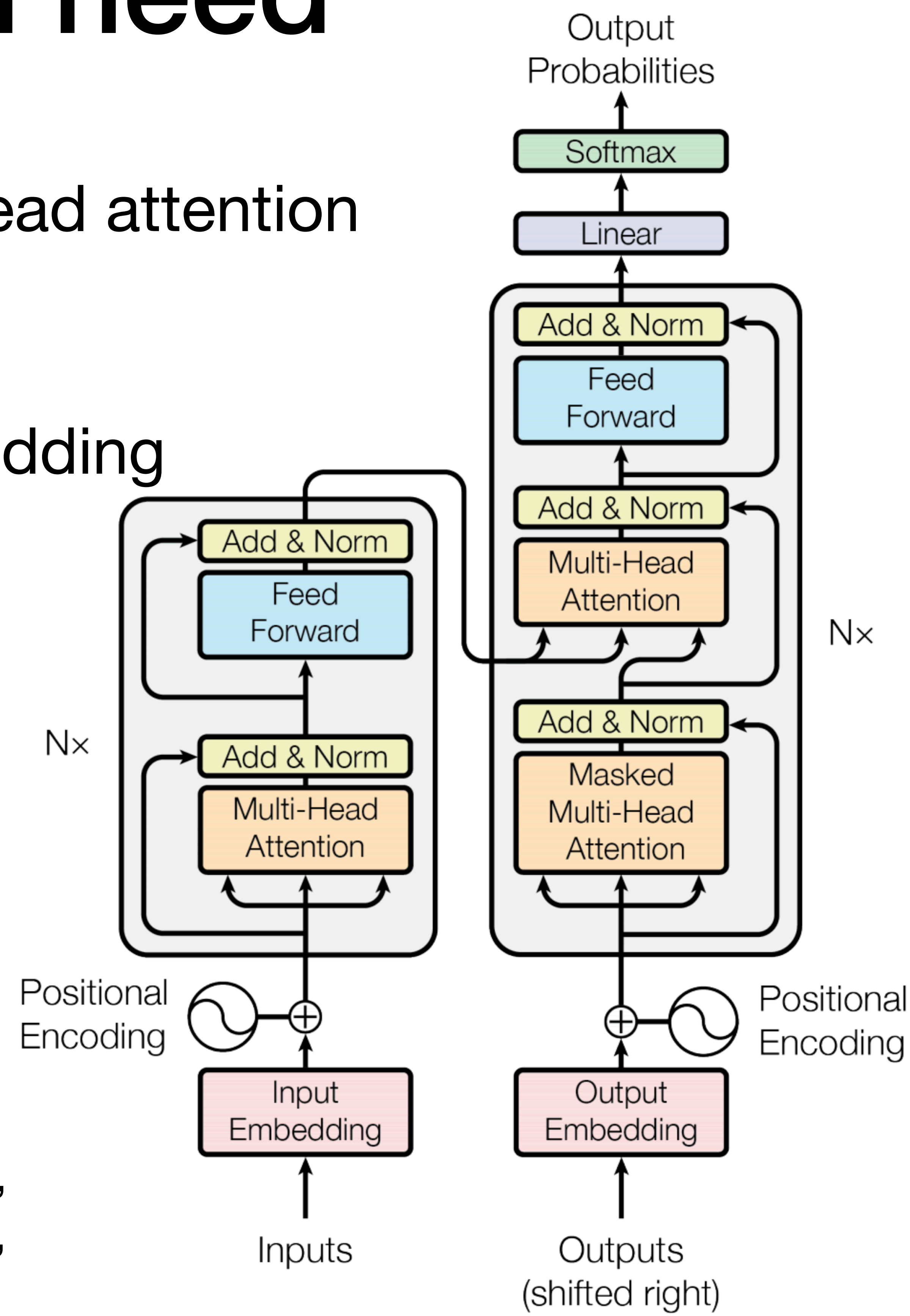
$$\Pr(\mathbf{f},\mathbf{a}|\mathbf{e}) = \Pr(m|\mathbf{e}) \prod_{j=1}^{m} \Pr(a_j|a_1^{j-1}, f_1^{j-1}, m, \mathbf{e}) \Pr(f_j|a_1^{j}, f_1^{j-1}, m, \mathbf{e})$$

IBM Models 1—5: A spectrum of simplifications

Brown, P.F., Pietra, V.J.D., Pietra, S.A.D. and Mercer, R.L., 1993. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2), pp.263-311.

