Conditional Language Modeling with Attention

Chris Dyer

DeepMind

Carnegie Mellon University

Review: Conditional LMs

A **conditional language model** assigns probabilities to sequences of words, $\boldsymbol{w} = (w_1, w_2, \ldots, w_\ell)$, given some conditioning context, $\boldsymbol{x}$.

As with unconditional models, it is again helpful to use the chain rule to decompose this probability:

$$p(\boldsymbol{w} \mid \boldsymbol{x}) = \prod_{t=1}^{\ell} p(w_t \mid \boldsymbol{x}, w_1, w_2, \ldots, w_{t-1})$$

*What is the probability of the next word, given the history of previously generated words **and** conditioning context $\boldsymbol{x}$?*

Sutskever et al. (2014)

明子は

ピムスが

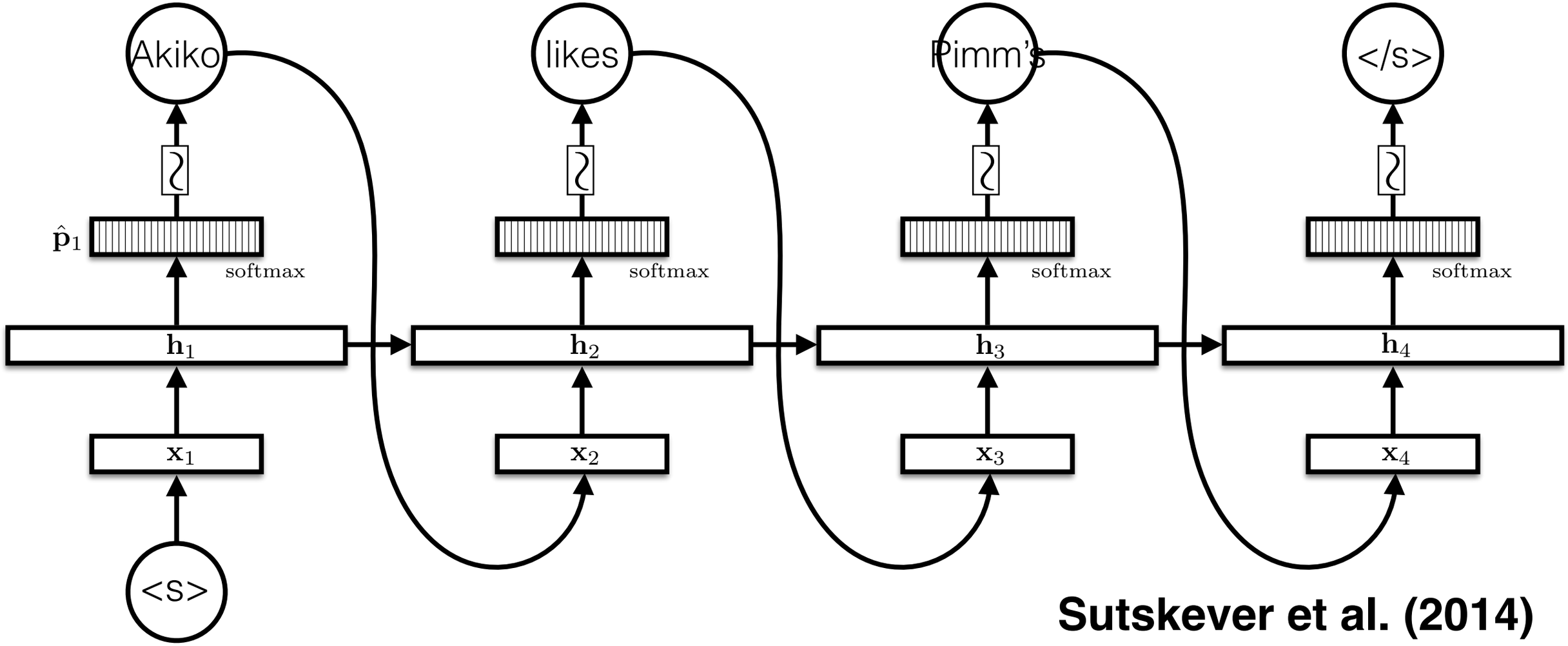
好きです

Sutskever et al. (2014)

明子は

ピムスが

好きです



Sutskever et al. (2014)

明子は

ピムスが

好きです

Akiko

likes

Pimm's

</s>

<S>

$\hat{\mathbf{p}}_1$

softmax

softmax

softmax

softmax

$\mathbf{h}_1$

$\mathbf{h}_2$

$\mathbf{h}_3$

$\mathbf{h}_4$

$\mathbf{x}_1$

$\mathbf{x}_2$

$\mathbf{x}_3$

$\mathbf{x}_4$

Sutskever et al. (2014)

Conditiong with vectors

We are compressing a lot of information in a finite-sized vector.

Conditiong with vectors

We are compressing a lot of information in a finite-sized vector.

Prof. Ray Mooney

“You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!”

Conditiong with vectors

We are compressing a lot of information in a finite-sized vector.

Gradients have a long way to travel. Even LSTMs forget!

Conditiong with vectors

We are compressing a lot of information in a finite-sized vector.

Gradients have a long way to travel. Even LSTMs forget!

What is to be done?

Outline of Lecture

- Machine translation with attention
- Image caption generation with attention

Solving the Vector Problem in Translation

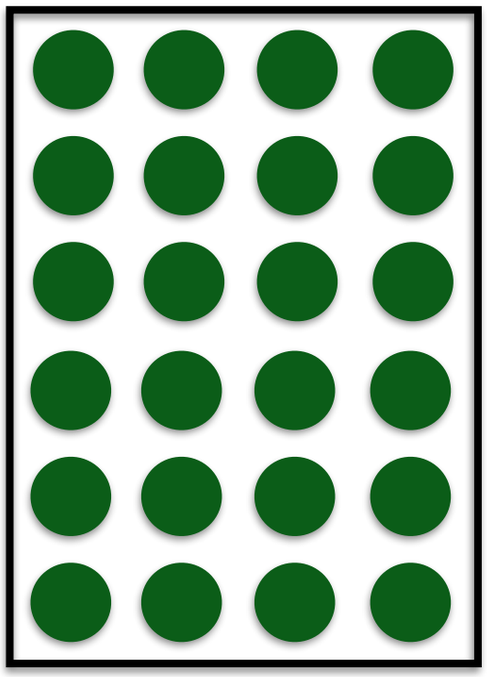
- Represent a source sentence as a matrix
- Generate a target sentence from a matrix

- This will
  - Solve the capacity problem
  - Solve the gradient flow problem

Sentences as Matrices

- Problem with the fixed-size vector model
  - Sentences are of different sizes but vectors are of the same size
- Solution: use matrices instead
  - Fixed number of rows, but number of columns depends on the number of words
  - Usually $|\boldsymbol{f}|$ = #cols

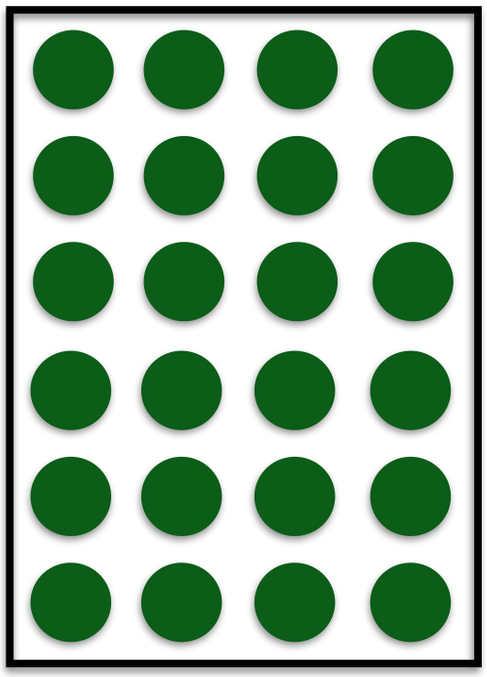
Sentences as Matrices



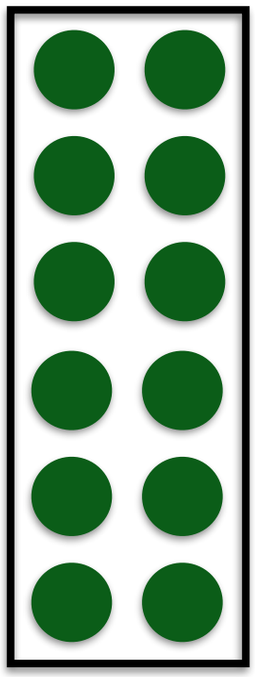
Ich möchte ein Bier

Sentences as Matrices

动态的matrices

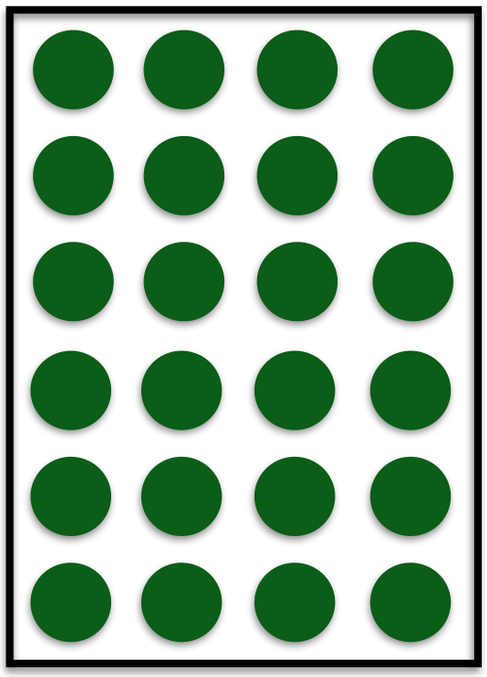


Ich möchte ein Bier

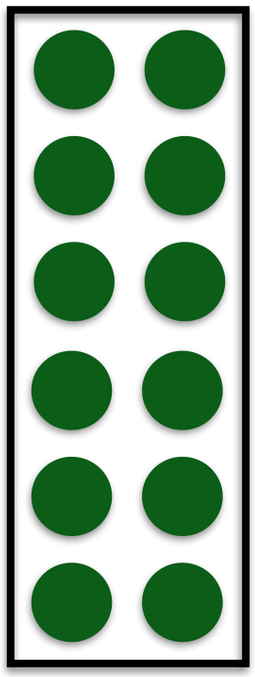


Mach's gut

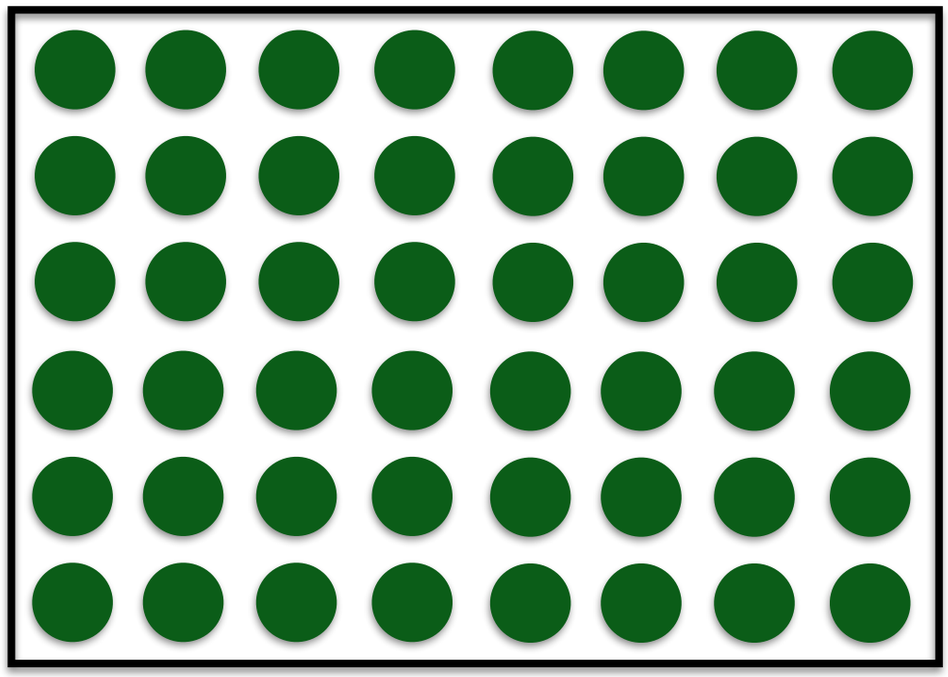
Sentences as Matrices



Ich möchte ein Bier

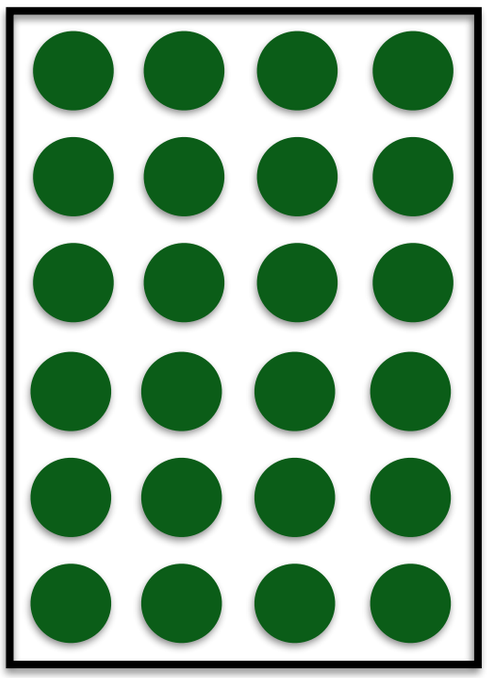


Mach's gut

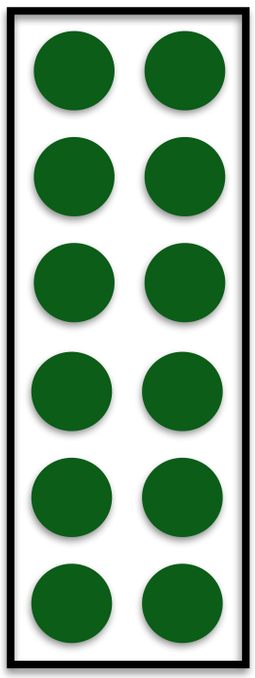


Die Wahrheiten der Menschen sind die unwiderlegbaren Irrtümer

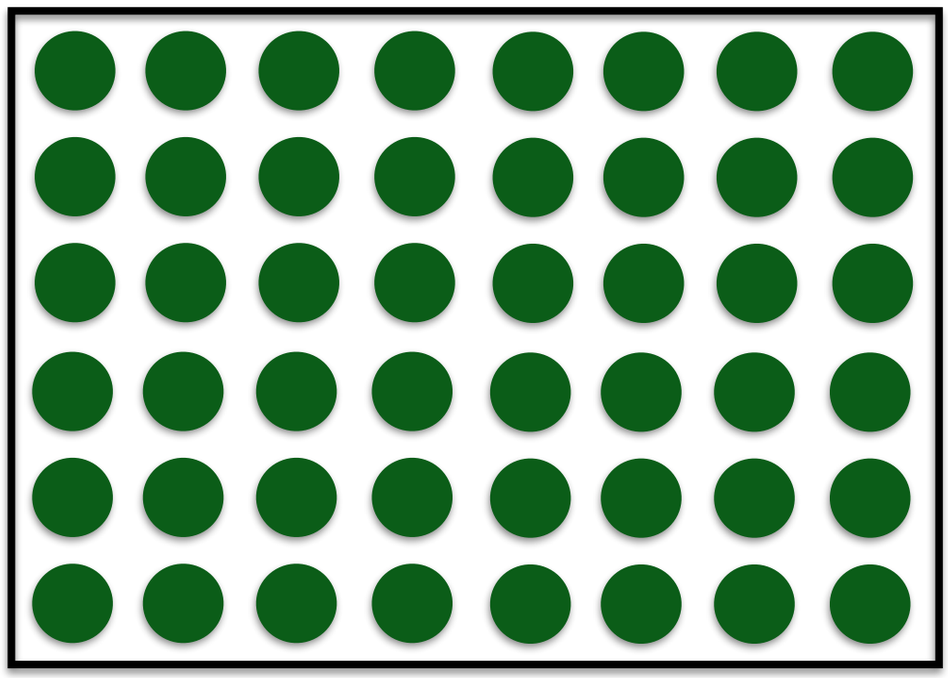
Sentences as Matrices



Ich möchte ein Bier



Mach's gut



Die Wahrheiten der Menschen sind die unwiderlegbaren Irrtümer

Question: How do we build these matrices?

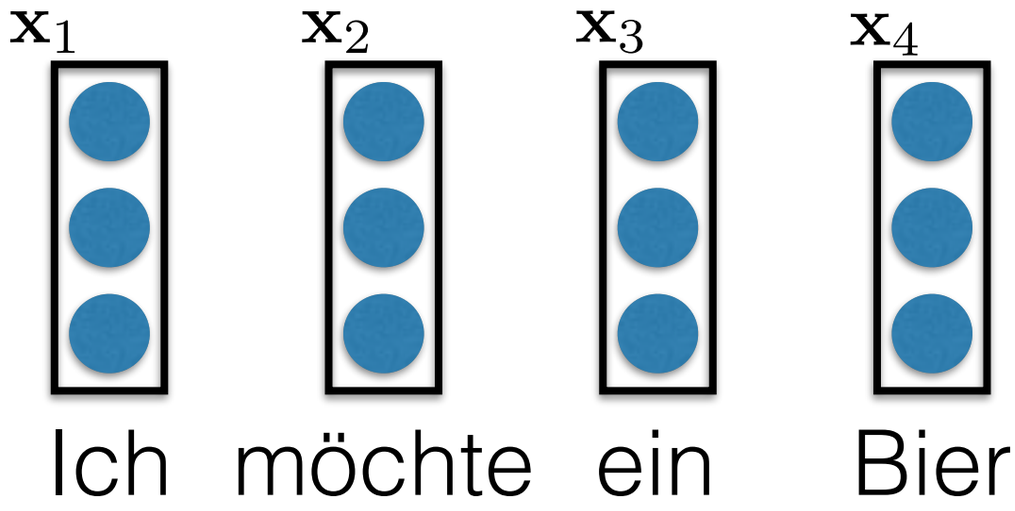
With Concatenation

- Each word type is represented by an n-dimensional vector

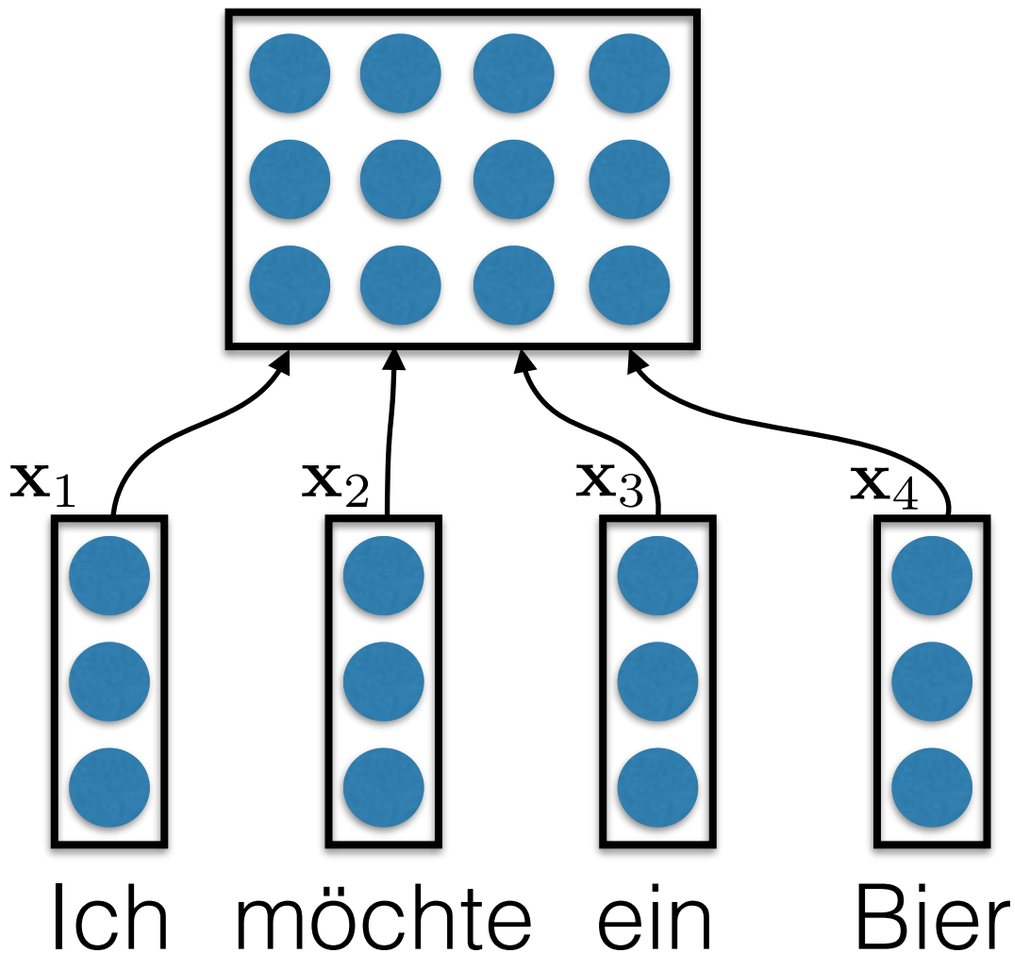
- Take all of the vectors for the sentence and concatenate them into a matrix

- Simplest possible model

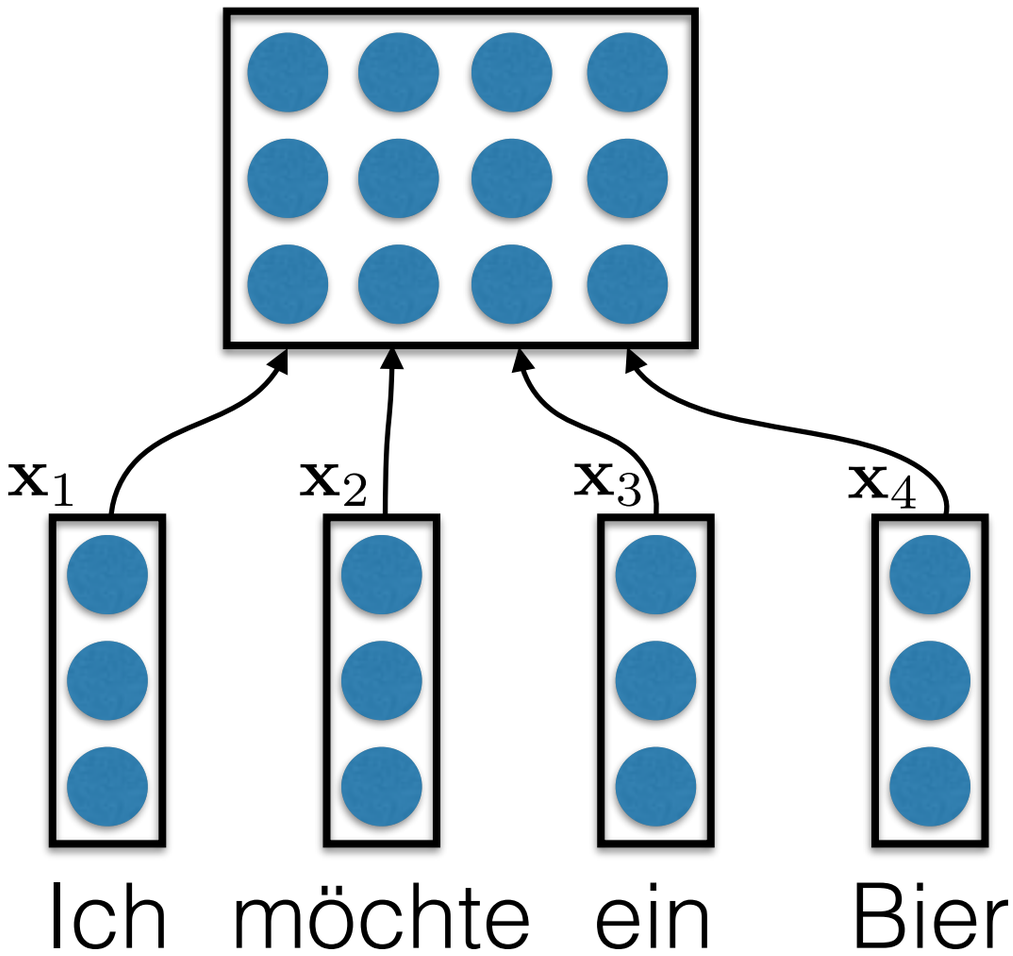
  - So simple, no one has bothered to publish how well/badly it works!


