Deep Learning for Natural Language Processing

Stephen Clark et al.
University of Cambridge and DeepMind

13. Sentence Representations

Edward Grefenstette
DeepMind & UCL

How do we represent sentence meaning?

- Is it a logical expression?
- Is it a vector? Many vectors?
- Is there just one notion of meaning, or is it task/context-dependent?

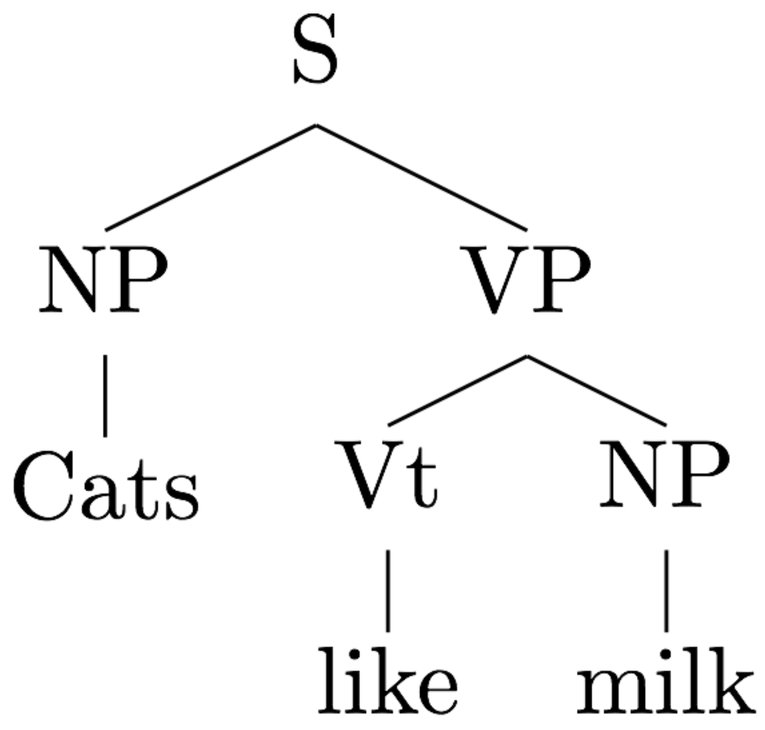
Montague Semantics

- Sentence representations are logical expressions.
- Sentence understanding is parsing and combining constituents to obtain logical form.
- Syntax guides semantics.

Montague Semantics

Syntactic Analysis	Semantic Interpretation
S $\Rightarrow$ NP VP	$[\![VP]\!]([\![NP]\!])$
NP $\Rightarrow$ cats, milk, etc.	$[\![\text{cats}]\!]$, $[\![\text{milk}]\!]$, $\ldots$
VP $\Rightarrow$ Vt NP	$[\![Vt]\!]([\![NP]\!])$
Vt $\Rightarrow$ like, hug, etc.	$\lambda yx.[\![\text{like}]\!](x, y)$, $\ldots$

Montague Semantics

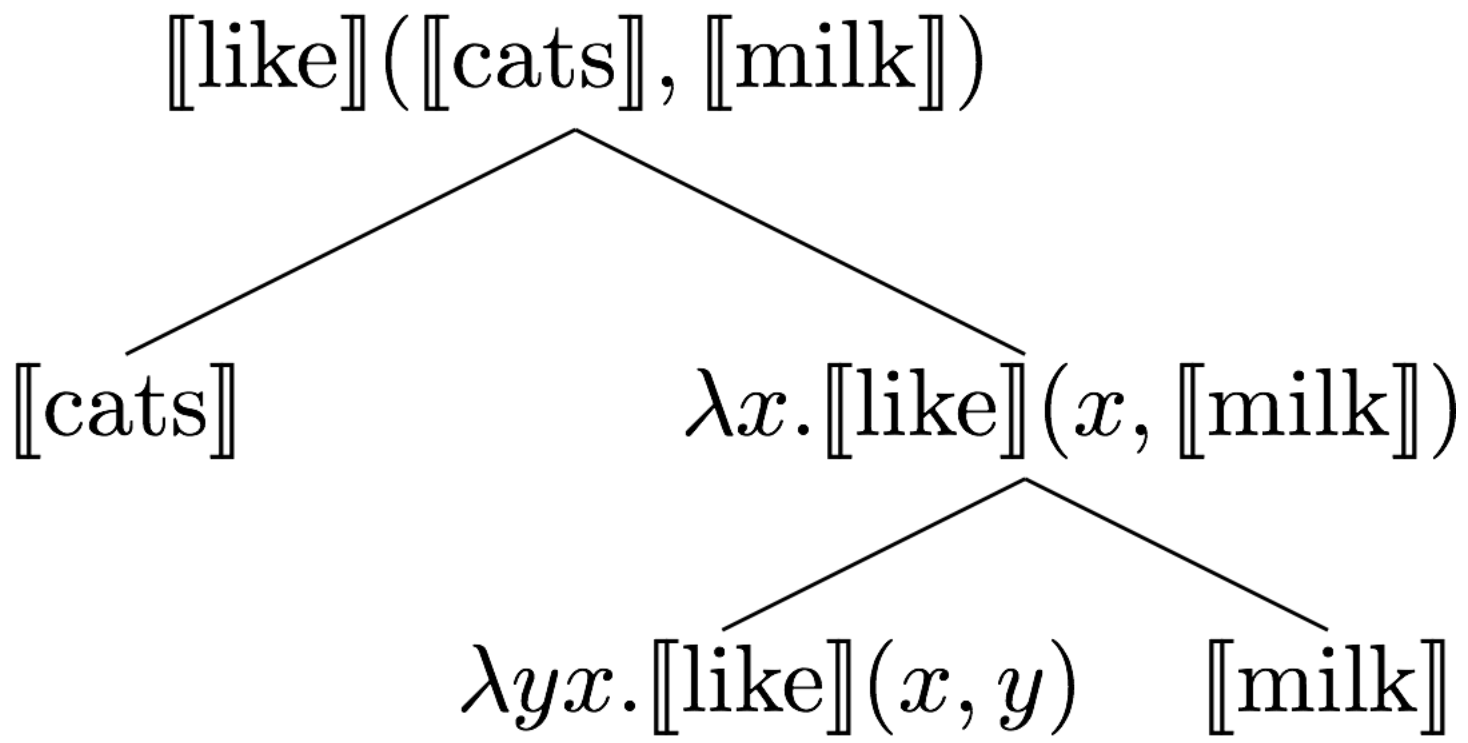


Cats like milk.

