UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020)· 课程资料包 @ShowMeAI

视频	课件	笔记	代码
中英双语字幕	一键打包下载	官方笔记翻译	作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1BL411t7RV

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/umass-cs685

NLP 语言模型 问答系统 文本生成 BERT 语义解析 GPT-3 知识推理 模型蒸馏 迁移学习 transformer 注意力机制

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n

 # Awesome AI Courses Notes Cheatsheets· 持续更新中

知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]

Attention mechanisms

CS 685, Fall 2020
Advanced Natural Language Processing

Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst

some slides from Richard Socher & Emma Strubell

stuff from last time…

- HW0 grading hopefully done by next week
- HW1 will be out within the next 1-2 weeks
- Project proposals due 9/21, all group assignments have been finalized

A RNN Language Model

output distribution

$$\hat{y} = \text{softmax}(W_2 h^{(t)} + b_2)$$

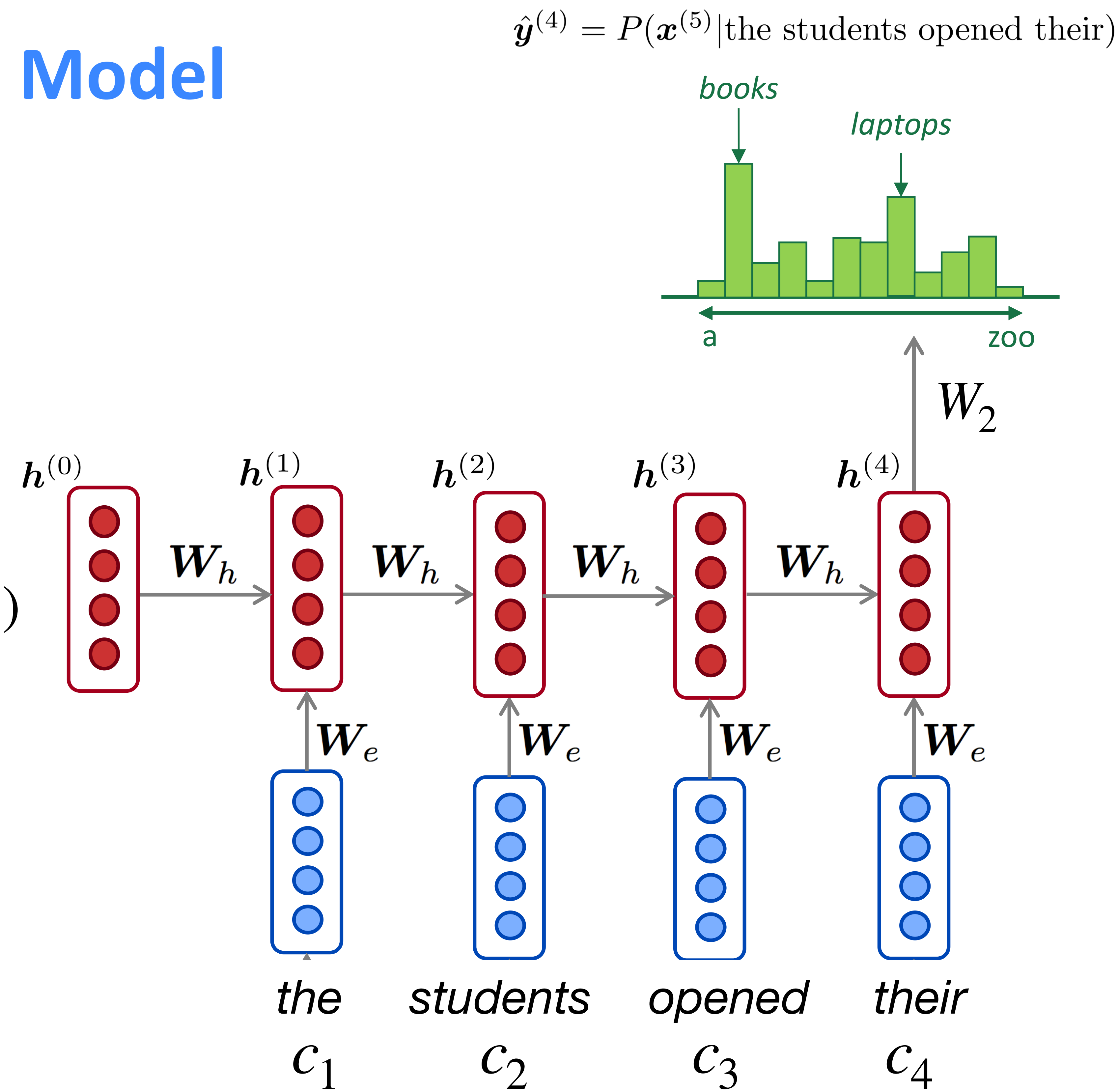
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t + b_1)$$

$h^{(0)}$ is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



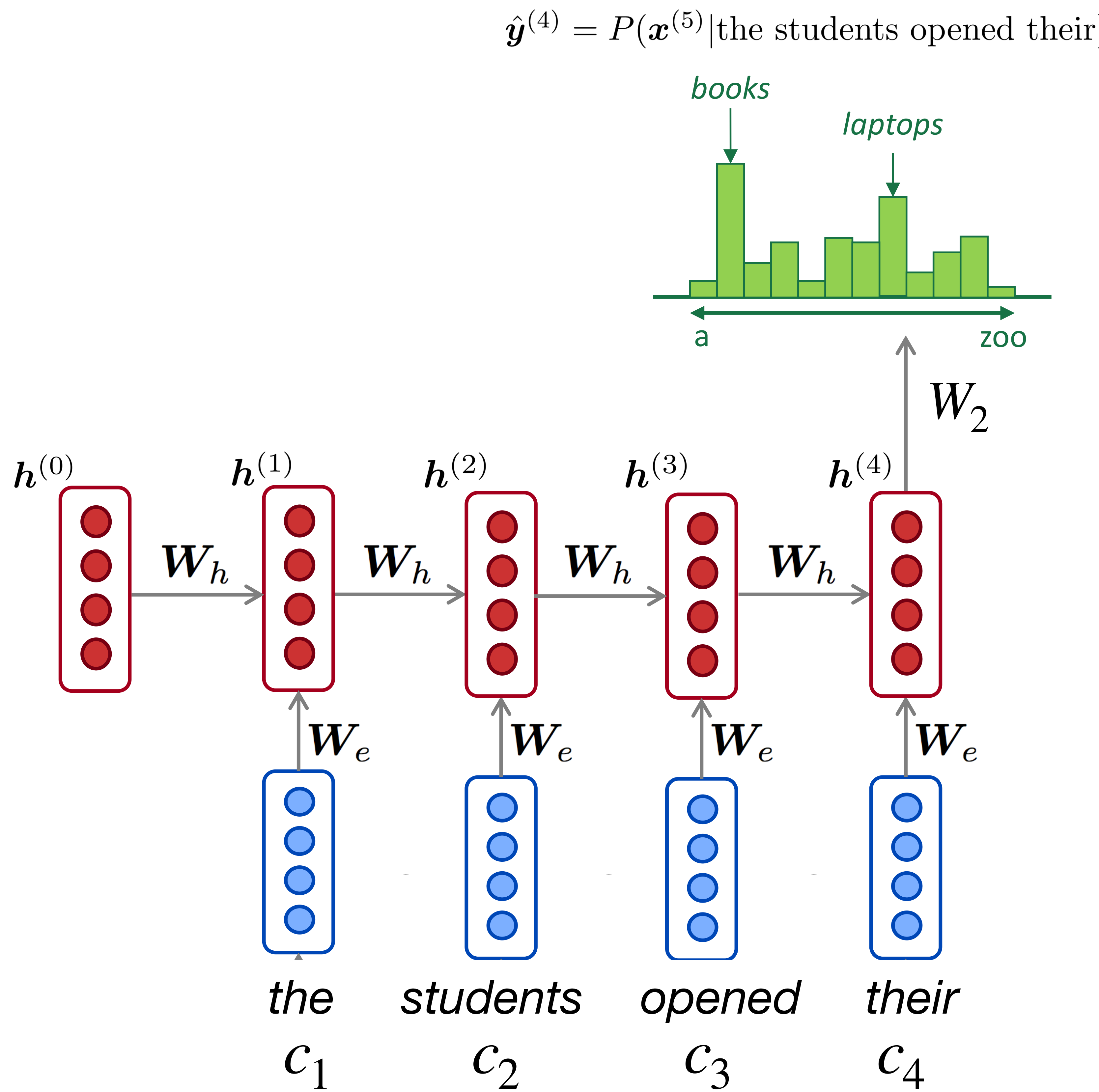
why is this good?

RNN **Advantages**:

- Can process any length input
- Model size doesn't increase for longer input
- Computation for step *t* can (in theory) use information from many steps back
- Weights are shared across timesteps → representations are shared

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back



Training a RNN Language Model

- Get a big corpus of text which is a sequence of words $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\boldsymbol{y}}^{(t)}$ for *every step t.*
  - i.e. predict probability dist of *every word*, given words so far

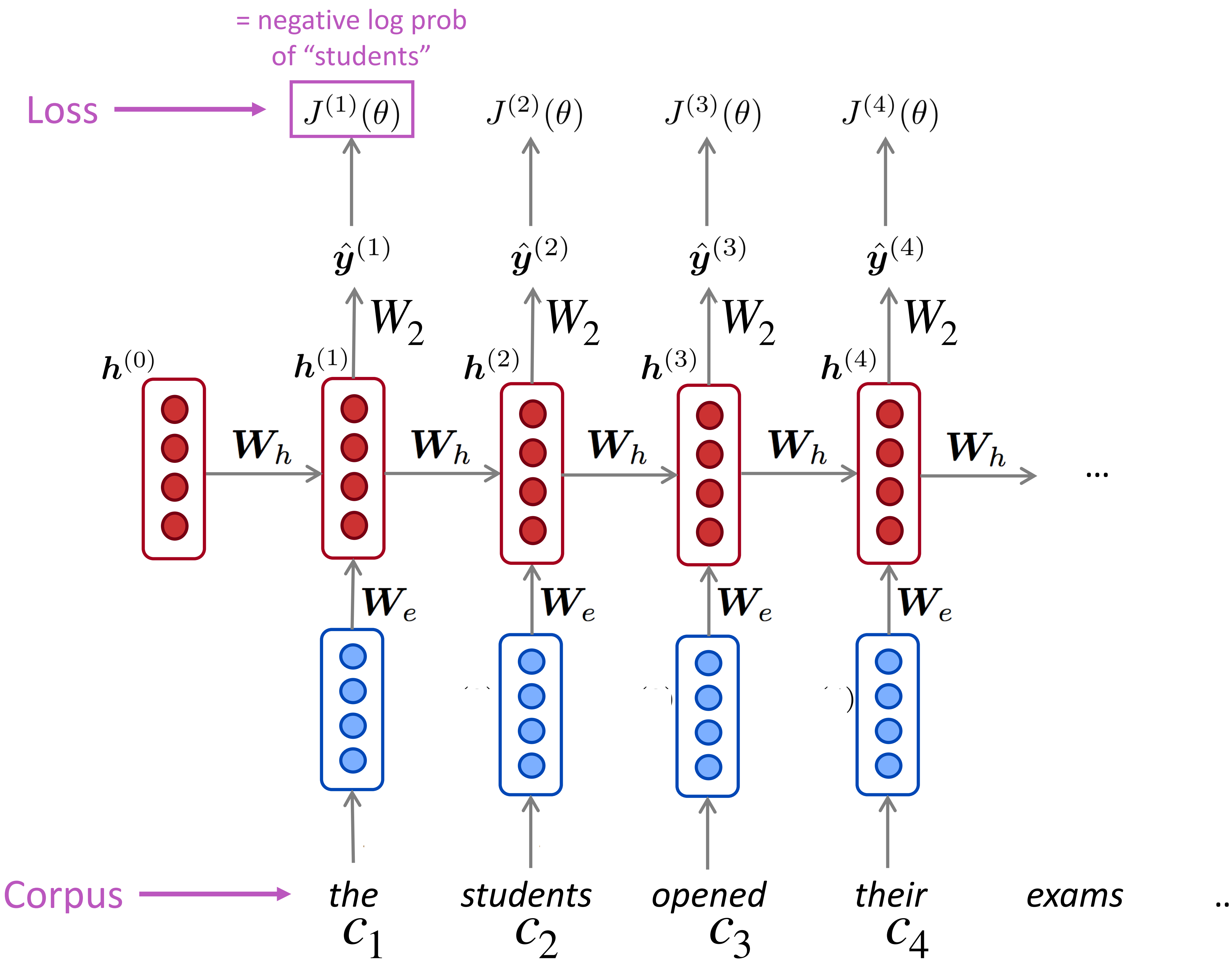
- Loss function on step *t* is usual cross-entropy between our predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)} = \boldsymbol{x}^{(t+1)}$:

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$

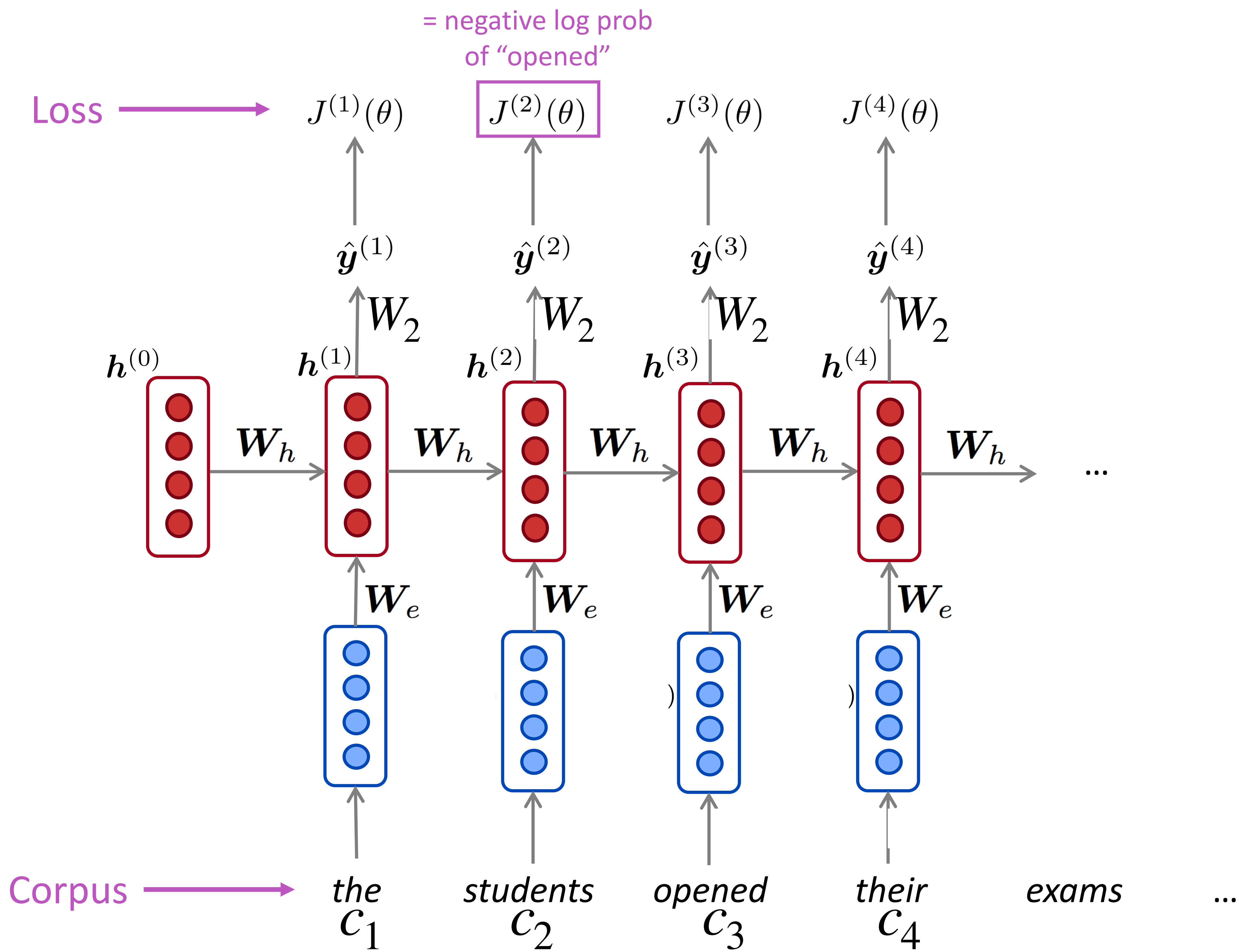
- Average this to get overall loss for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta)$$