$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

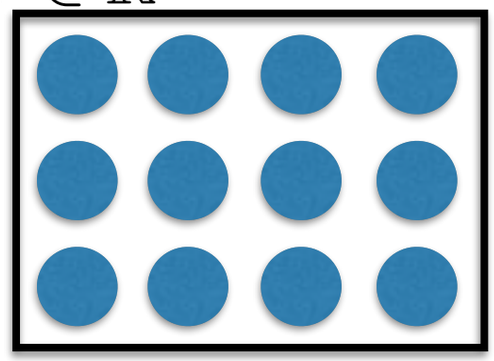
$$\mathbf{f}_i = \mathbf{x}_i$$



$$\mathbf{f}_i = \mathbf{x}_i$$



$\mathbf{F} \in \mathbb{R}^{n \times |\boldsymbol{f}|}$



Ich möchte ein Bier

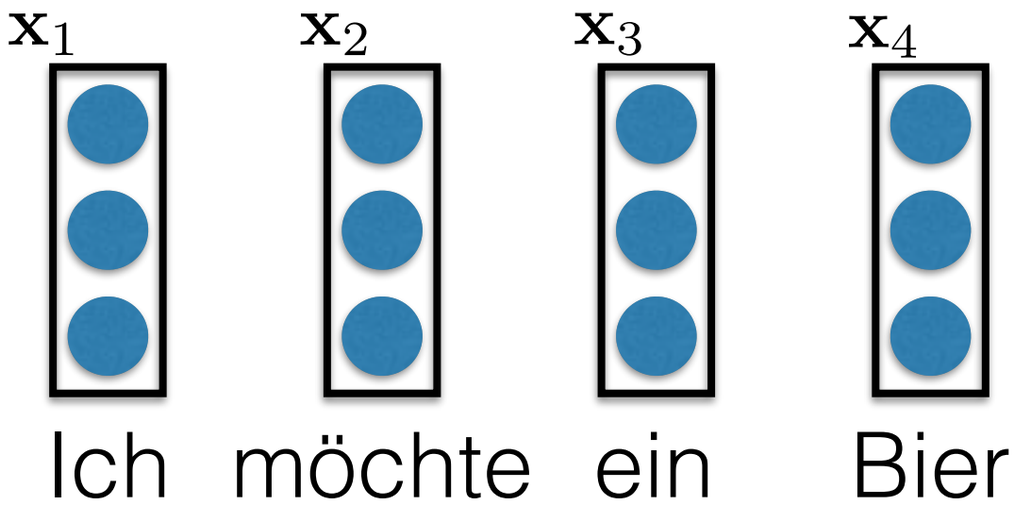
With Convolutional Nets

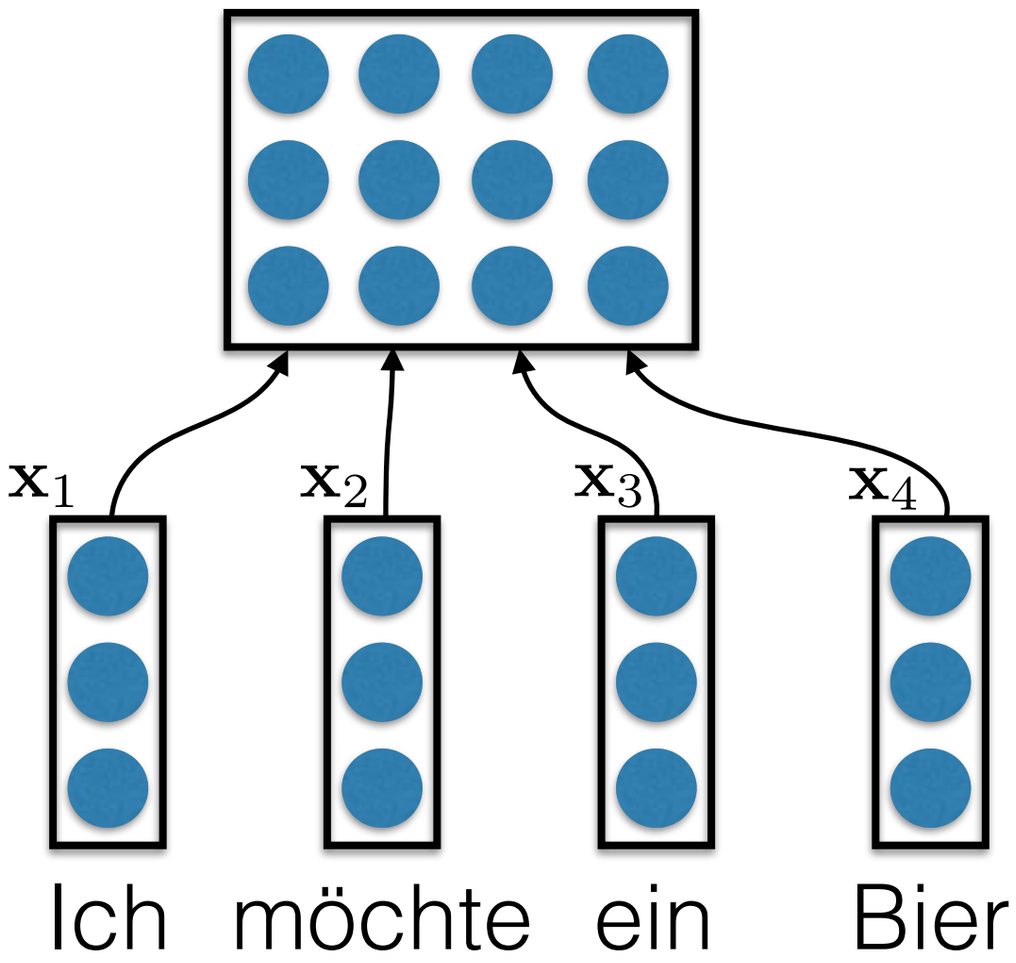
- Apply convolutional networks to transform the naive concatenated matrix to obtain a context-dependent matrix

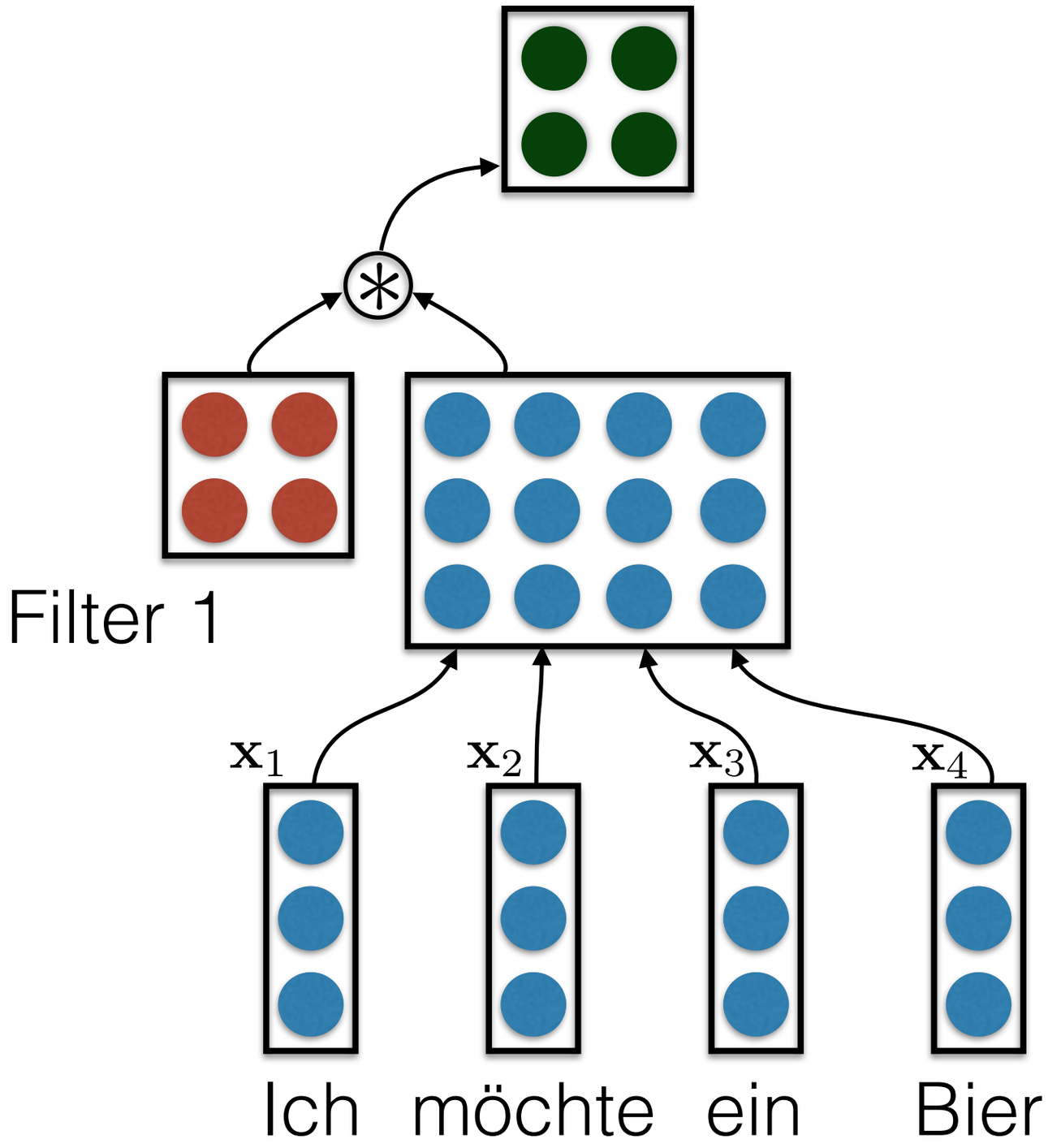
- Explored in a recent ICLR submission by Gehring et al., 2016 (from FAIR)

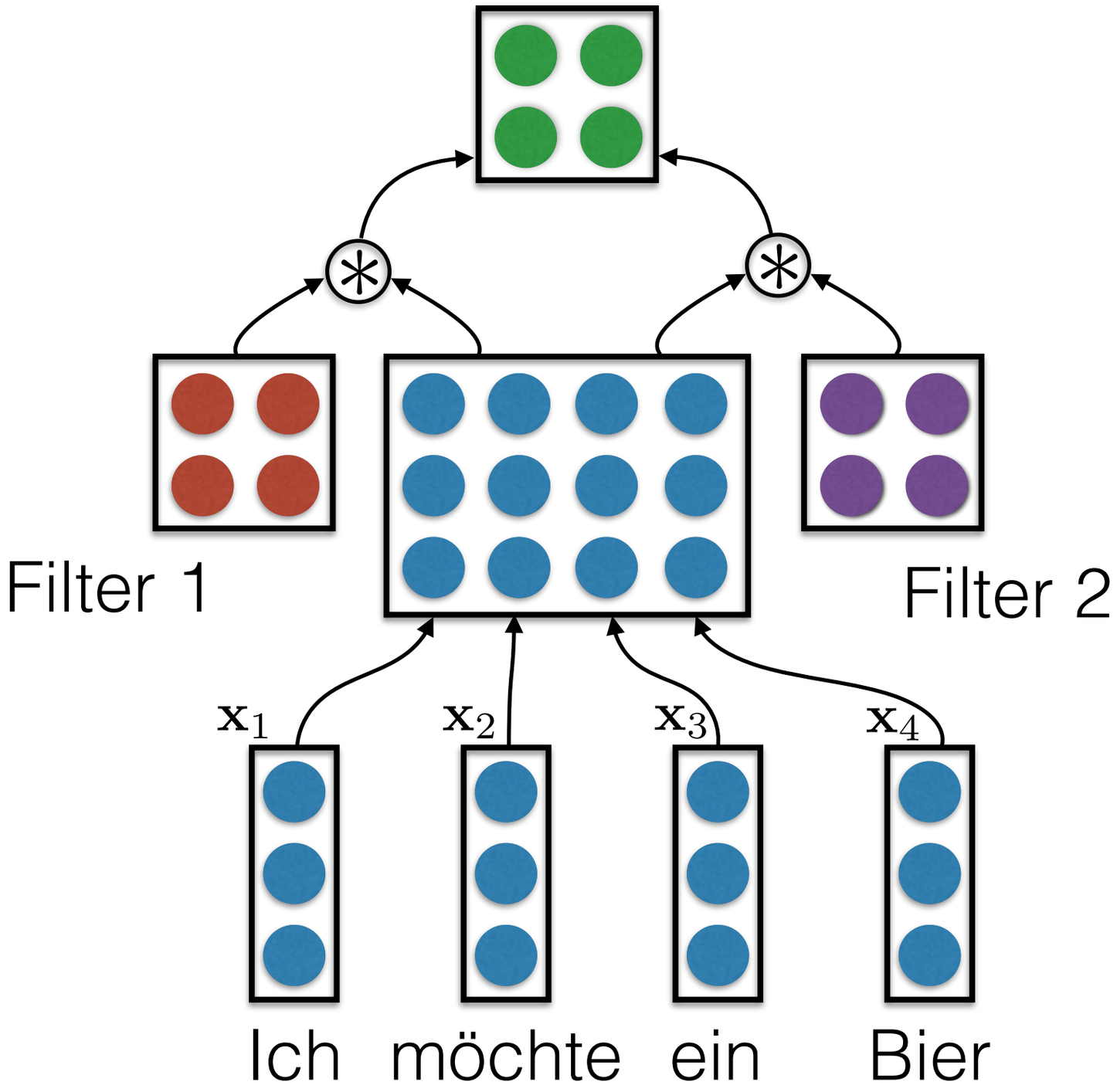
    - Closely related to the neural translation model proposed by Kalchbrenner and Blunsom, 2013

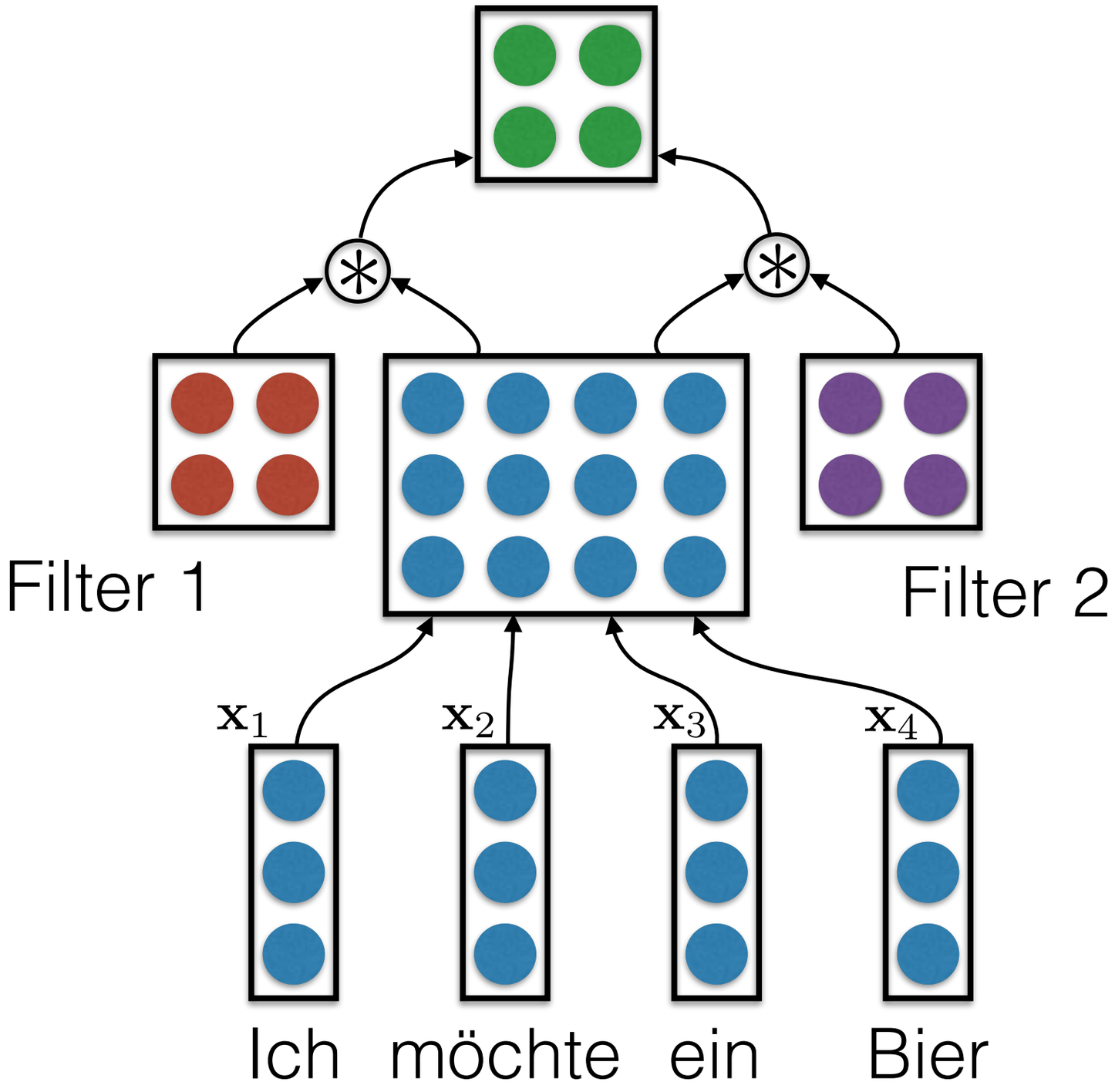
- Note: convnets usually have a “pooling” operation at the top level that results in a fixed-sized representation. For sentences, leave this out.


$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

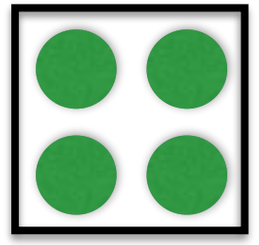

$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


Filter 1
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


Filter 1
Filter 2
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier



$$\mathbf{F} \in \mathbb{R}^{f(n) \times g(|\boldsymbol{f}|)}$$



Ich möchte ein Bier

With Bidirectional RNNs

- By far the most widely used matrix representation, due to Bahdanau et al (2015)

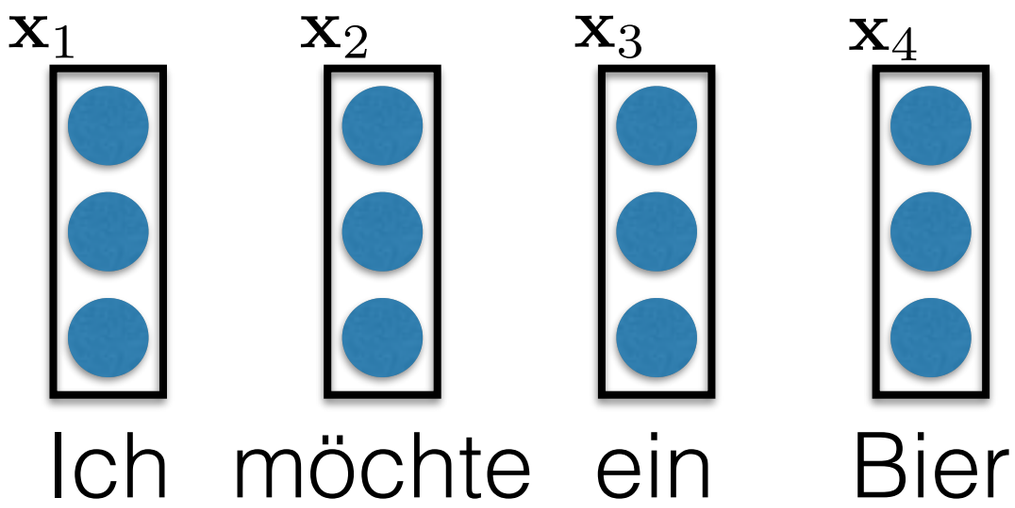
- One column per word

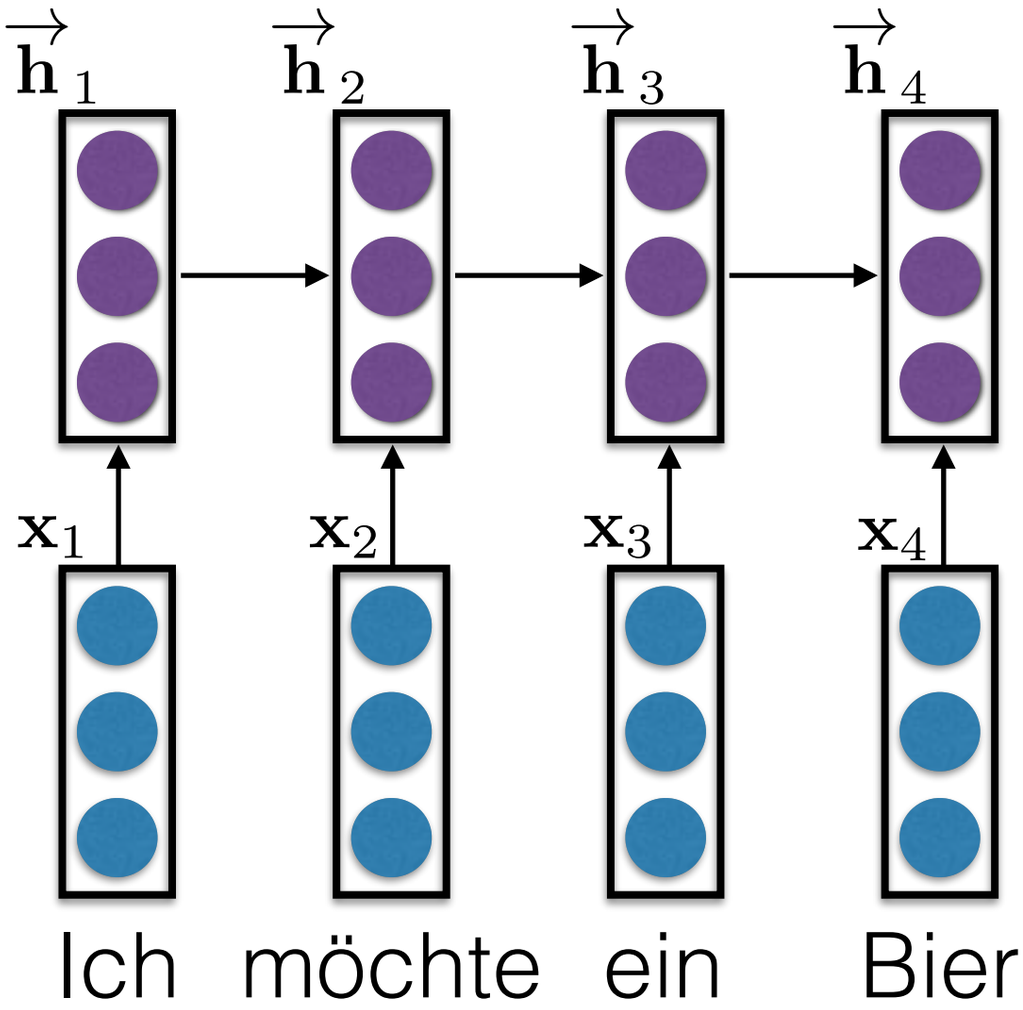
- Each column (word) has two halves concatenated together:

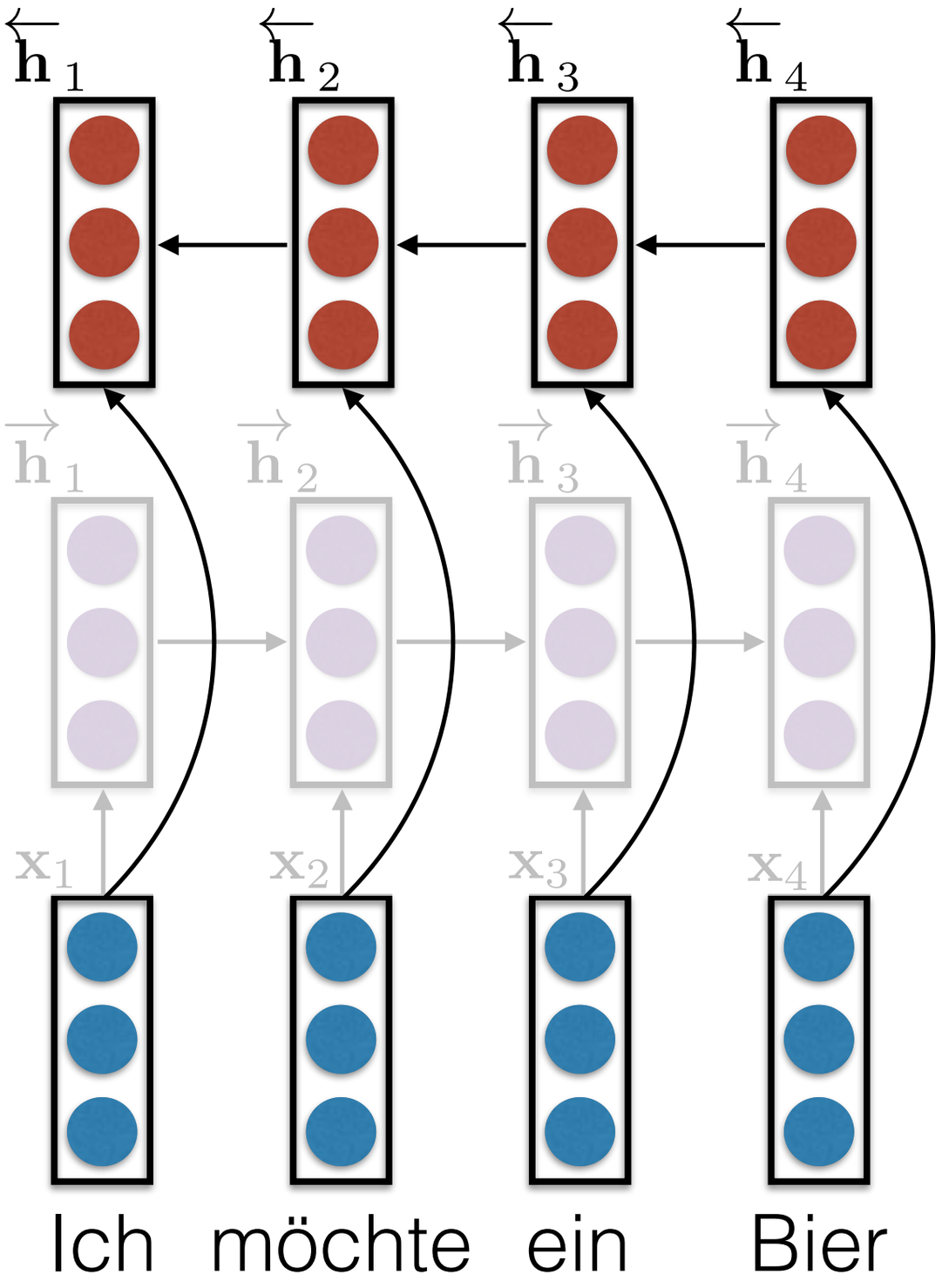
  - a “forward representation”, i.e., a word and its left context

  - a “reverse representation”, i.e., a word and its right context

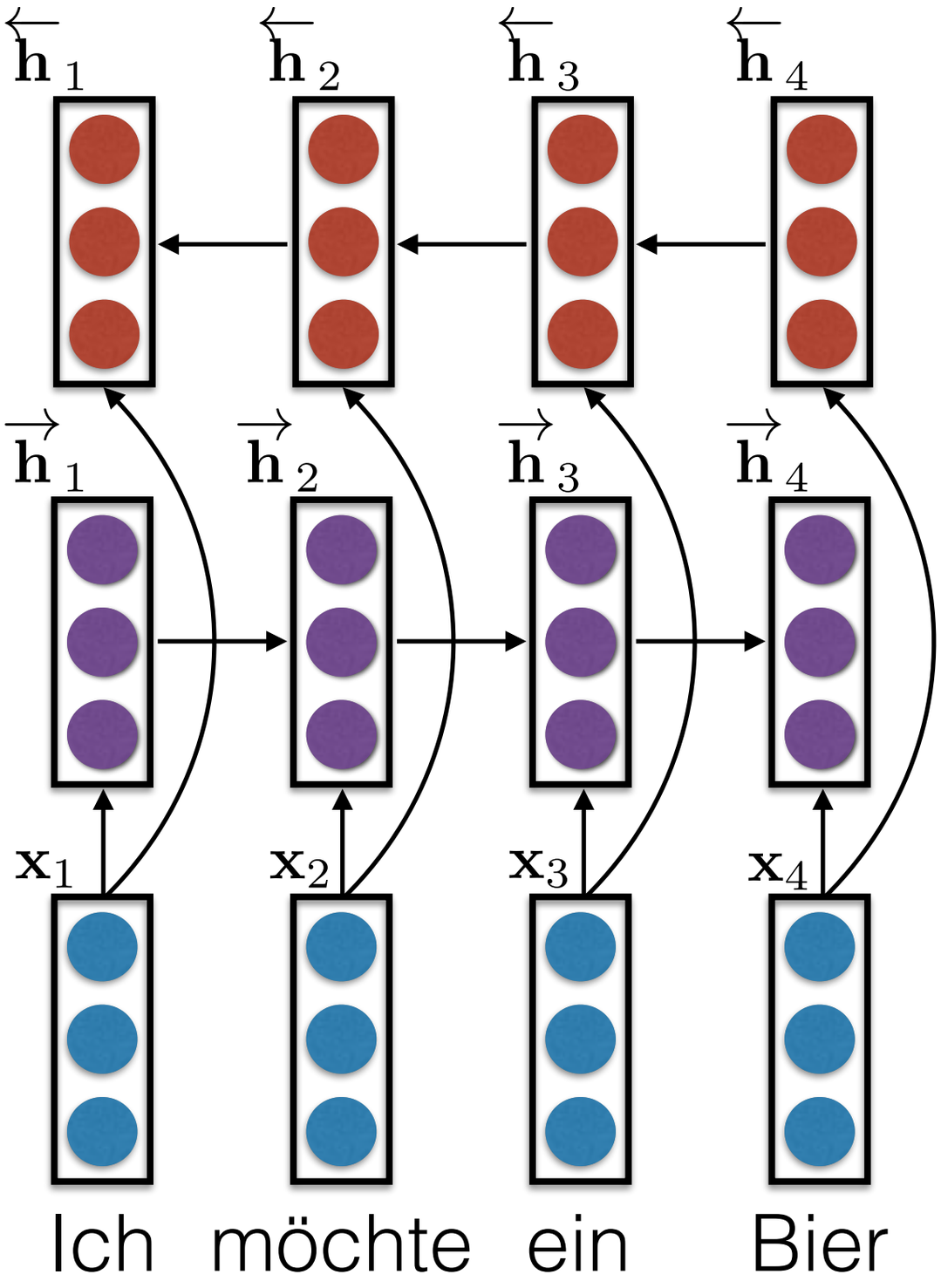
- Implementation: bidirectional RNNs (GRUs or LSTMs) to read ***f*** from left to right and right to left, concatenate representations

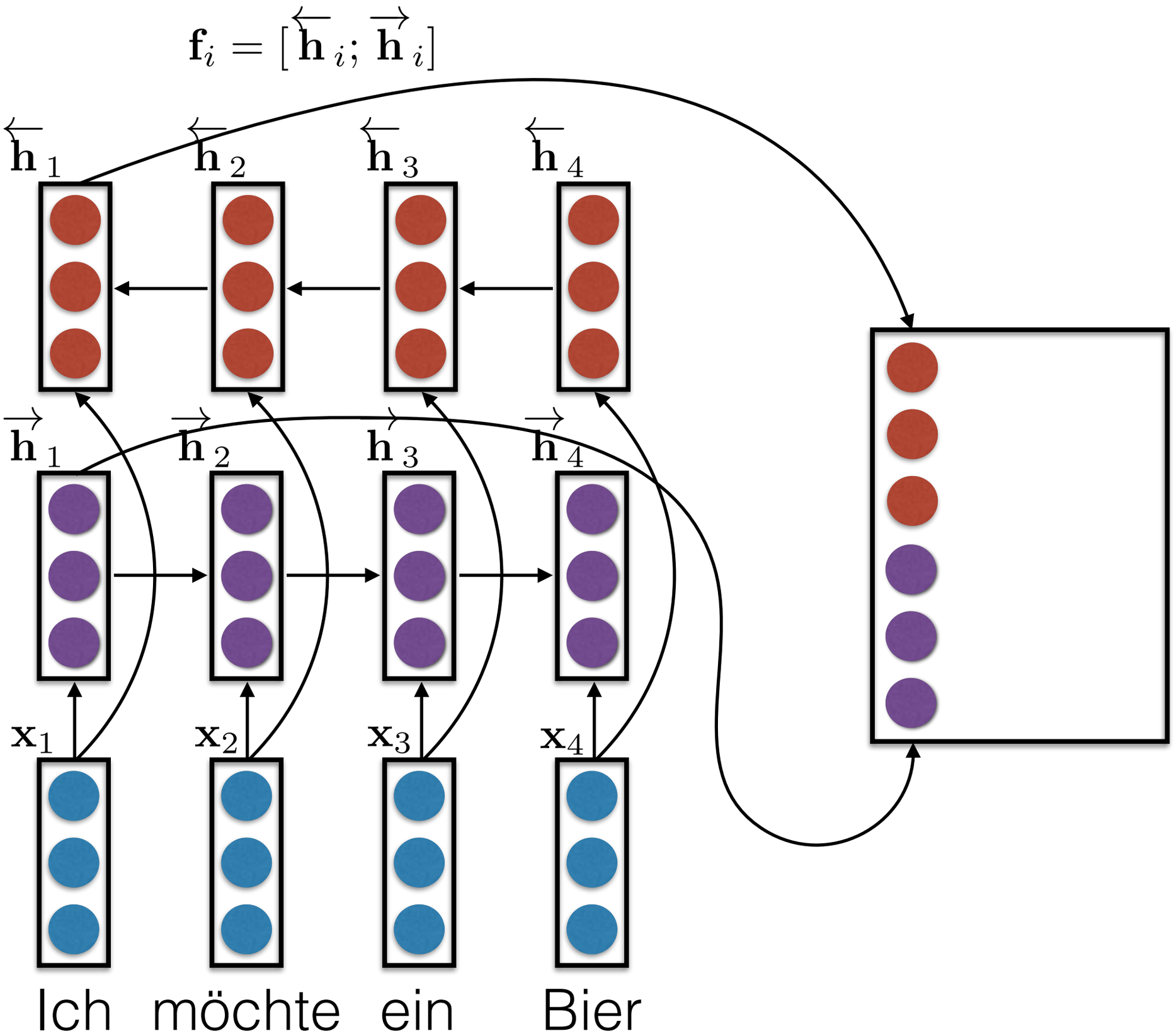

$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

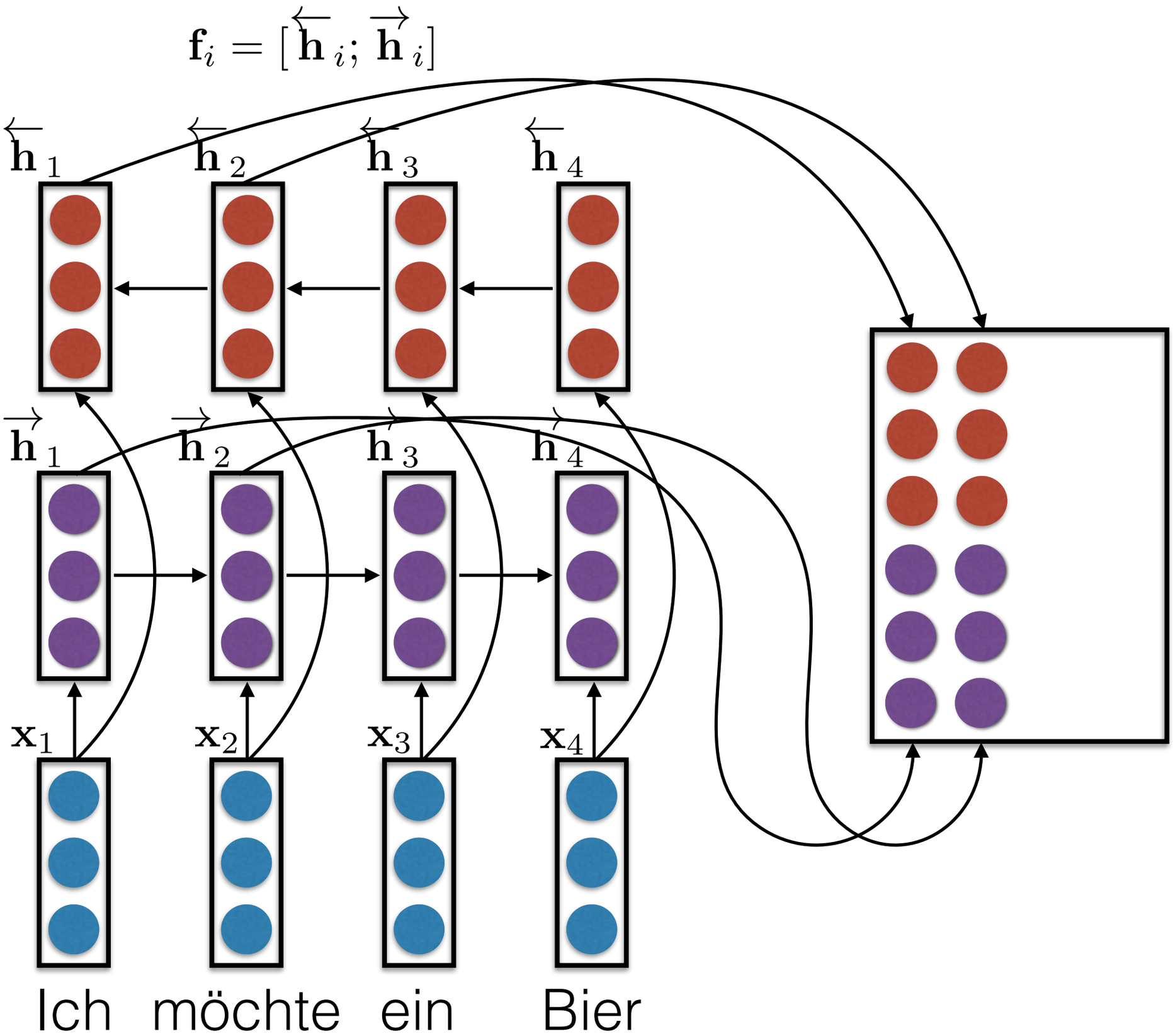

$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

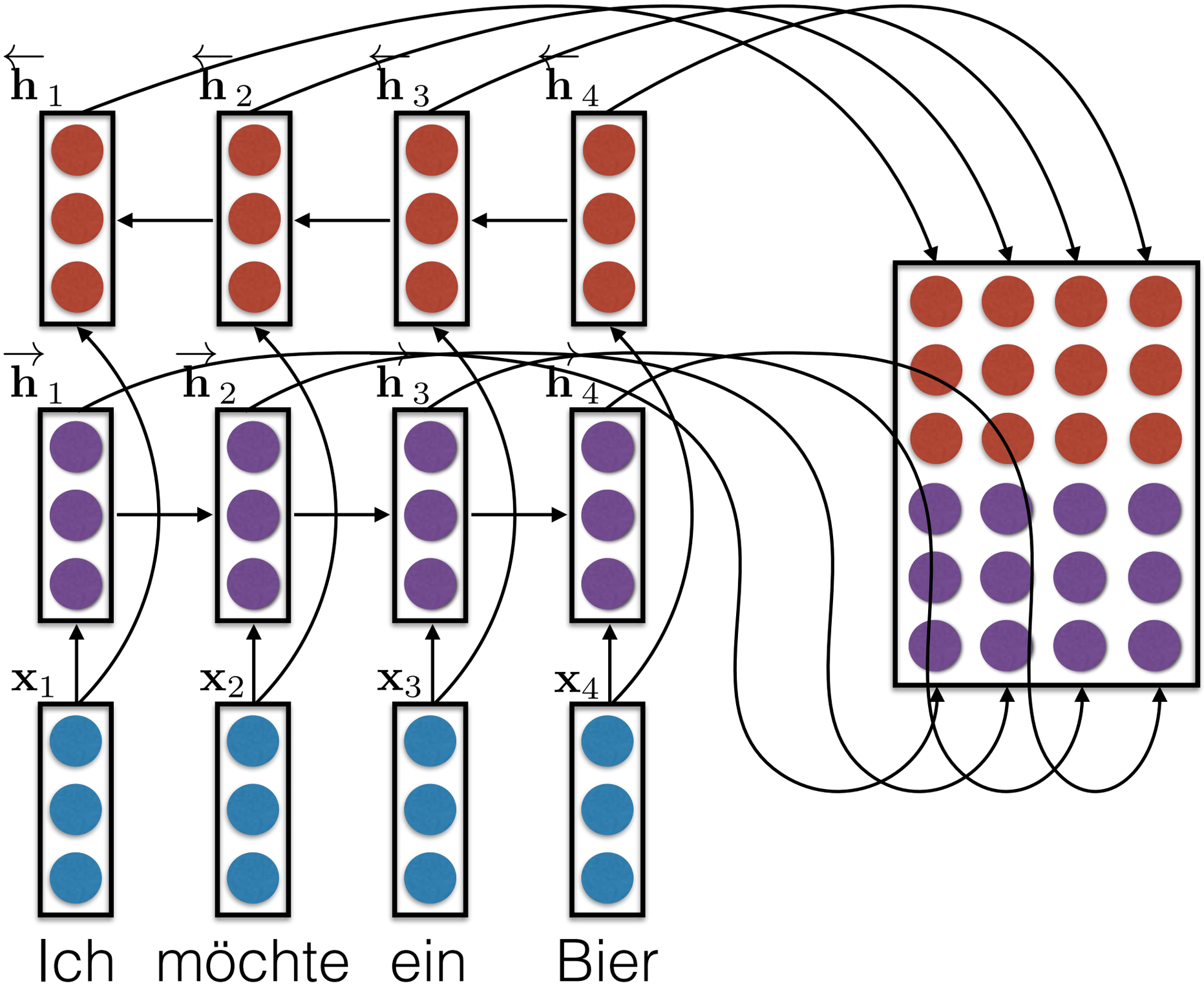
$$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$$



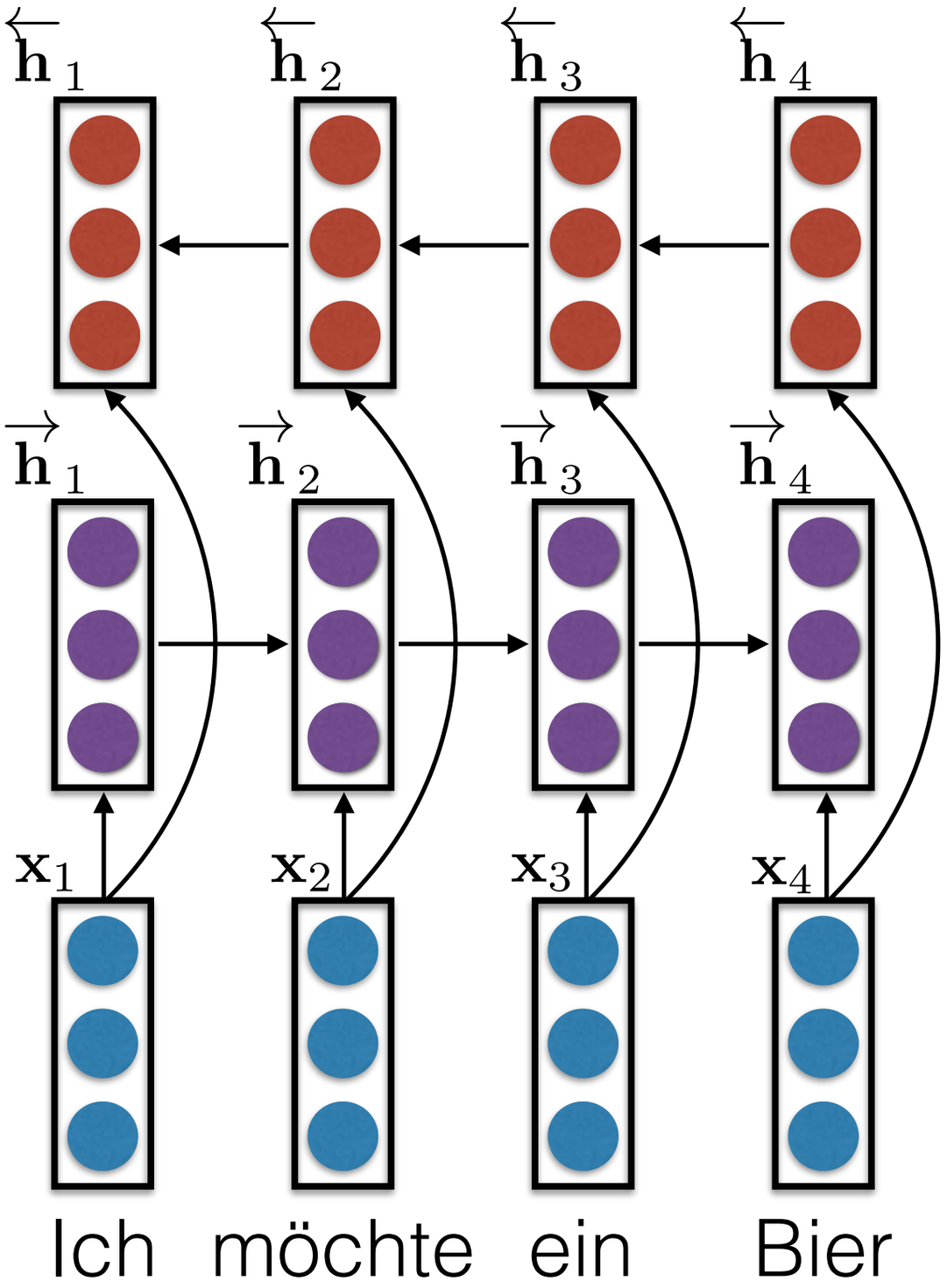

$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$
$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$
$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

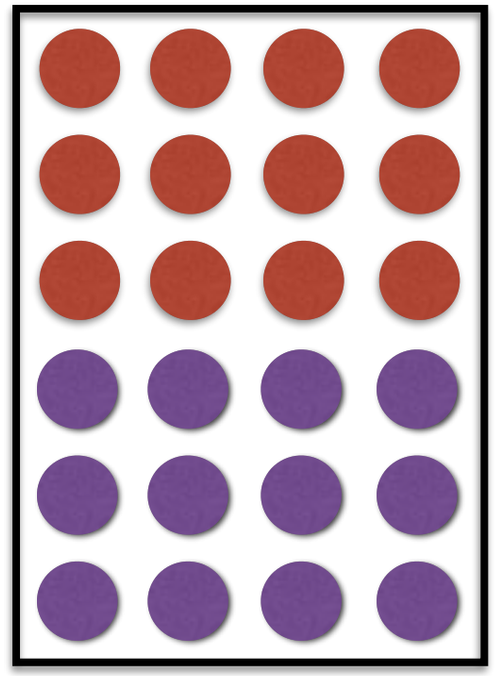
$$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$$



$$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$$



$$\mathbf{F} \in \mathbb{R}^{2n \times |\boldsymbol{f}|}$$



Ich möchte ein Bier

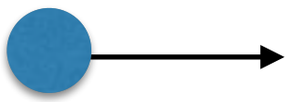
Where are we in 2017?

- There are lots of ways to construct **F**
  - Very little systematic work comparing them
  - There are many more undiscovered things out there
    - convolutions are particularly interesting and under-explored
    - syntactic information can help (Sennrich & Haddow, 2016; Nadejde et al., 2017), but many more integration strategies are possible
    - try something with phrase types instead of word types?