DeepMind

UNIVERSITY OF CAMBRIDGE

Montague Semantics


$[\![\text{like}]\!]([\![\text{cats}]\!], [\![\text{milk}]\!])$
$[\![\text{cats}]\!]$
$\lambda x.[\![\text{like}]\!](x, [\![\text{milk}]\!])$
$\lambda yx.[\![\text{like}]\!](x, y)$
$[\![\text{milk}]\!]$

Cats like milk.

Montague Semantics

Pros:

- Intuitive and interpretable(?) representations.
- Leverage the power of predicate logic to model semantics.
- Evaluate the truth of statements, derive conclusions, etc.

Montague Semantics

Cons:

- Brittle, requires robust parsers.
- Extensive logical model required for evaluation of clauses.
- Extensive set of rules required to do anything useful.
- Overall, an intractable (or unappealing) learning problem.

Neural Networks as Models of Composition

- **Sentence Classification:** produce sentence representation, classify based on this representation

- **Seq2Seq:** Produce sentence representation, generate another sentence conditioned on this representation

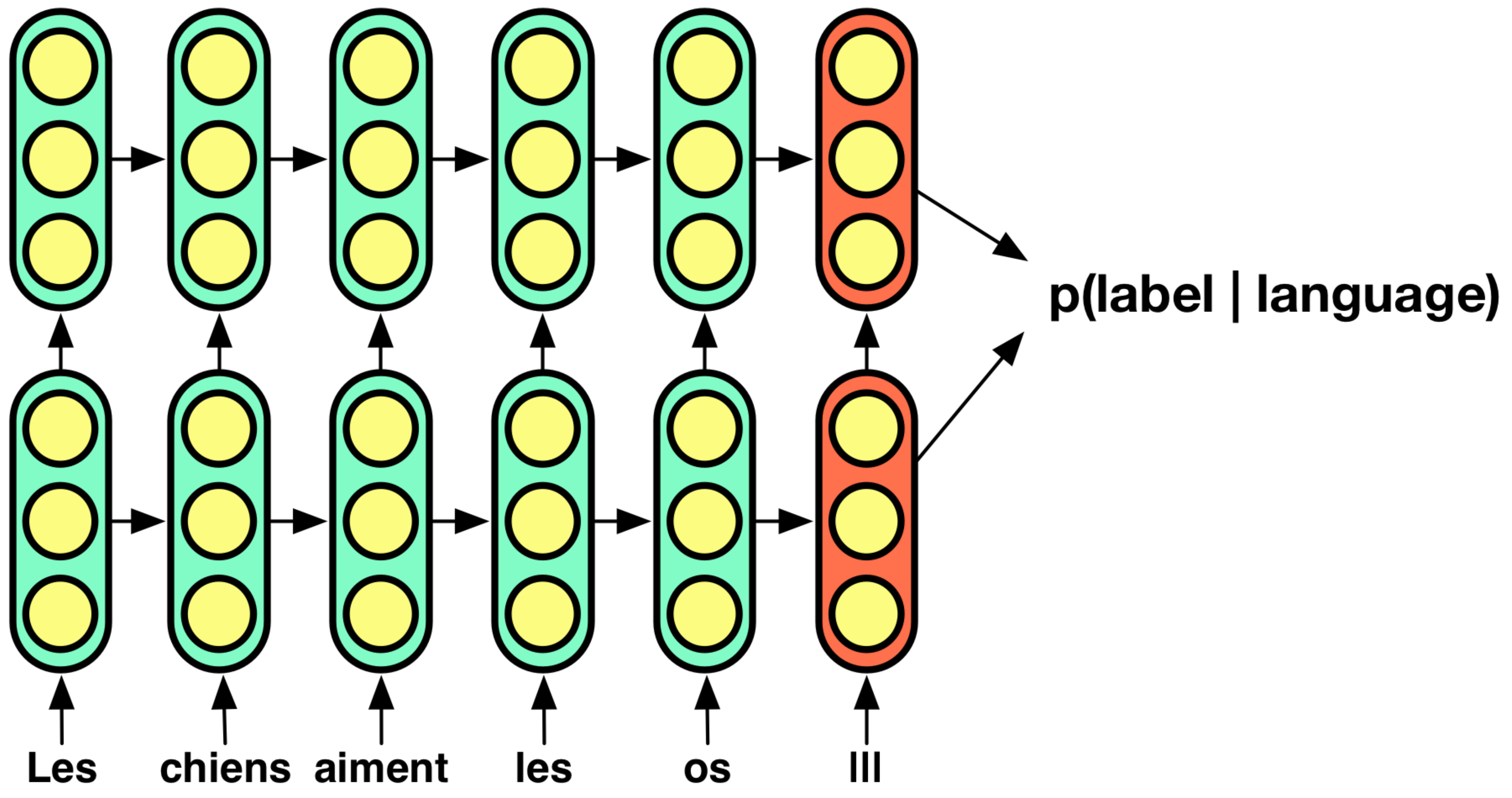
What models of composition?

What <u>objectives</u>?