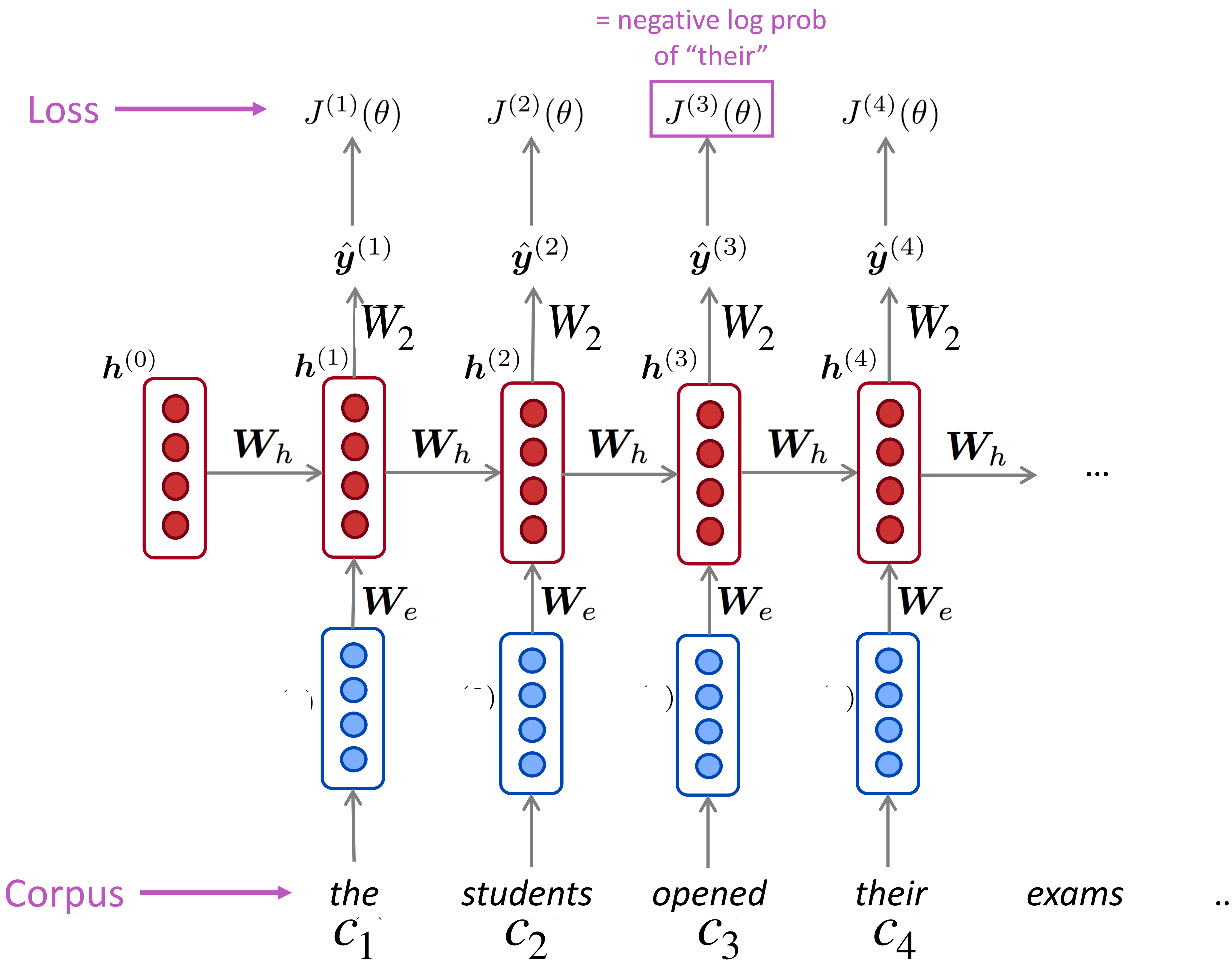
Training a RNN Language Model



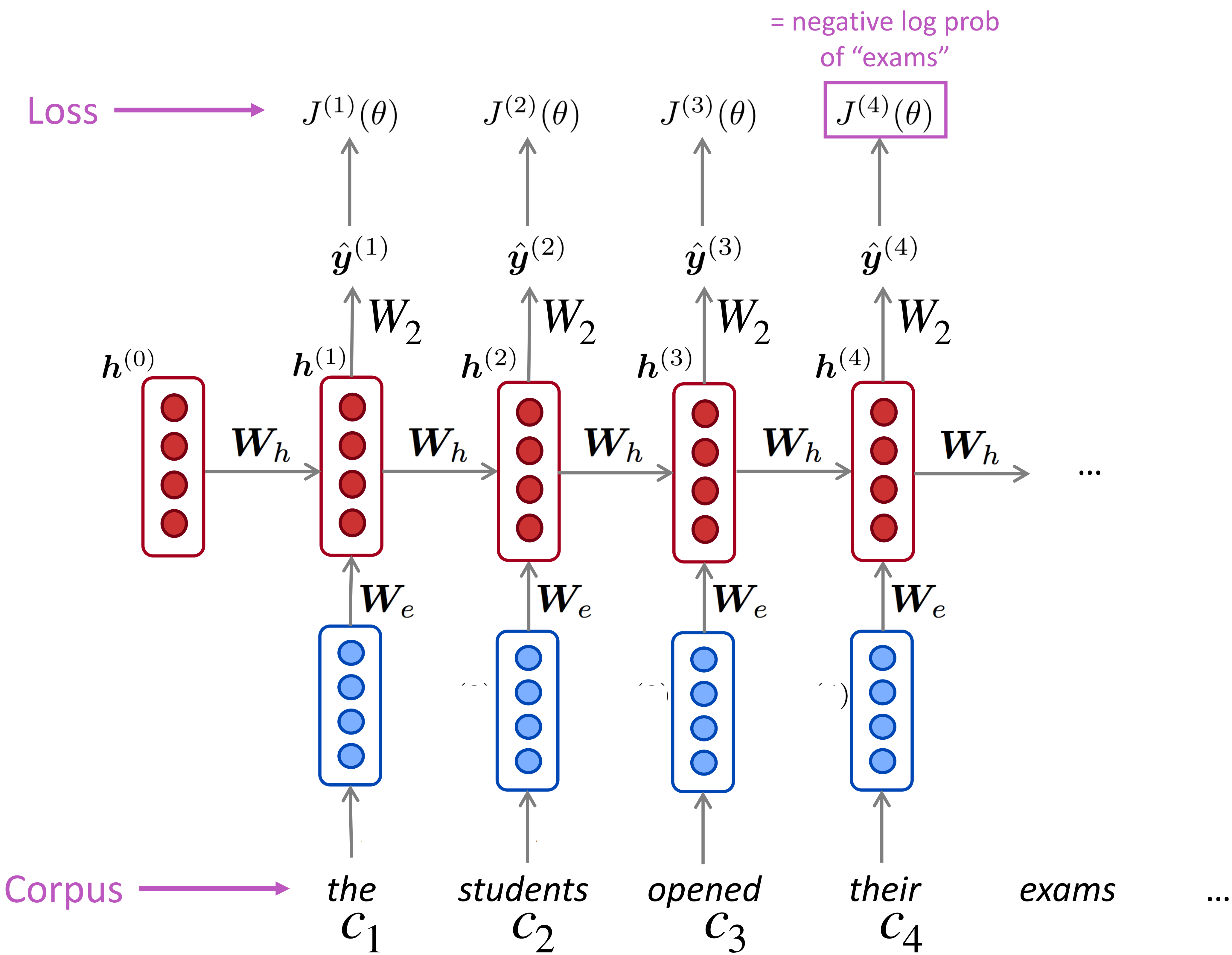
Training a RNN Language Model



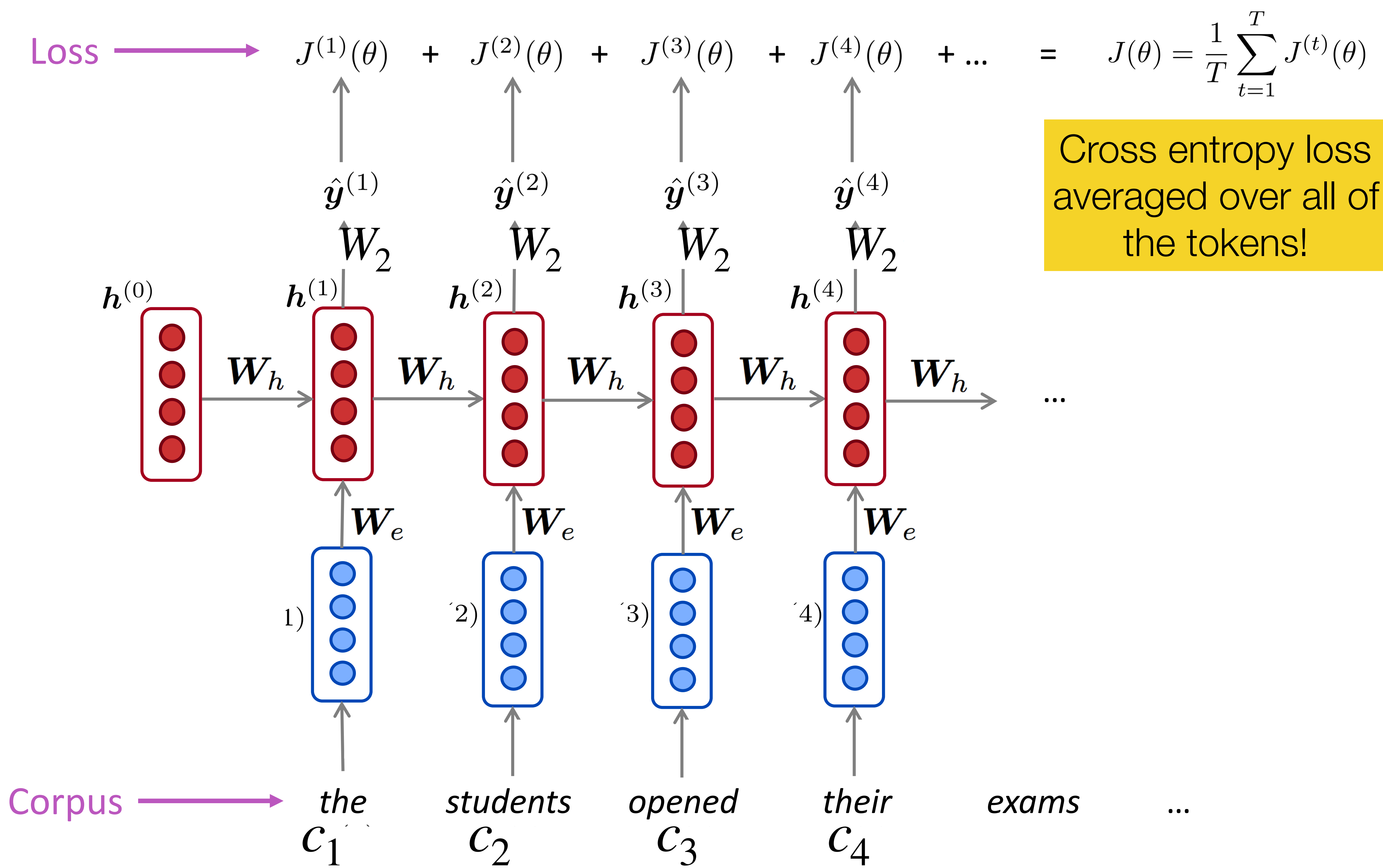
Training a RNN Language Model



Training a RNN Language Model

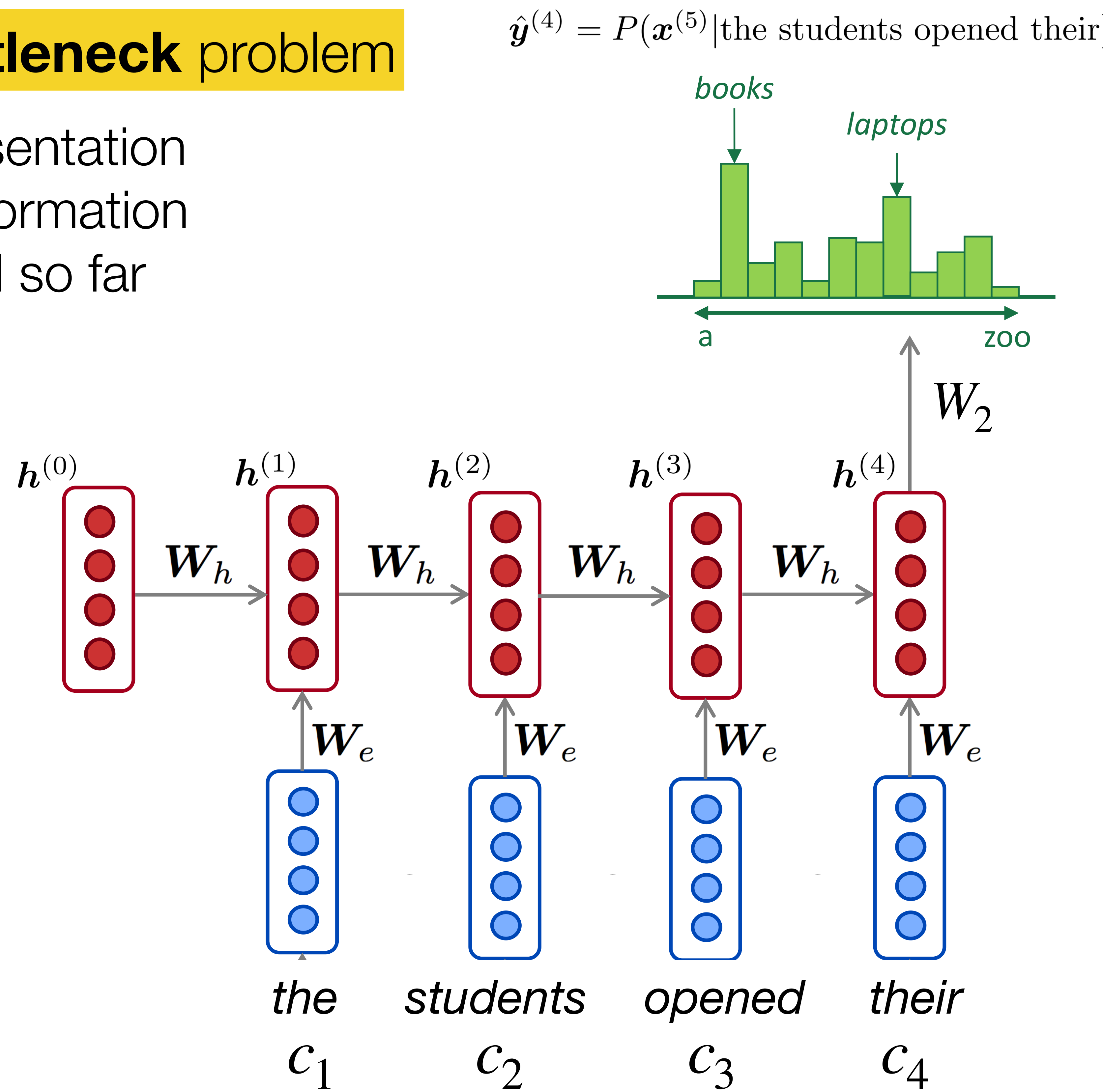


Training a RNN Language Model



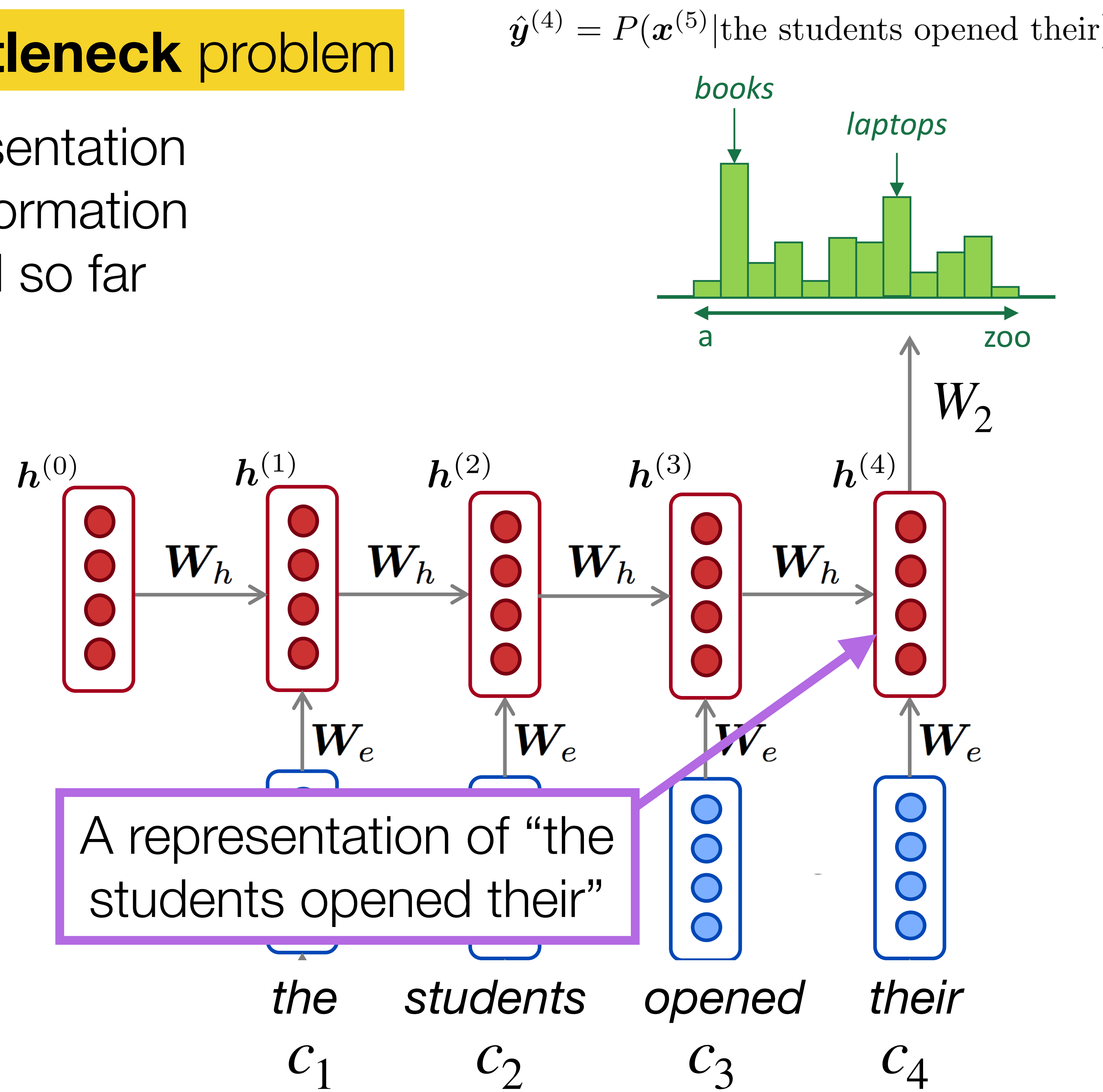
RNNs suffer from a **bottleneck** problem

The current hidden representation must encode all of the information about the text observed so far



RNNs suffer from a **bottleneck** problem

The current hidden representation must encode all of the information about the text observed so far

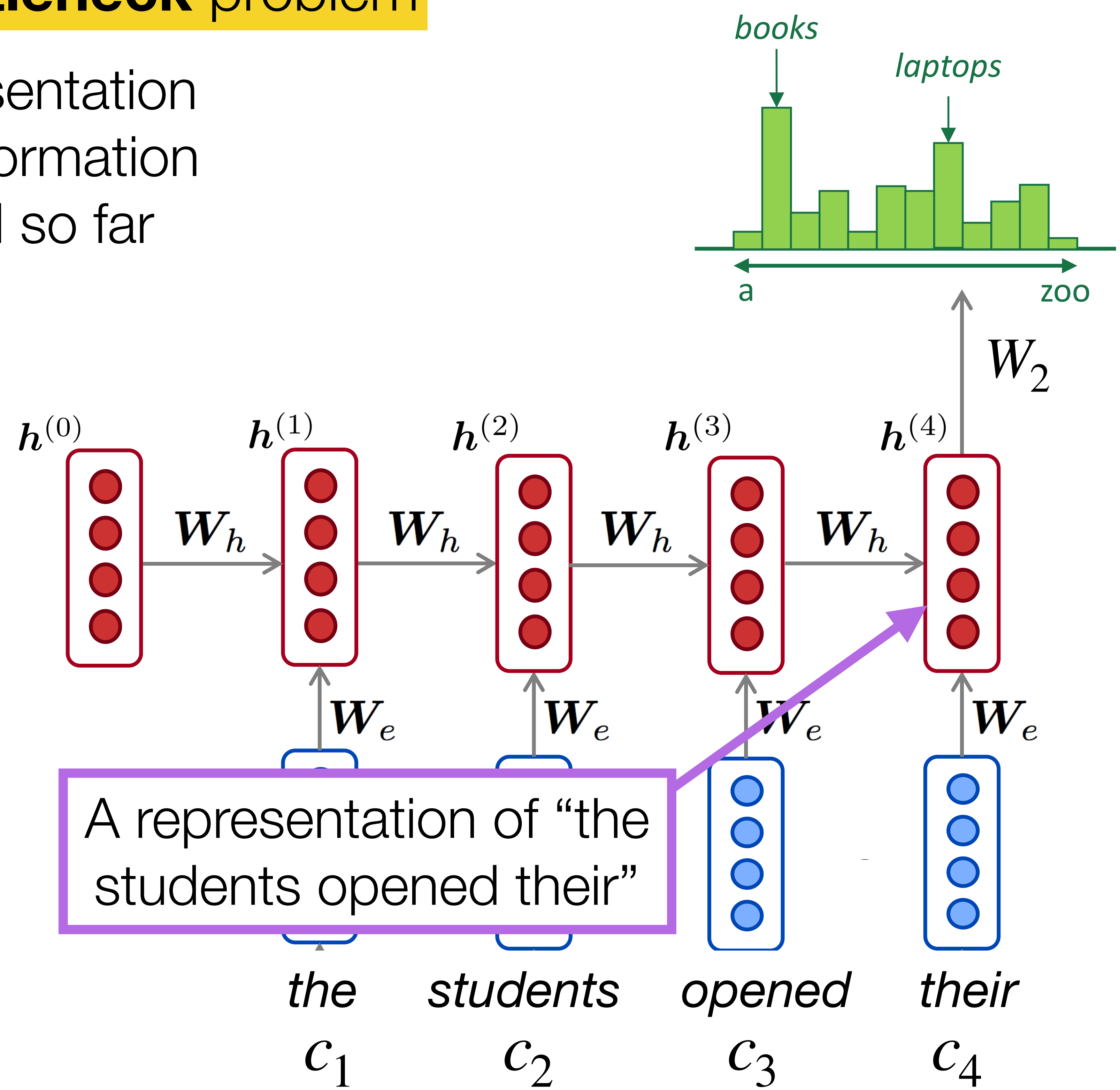


RNNs suffer from a **bottleneck** problem

The current hidden representation must encode all of the information about the text observed so far