Attention is all you need

- Information processed by multi-head attention
- Sinusoidal position embedding
  - In BERT: learned position embedding



Transformer

(Horrible terminology)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. Attention is all you need. In NIPS, 2017.

Attention beyond MT

- Attention probability is essentially a softmax
  - with varying # of target classes

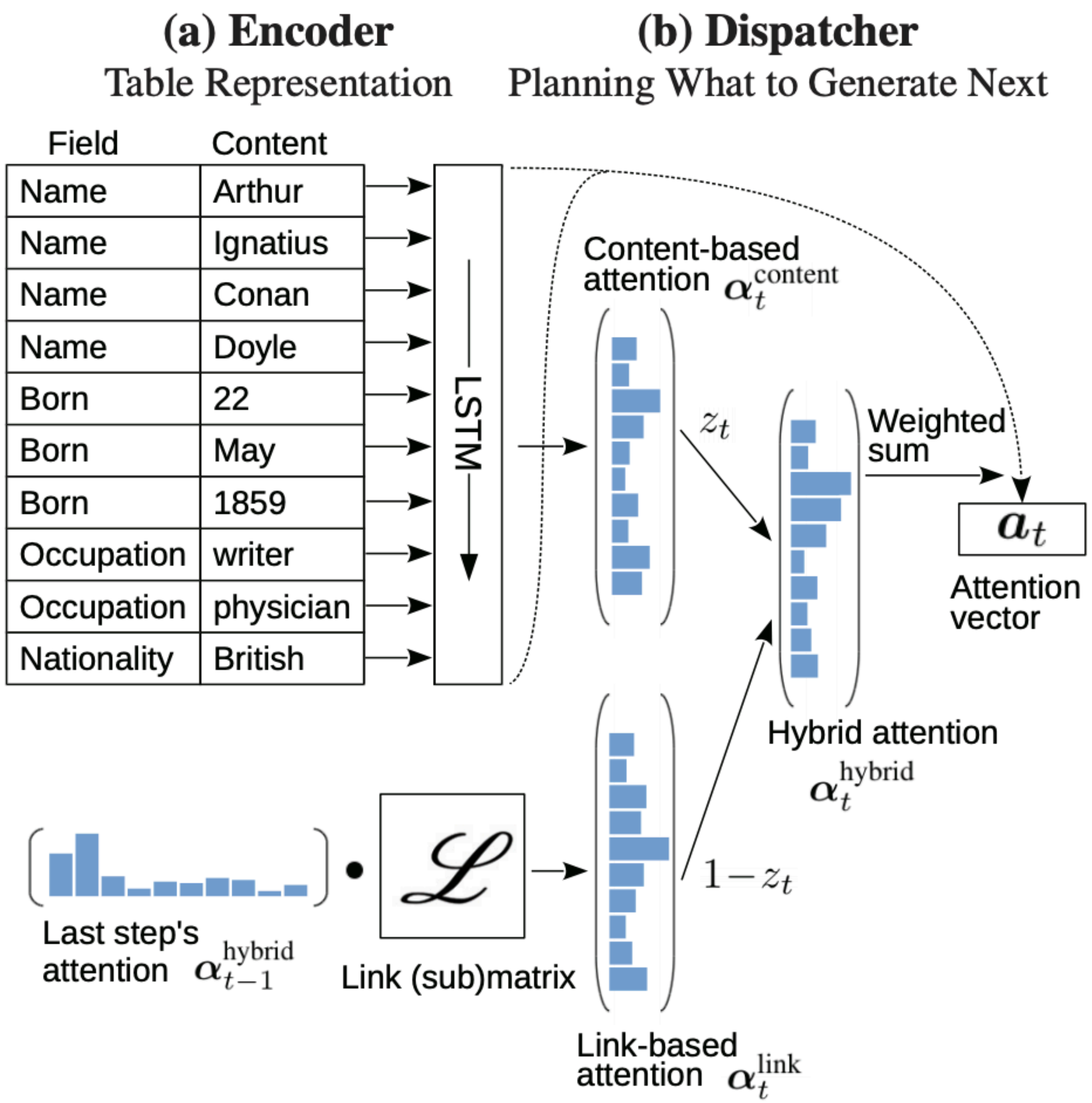
- Attention content is aggregating information by weighted sum
  - Especially # of entities may change

More Applications

- Dialogue systems
- Summarization
- Paraphrase generation
- etc.