Algebraic Encoders

$$
\begin{aligned}
\text{General form:} \quad & \mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K) \\
\text{Add:} \quad & \mathbf{p} = \mathbf{u} + \mathbf{v} \\
\text{WeightAdd:} \quad & \mathbf{p} = \alpha^T \mathbf{u} + \beta^T \mathbf{v} \\
\text{Multiplicative:} \quad & \mathbf{p} = \mathbf{u} \otimes \mathbf{v} \\
\text{Combined:} \quad & \mathbf{p} = \alpha^T \mathbf{u} + \beta^T \mathbf{v} + \gamma^T (\mathbf{u} \otimes \mathbf{v})
\end{aligned}
$$

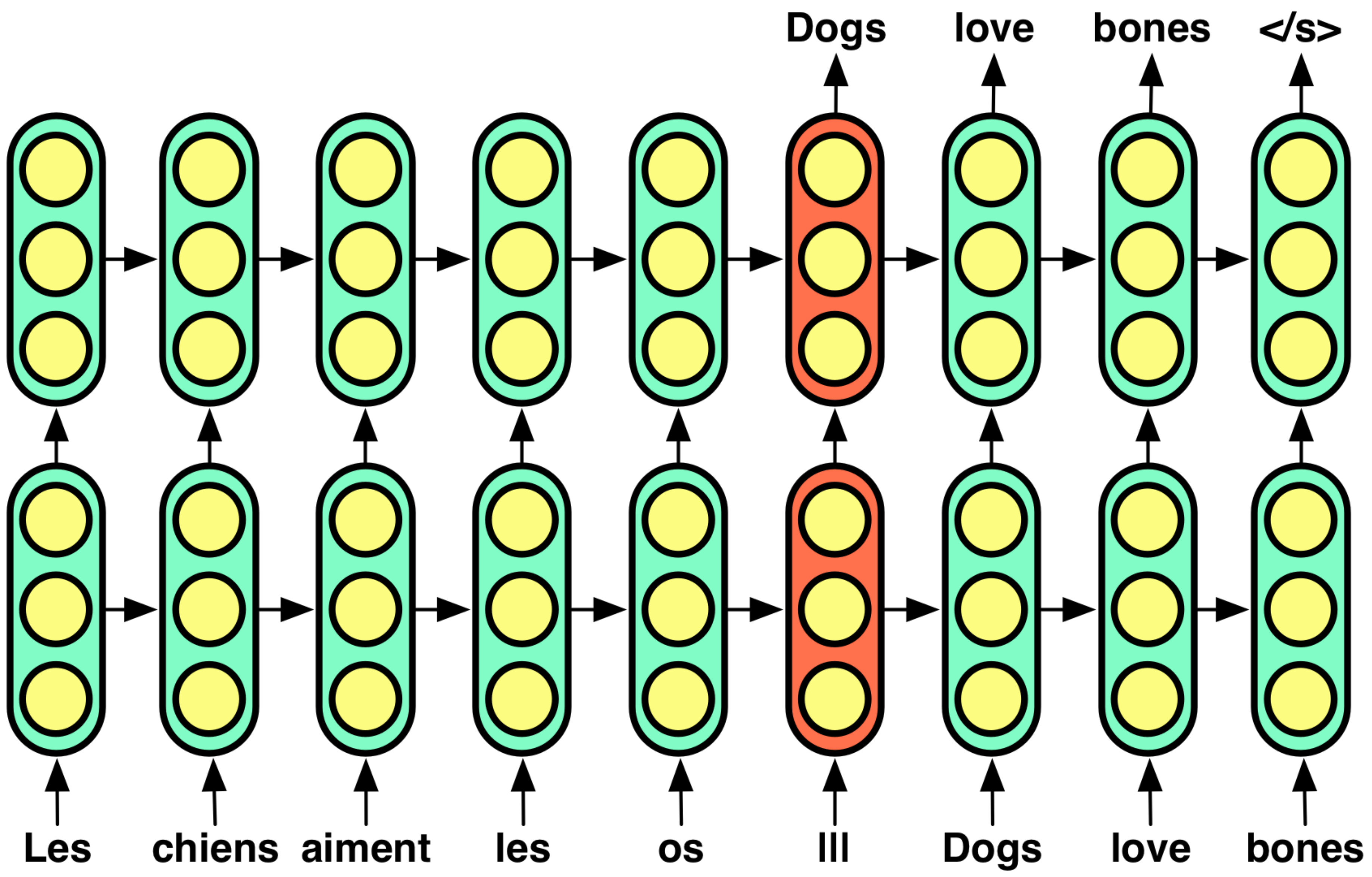
Sequence Encoders in Sequence Classification