This becomes difficult especially with longer sequences

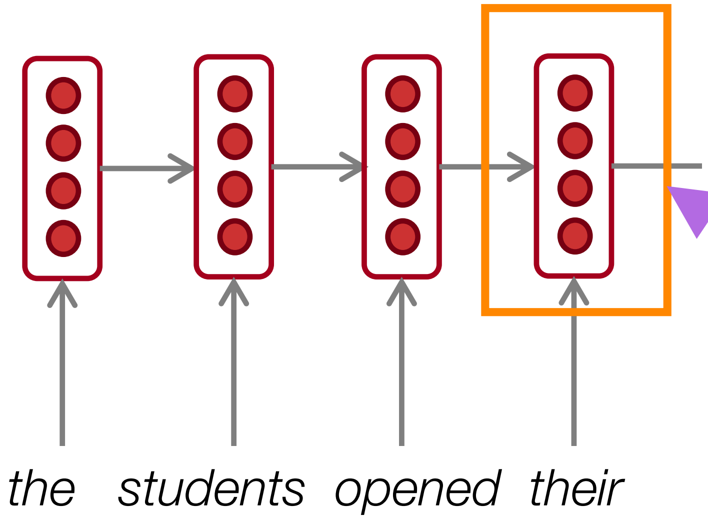
$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$



“you can’t cram the meaning of a whole %&@#&ing sentence into a single $*(&@ing vector!”

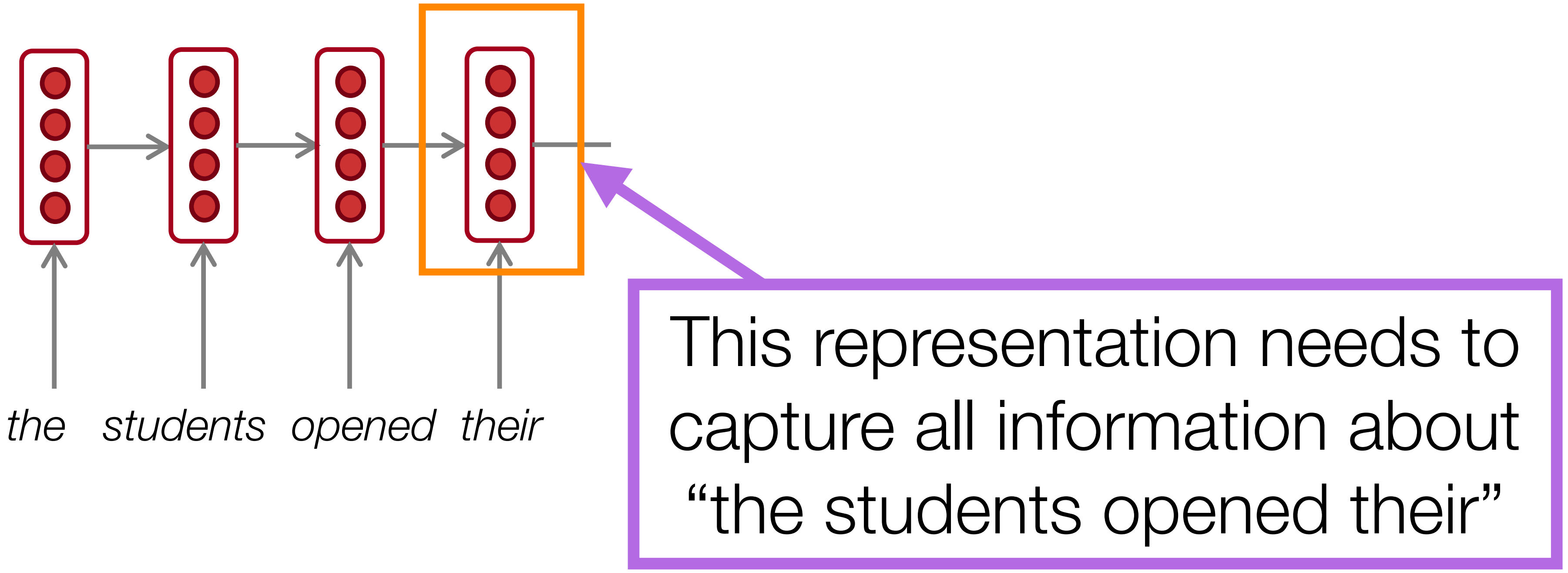
— Ray Mooney (NLP professor at UT Austin)

idea: what if we use multiple vectors?

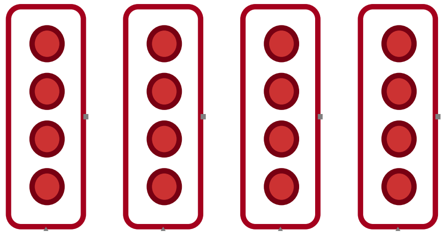


This representation needs to capture all information about "the students opened their"

idea: what if we use multiple vectors?



Instead of this, let's try:

the students opened their =  (all 4 hidden states!)

The solution: **attention**

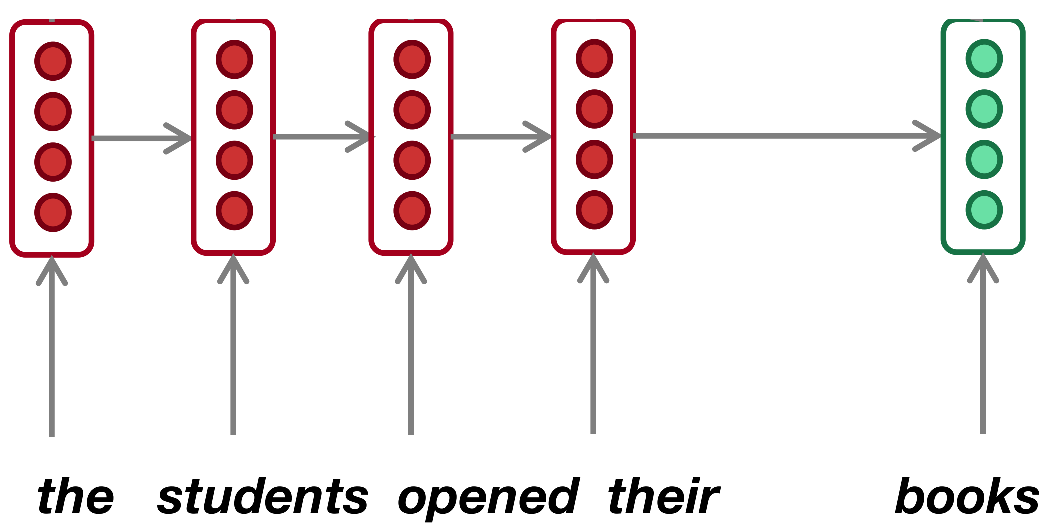
- **Attention mechanisms** (Bahdanau et al., 2015) allow language models to focus on a particular part of the observed context at each time step
    - Originally developed for machine translation, and intuitively similar to *word alignments* between different languages

How does it work?

- in general, we have a single *query* vector and multiple *key* vectors. We want to score each query-key pair

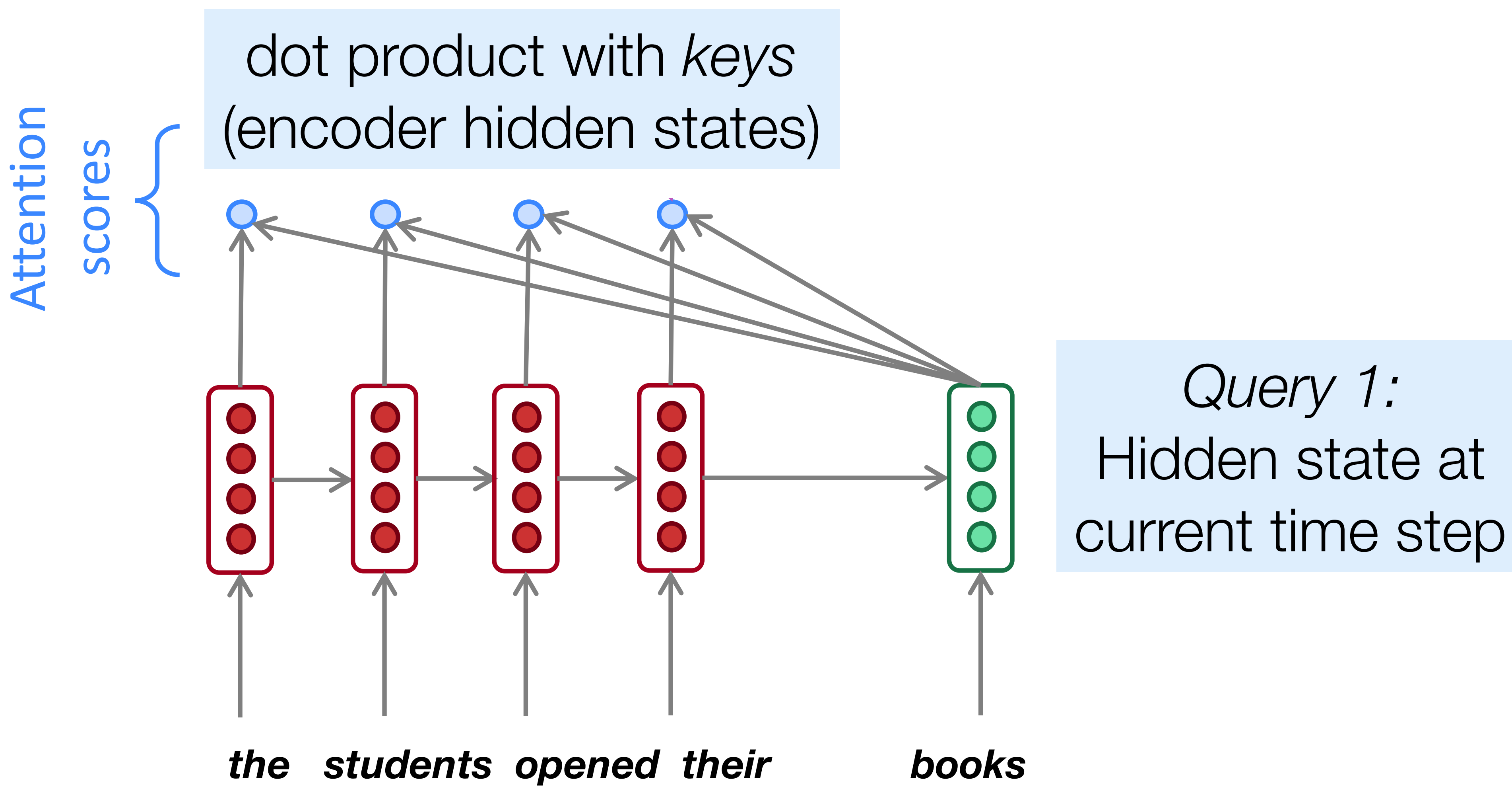
in a neural language model, what are the queries and keys?