Encoder does not have to be a sequence model

- Table-to-test generation
- Graph-to-text generation



Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, Zhifang Sui. Order-planning neural text generation from structure data. In *AAAI*, 2018.

Memory-Based Network

Question answering (synthetic dataset)

```
Sam walks into the kitchen.
Sam picks up an apple.
Sam walks into the bedroom.
Sam drops the apple.
Q: Where is the apple?
A. Bedroom
```

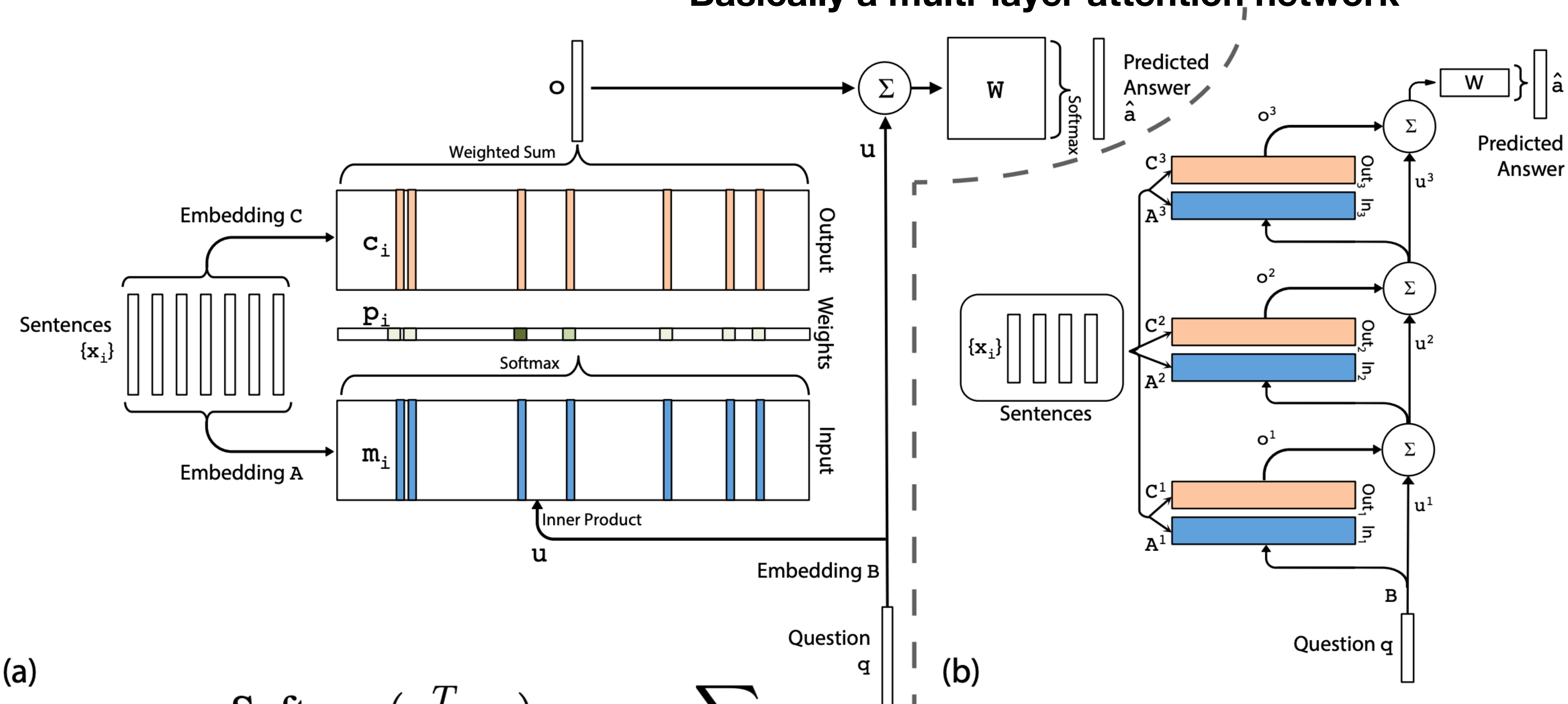
```
Brian is a lion.
Julius is a lion.
Julius is white.
Bernhard is green.
Q: What color is Brian?
A. White
```

```
Mary journeyed to the den.
Mary went back to the kitchen.
John journeyed to the bedroom.
Mary discarded the milk.
Q: Where was the milk before the den?
A. Hallway
```

Weston, Jason, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.
Sukhbaatar, S., Weston, J. and Fergus, R., 2015. End-to-end memory networks. In *NIPS*, 2015.