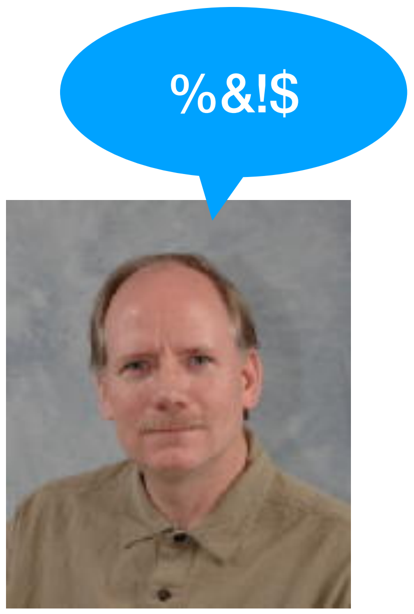
DeepMind

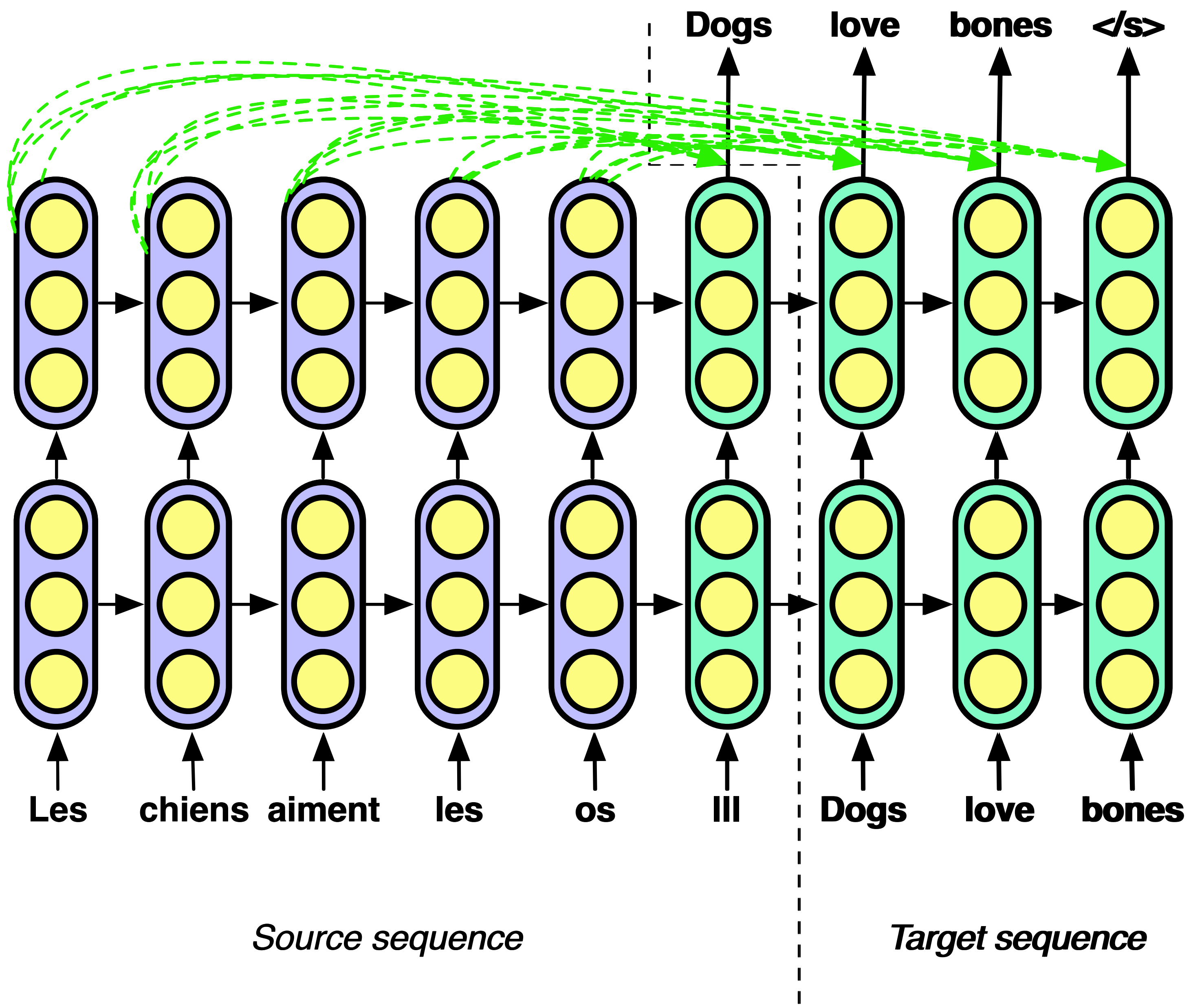
UNIVERSITY OF CAMBRIDGE

Sequence Encoders in Seq2Seq



Sentence Matrices as Sentence Representations





DeepMind

UNIVERSITY OF CAMBRIDGE

Incorporating Structure

Basic idea: use sampled or maximally likely parse tree to guide composition:

$$\mathbf{parent} = f_\theta(\mathbf{child}_1, \ldots, \mathbf{child}_n)$$

Perhaps additionally condition on the production rule:

$$\mathbf{parent} = f_\theta(\mathbf{child}_1, \ldots, \mathbf{child}_n; \mathrm{A} \rightarrow \mathrm{B\ C})$$

f can be any differentiable function. There are RNN-style and LSTM-style updates in various papers.

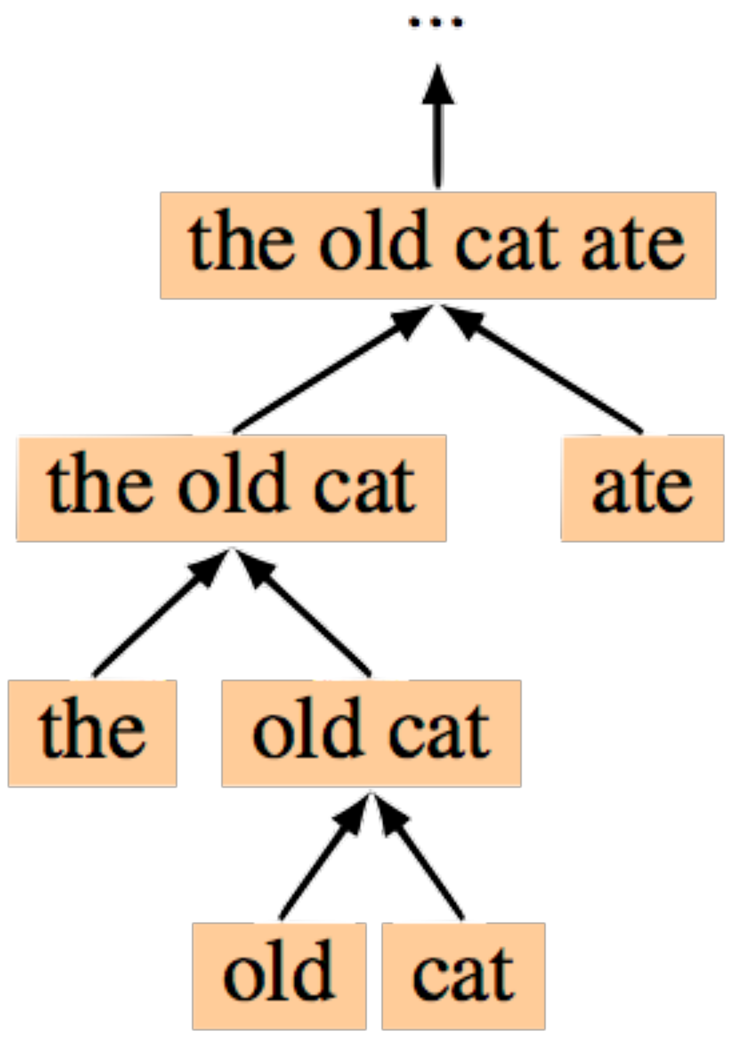
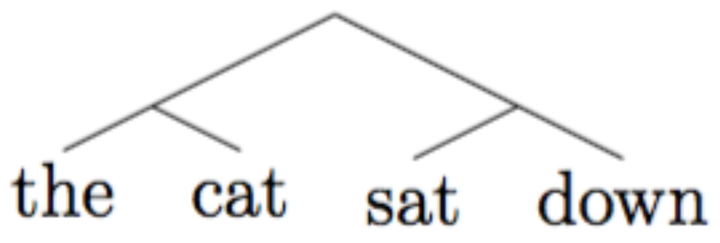
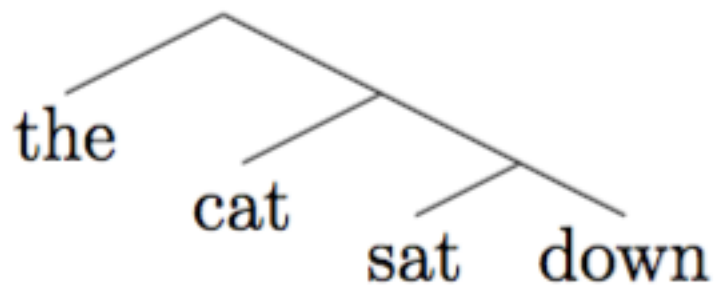
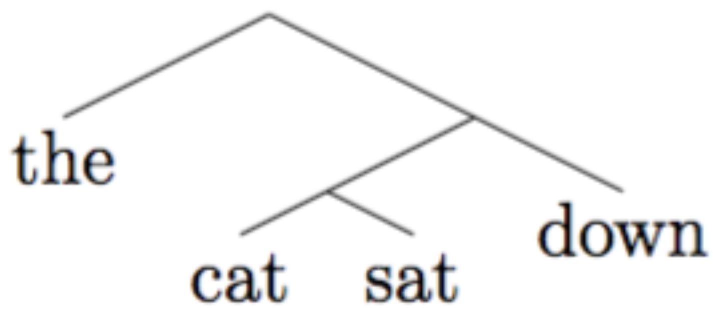