Attention mechanisms in neural language models

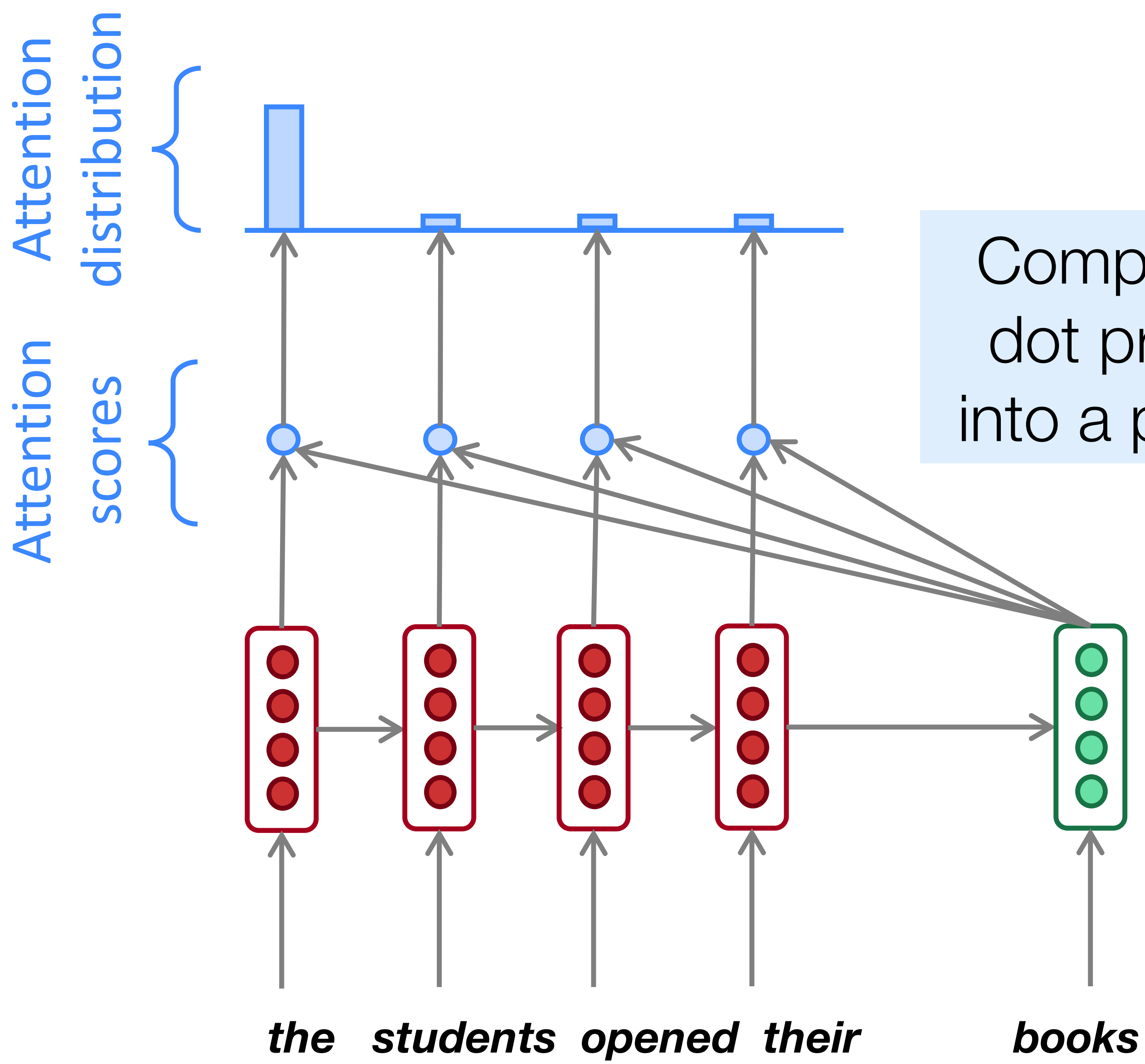


Query 1:
Hidden state at current time step

Attention mechanisms in neural language models



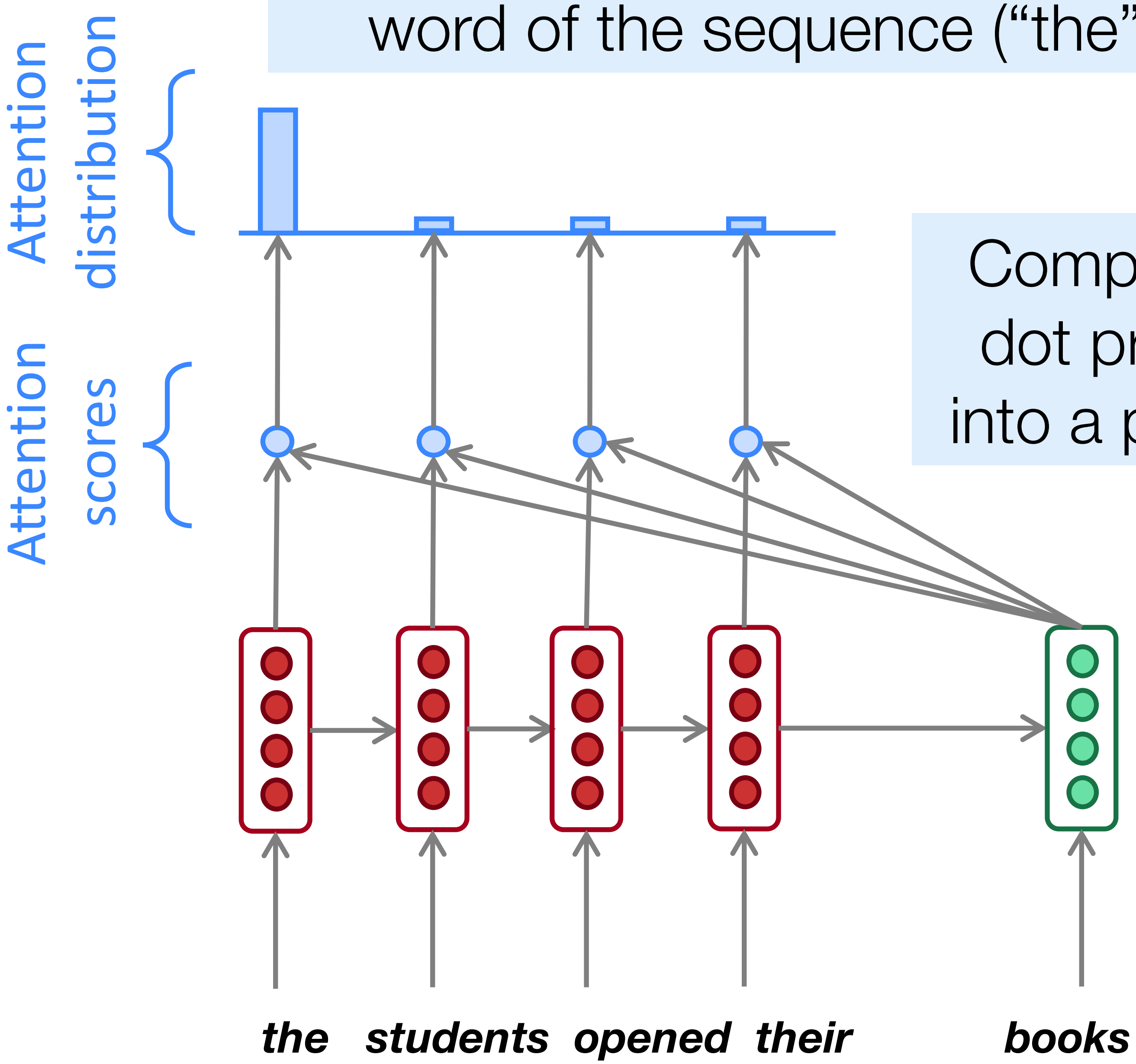
Attention mechanisms in neural language models



Compute softmax over the dot products to turn them into a probability distribution

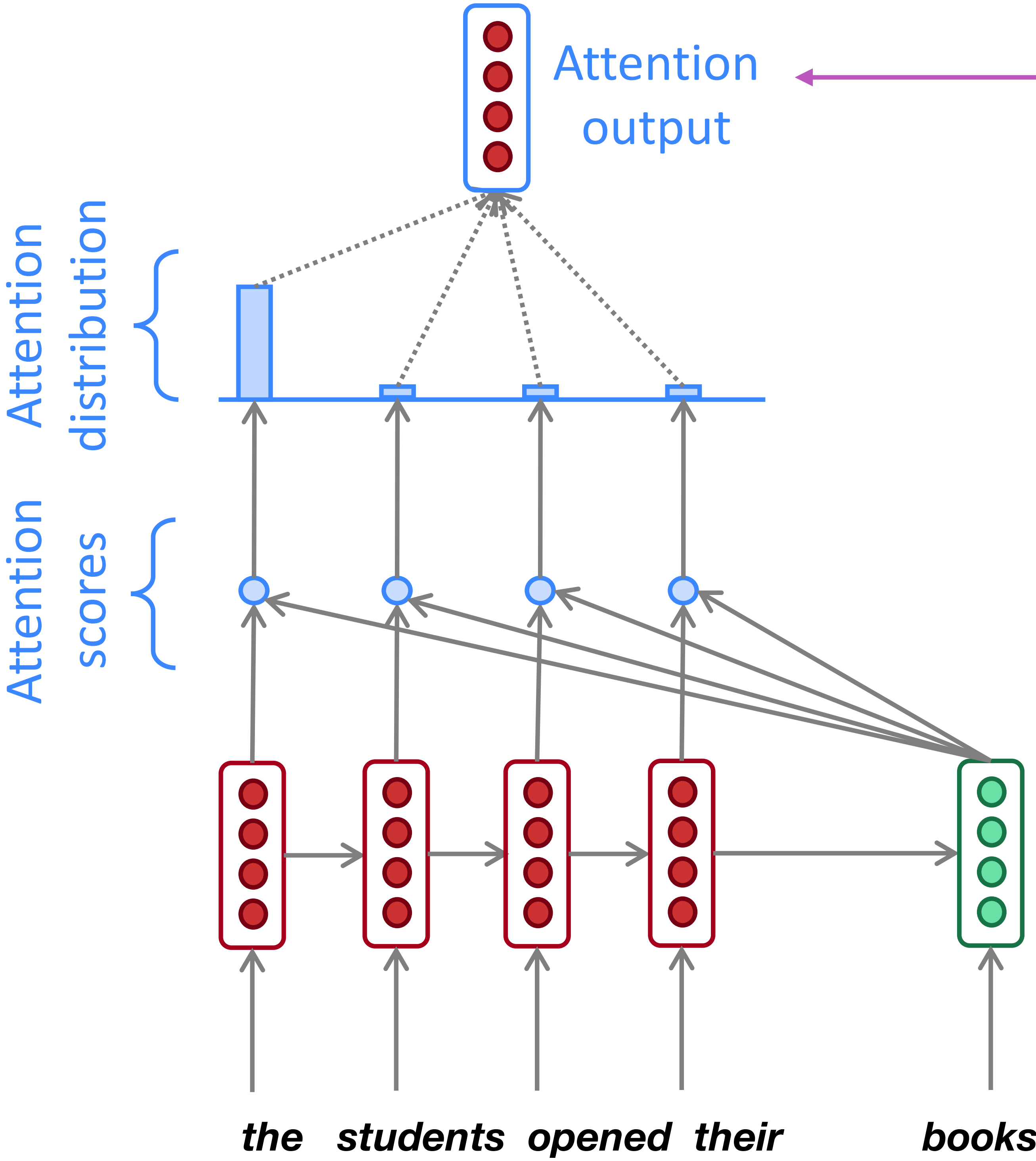
Attention mechanisms in neural language models

At this time step, the attention distribution is focused on the first word of the sequence ("the")



Compute softmax over the dot products to turn them into a probability distribution

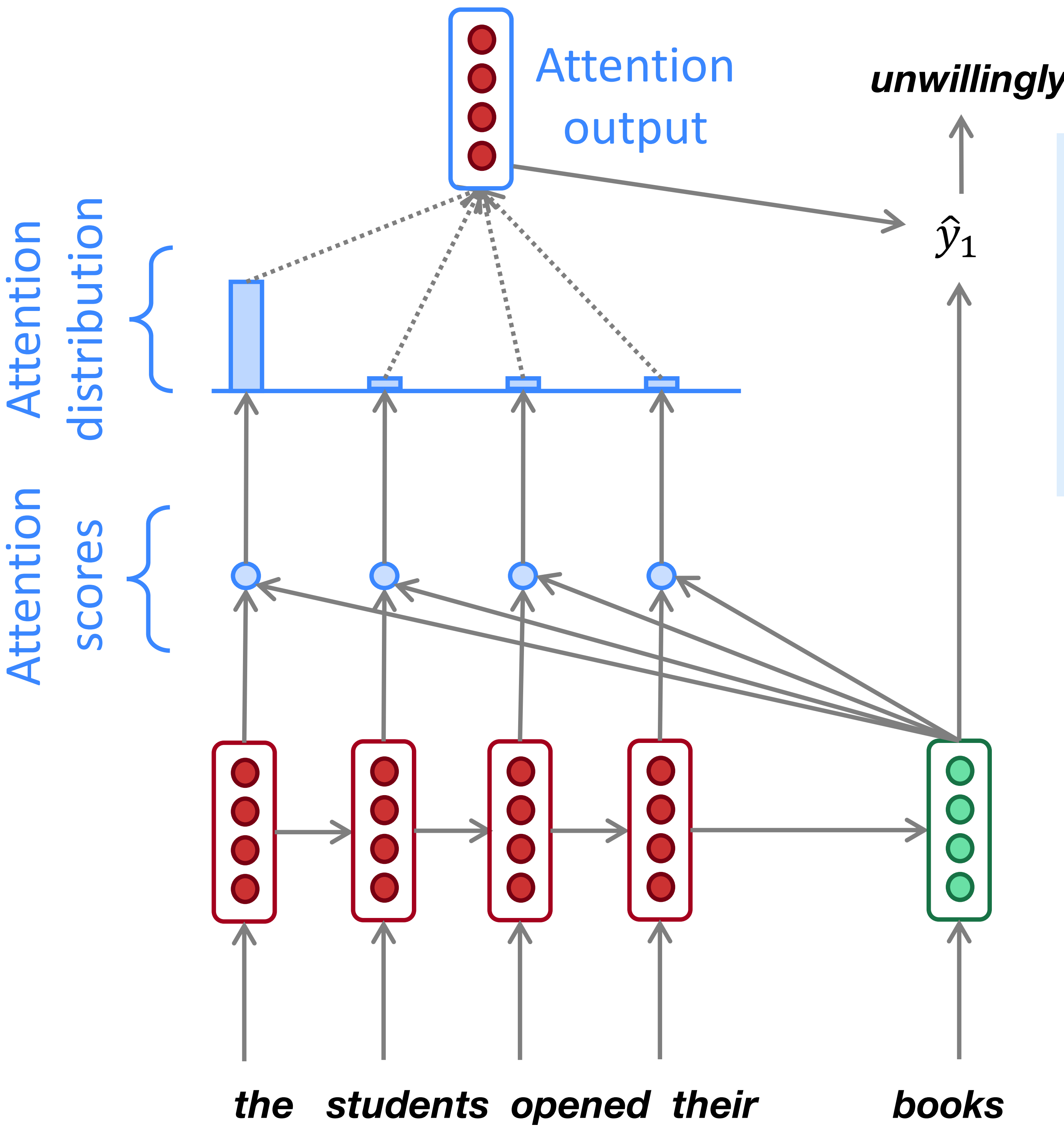
Attention mechanisms in neural language models



We use the attention distribution to compute a weighted average of the hidden states.