Memory-Based Network

Basically a multi-layer attention network



$$p_i = \mathrm{Softmax}(u^T m_i) \qquad o = \sum_i p_i c_i$$

Weston, Jason, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

Sukhbaatar, S., Weston, J. and Fergus, R., 2015. End-to-end memory networks. In *NIPS*, 2015.

Neural Turing Machine

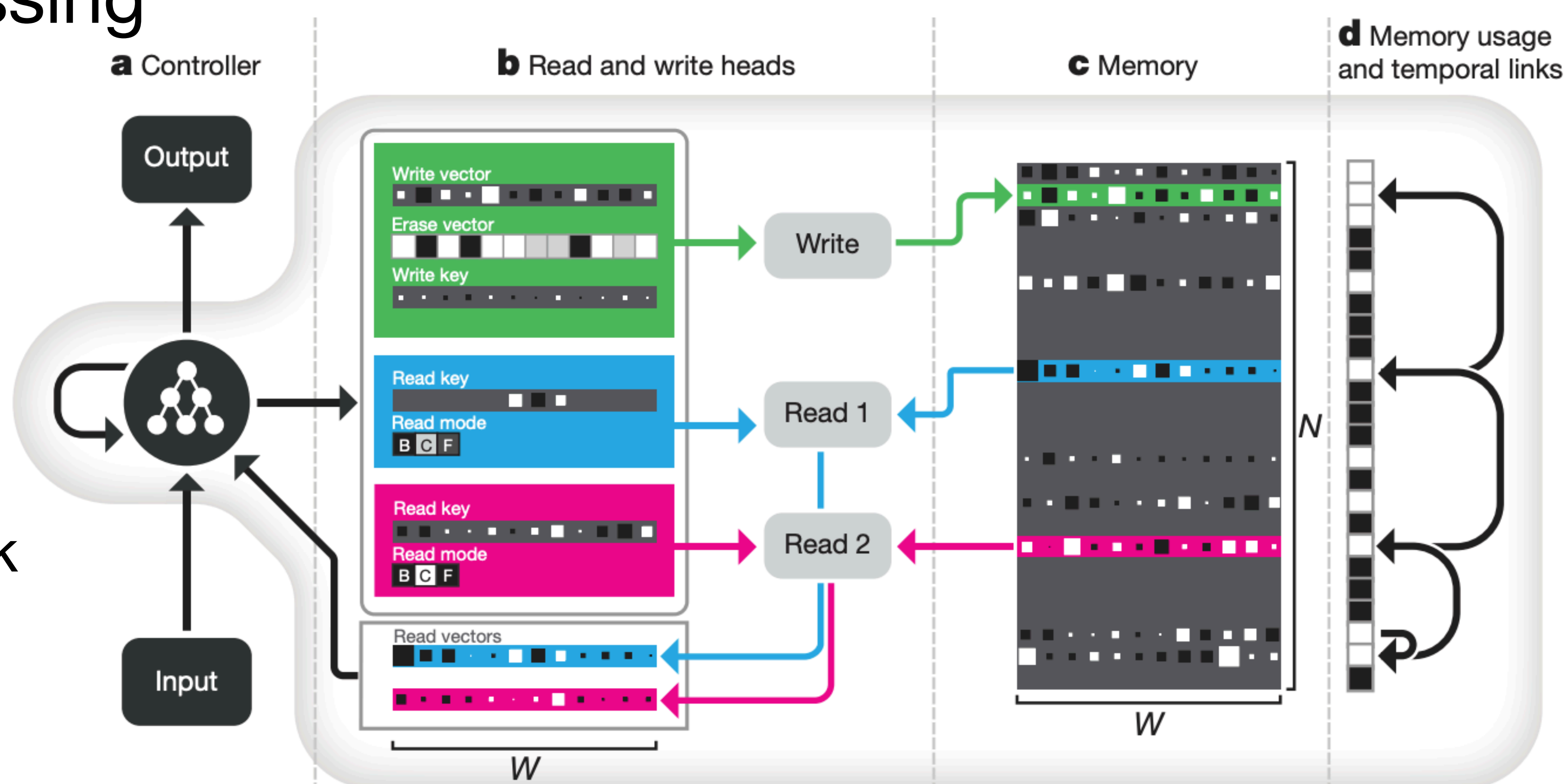
Chomsky hierarchy	Grammar	Language	Automata	Neural analog
Type-3	$A \rightarrow aB \mid a$	Regular expression	Finite state machine	RNN
Type-2	$A \rightarrow a$	Context-free	ND Pushdown automata	
Type-1	$\alpha A \beta \rightarrow \alpha \gamma \beta$	Context-sensitive	Esoteric	
Type-0	$\alpha A \beta \rightarrow \gamma$	Recursive enumerable	Turing machine	NTM