Figure due to Sam Bowman.
Reproduced with author's permission.

What if you don't have parses at test time?







[SHIFT, SHIFT, REDUCE, SHIFT, SHIFT, REDUCE, REDUCE]

[SHIFT, SHIFT, SHIFT, SHIFT, REDUCE, REDUCE, REDUCE]

[SHIFT, SHIFT, SHIFT, REDUCE, SHIFT, REDUCE, REDUCE]

Figure due to Sam Bowman.
Reproduced with author's permission.

DeepMind

UNIVERSITY OF CAMBRIDGE

SPINN

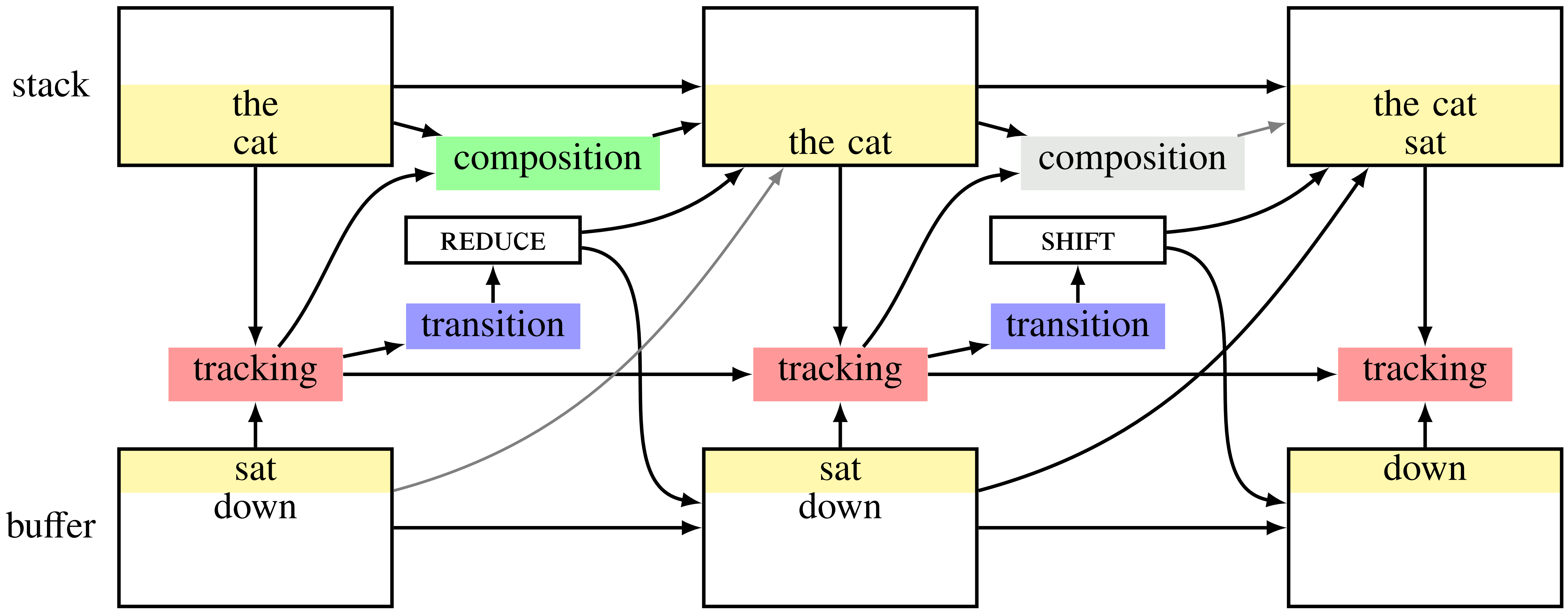


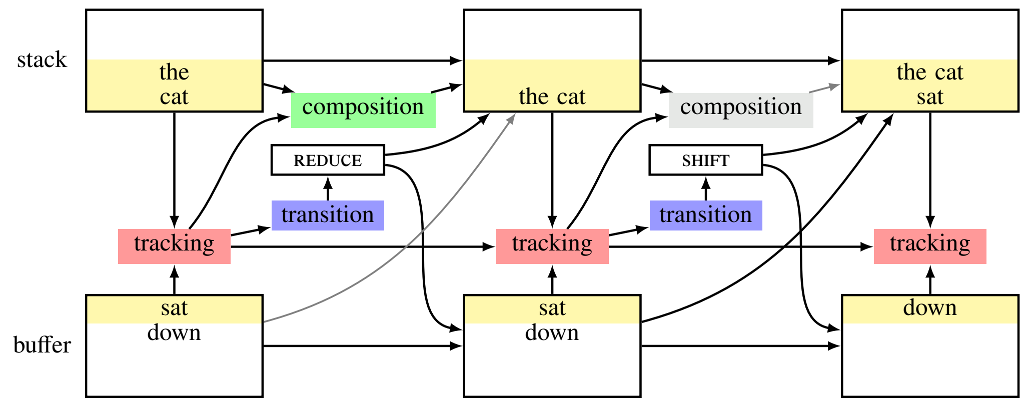
Figure due to Sam Bowman.
Reproduced with author's permission.

DeepMind

UNIVERSITY OF CAMBRIDGE

SPINN

Figure due to Sam Bowman.
Reproduced with author's permission.



Word vectors start on buffer *b*
(top: first word in sentence).

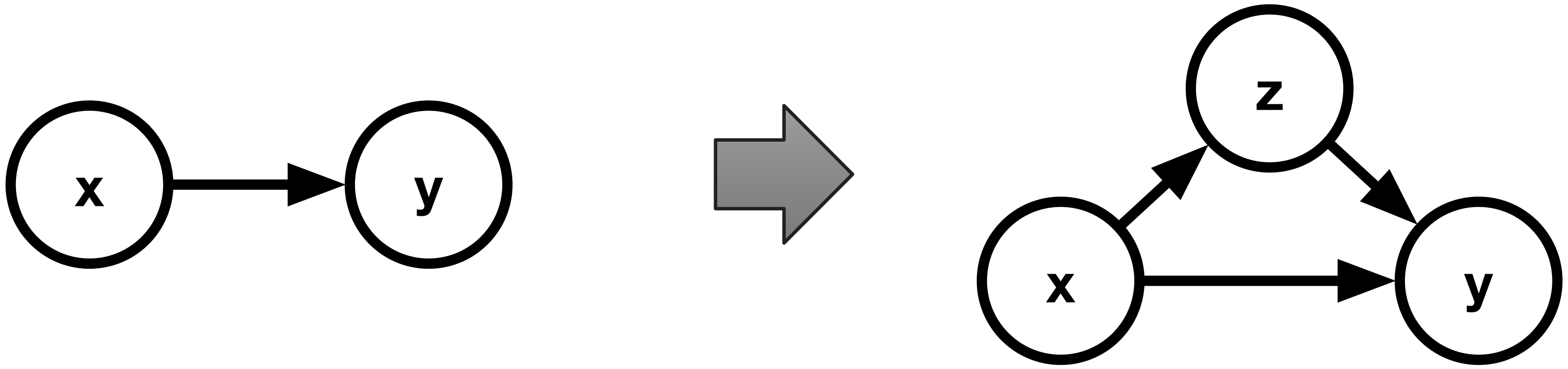
Shift moves word vectors from buffer to stack *s*.