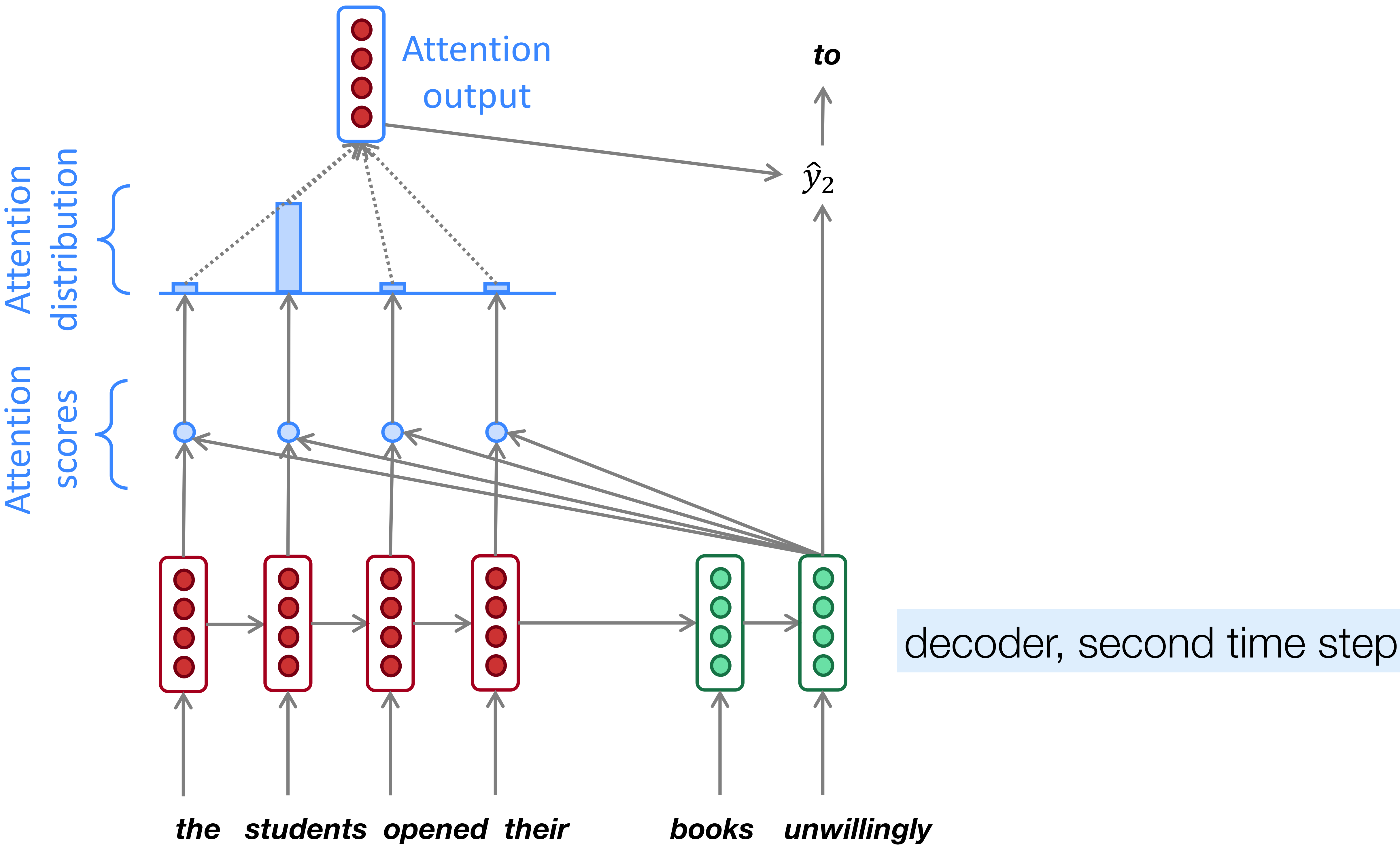
Intuitively, the resulting attention output contains information from hidden states that received high attention scores

Sequence-to-sequence with attention

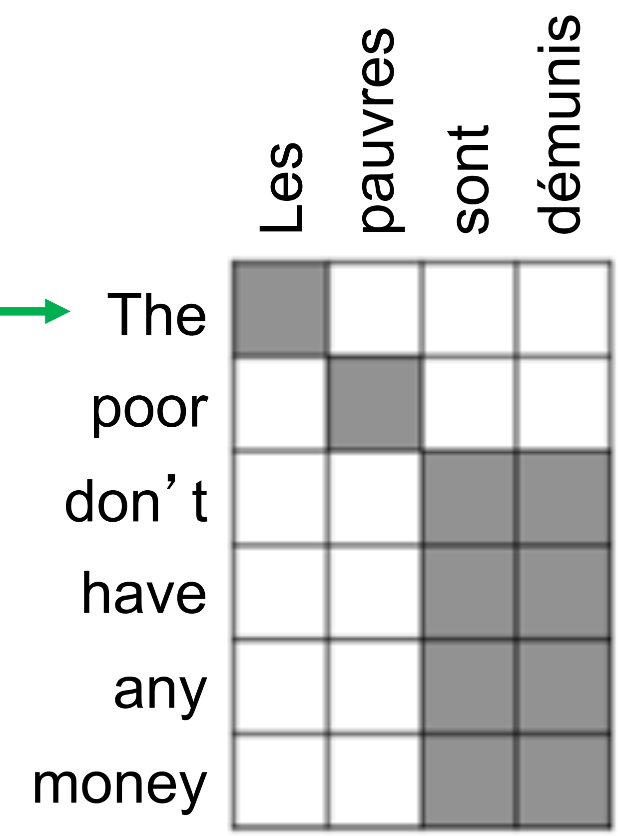


Concatenate (or otherwise compose) the attention output with the current hidden state, then pass through a softmax layer to predict the next word

Sequence-to-sequence with attention



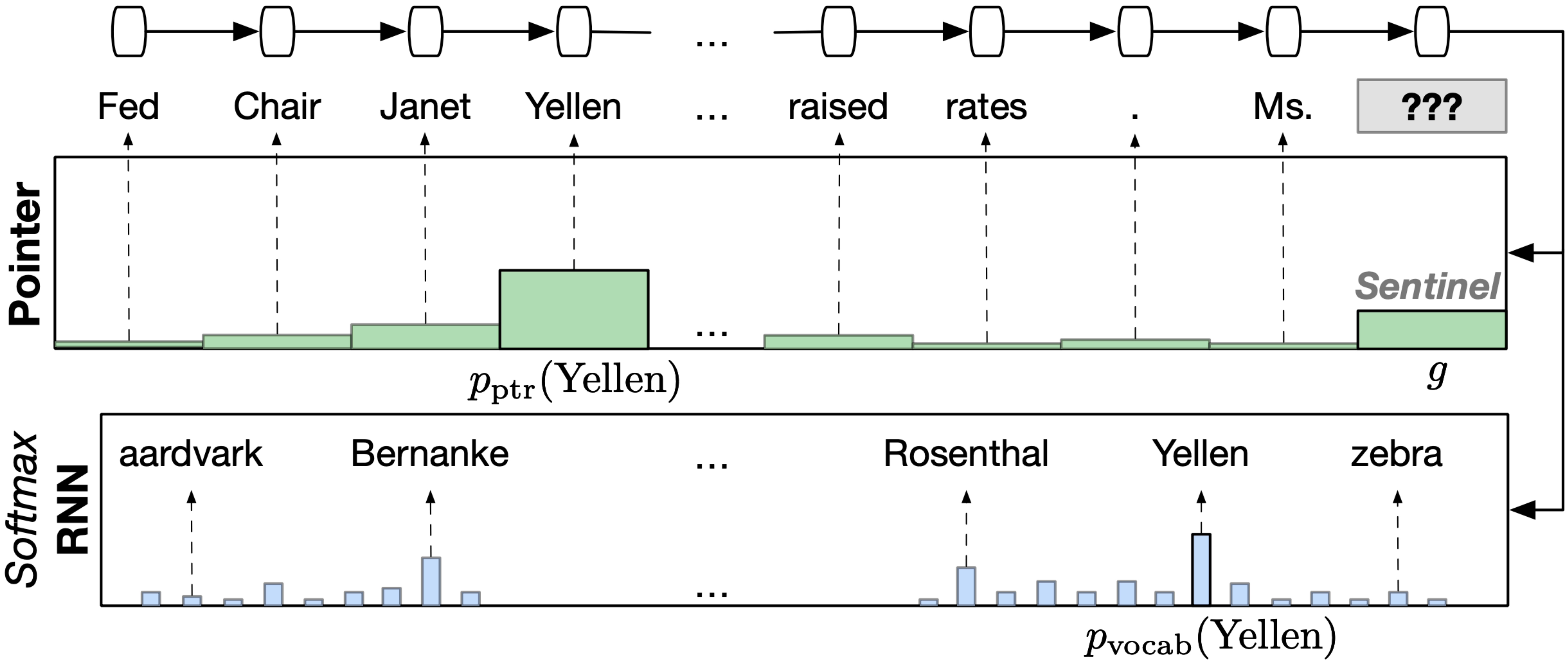
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Many variants of attention

- Original formulation: $a(\mathbf{q}, \mathbf{k}) = w_2^T \tanh(W_1[\mathbf{q}; \mathbf{k}])$
- Bilinear product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T W \mathbf{k}$ Luong et al., 2015
- Dot product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k}$ Luong et al., 2015
- Scaled dot product: $a(\mathbf{q}, \mathbf{k}) = \dfrac{\mathbf{q}^T \mathbf{k}}{\sqrt{|\mathbf{k}|}}$ Vaswani et al., 2017

Attention can also be used to copy tokens from the context!

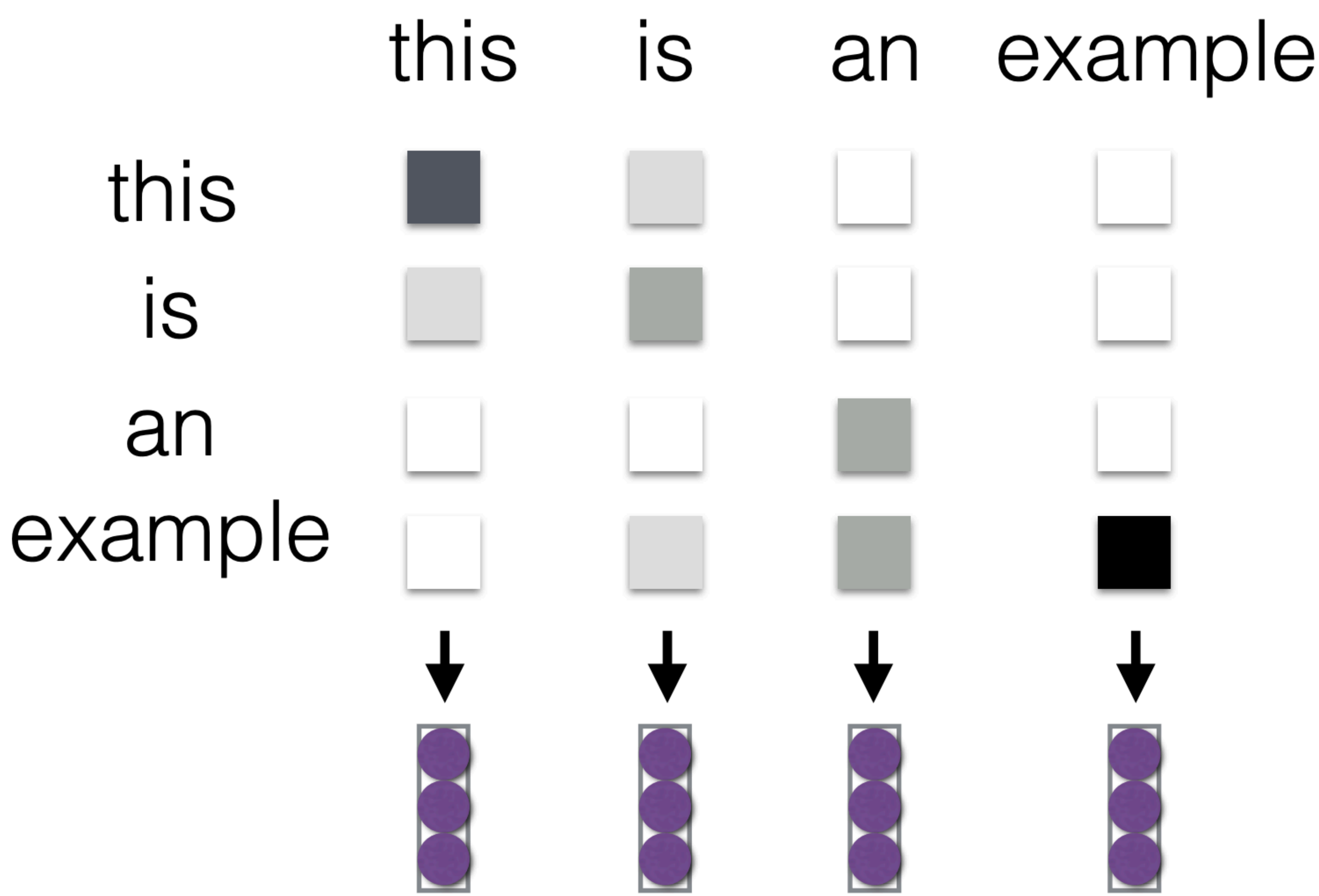


$$p(\mathrm{Yellen}) = g\, p_{\mathrm{vocab}}(\mathrm{Yellen}) + (1-g)\, p_{\mathrm{ptr}}(\mathrm{Yellen})$$

Merity et al., 2016

Self-attention can completely replace recurrence!