A Rough Thought on RNN Computational Power

- If states are discrete, RNN is FSM
- # distinct states ∝ exp(units)
- However, they are not free states
  - Transitions subject to a parametric function
- Unknown (at least to me) how real-valued states add to computational power
- At least, using denseness of real numbers to express potentially infinite steps of recursion is inefficient

Neural Turing Machine

- Augment RNN with a memory pad
- Read & write by memory addressing
- Attention-based memory addressing
  - Content-based addressing
  - Allocation-based addressing
  - Link-based addressing



Graves, A., et al., 2016. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626), p.471.

Neural Turing Machine

Content-based memory addressing

$$\mathcal{C}(M, \boldsymbol{k}, \beta)[i] = \frac{\exp\{\mathcal{D}(\boldsymbol{k}, M[i, \cdot\,])\beta\}}{\sum_j \exp\{\mathcal{D}(\boldsymbol{k}, M[j, \cdot\,])\beta\}}$$

Dynamic memory allocation

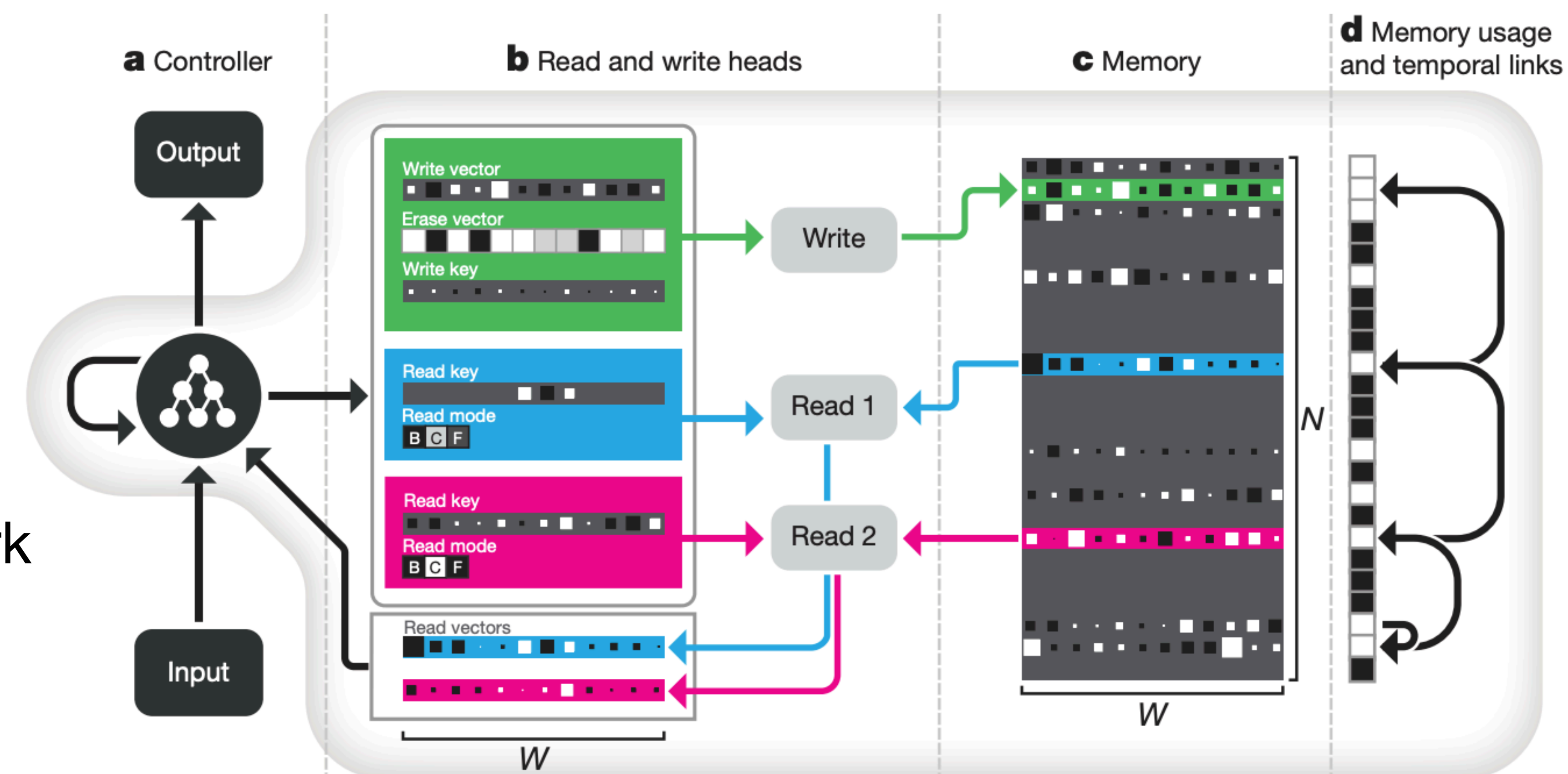
$$\psi_t = \prod_{i=1}^{R} \left(\mathbf{1} - f_t^i \mathbf{w}_{t-1}^{\mathrm{r},i}\right)$$