Reduce pops top two vectors off stack, applies a function **f^R** to them, and pushes the result back on the stack.

Tracker LSTM tracks parser/composer state, decides shift/reduce actions *a*, is supervised by both observed shift-reduce operations and end task:

$$h_t = LSTM(f^C(b_{t-1}[0], s_{t-1}[0], s_{t-1}[1]), h_{t-1}) \qquad a_t \sim f^A(h_t)$$

What if we don't observe actions?



$$p(y|x) = \mathbb{E}_{p_\theta(z|x)}\left[f_\phi(z,x)\right] \quad \text{s.t. } y \sim f_\phi(z,x) \text{ or } y = f_\phi(z,x)$$

$$\nabla_\phi p(y|x) = \sum_z p_\theta(z|x)\nabla_\phi f_\phi(z,x) = \mathbb{E}_{p_\theta(z|x)}\left[\nabla_\phi f_\phi(z,x)\right]$$

$$\nabla_\theta p(y|x) = \sum_z f_\phi(z,x)\nabla_\theta p_\theta(z|x) = ???$$

DeepMind

UNIVERSITY OF CAMBRIDGE

What if we don't observe actions?

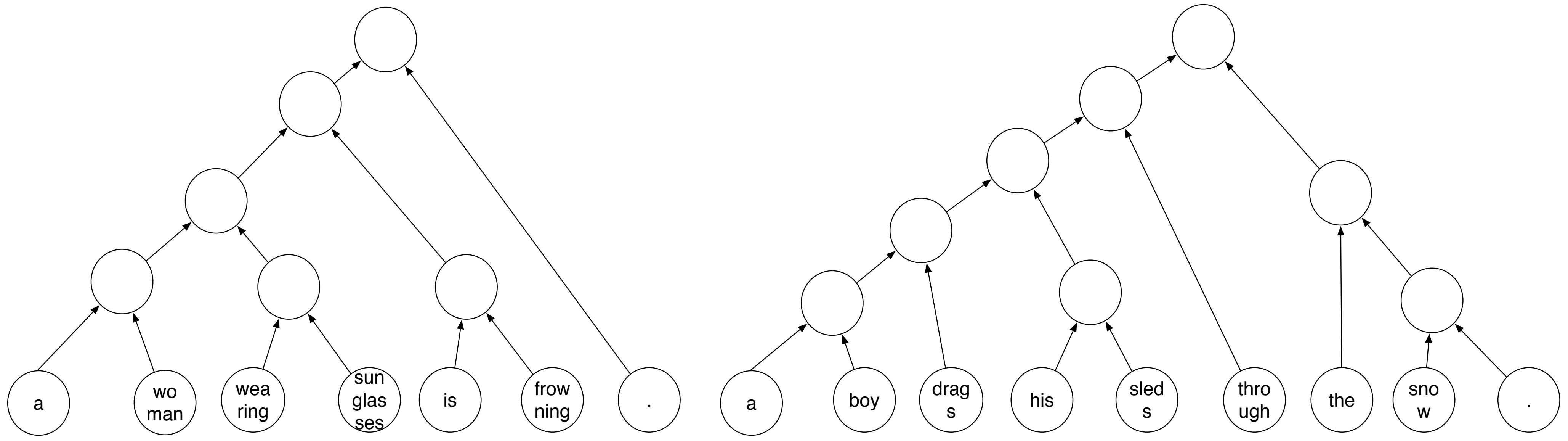
REINFORCE Log-Trick:

$$\nabla_\theta \log p_\theta(z|x) = \frac{\nabla_\theta p_\theta(z|x)}{p_\theta(z|x)} \quad \Rightarrow \quad \nabla_\theta p_\theta(z|x) = p_\theta(z|x) \nabla_\theta \log p_\theta(z|x)$$

$$\begin{aligned}
\nabla_\theta p(y|x) &= \sum_z f_\phi(z,x) \nabla_\theta p_\theta(z|x) \\
&= \sum_z f_\phi(z,x) p_\theta(z|x) \nabla_\theta \log p_\theta(z|x) \\
&= \mathbb{E}_{p_\theta(z|x)} \left[f_\phi(z,x) \nabla_\theta \log p_\theta(z|x) \right]
\end{aligned}$$

SPINN+REINFORCE

- Treat $\mathbf{a}_t \sim f^A(h_t)$ as a policy trained by REINFORCE.
- Reward is negated loss of the end task, e.g. NLL of correct label.
- Everything else is trained by backpropagation against the end task: tracker LSTM, representations, etc. receive gradient both from the supervised objective, and from Reinforce via the shift-reduce policy.

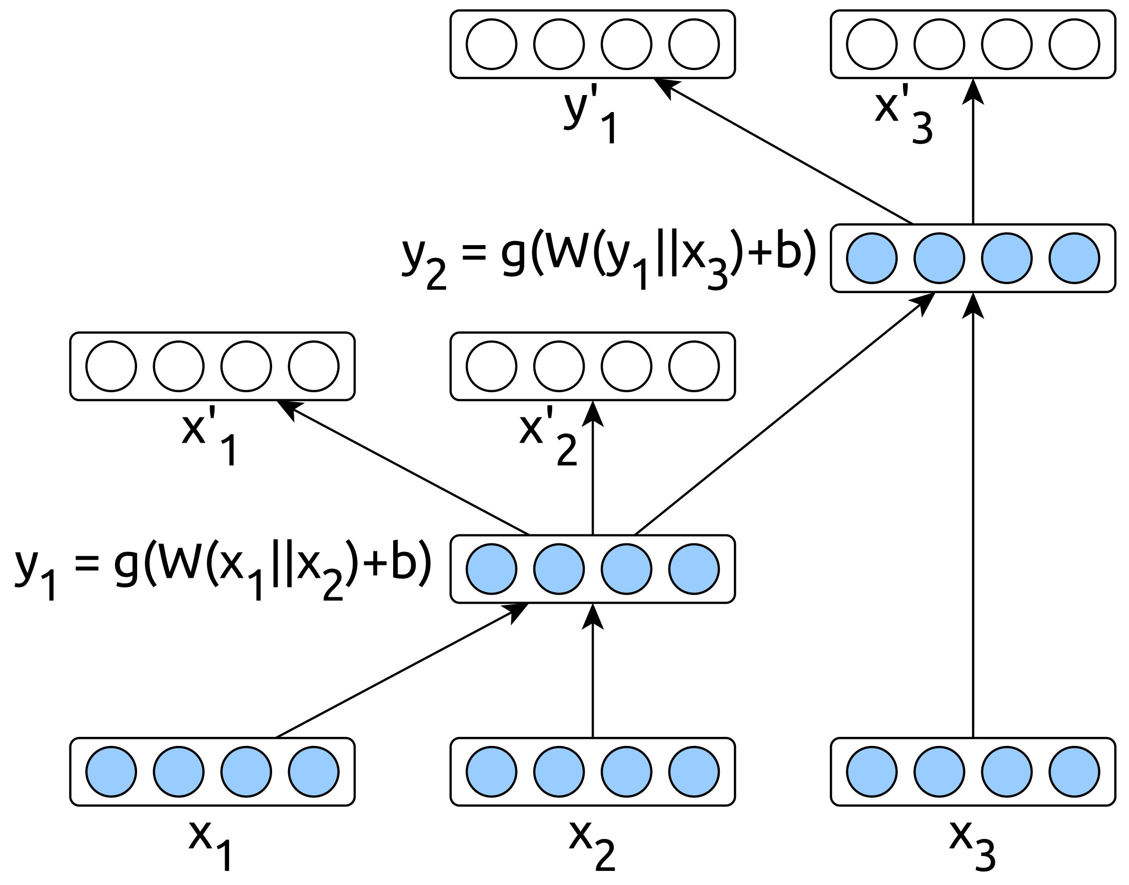


What objective?

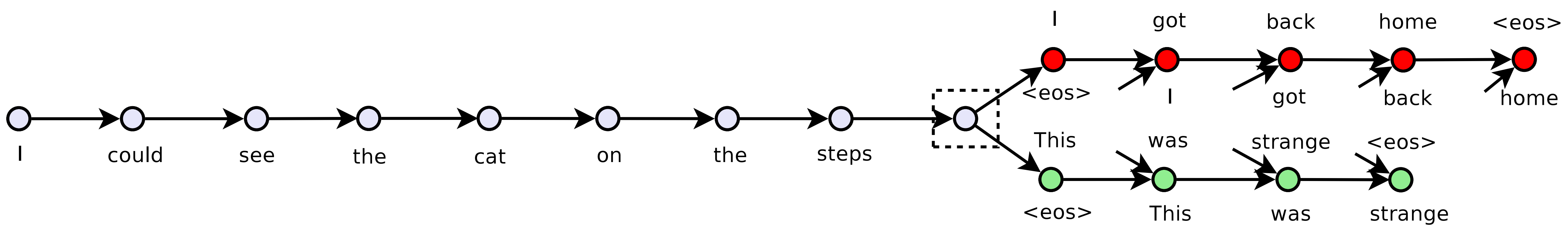
- Seq2Seq?
- Classification?
- Training against a specific objective will (trivially) produce representations that are useful for that objective?
- Can we get something more general?