Each element in the sentence attends to the other elements

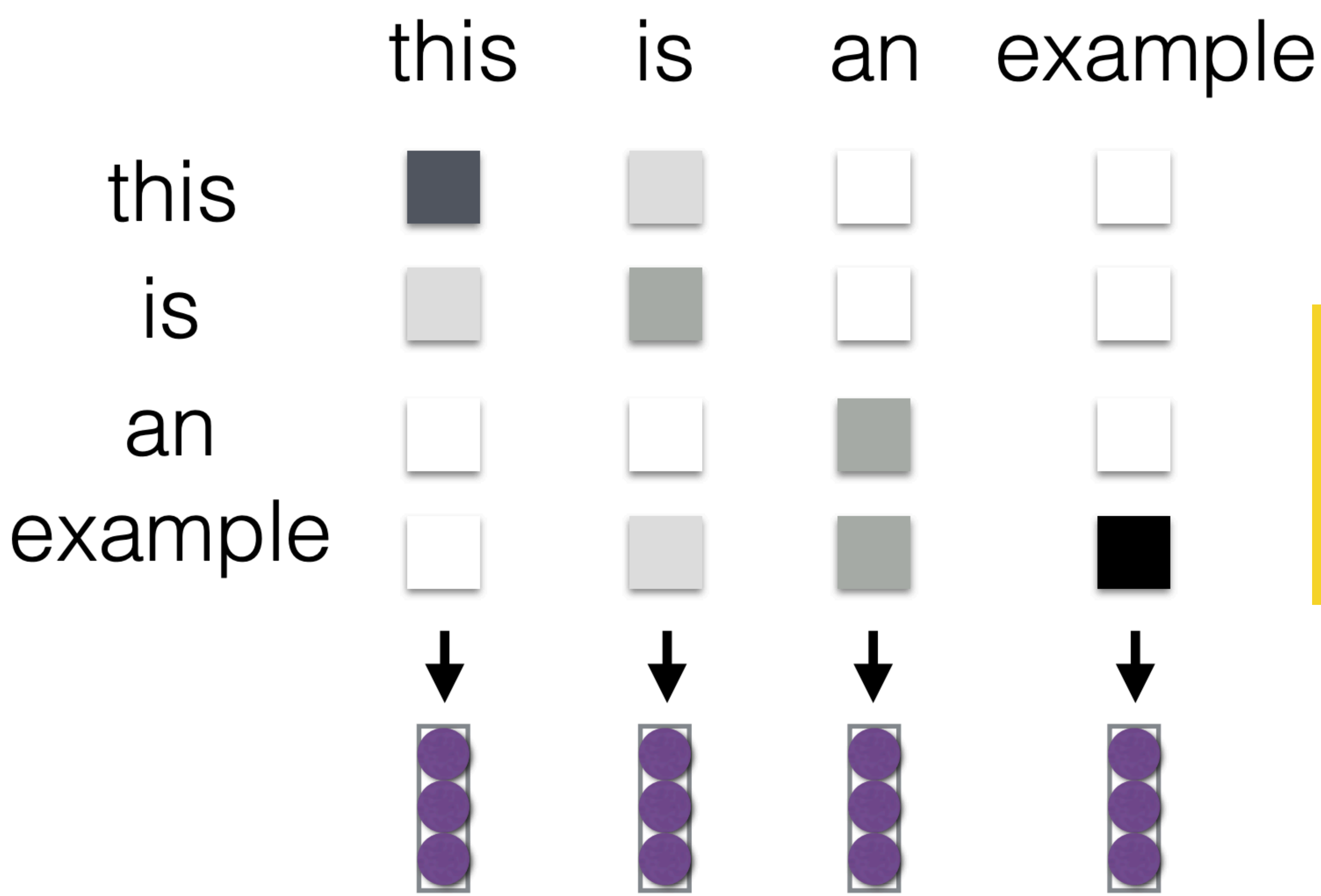


Cheng et al., 2016

Figure: Graham Neubig

Self-attention can completely replace recurrence!

Each element in the sentence attends to the other elements



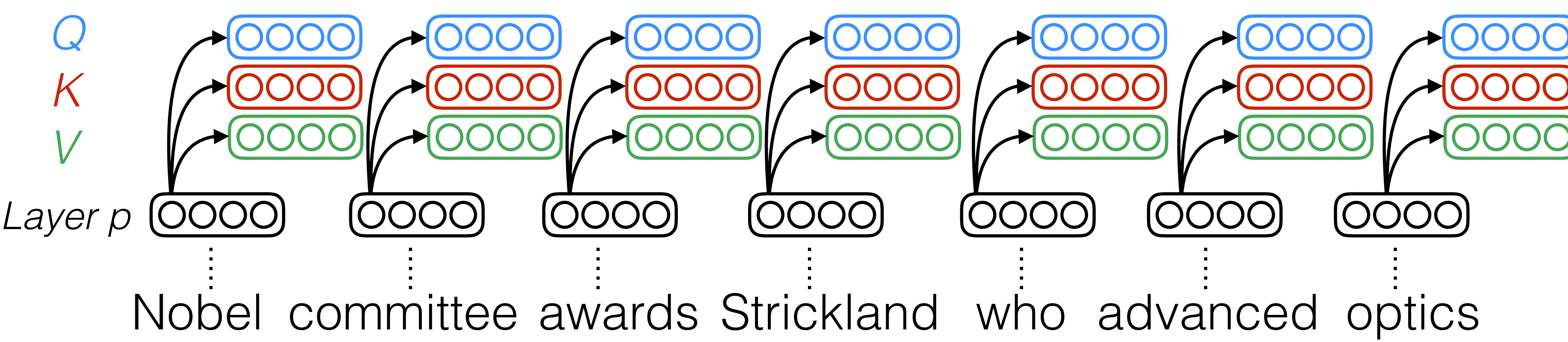
What are the queries and keys here?

Cheng et al., 2016

Figure: Graham Neubig

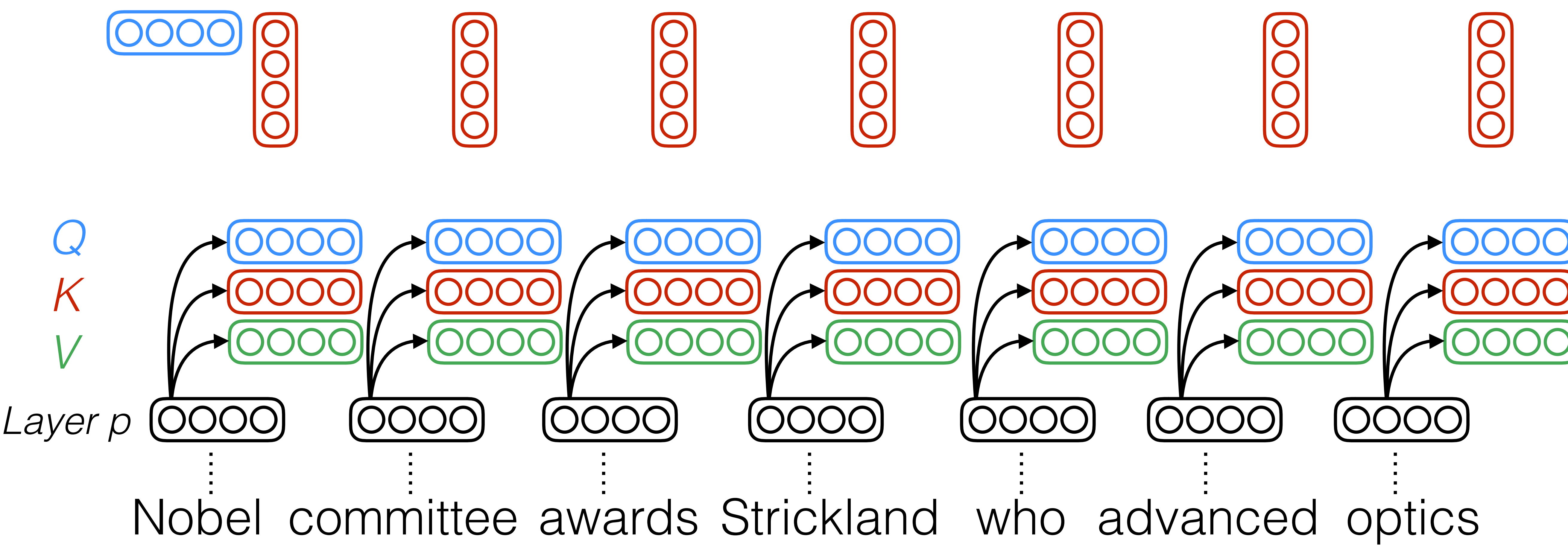
Self-attention

[Vaswani et al. 2017]

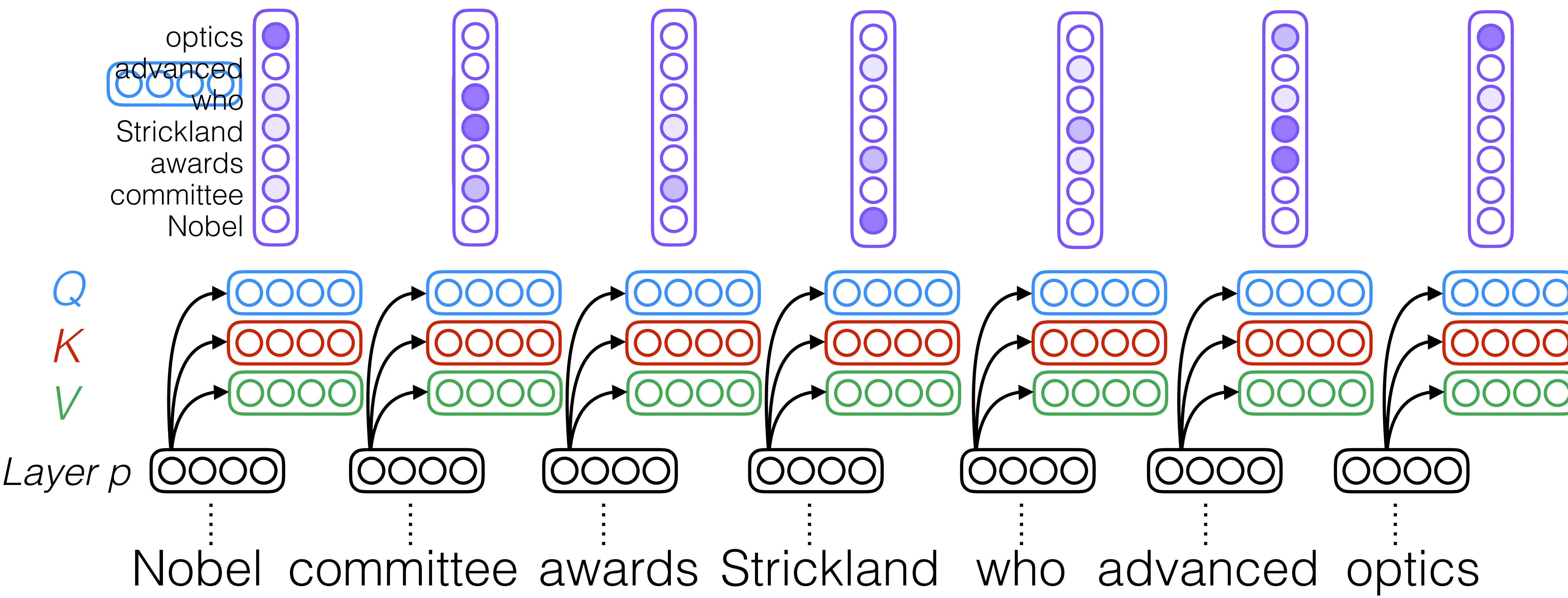


Self-attention

[Vaswani et al. 2017]