Temporal linkage-based memory addressing

$$L_0[i,j] = 0 \quad \forall\, i,j$$

$$L_t[i,i] = 0 \quad \forall\, i$$

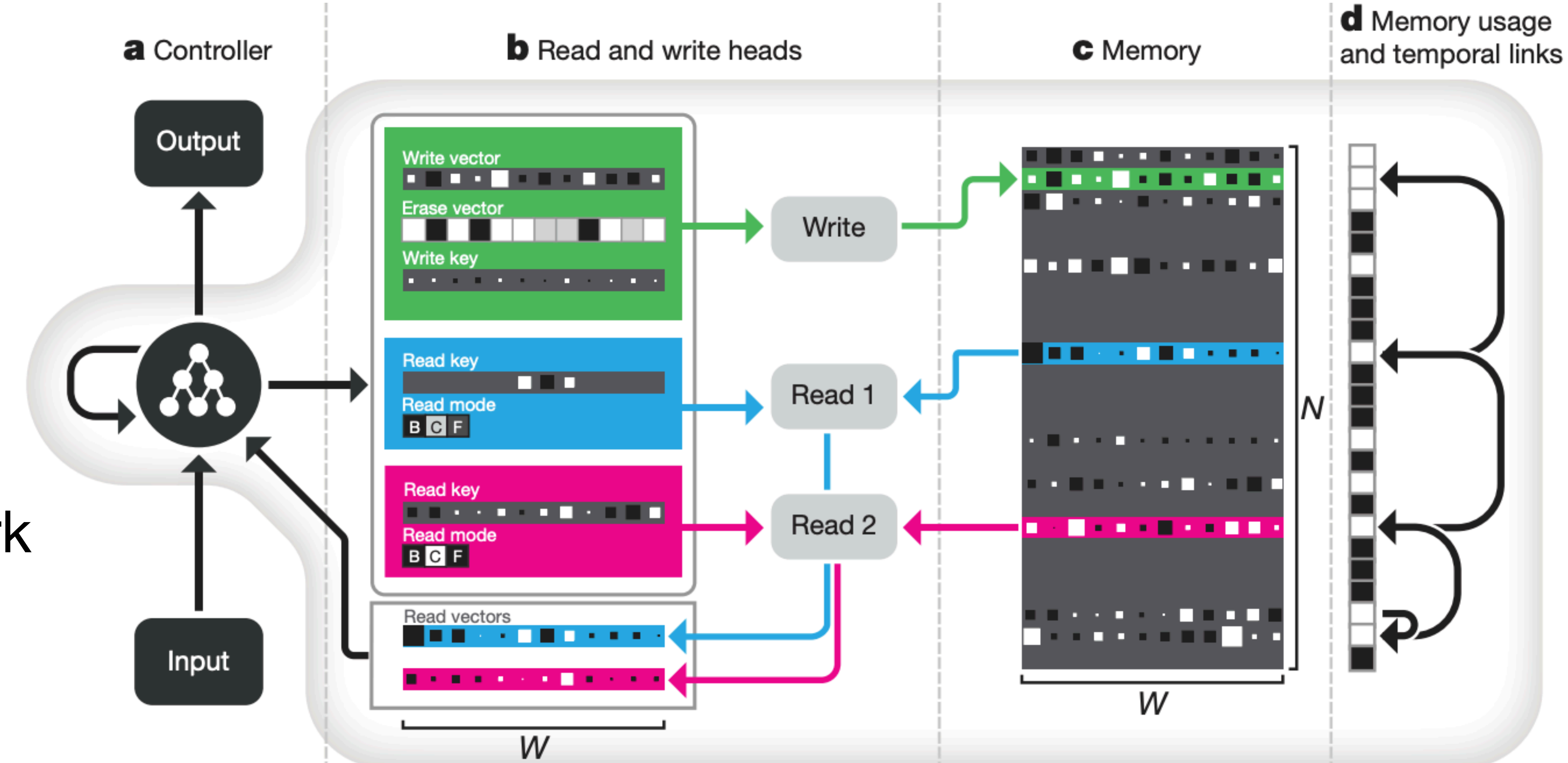
$$L_t[i,j] = (1 - \boldsymbol{w}_t^{\mathrm{w}}[i] - \boldsymbol{w}_t^{\mathrm{w}}[j]) L_{t-1}[i,j] + \boldsymbol{w}_t^{\mathrm{w}}[i] \boldsymbol{p}_{t-1}[j]$$