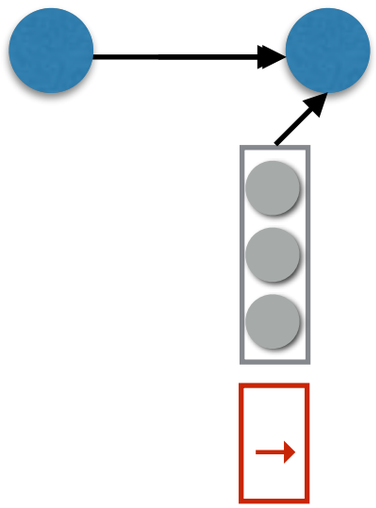
Multi-word expressions are a pain in the neck .

Generation from Matrices

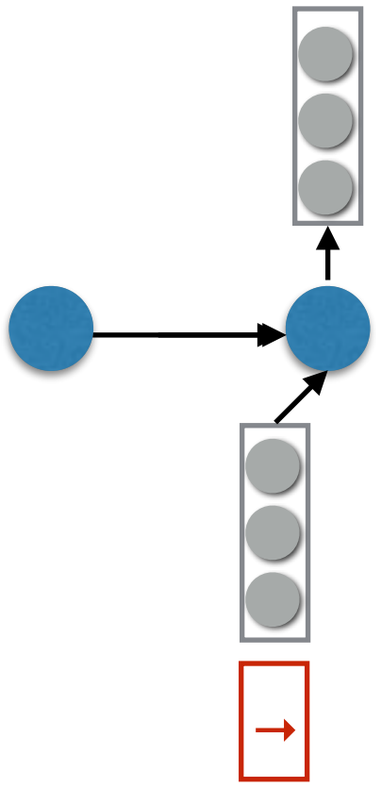
- We have a matrix **F** representing the input, now we need to generate from it
- Bahdanau et al. (2015) were the first to propose using ***attention*** for translating from matrix-encoded sentences
- High-level idea
  - Generate the output sentence word by word using an RNN
  - At each output position *t*, the RNN receives **two** inputs (in addition to any recurrent inputs)
    - a fixed-size vector embedding of the previously generated output symbol $e_{t\text{-}1}$
    - a fixed-size vector encoding a "view" of the input matrix
  - How do we get a fixed-size vector from a matrix that changes over time?
    - Bahdanau et al: do a weighted sum of the columns of **F** (i.e., words) based on how important they are *at the current time step*. (i.e., just a matrix-vector product $\mathbf{Fa}_t$)
    - The weighting of the input columns at each time-step ($\mathbf{a}_t$) is called ***attention***



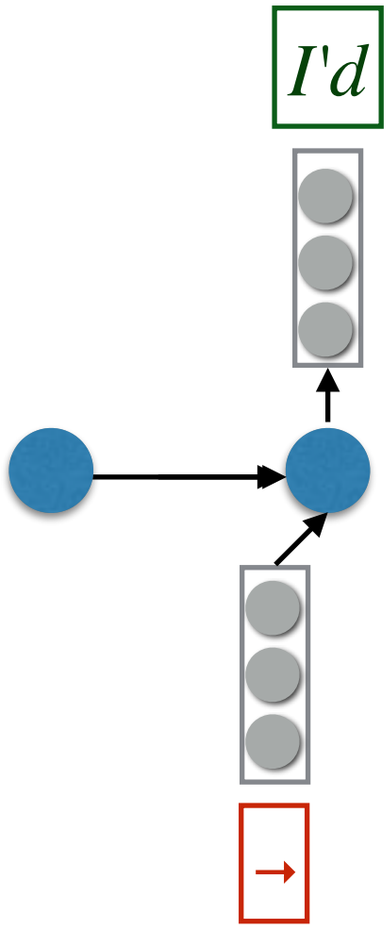
Recall RNNs…



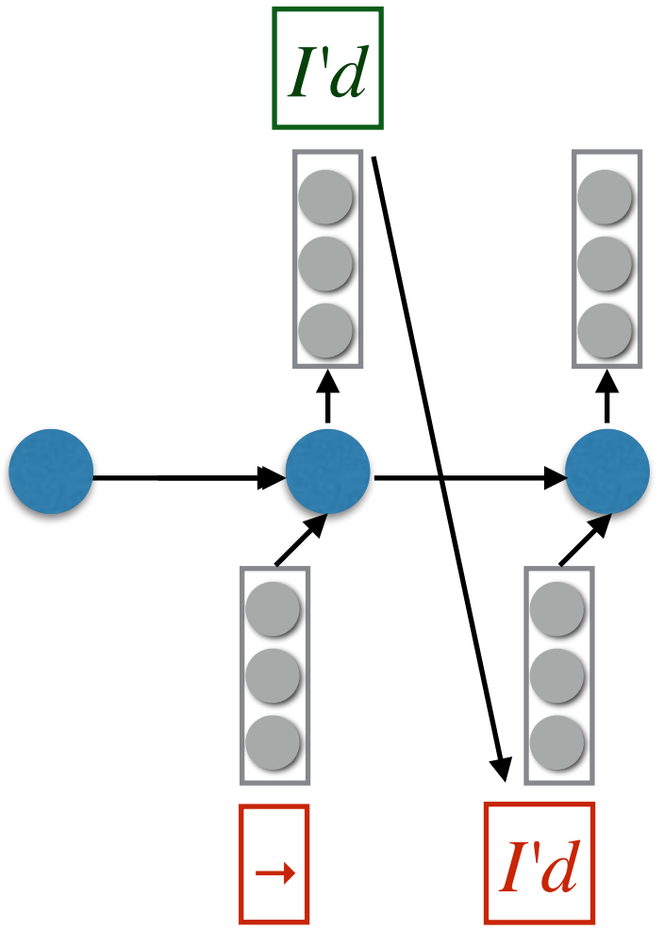
Recall RNNs…



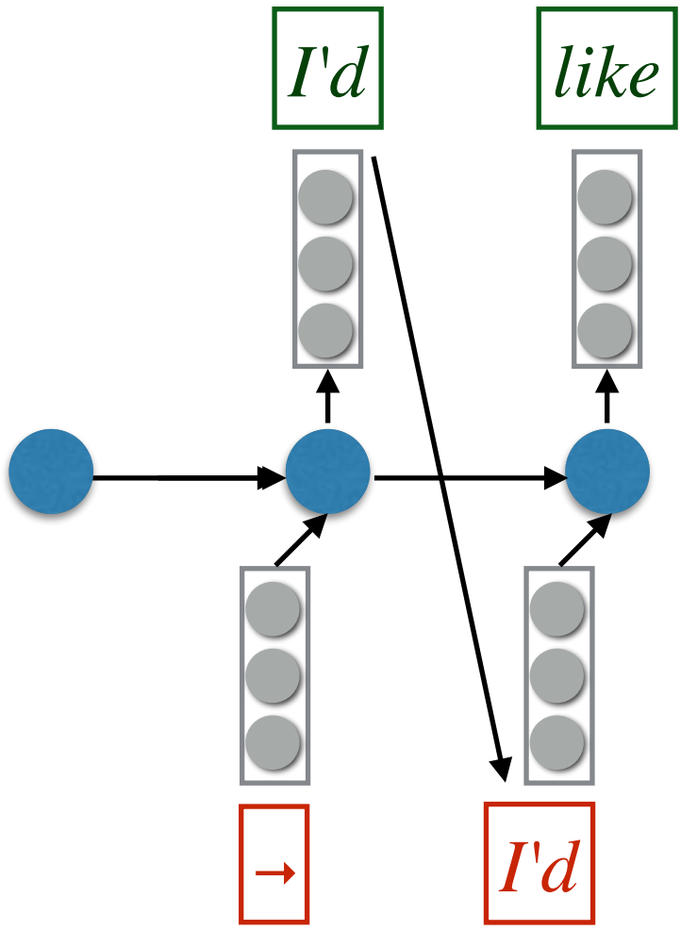
Recall RNNs…



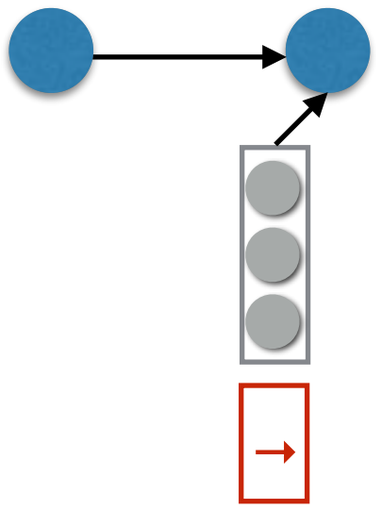
Recall RNNs…

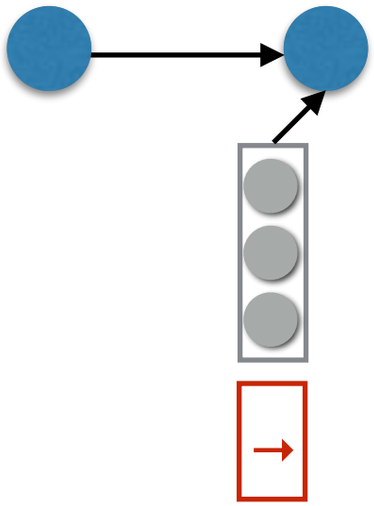


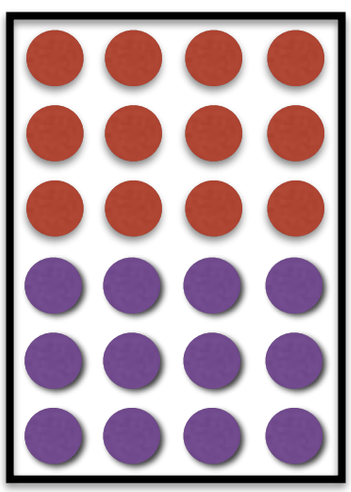
Recall RNNs…



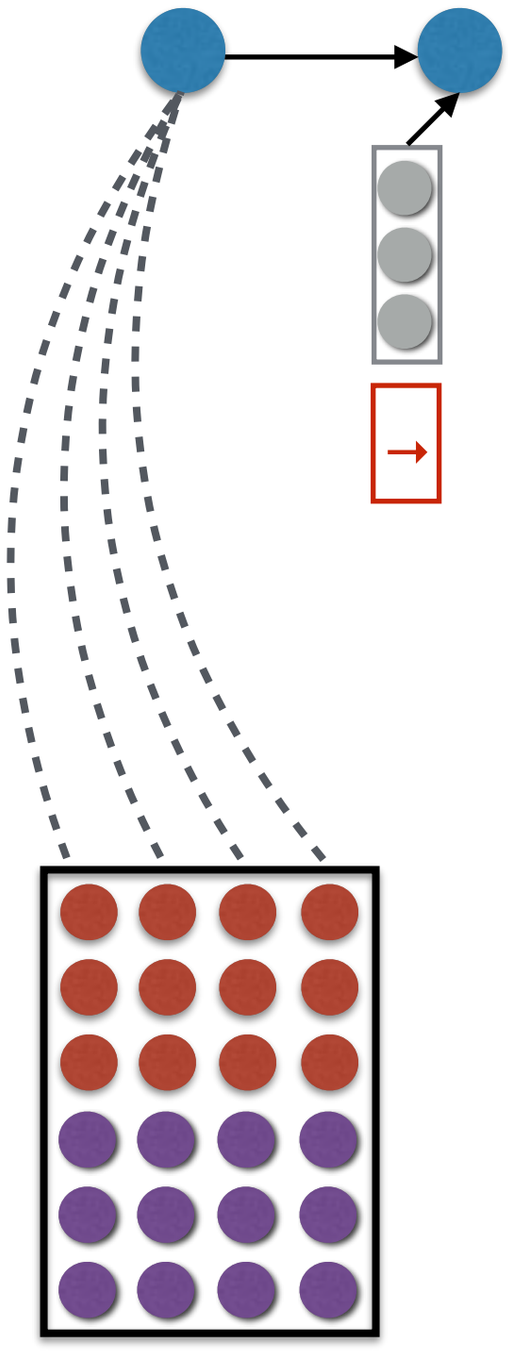
Recall RNNs…



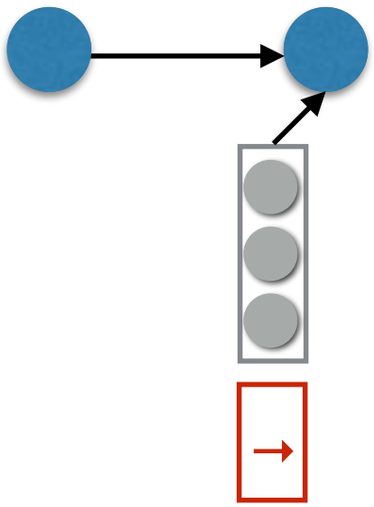


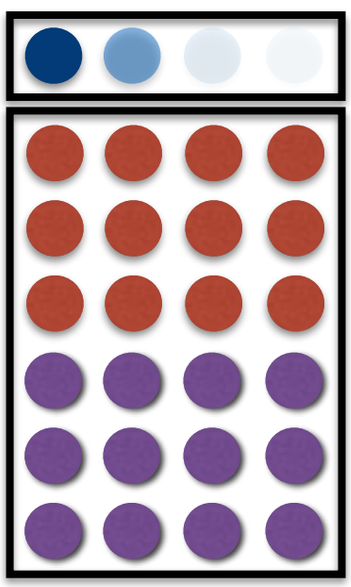


Ich möchte ein Bier



Ich möchte ein Bier

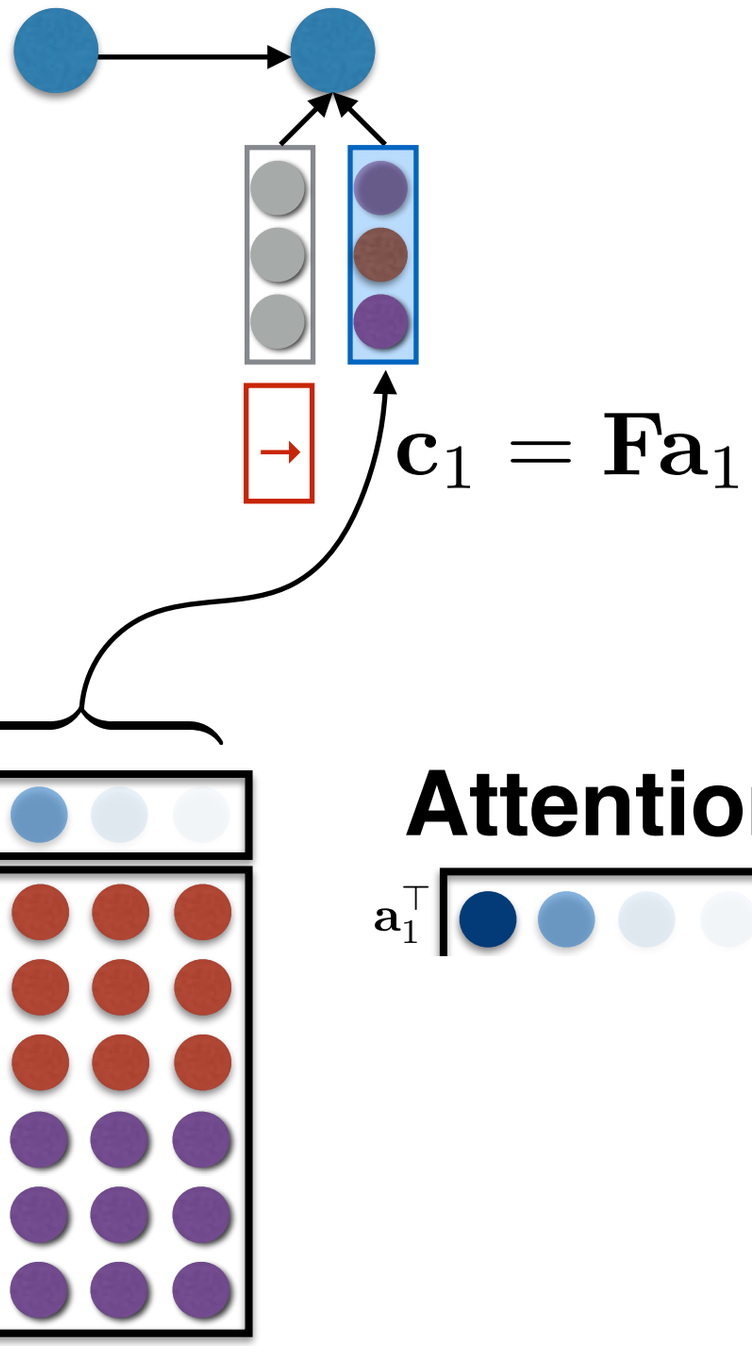


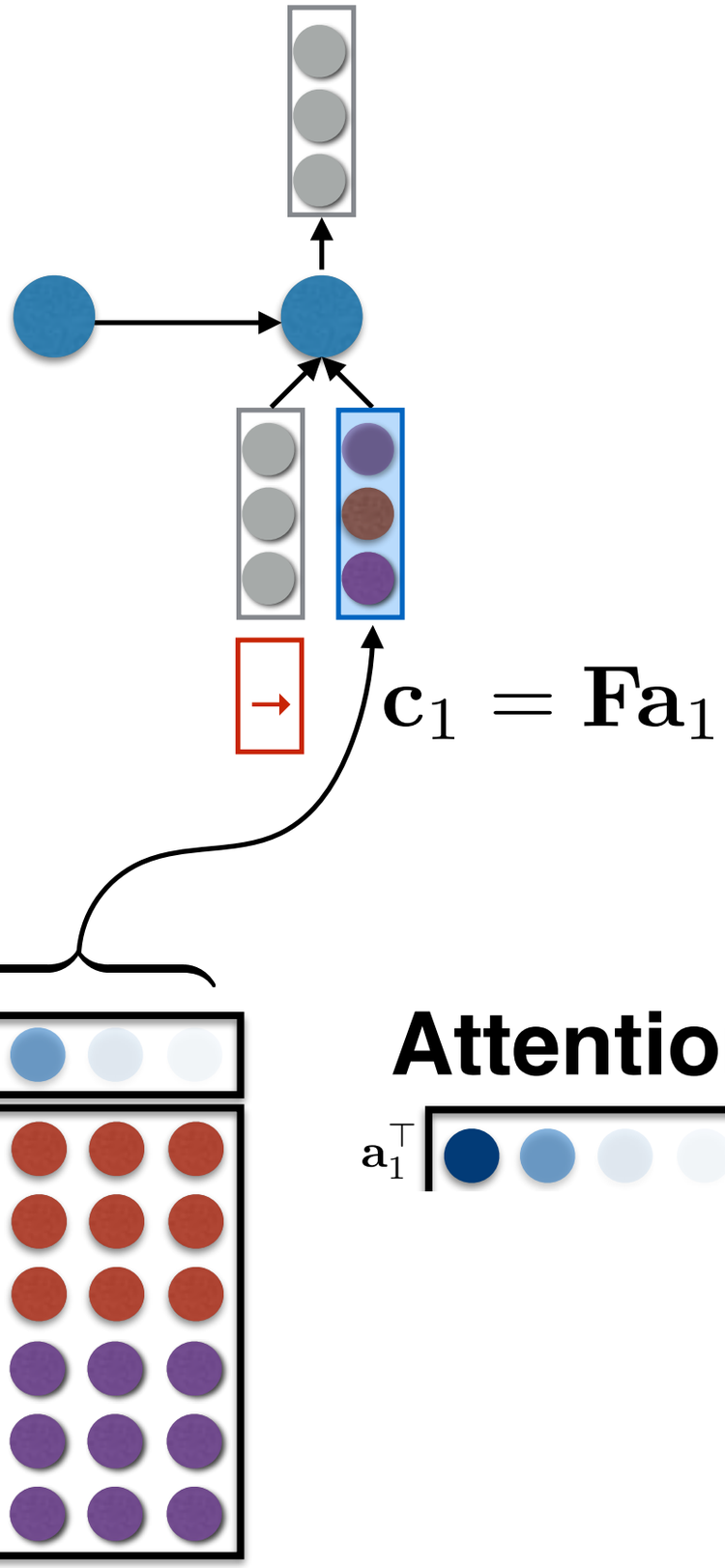


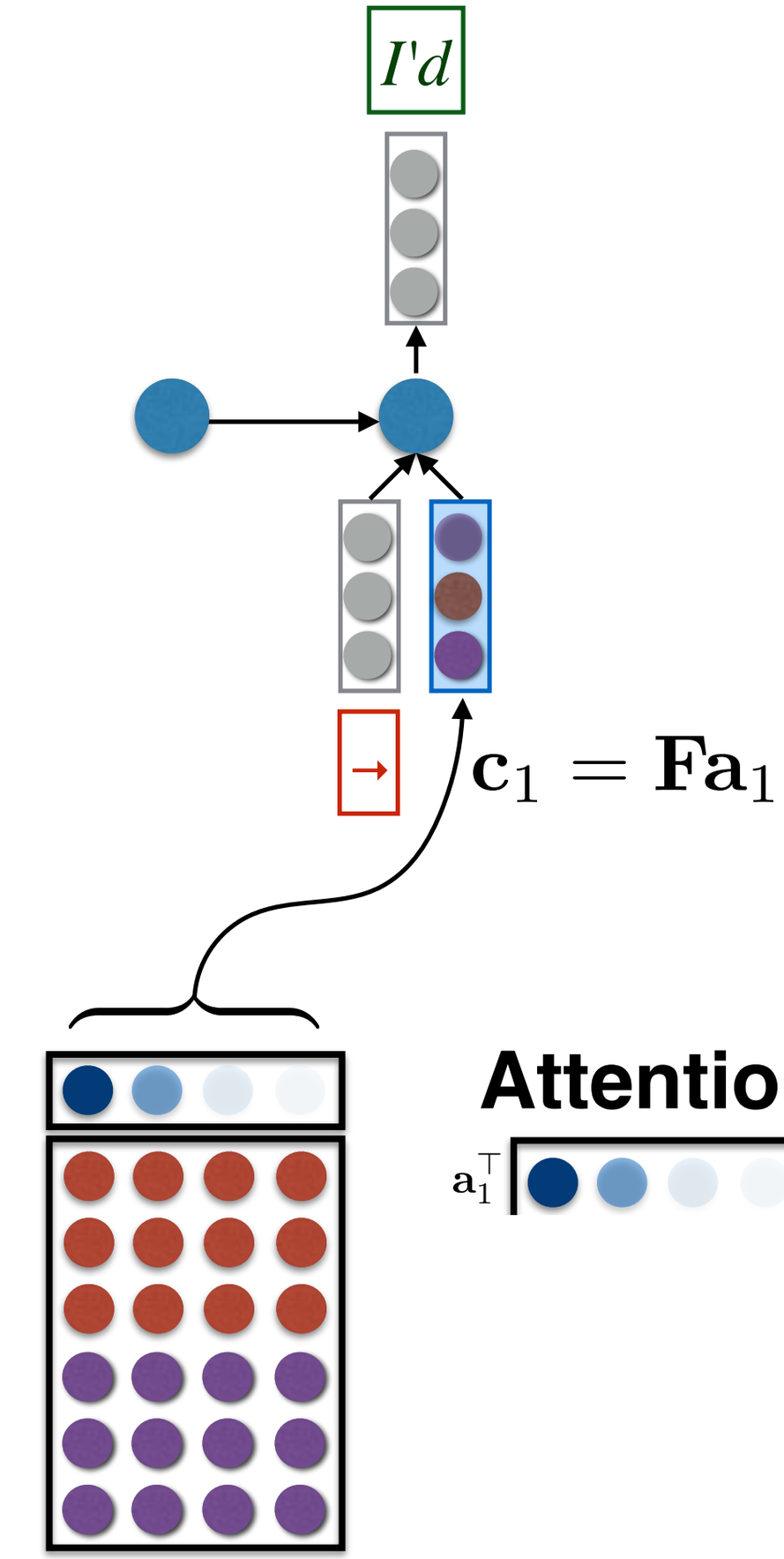
Ich möchte ein Bier

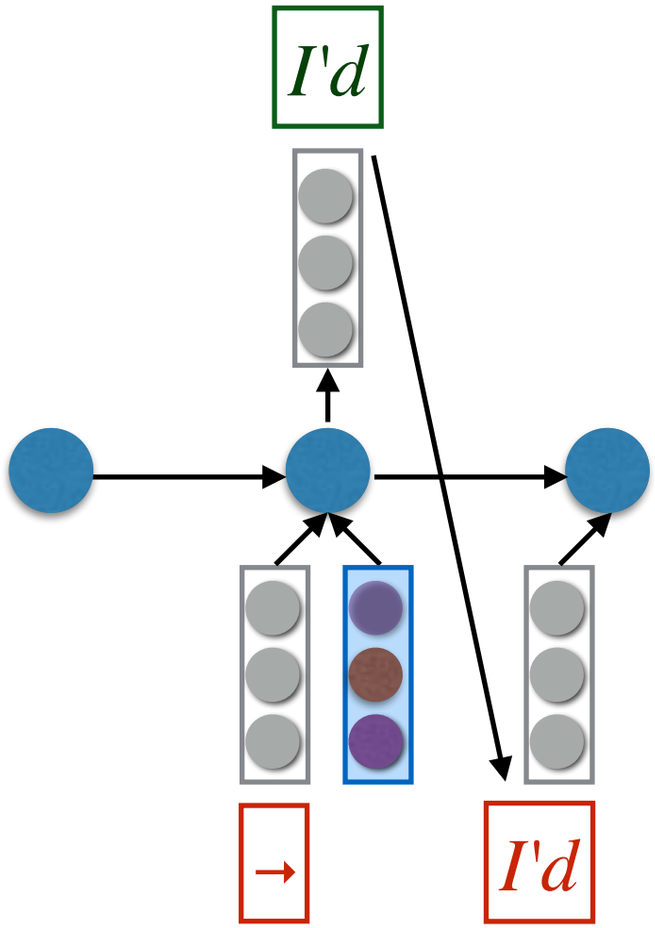
Attention history:

$\mathbf{a}_1^\top$

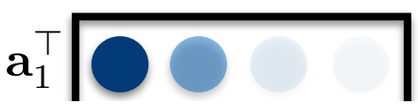

$\mathbf{c}_1 = \mathbf{F}\mathbf{a}_1$
Attention history:
$\mathbf{a}_1^\top$
Ich möchte ein Bier

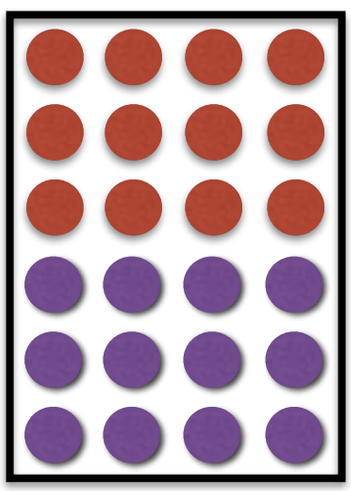

$\mathbf{c}_1 = \mathbf{F}\mathbf{a}_1$
Attention history:
$\mathbf{a}_1^\top$
Ich möchte ein Bier


I'd
→
$\mathbf{c}_1 = \mathbf{F}\mathbf{a}_1$
Attention history:
$\mathbf{a}_1^\top$
Ich möchte ein Bier

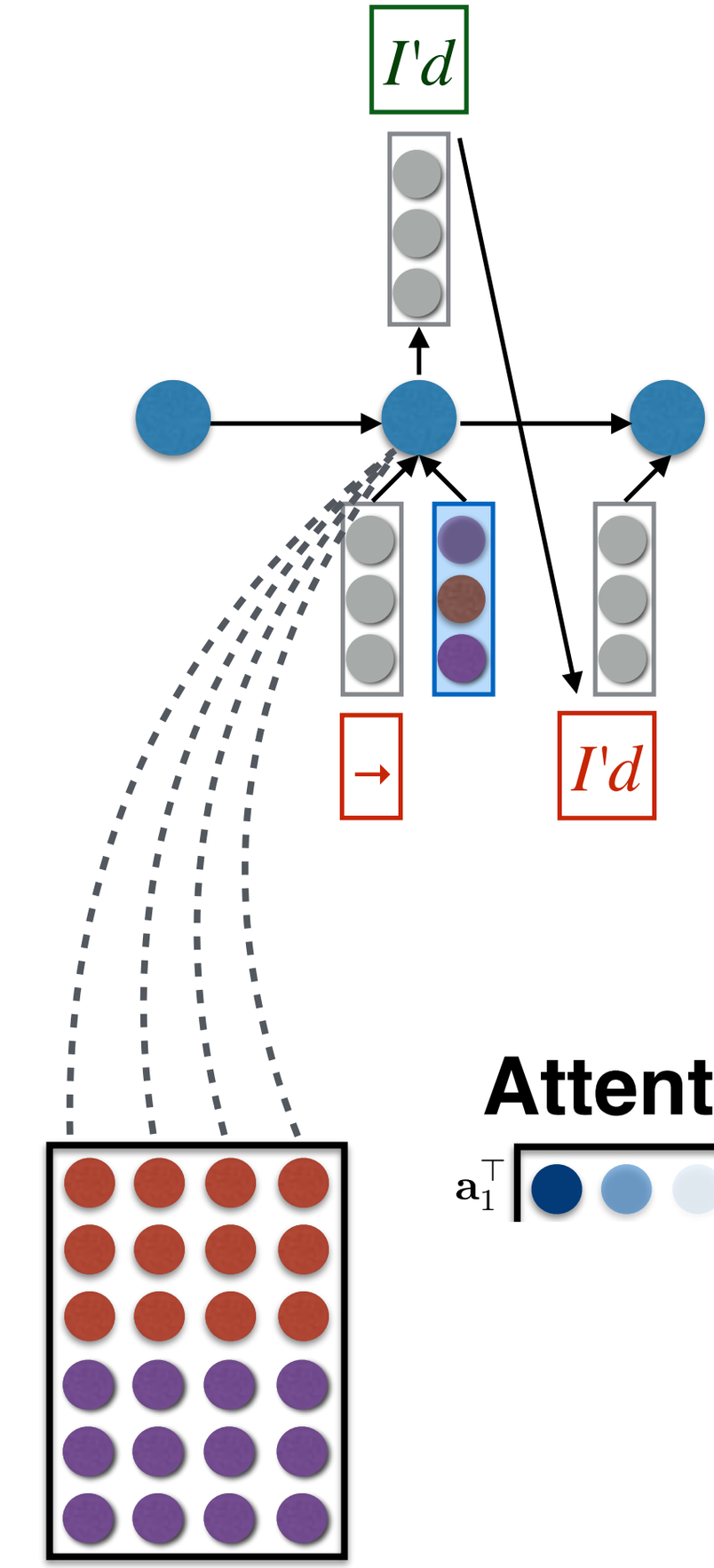


Attention history:

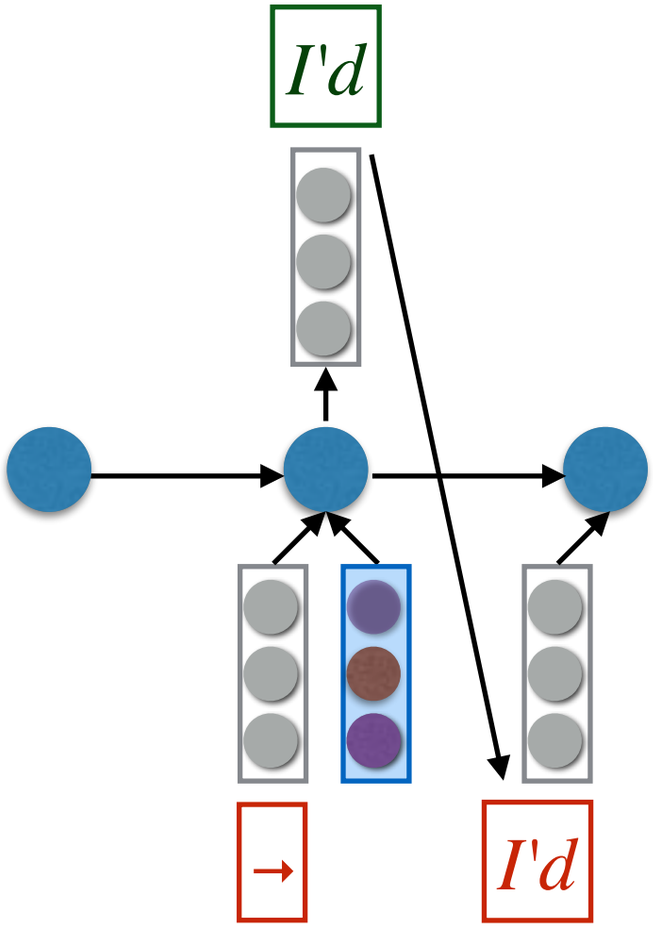


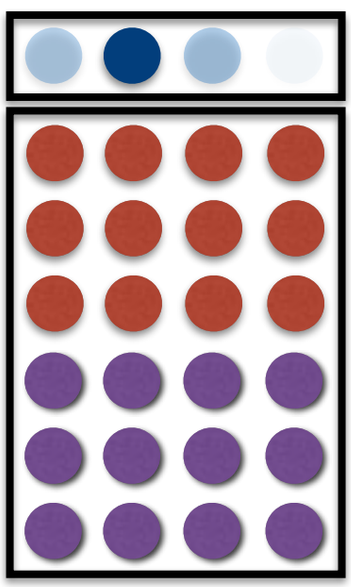


Ich möchte ein Bier



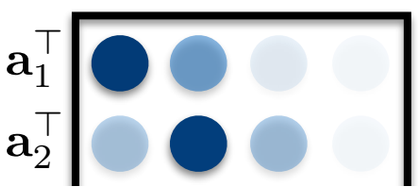
Attention history:

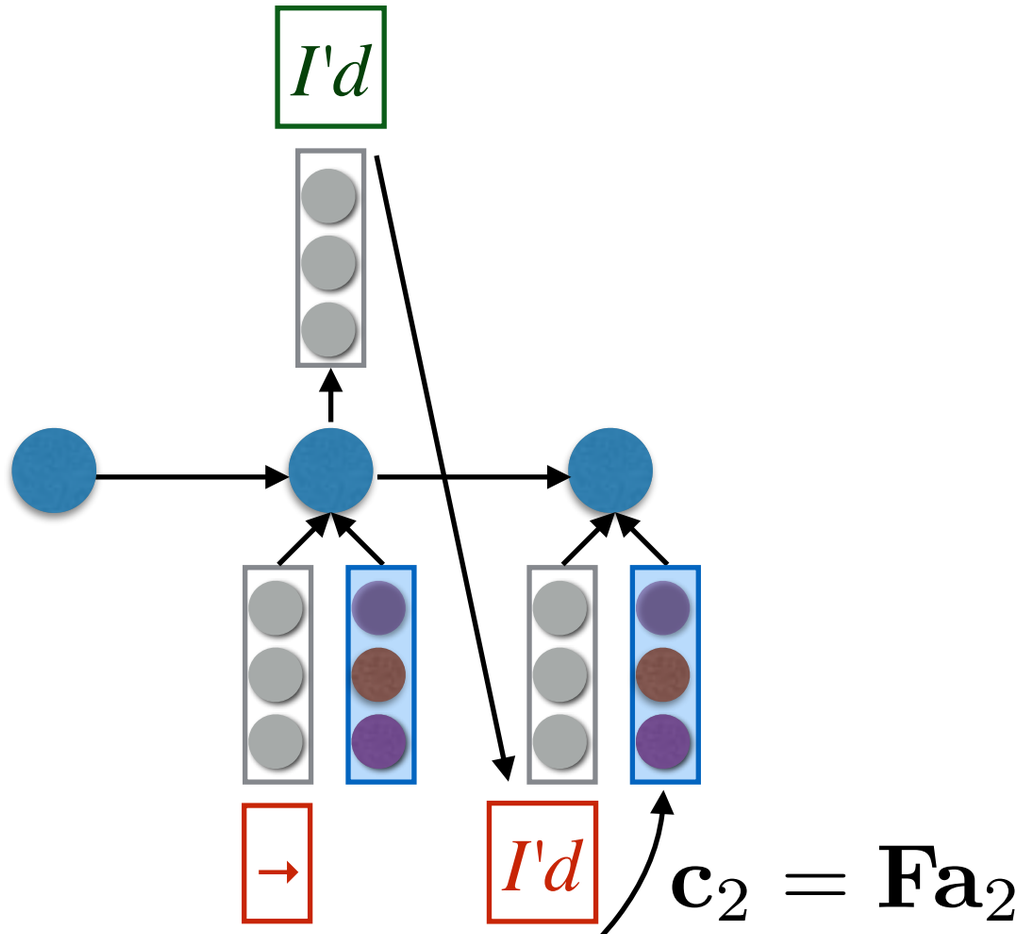


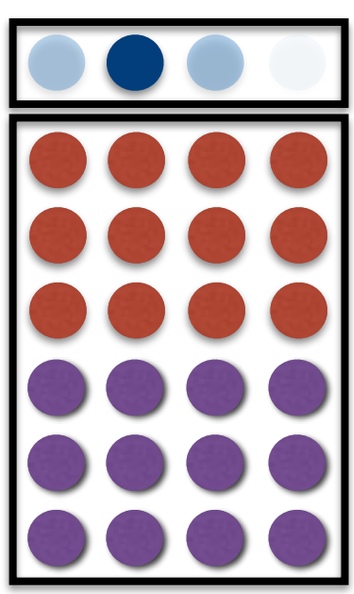


Ich möchte ein Bier

Attention history:

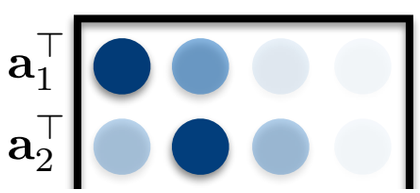


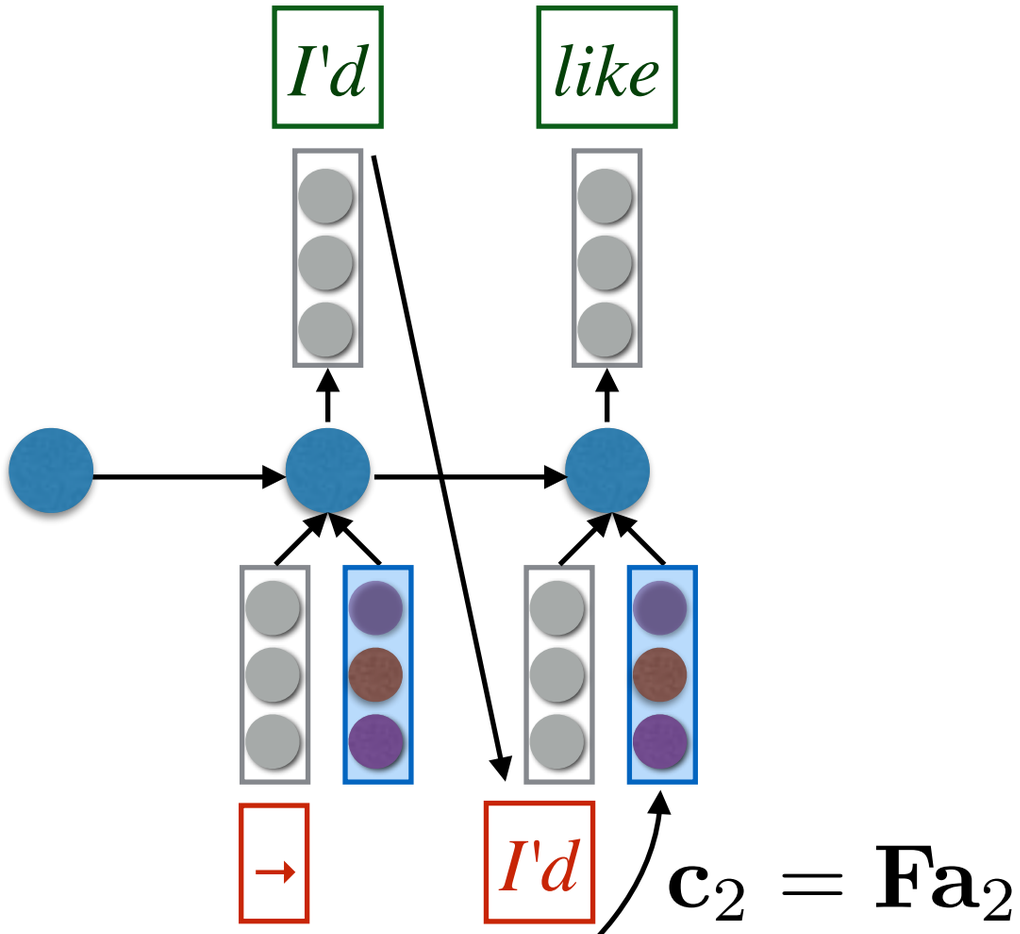


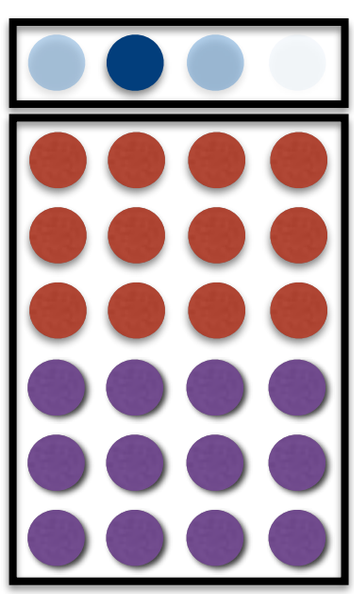


Ich möchte ein Bier

Attention history:

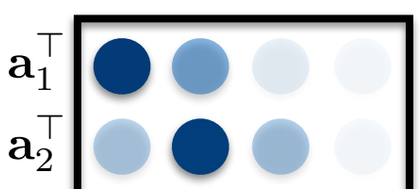


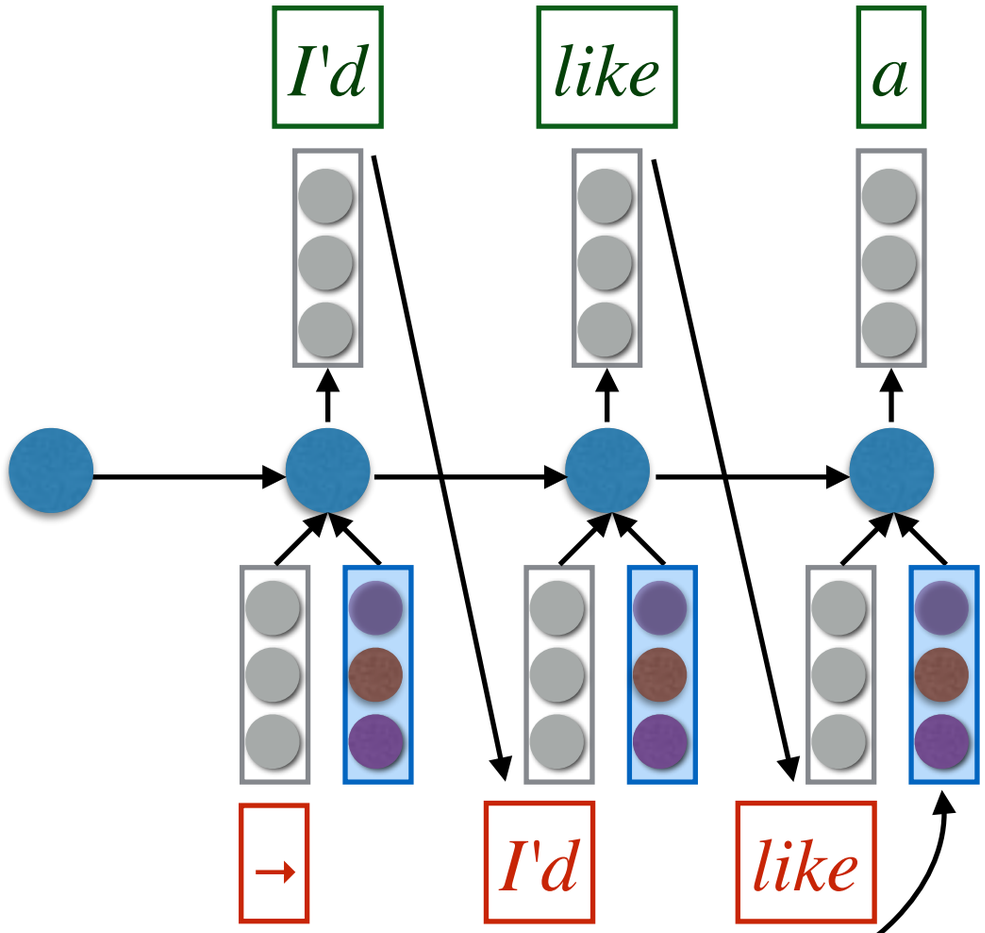


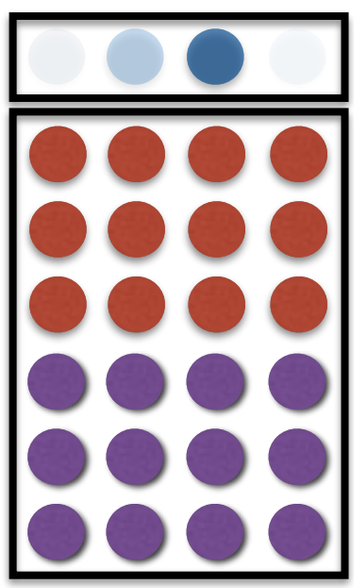


Ich möchte ein Bier

Attention history:

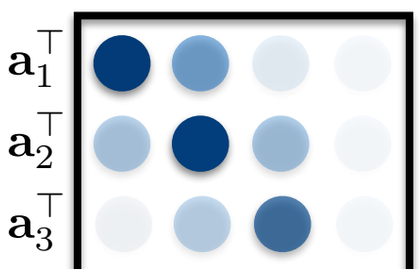


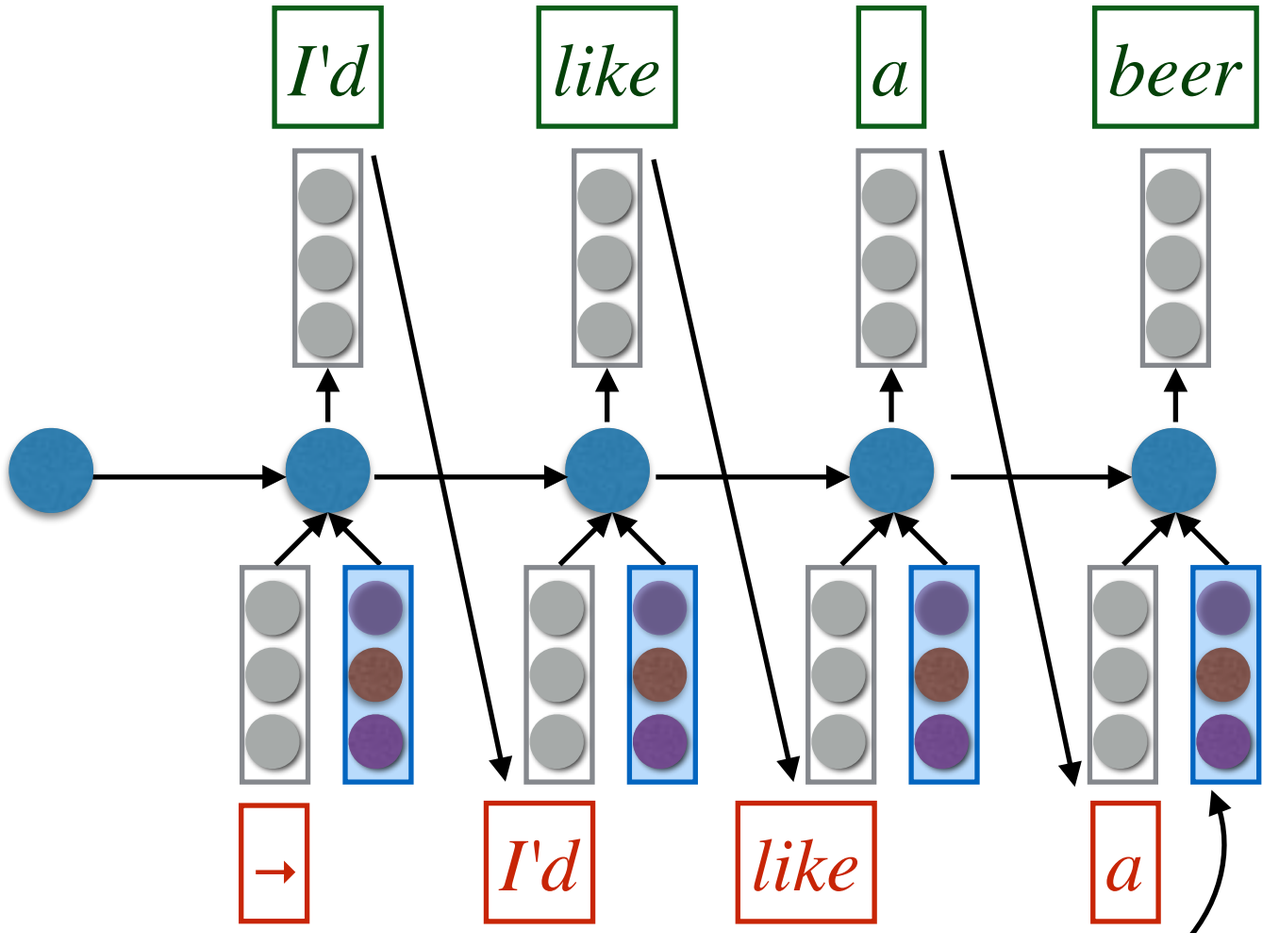


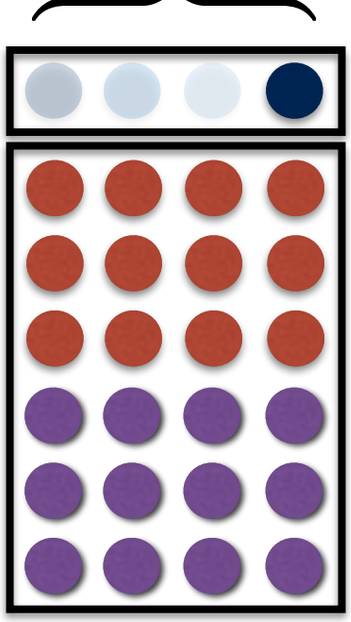


Ich möchte ein Bier

Attention history:

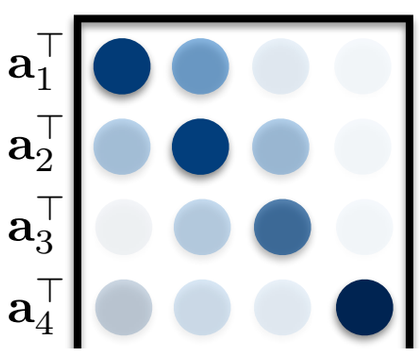


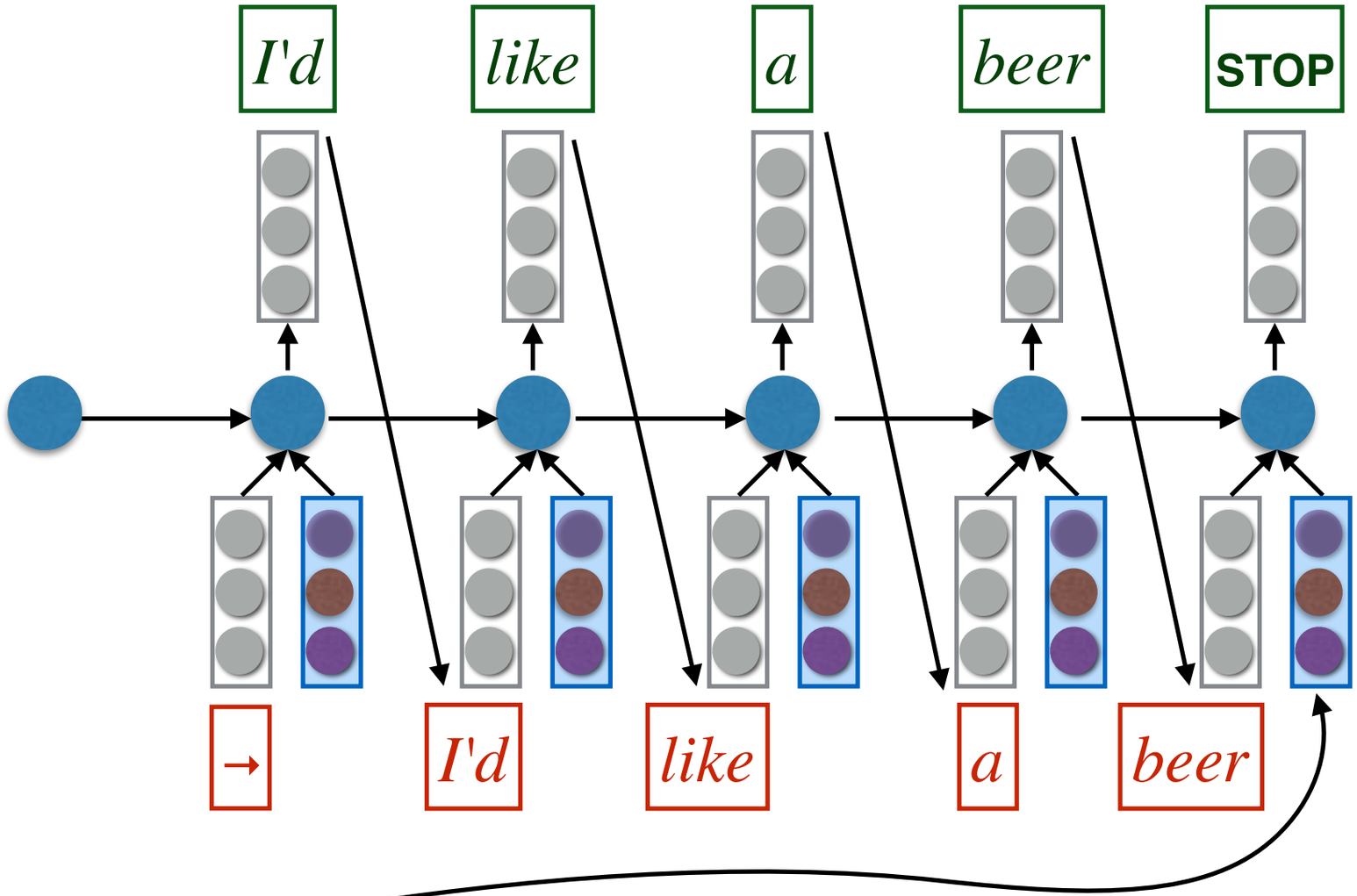


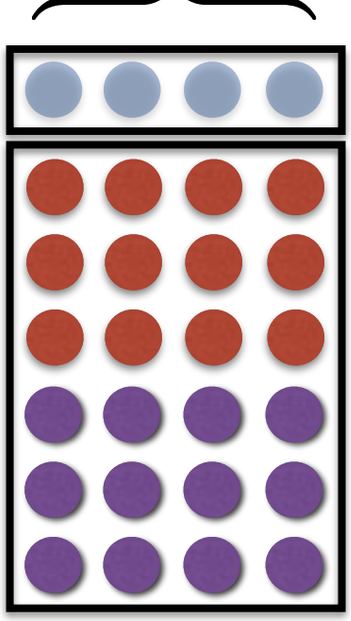


Ich möchte ein Bier

Attention history:

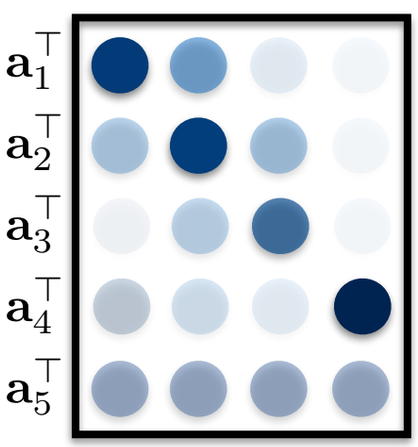






Ich möchte ein Bier

Attention history:



Attention

- How do we know what to attend to at each time-step?

- That is, how do we compute $\mathbf{a}_t$?

Computing Attention

每个time step，都会有个|f|长的向量，决定每个词的权重

- At each time step (one time step = one output word), we want to be able to "attend" to different words in the source sentence
  - We need a weight for every column: this is an $|\boldsymbol{f}|$-length vector $\mathbf{a}_t$
  - Here is a simplified version of Bahdanau et al.'s solution
    - Use an RNN to predict model output, call the hidden states $\mathbf{s}_t$ ($\mathbf{s}_t$ has a fixed dimensionality, call it m)

Computing Attention

- At each time step (one time step = one output word), we want to be able to "attend" to different words in the source sentence
  - We need a weight for every column: this is an $|\boldsymbol{f}|$-length vector $\mathbf{a}_t$
  - Here is a simplified version of Bahdanau et al.'s solution
    - Use an RNN to predict model output, call the hidden states $\mathbf{s}_t$ ($\mathbf{s}_t$ has a fixed dimensionality, call it *m*)
    - At time *t* compute the ***expected input embedding*** $\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$ ($\mathbf{V}$ is a learned parameter)

Computing Attention

- At each time step (one time step = one output word), we want to be able to "attend" to different words in the source sentence
  - We need a weight for every column: this is an $|\boldsymbol{f}|$-length vector $\mathbf{a}_t$
  - Here is a simplified version of Bahdanau et al.'s solution
    - Use an RNN to predict model output, call the hidden states $\mathbf{s}_t$
      ($\mathbf{s}_t$ has a fixed dimensionality, call it *m*)
    - At time *t* compute the ***expected input embedding*** $\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$
      ($\mathbf{V}$ is a learned parameter)
    - Take the dot product with every column in the source matrix to compute the ***attention energy***. $\mathbf{u}_t = \mathbf{F}^\top \mathbf{r}_t$ (called $\mathbf{e}_t$ in the paper)
      (Since **F** has $|\boldsymbol{f}|$ columns, $\mathbf{u}_t$ has $|\boldsymbol{f}|$ rows)

Computing Attention

- At each time step (one time step = one output word), we want to be able to "attend" to different words in the source sentence
  - We need a weight for every column: this is an $|\boldsymbol{f}|$-length vector $\mathbf{a}_t$
  - Here is a simplified version of Bahdanau et al.'s solution
    - Use an RNN to predict model output, call the hidden states $\mathbf{s}_t$
      ($\mathbf{s}_t$ has a fixed dimensionality, call it *m*)
    - At time *t* compute the ***expected input embedding*** $\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$
      ($\mathbf{V}$ is a learned parameter)
    - Take the dot product with every column in the source matrix to compute the ***attention energy***. $\mathbf{u}_t = \mathbf{F}^\top \mathbf{r}_t$ (called $\mathbf{e}_t$ in the paper)
      (Since **F** has $|\boldsymbol{f}|$ columns, $\mathbf{u}_t$ has $|\boldsymbol{f}|$ rows)
    - Exponentiate and normalize to 1: $\mathbf{a}_t = \mathrm{softmax}(\mathbf{u}_t)$
      (called $\boldsymbol{\alpha}_t$ in the paper)

Computing Attention

- At each time step (one time step = one output word), we want to be able to "attend" to different words in the source sentence
  - We need a weight for every column: this is an $|\boldsymbol{f}|$-length vector $\mathbf{a}_t$
  - Here is a simplified version of Bahdanau et al.'s solution
    - Use an RNN to predict model output, call the hidden states $\mathbf{s}_t$ ($\mathbf{s}_t$ has a fixed dimensionality, call it *m*)
    - At time *t* compute the ***expected input embedding*** $\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$ ($\mathbf{V}$ is a learned parameter)
    - Take the dot product with every column in the source matrix to compute the ***attention energy***. $\mathbf{u}_t = \mathbf{F}^\top \mathbf{r}_t$ (called $\mathbf{e}_t$ in the paper) (Since **F** has $|\boldsymbol{f}|$ columns, $\mathbf{u}_t$ has $|\boldsymbol{f}|$ rows)
    - Exponentiate and normalize to 1: $\mathbf{a}_t = \mathrm{softmax}(\mathbf{u}_t)$ (called $\boldsymbol{\alpha}_t$ in the paper)
    - Finally, the ***input source vector*** for time *t* is $\mathbf{c}_t = \mathbf{F}\mathbf{a}_t$

Nonlinear Attention-Energy Model

- In the actual model, Bahdanau et al. replace the dot product between the columns of **F** and $\mathbf{r}_t$ with an MLP:

$$\mathbf{u}_t = \mathbf{F}^\top \mathbf{r}_t \qquad \text{(simple model)}$$

Nonlinear Attention-Energy Model

- In the actual model, Bahdanau et al. replace the dot product between the columns of **F** and $\mathbf{r}_t$ with an MLP:

$$\mathbf{u}_t = \mathbf{F}^\top \mathbf{r}_t \quad \text{(simple model)}$$

$$\mathbf{u}_t = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{F} + \mathbf{r}_t) \quad \text{(Bahdanau et al)}$$

Nonlinear Attention-Energy Model

- In the actual model, Bahdanau et al. replace the dot product between the columns of **F** and $\mathbf{r}_t$ with an MLP:

$$\mathbf{u}_t = \mathbf{F}^\top \mathbf{r}_t \quad \text{(simple model)}$$

$$\mathbf{u}_t = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{F} + \mathbf{r}_t) \quad \text{(Bahdanau et al)}$$

- Here, **W** and **v** are learned parameters of appropriate dimension and + "broadcasts" over the $|\boldsymbol{f}|$ columns in **WF**

- This can learn more complex interactions

  - It is unclear if the added complexity is necessary for good performance

Putting it all together

$\mathbf{F} = \text{EncodeAsMatrix}(\boldsymbol{f})$ (Part 1 of lecture)

$e_0 = \langle \mathbf{s} \rangle$

$\mathbf{s}_0 = \mathbf{w}$ (Learned initial state; Bahdanau uses $\mathbf{U}\overleftarrow{\mathbf{h}}_1$)

$t = 0$

while $e_t \neq \langle /\mathbf{s} \rangle$:

$t = t + 1$

$\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$

$\mathbf{u}_t = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{F} + \mathbf{r}_t)$

$\mathbf{a}_t = \text{softmax}(\mathbf{u}_t)$

} (Compute attention; part 2 of lecture)

$\mathbf{c}_t = \mathbf{F}\mathbf{a}_t$

$\mathbf{s}_t = \text{RNN}(\mathbf{s}_{t-1}, [\mathbf{e}_{t-1}; \mathbf{c}_t])$ ($\mathbf{e}_{t-1}$ is a learned embedding of e_t)

$\mathbf{y}_t = \text{softmax}(\mathbf{P}\mathbf{s}_t + \mathbf{b})$ ($\mathbf{P}$ and $\mathbf{b}$ are learned parameters)

$e_t \mid \boldsymbol{e}_{<t} \sim \text{Categorical}(\mathbf{y}_t)$

Putting it all together

$\mathbf{F} = \mathrm{EncodeAsMatrix}(\boldsymbol{f})$ (Part 1 of lecture)

$e_0 = \langle \mathbf{s} \rangle$

$\mathbf{s}_0 = \mathbf{w}$ (Learned initial state; Bahdanau uses $\mathbf{U}\overleftarrow{\mathbf{h}}_1$)

$t = 0$

while $e_t \neq \langle /\mathbf{s} \rangle$:

$t = t + 1$

doesn't depend on output decisions

$\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$

$\mathbf{u}_t = \mathbf{v}^\top \tanh(\mathbf{WF} + \mathbf{r}_t)$

$\mathbf{a}_t = \mathrm{softmax}(\mathbf{u}_t)$

} (Compute attention; part 2 of lecture)

$\mathbf{c}_t = \mathbf{F}\mathbf{a}_t$

$\mathbf{s}_t = \mathrm{RNN}(\mathbf{s}_{t-1}, [\mathbf{e}_{t-1}; \mathbf{c}_t])$ ($\mathbf{e}_{t-1}$ is a learned embedding of e_t)

$\mathbf{y}_t = \mathrm{softmax}(\mathbf{P}\mathbf{s}_t + \mathbf{b})$ ($\mathbf{P}$ and $\mathbf{b}$ are learned parameters)

$e_t \mid \boldsymbol{e}_{<t} \sim \mathrm{Categorical}(\mathbf{y}_t)$

Putting it all together

$\mathbf{F} = \text{EncodeAsMatrix}(\boldsymbol{f})$ (Part 1 of lecture)

$e_0 = \langle \mathbf{s} \rangle$

$\mathbf{s}_0 = \mathbf{w}$ (Learned initial state; Bahdanau uses $\mathbf{U}\overleftarrow{\mathbf{h}}_1$)

$t = 0$

$\mathbf{X} = \mathbf{WF}$

while $e_t \neq \langle /\mathbf{s} \rangle$:

$t = t + 1$

$\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$

$\mathbf{u}_t = \mathbf{v}^\top \tanh(\mathbf{WF} + \mathbf{r}_t)$

$\mathbf{a}_t = \text{softmax}(\mathbf{u}_t)$

} (Compute attention; part 2 of lecture)

$\mathbf{c}_t = \mathbf{F}\mathbf{a}_t$

$\mathbf{s}_t = \text{RNN}(\mathbf{s}_{t-1}, [\mathbf{e}_{t-1}; \mathbf{c}_t])$ ($\mathbf{e}_{t-1}$ is a learned embedding of e_t)

$\mathbf{y}_t = \text{softmax}(\mathbf{P}\mathbf{s}_t + \mathbf{b})$ ($\mathbf{P}$ and $\mathbf{b}$ are learned parameters)

$e_t \mid \boldsymbol{e}_{<t} \sim \text{Categorical}(\mathbf{y}_t)$

Putting it all together

$\mathbf{F} = \text{EncodeAsMatrix}(\boldsymbol{f})$ (Part 1 of lecture)

$e_0 = \langle \mathbf{s} \rangle$

$\mathbf{s}_0 = \mathbf{w}$ (Learned initial state; Bahdanau uses $\mathbf{U}\overleftarrow{\mathbf{h}}_1$)

$t = 0$

$\mathbf{X} = \mathbf{WF}$

while $e_t \neq \langle \mathbf{/s} \rangle$:

$t = t + 1$

$\mathbf{r}_t = \mathbf{V}\mathbf{s}_{t-1}$

$\mathbf{u}_t = \mathbf{v}^\top \tanh(\mathbf{X} + \mathbf{r}_t)$

$\mathbf{a}_t = \text{softmax}(\mathbf{u}_t)$

} (Compute attention; part 2 of lecture)

$\mathbf{c}_t = \mathbf{F}\mathbf{a}_t$

$\mathbf{s}_t = \text{RNN}(\mathbf{s}_{t-1}, [\mathbf{e}_{t-1}; \mathbf{c}_t])$ ($\mathbf{e}_{t-1}$ is a learned embedding of e_t)

$\mathbf{y}_t = \text{softmax}(\mathbf{P}\mathbf{s}_t + \mathbf{b})$ ($\mathbf{P}$ and $\mathbf{b}$ are learned parameters)

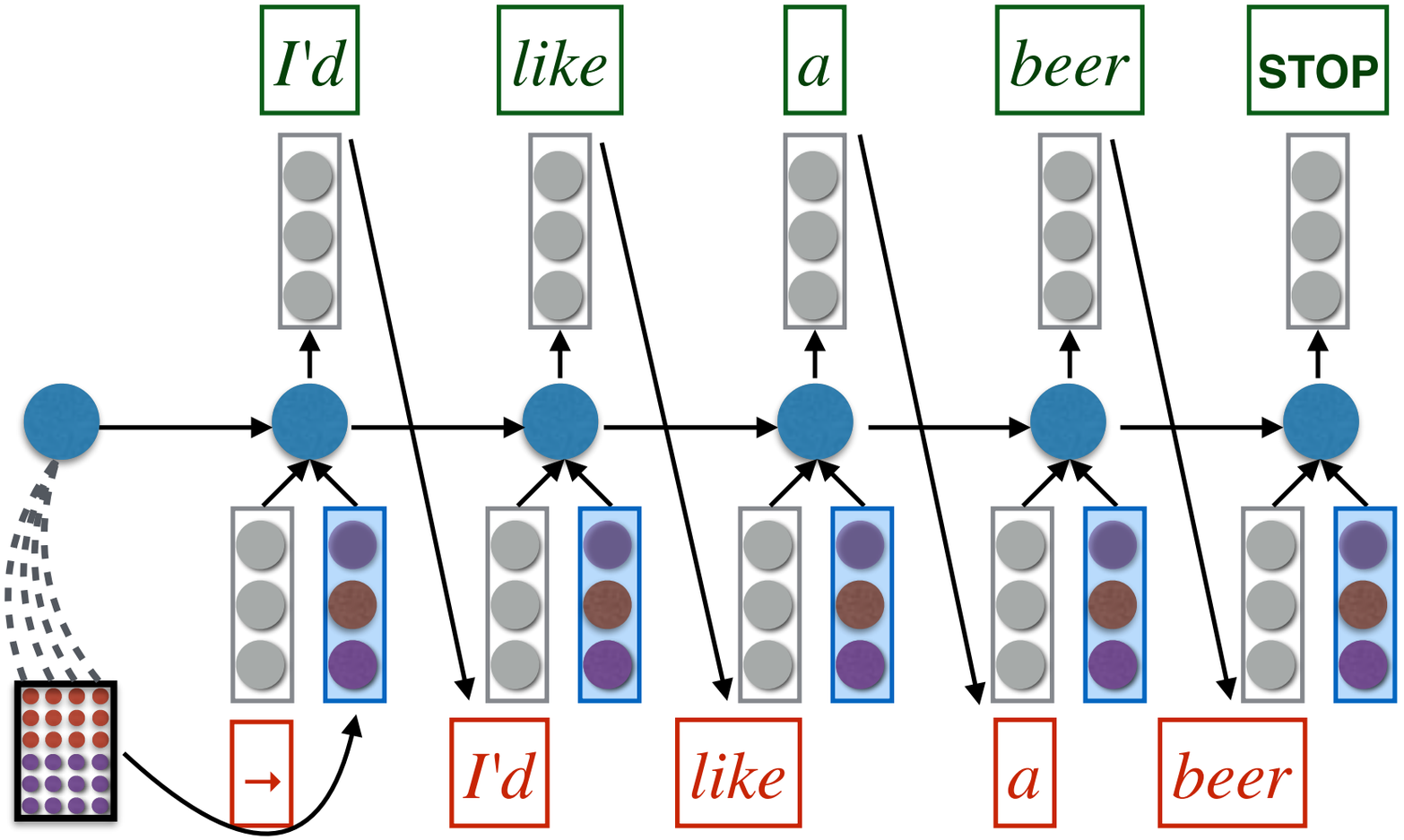
$e_t \mid \boldsymbol{e}_{<t} \sim \text{Categorical}(\mathbf{y}_t)$