Auto-Encoders

- Reconstruction objective includes nothing about distance preservation in latent space.
- Conversely, little incentive for similar latent codes to generate radically different (but semantically equivalent) observations.
- Generally, auto-encoders sparsely encode or densely compress information. No pressure to ensure similarity continuum amongst codes.



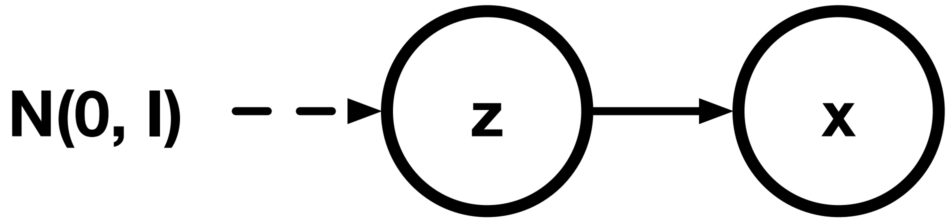
Skip Thought



- Similar to auto-encoding objective: encode sentence, but decode neighbouring sentences.
- Pair of LSTM-based seq2seq models with share encoder, but alternative formulations are possible.
- Conceptually similar to distributional semantics: a unit's representation is a function of its neighbouring units, except units are sentence instead of words.

Variational Auto-Encoders

$$\begin{aligned} p(x) &= \int p(x, z) dz \\ &= \int p(x|z)p(z) dz \\ &= \mathbb{E}_{p(z)}\left[p(x|z)\right] \end{aligned}$$



Prior on z enforces semantic continuum (e.g. no arbitrarily unrelated codes for similar data), but expectation is typically intractable to compute exactly, and Monte Carlo estimate of gradients will be high variance.

Variational Auto-Encoders

Goal is to estimate, by maximising $p(x)$:

- The parameters θ of a function modelling the part of the generative process $p_\theta(x|z)$ given samples from a fixed prior $z \sim p(z)$.
- The parameters ϕ of a *proposal distribution* $q_\phi(z|x)$ approximating the true posterior $p(z|x)$.

Variational Auto-Encoders

How do we do it?

We maximise p(x) via a variational lower bound (VLB):

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x) \| p(z)\right)$$

Equivalently we can maximise the NLL(x):

$$NLL(x) \leq \mathbb{E}_{q_\phi(z|x)}[NLL_\theta(x|z)] + D_{KL}\left(q_\phi(z|x) \| p(z)\right)$$

DeepMind

UNIVERSITY OF CAMBRIDGE

Deriving the VLB

$$\begin{aligned}
\log p(x) &= \log \int 1 \cdot p_\theta(x|z)p(z)dz \\
&= \log \int \frac{q_\phi(z|x)}{q_\phi(z|x)} p_\theta(x|z)p(z)dz \\
&= \log \mathbb{E}_{q_\phi(z|x)}\left[\frac{p(z)}{q_\phi(z|x)} p_\theta(x|z)\right] \\
&\geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(z)}{q_\phi(z|x)} + \log p_\theta(x|z)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x) \| p(z)\right)
\end{aligned}$$

For right $q_\phi(z|x)$ *and* $p(z)$ (e.g. Gaussians) there is a closed form expression of DKL($q_\phi(z|x)||p(z)$).

DeepMind

UNIVERSITY OF CAMBRIDGE

Variational Auto-Encoders

Estimating $\frac{\partial}{\partial \phi}\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$ requires backpropagating through samples. For some choices of *q* (e.g. Gaussians) this can be done through the use of *reparameterization tricks*.

$$\begin{aligned} z &\sim N(z;\mu,\sigma^2) \\ \text{equivalent to } z &= \mu + \sigma\epsilon \\ \text{where } \epsilon &\sim N(\epsilon;\mathbf{0},\mathbf{I}) \end{aligned}$$

Variational Auto-Encoders for Text

1. Observe a sentence $w_1, \ldots, w_n$. Encode it, e.g. with an LSTM: $h^e = LSTM^e(w_1, \ldots, w_n)$
2. Predict $\mu = f^\mu(h^e)$ and $\sigma^2 = f^\sigma(h^e)$ (in practice we operate in log space for σ^2 by determining $\log \sigma$).
3. Sample $z \sim q(z|x) = N(z; \mu, \sigma^2)$
4. Use conditional RNN to decode and measure $\log p(x|z)$. Use closed-form formula of KL divergence of two Gaussians to calculate $-D_{KL}(q_\phi(z|x) \| p(z))$. Add both to obtain maximisation objective.
5. Backpropagate gradient through decoder normally based on log component of the objective, and use reparameterisation trick to backpropagate through sampling operation back to encoder.
6. Gradient of the KL divergence component of the loss with regard to the encoder parameters is straightforward backpropagation.

DeepMind

UNIVERSITY OF CAMBRIDGE

Some issues

- If decoder is powerful enough to model p(x) without exploiting the latent variable, the model will learn to ignore z and the VAE-encoder is useless.
- This is what happens for LSTMs and other powerful autoregressive models.
- Some “hacky” solutions (e.g. KL annealing) exist, but this is an open research problem.
- In short, VAEs are promising, but not (yet) the solution for unsupervised sentence representation learning.

DeepMind

UNIVERSITY OF CAMBRIDGE

Reading

Montague, R. (1970). English as a formal language.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. *proceedings of ACL-08: HLT, 236-244*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631-1642).

Bowman, S. R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C. D., & Potts, C. (2016). *A fast unified model for parsing and sentence understanding. arXiv preprint arXiv:1603.06021.*

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning* (pp. 5-32). Springer, Boston, MA.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., & Ling, W. (2016). Learning to compose words into sentences with reinforcement learning. *ICLR 2017*.

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., & Ng, A. Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems* (pp. 801-809).

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349.