Graves, A., et al., 2016. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626), p.471.

Issues with NTM

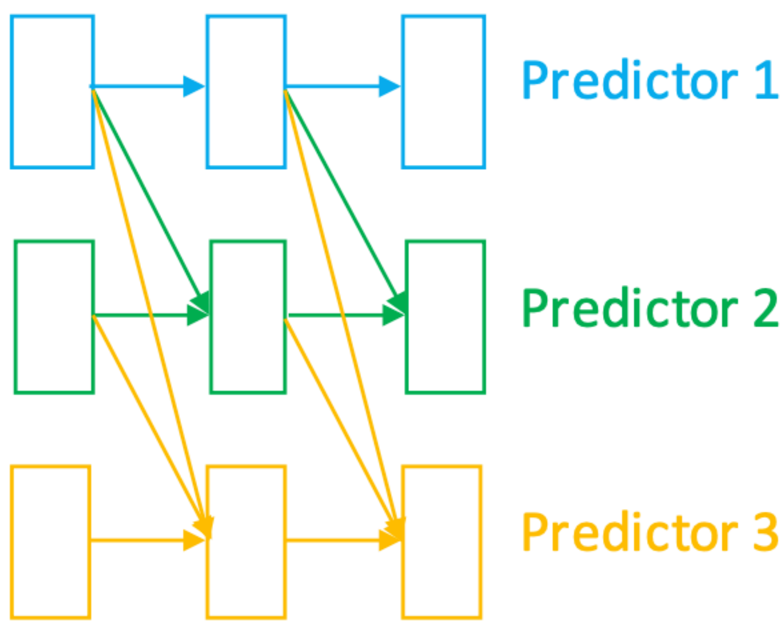
- Memory addressing is purely hypothetical
- May not learn true "programs"
- Thoughts for future work
  - Learn from restricted class of automata (e.g., PDA)
  - Make intermediate execution results Markov blanket