Attention in MT

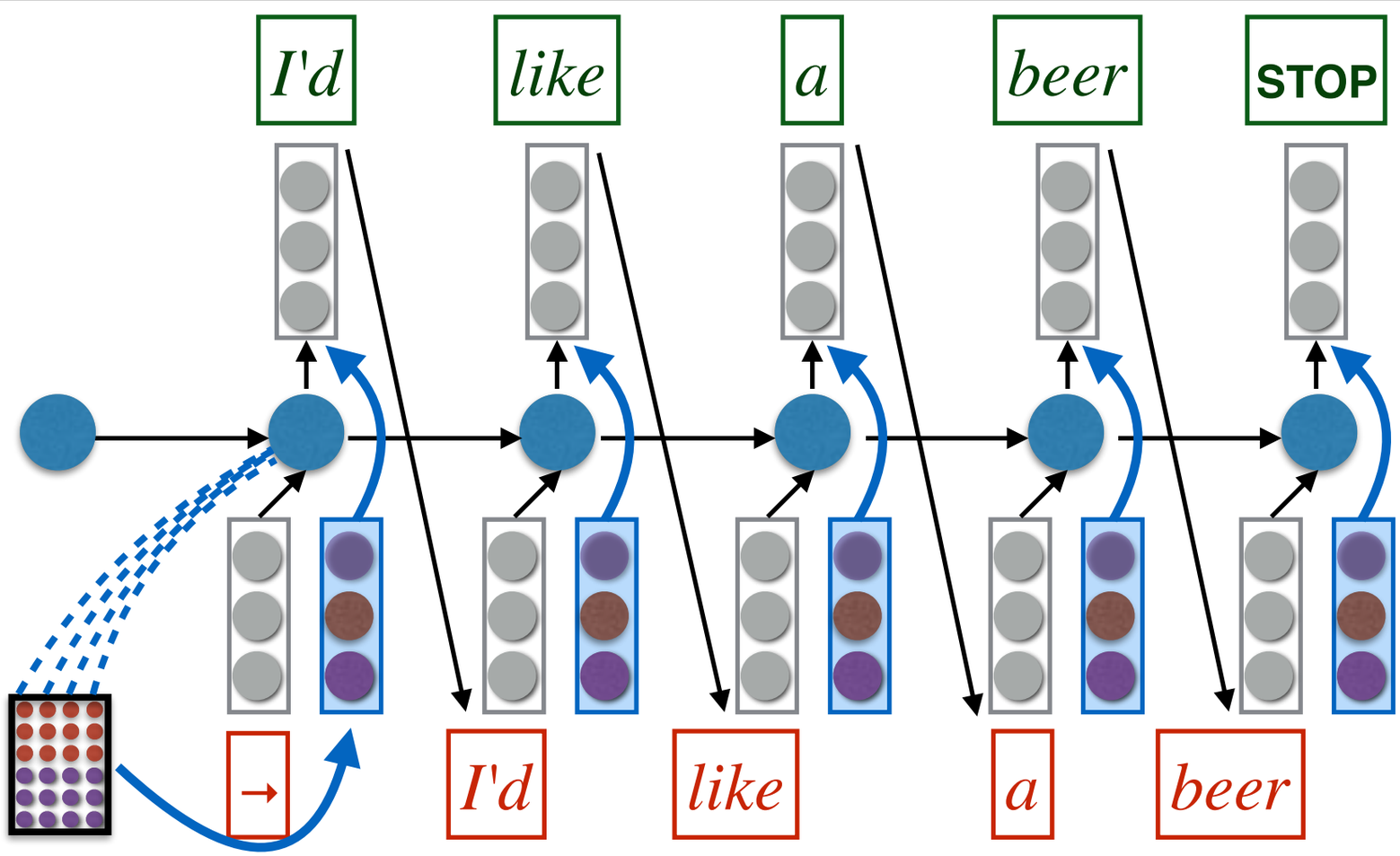
Add attention to seq2seq translation: **+11 BLEU**

Model Variant

“Early binding”



“Late binding”



Model Variant

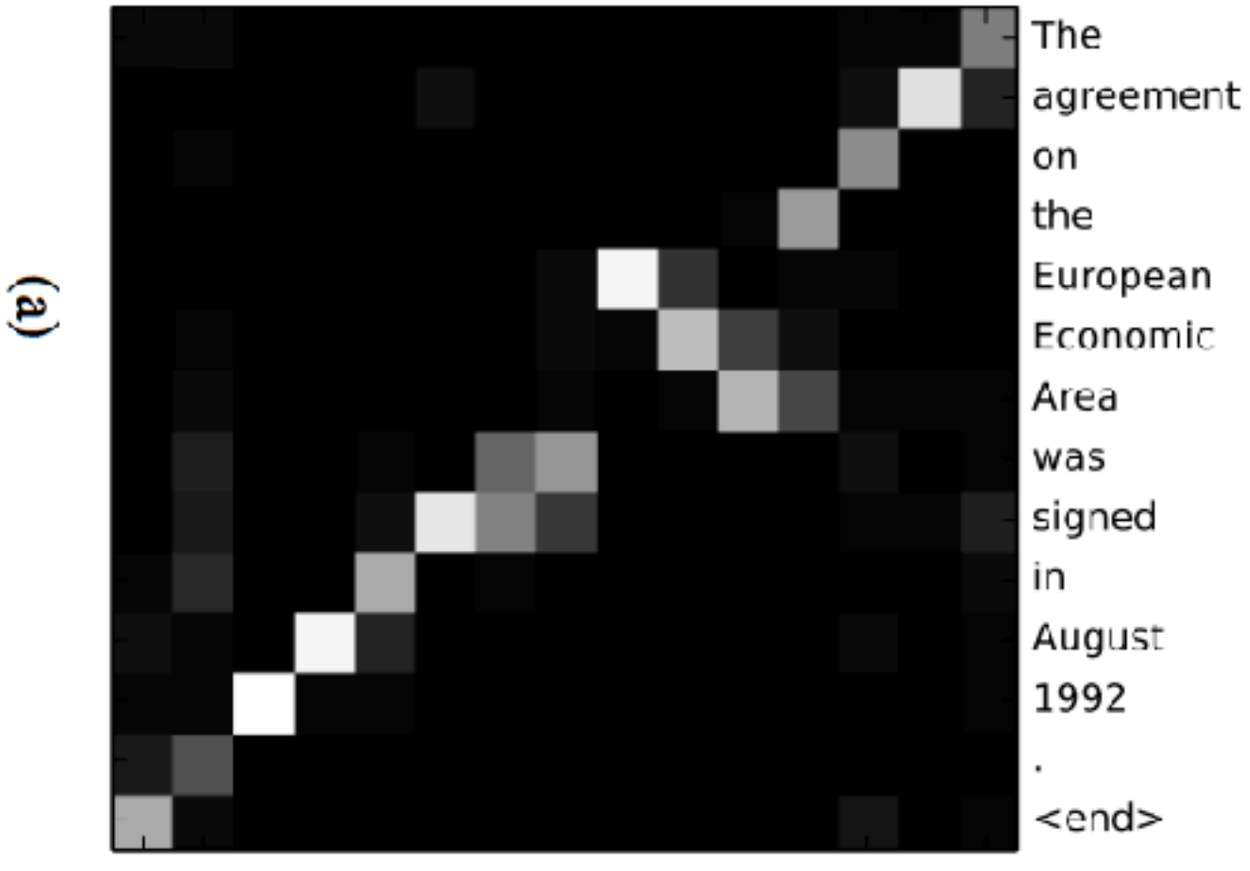
- What are the relative advantages of early binding versus late binding?

Summary

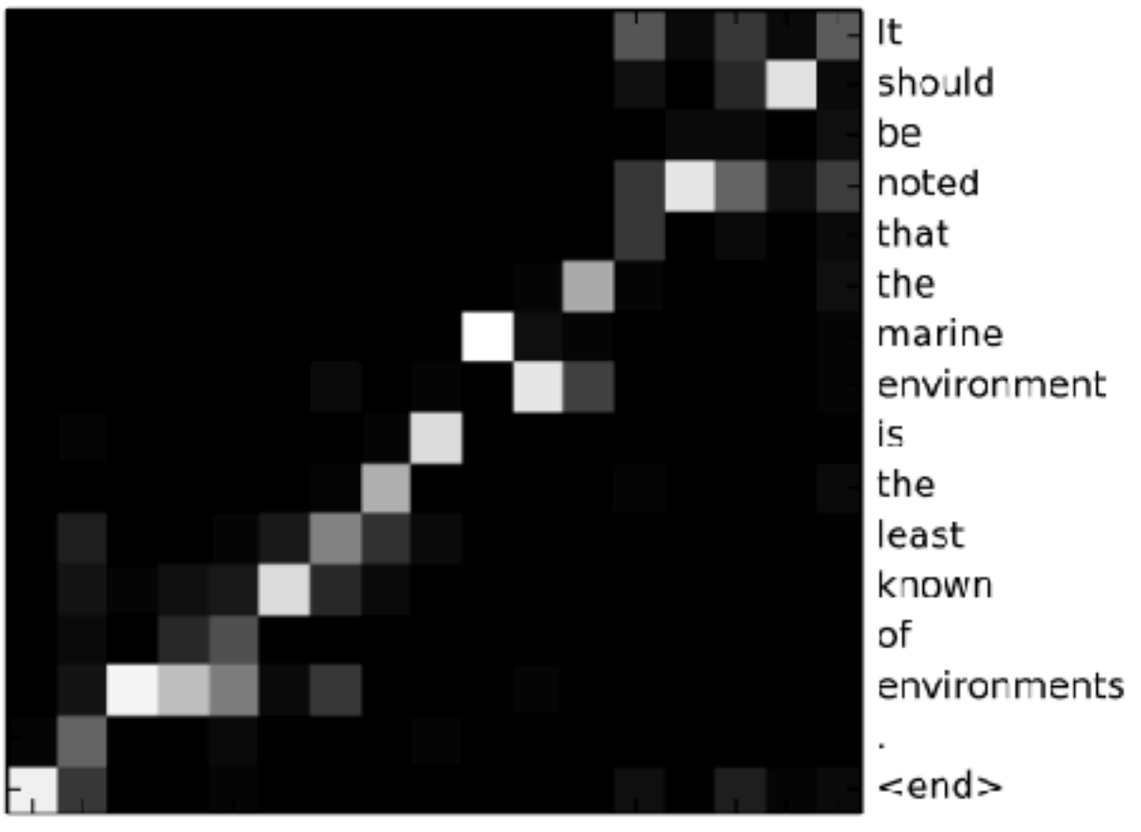
- Attention is closely related to “pooling” operations in convnets (and other architectures)

- Bahdanau’s attention model seems to only cares about “content”
    - No obvious bias in favor of diagonals, short jumps, fertility, etc.
    - Some work has begun to add other “structural” biases (Luong et al., 2015; Cohn et al., 2016), but there are lots more opportunities

- Attention weights provide interpretation you can look at

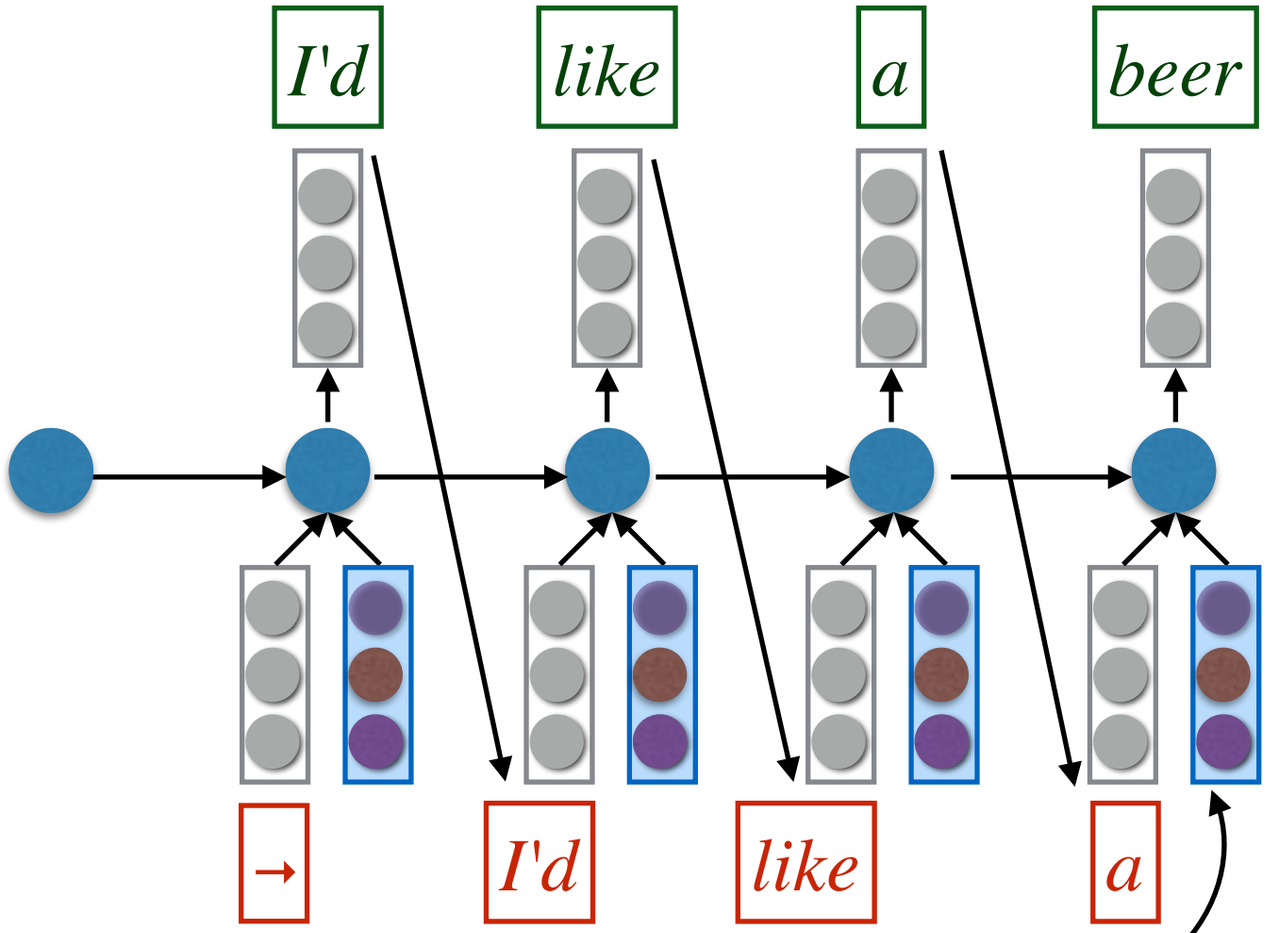

L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>
The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>

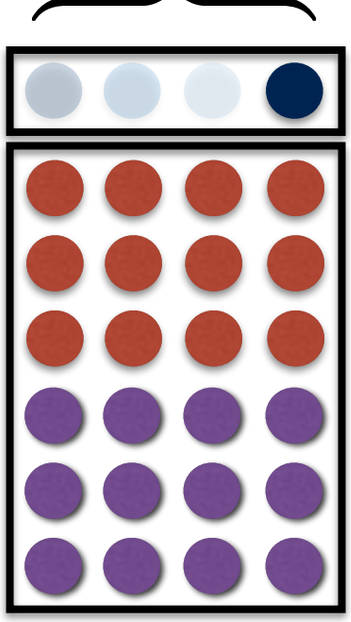
(a)


Il
convient
de
noter
que
l'
environnement
marin
est
le
moins
connu
de
l'
environnement
.
<end>
It
should
be
noted
that
the
marine
environment
is
the
least
known
of
environments
.
<end>

(b)

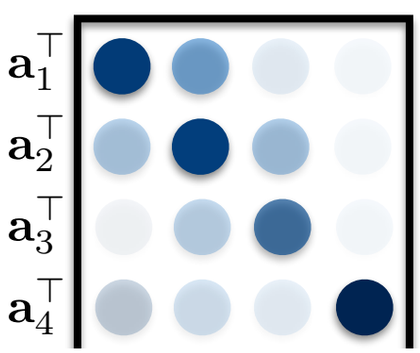
A word about gradients

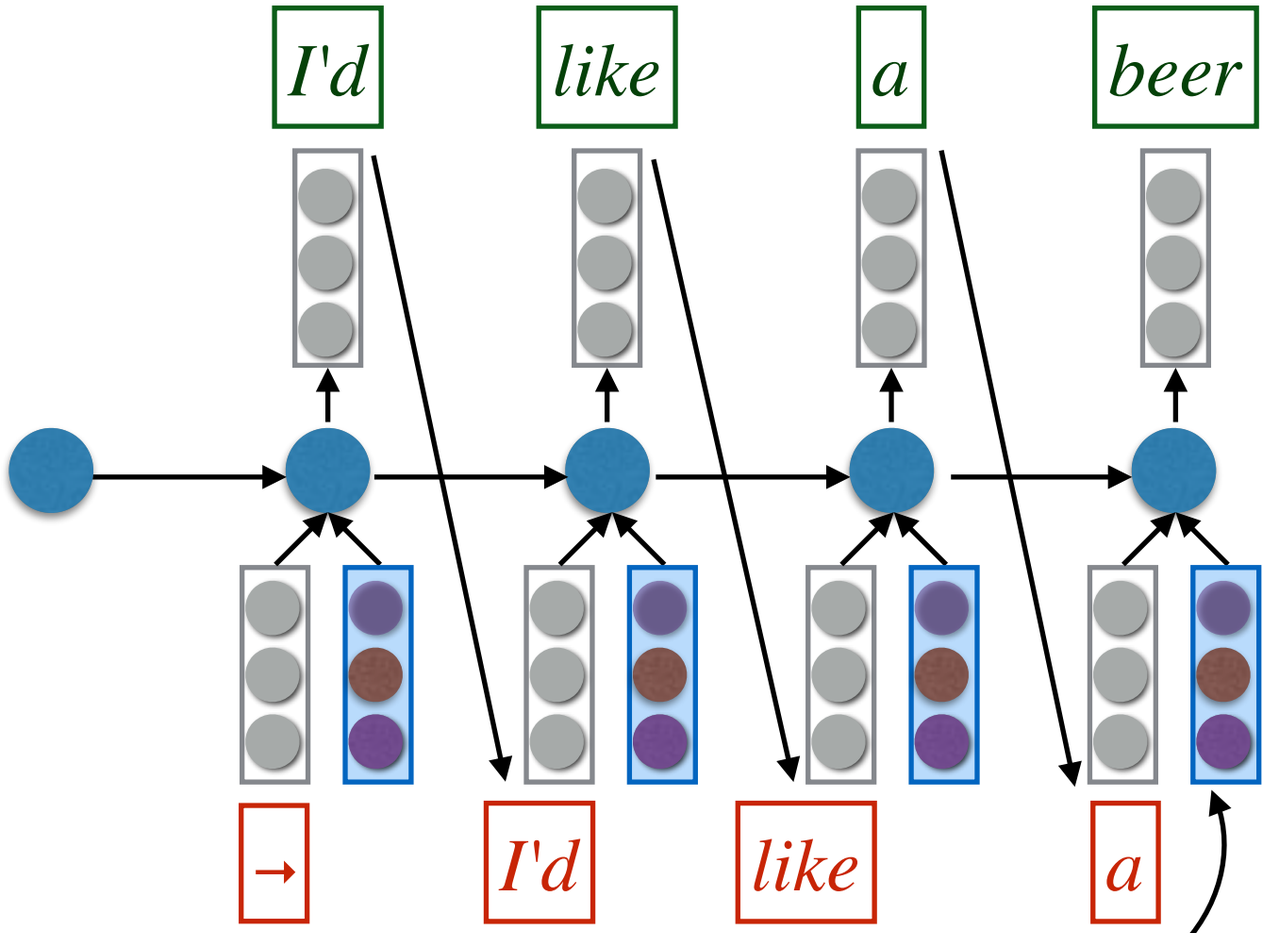


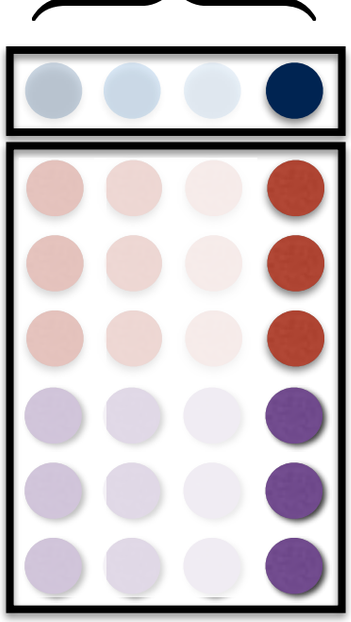


Ich möchte ein Bier

Attention history:

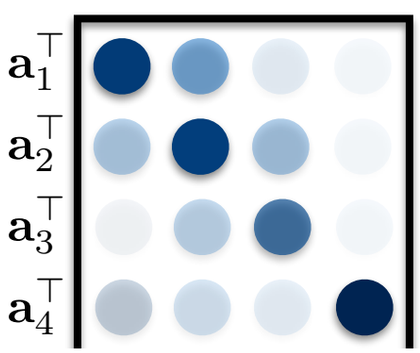


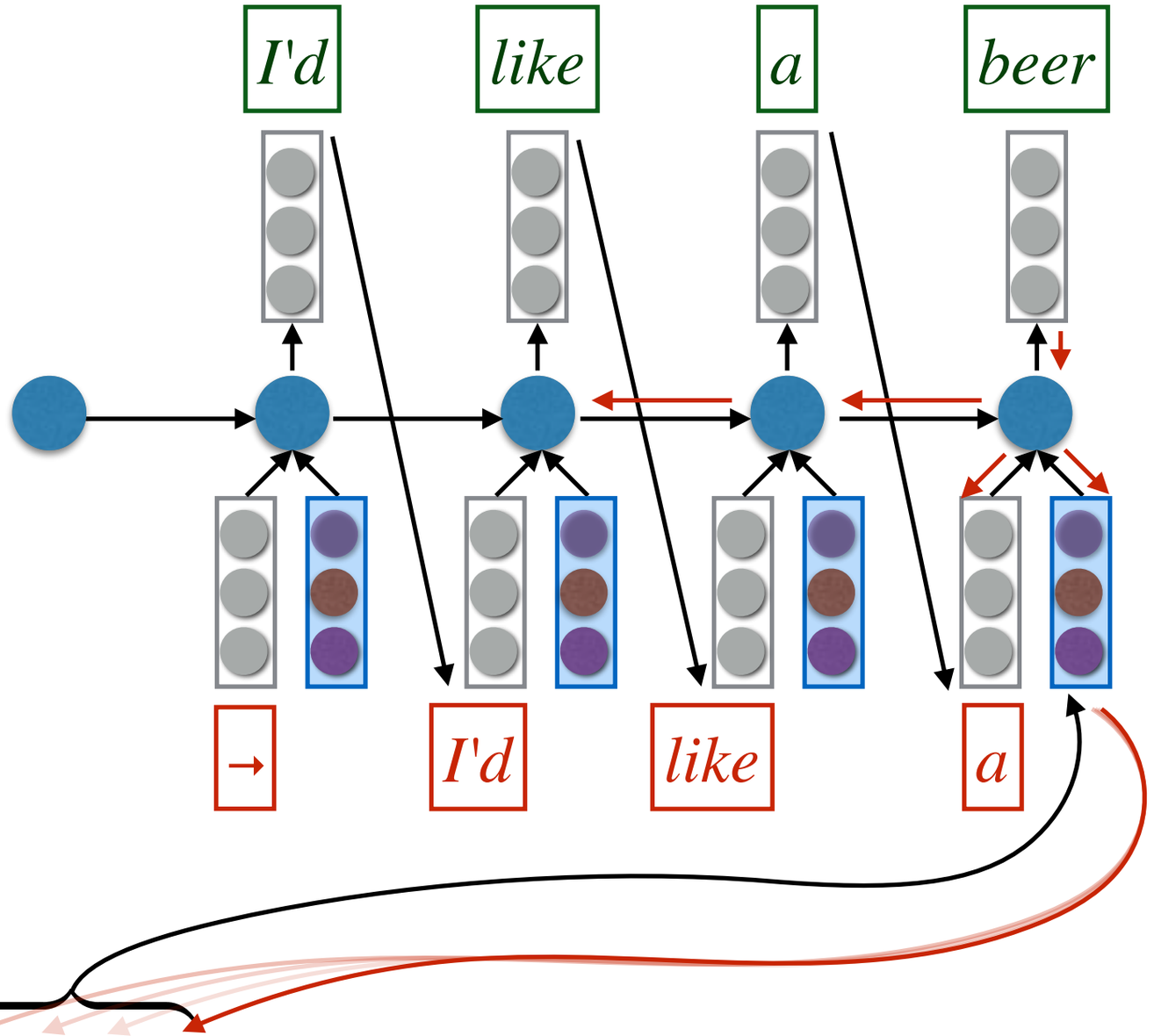


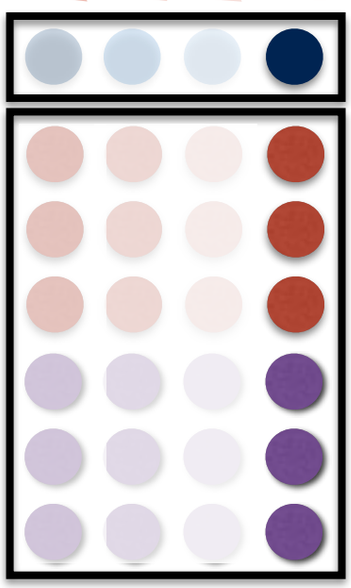


Ich möchte ein Bier

Attention history:

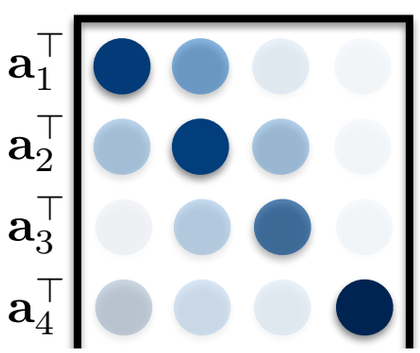






Ich möchte ein Bier

Attention history:



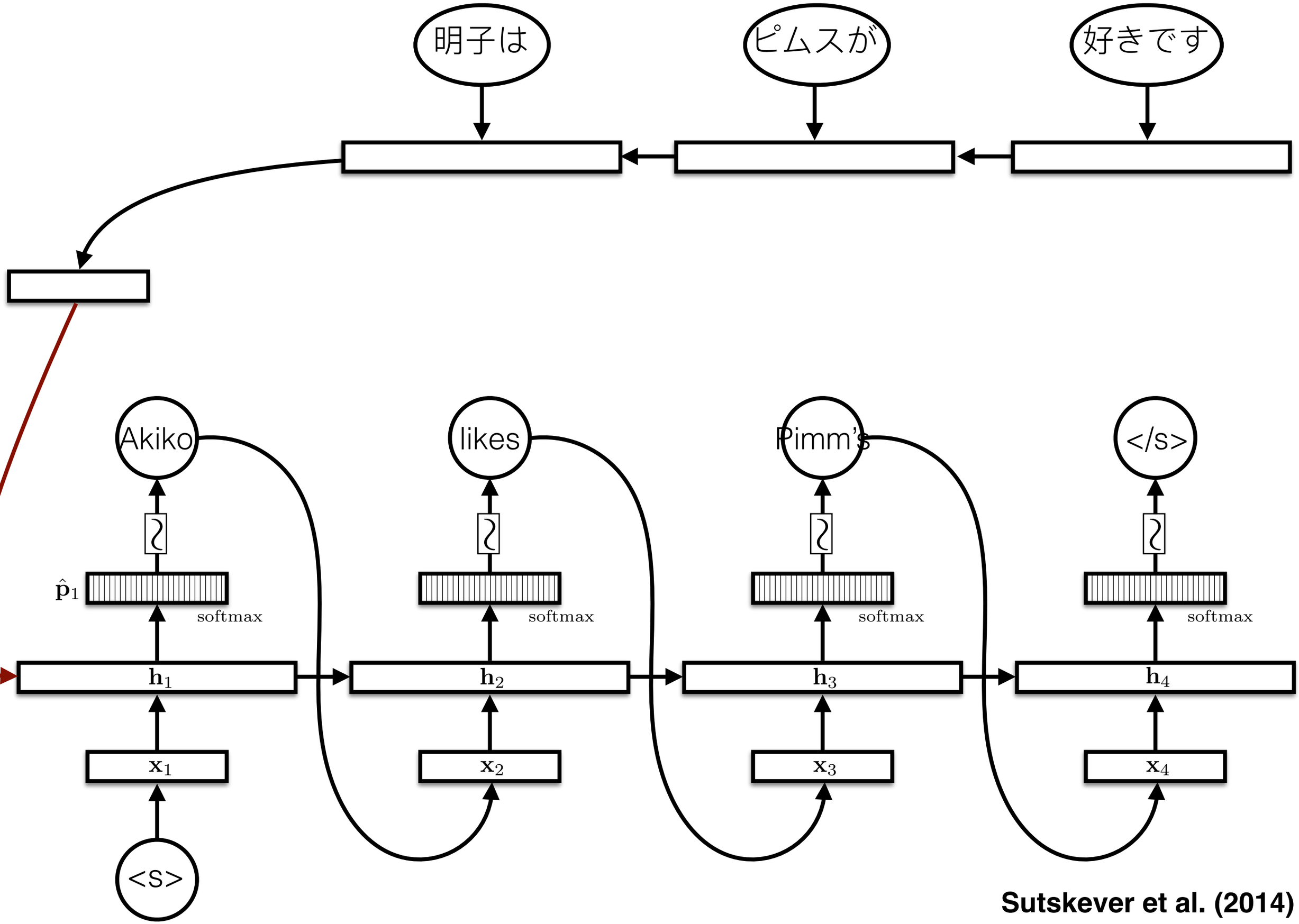
Attention and Translation

- Cho's question: does a translator read and memorize the input sentence/document and then generate the output?
  - Compressing the entire input sentence into a vector basically says "memorize the sentence"
  - Common sense experience says translators refer back and forth to the input. (also backed up by eye-tracking studies)
- Should humans be a model for machines?

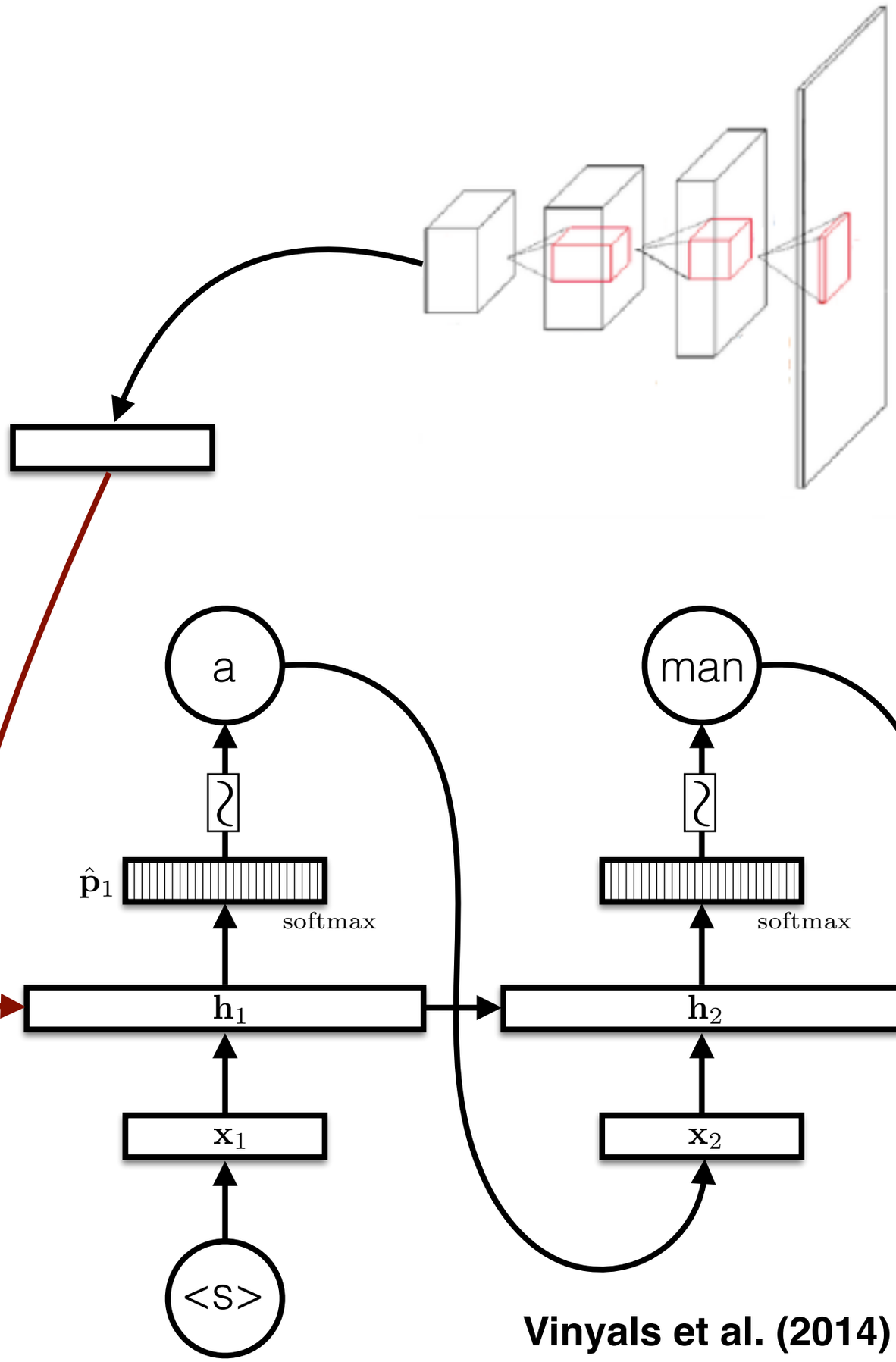
Outline of Lecture

- Machine translation with attention

- Image caption generation with attention



Sutskever et al. (2014)



Vinyals et al. (2014) Show and Tell: A Neural Image Caption Generator

Image Caption Generation

- Can attention help caption modeling?

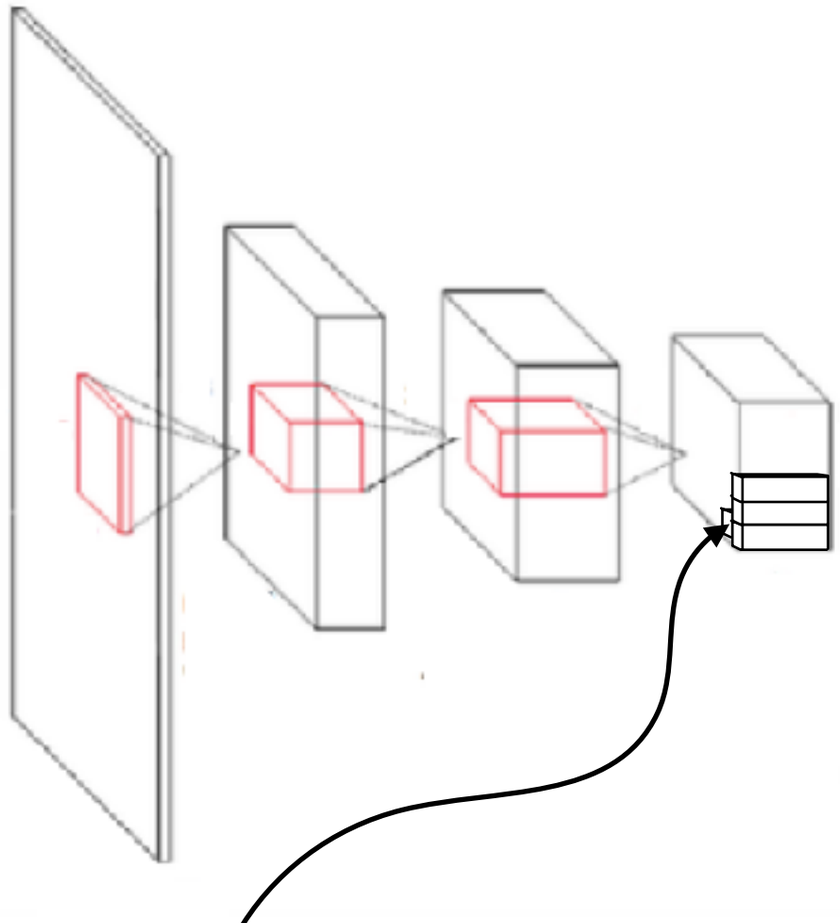
Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

---

Kelvin Xu KELVIN.XU@UMONTREAL.CA
Jimmy Lei Ba JIMMY@PSI.UTORONTO.CA
Ryan Kiros RKIROS@CS.TORONTO.EDU
Kyunghyun Cho KYUNGHYUN.CHO@UMONTREAL.CA
Aaron Courville AARON.COURVILLE@UMONTREAL.CA
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Richard S. Zemel ZEMEL@CS.TORONTO.EDU
Yoshua Bengio FIND-ME@THE.WEB

Xu et al. (2015, ICML)

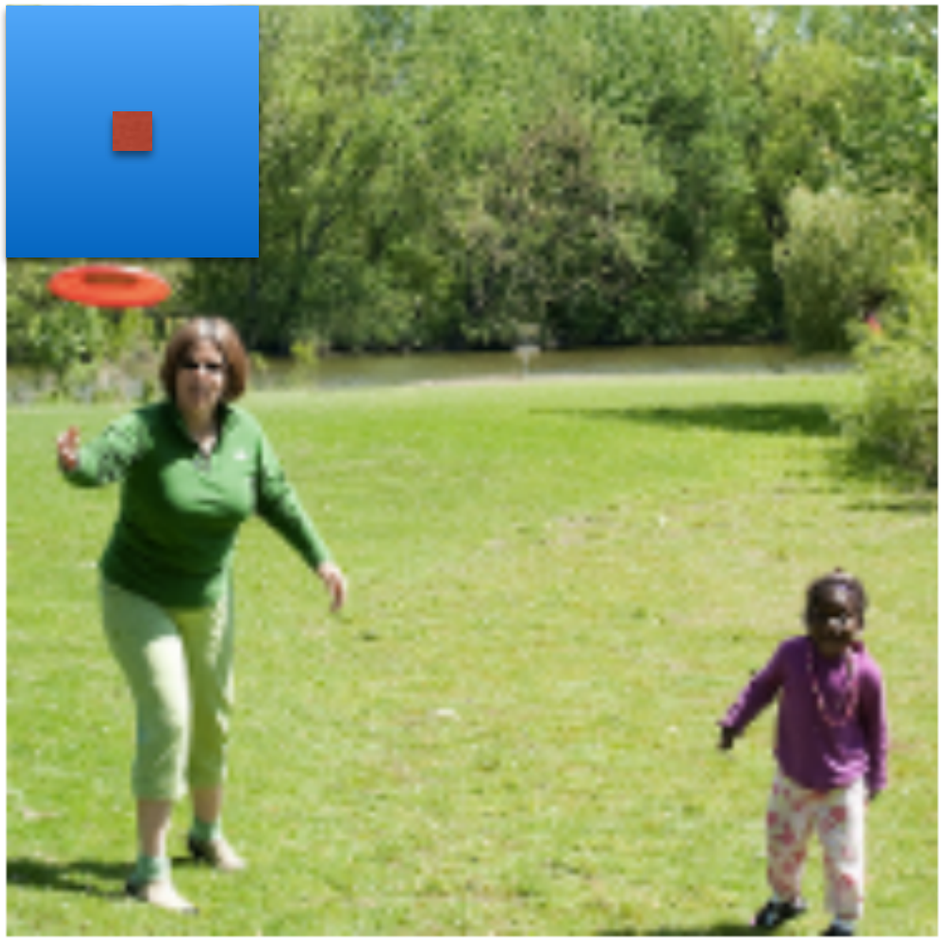
Regions in ConvNets



Each point in a “higher” level of a convnet
defines spatially localised feature vectors(/matrices).

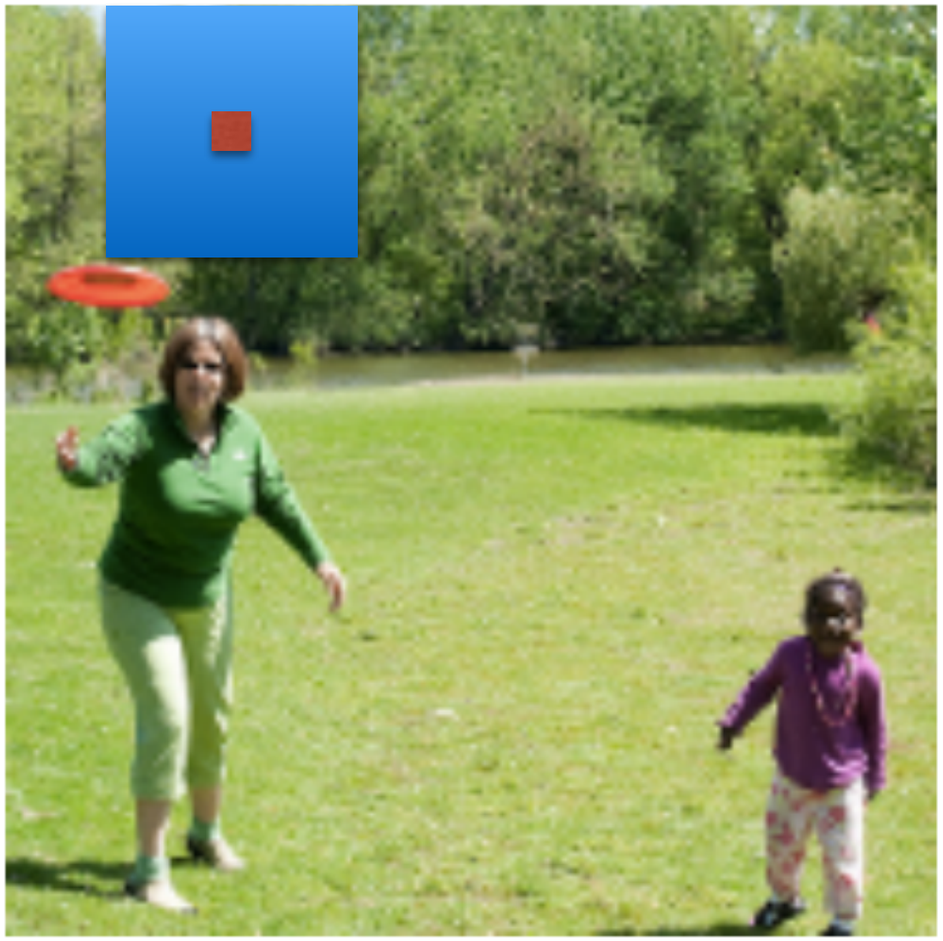
Xu et al. calls these “annotation vectors”, $\mathbf{a}_i,\ i \in \{1, \ldots, L\}$

$\mathbf{a}_1$



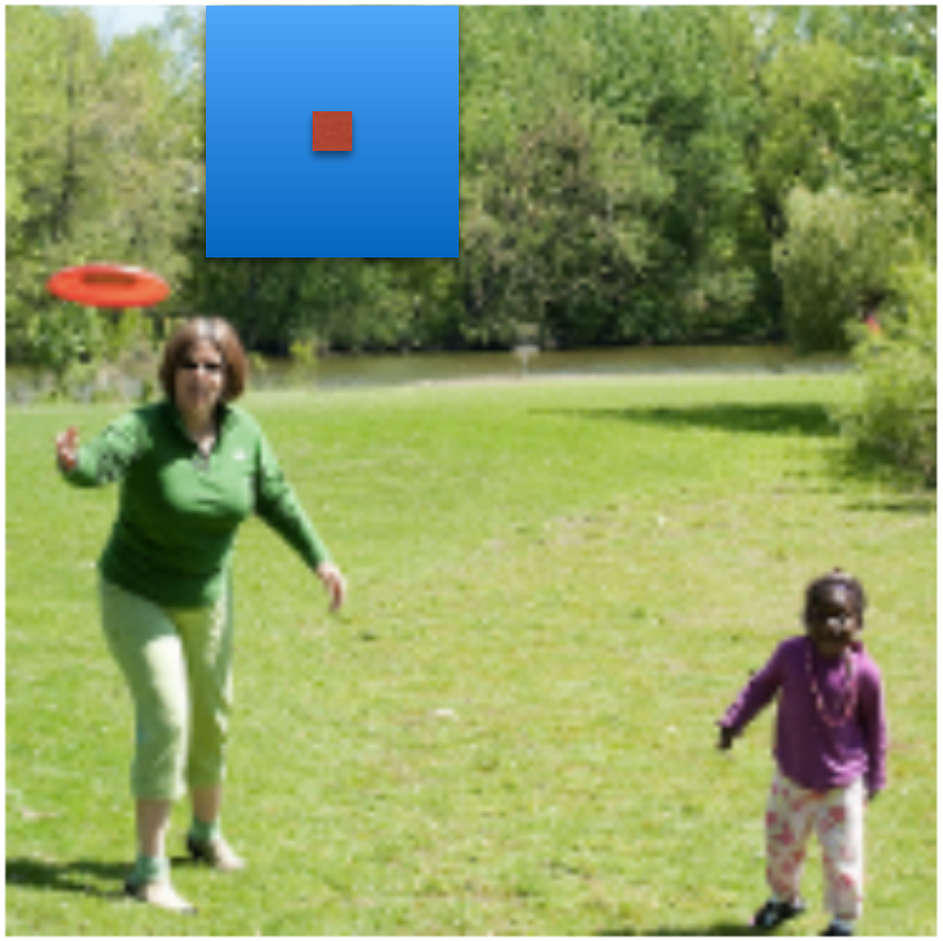
$$\mathbf{F} = \begin{bmatrix} | \\ \mathbf{a}_1 \\ | \end{bmatrix}$$

$\mathbf{a}_2$



$$\mathbf{F} = \begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix}$$

$\mathbf{a}_3$



$$\mathbf{F} = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \cdots \\ | & | & | \end{bmatrix}$$

Attention

- Attention "weights" ($\mathbf{a}_t$) are computed using exactly the same technique as discussed above

Attention

- Attention "weights" ($\mathbf{a}_t$) are computed using exactly the same technique as discussed above

- Deterministic soft attention (Bahdanau et al., 2014)

$$\mathbf{c}_t = \mathbf{F}\mathbf{a}_t$$ (weighted average)

Attention

- Attention "weights" ($\mathbf{a}_t$) are computed using exactly the same technique as discussed above

- Deterministic soft attention (Bahdanau et al., 2014)

$$\mathbf{c}_t = \mathbf{F}\mathbf{a}_t \qquad \text{(weighted average)}$$

- Stochastic hard attention (Xu et al., 2015)

$$s_t \sim \text{Categorical}(\mathbf{a}_t)$$

$$\mathbf{c}_t = \mathbf{F}_{:,s_t} \qquad \text{(sample a column)}$$

  - What are the benefits of this model?

  - What are the challenges of learning the parameters of this model?

Learning Hard Attention

$$\mathcal{L} = -\log p(\boldsymbol{w} \mid \boldsymbol{x})$$

$$= -\log \sum_{\boldsymbol{s}} p(\boldsymbol{w}, \boldsymbol{s} \mid \boldsymbol{x})$$

$$= -\log \sum_{\boldsymbol{s}} p(\boldsymbol{s} \mid \boldsymbol{x}) p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{s})$$

Learning Hard Attention

$$\begin{aligned}
\mathcal{L} &= -\log p(\boldsymbol{w} \mid \boldsymbol{x}) \\
&= -\log \sum_{\boldsymbol{s}} p(\boldsymbol{w}, \boldsymbol{s} \mid \boldsymbol{x}) \\
&= -\log \sum_{\boldsymbol{s}} p(\boldsymbol{s} \mid \boldsymbol{x}) p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{s}) \\
&\leq -\sum p(\boldsymbol{s} \mid \boldsymbol{x}) \log p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{s}) \quad \text{(Jensen's inequality)}
\end{aligned}$$

Learning Hard Attention

$$
\begin{aligned}
\mathcal{L} &= -\log p(\boldsymbol{w} \mid \boldsymbol{x}) \\
&= -\log \sum_{\boldsymbol{s}} p(\boldsymbol{w}, \boldsymbol{s} \mid \boldsymbol{x}) \\
&= -\log \sum_{\boldsymbol{s}} p(\boldsymbol{s} \mid \boldsymbol{x}) p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{s}) \\
&\leq -\sum_{\boldsymbol{s}} p(\boldsymbol{s} \mid \boldsymbol{x}) \log p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{s}) \quad \text{(Jensen's inequality)} \\
&\overset{\text{MC}}{\approx} -\frac{1}{N} \sum_{i=1}^{N} p(\boldsymbol{s}^{(i)} \mid \boldsymbol{x}) \log p(\boldsymbol{w} \mid \boldsymbol{x}, \boldsymbol{s})
\end{aligned}
$$

Learning Hard Attention

- Sample *N* sequences of attention decisions from the model

- The gradient is the probability of the gradient of the probability of this sequence scaled by the log probability of generating the target words using that sequence of attention decisions

- This is equivalent to using the REINFORCE algorithm (Williams, 1992) using the log probability of the observed words as a “reward function”. REINFORCE a policy gradient algorithm used for reinforcement learning.

A(0.98)

woman(0.54)

is(0.37)

throwing(0.33)

a(0.28)

frisbee(0.37)

in(0.21)

a(0.18)

park(0.35)

.(0.33)

A woman holding a clock in her hand.

A large white bird standing in a forest.

Attention in Captioning

Add soft attention to image captioning: **+2 BLEU**

Add hard attention to image captioning: **+4 BLEU**

Summary

- Significant performance improvements
  - Better performance over vector-based encodings
  - Better performance with smaller training data sets
- Model interpretability
- Better gradient flow
- Better capacity (especially obvious for translation)

Questions?