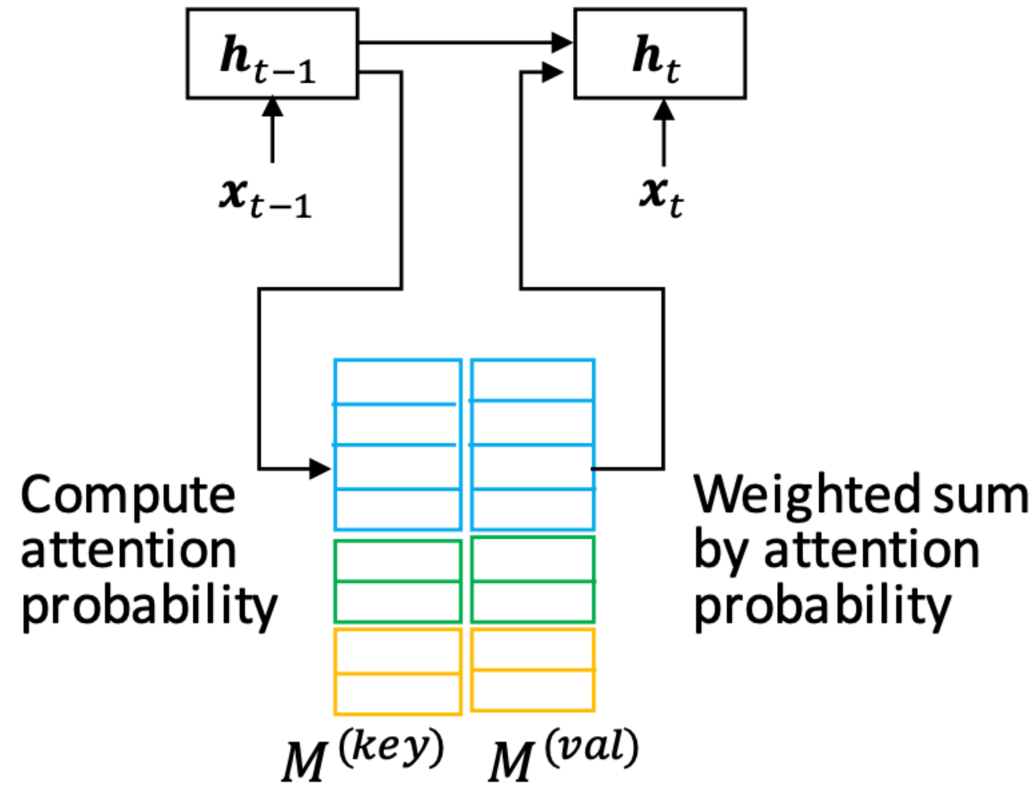
Graves, A., et al., 2016. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626), p.471.

Memory for Domain Adaptation

(a) Progressive neural network



(b) Progressive memory



Theorem 1. *Let RNN have vanilla transition with the linear activation function, and let the RNN state at the last step $\boldsymbol{h}_{i-1}$ be fixed. For a particular data point, if the memory attention satisfies $\sum_{j=N+1}^{N+M} \widetilde{\alpha}_{i,j} \leq \sum_{j=1}^{N} \widetilde{\alpha}_{i,j}$, then memory expansion yields a lower expected mean squared difference in $\boldsymbol{h}_i$ than RNN state expansion. That is,*

$$\mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{m})} - \boldsymbol{h}_i\|^2\right] \leq \mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{s})} - \boldsymbol{h}_i\|^2\right] \tag{9}$$

where $\boldsymbol{h}_i^{(\mathrm{m})}$ refers to the hidden states if the memory is expanded. $\boldsymbol{h}_i^{(\mathrm{s})}$ refers to the original dimensions of the RNN states, if we expand the size of RNN states themselves. Here, we compute the expectation by assuming weights and hidden states are iid from a zero-mean Gaussian distribution (with variance σ^2).

Asghar, N., Mou, L., Selby, K.A., Pantasdo, K.D., Poupart, P. and Jiang, X., 2018. Progressive Memory Banks for Incremental Domain Adaptation. *arXiv preprint arXiv:1811.00239*.

Conclusion & Take-Home Msg

- Sequence-to-sequence training
  - Training: Step-by-step supervised learning
  - Inference: Greedy, Beam search, sampling
- Attention
  - Adaptive weighted sum of source information
  - Alignment in MT
  - Aggregated information

Suggested Reading

- A course on automata theory
- Brown, P.F., Pietra, V.J.D., Pietra, S.A.D. and Mercer, R.L., 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2), pp.263-311.
- Graves, A., et al., 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626), p.471.

More References

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. Attention is all you need. In *NIPS*, 2017.
- Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, Zhifang Sui. Order-planning neural text generation from structure data. In *AAAI*, 2018.
- Weston, Jason, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.
- Sukhbaatar, S., Weston, J. and Fergus, R., 2015. End-to-end memory networks. In *NIPS*, 2015.
- Asghar, N., Mou, L., Selby, K.A., Pantasdo, K.D., Poupart, P. and Jiang, X., 2018. Progressive Memory Banks for Incremental Domain Adaptation. *arXiv preprint arXiv: 1811.00239*.

Thank you!

Q&A