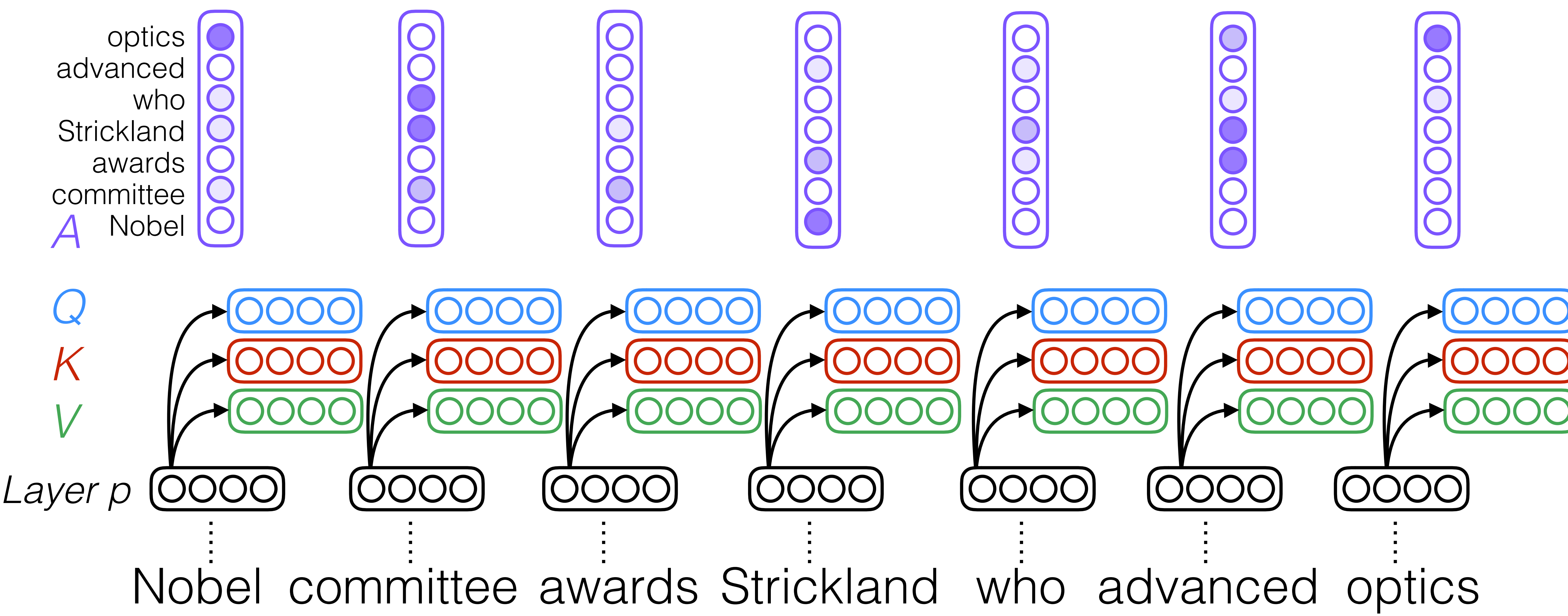


Self-attention



[Vaswani et al. 2017]

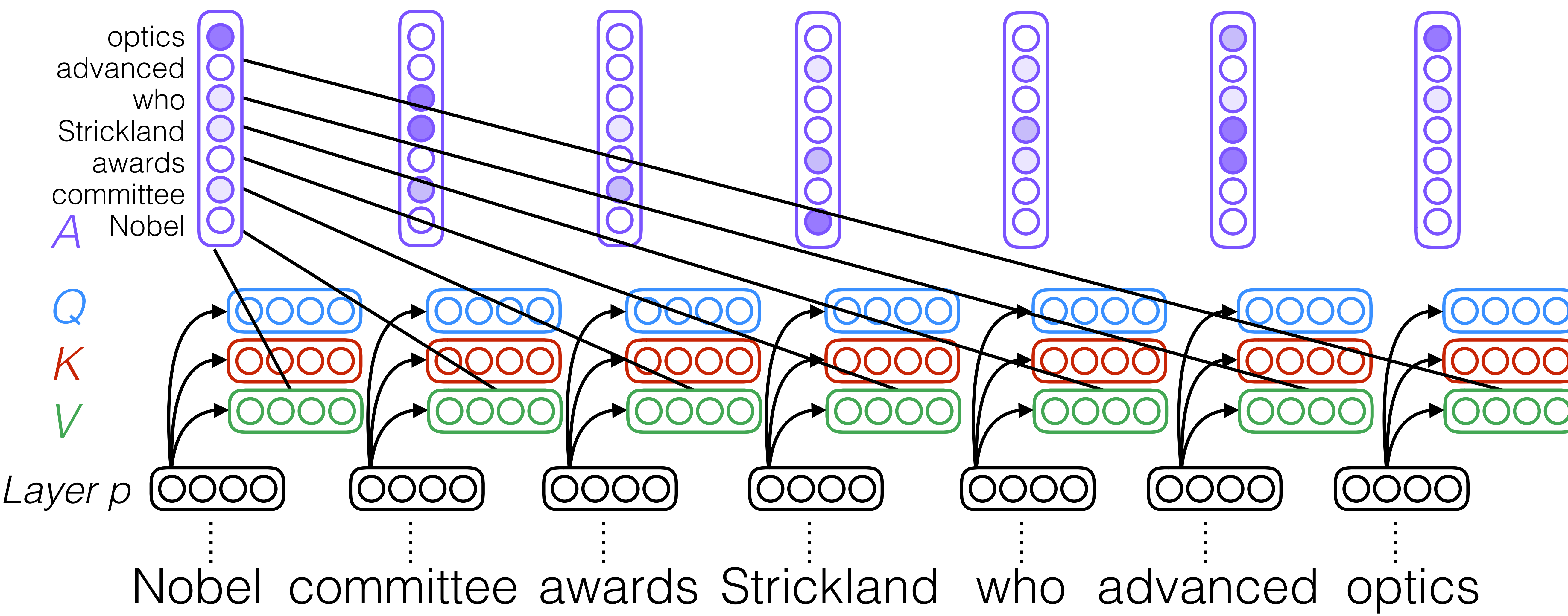
Self-attention



Slides by Emma Strubell!

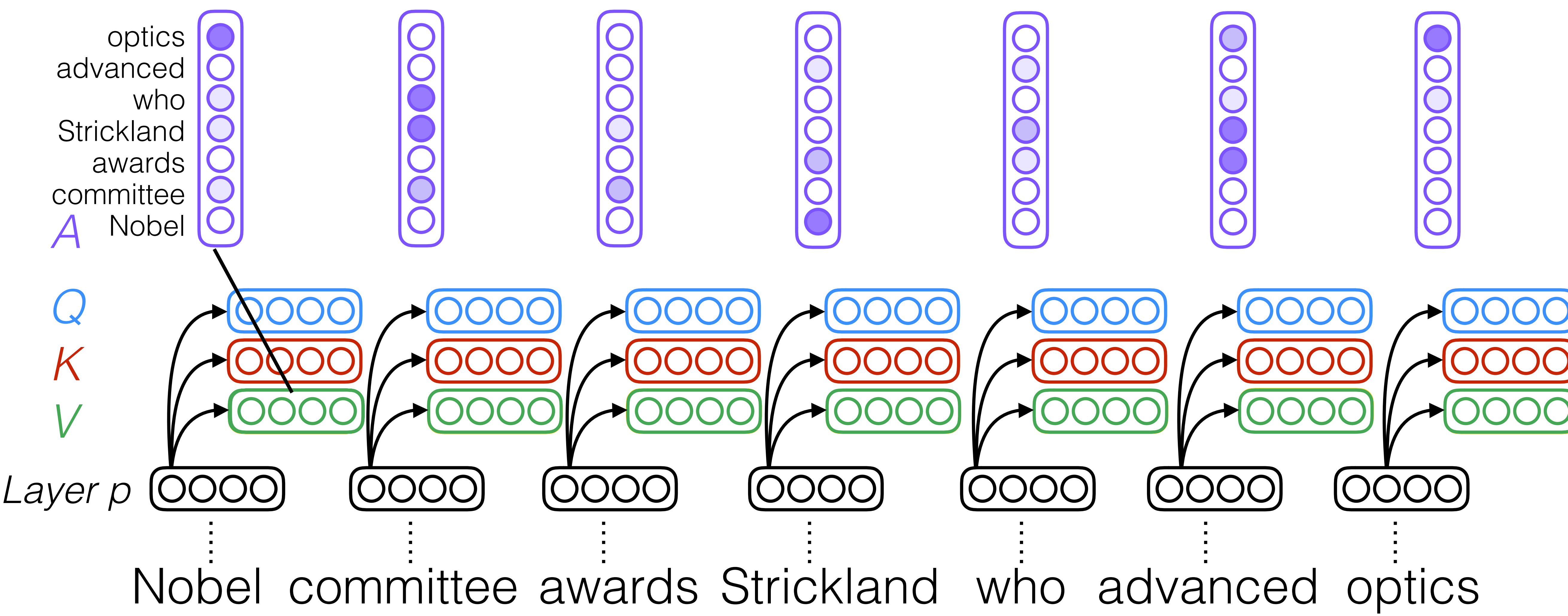
[Vaswani et al. 2017]

Self-attention



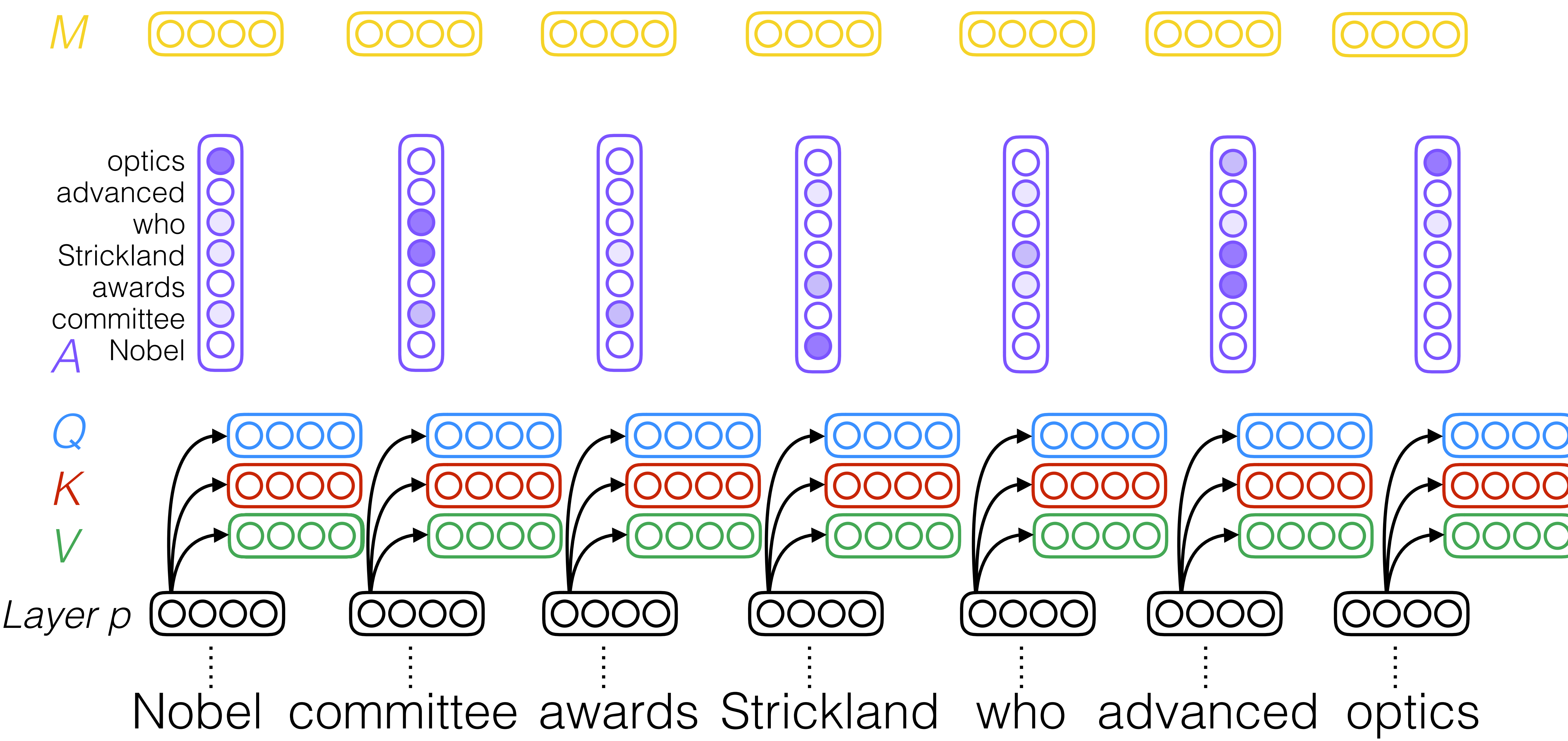
[Vaswani et al. 2017]

Self-attention



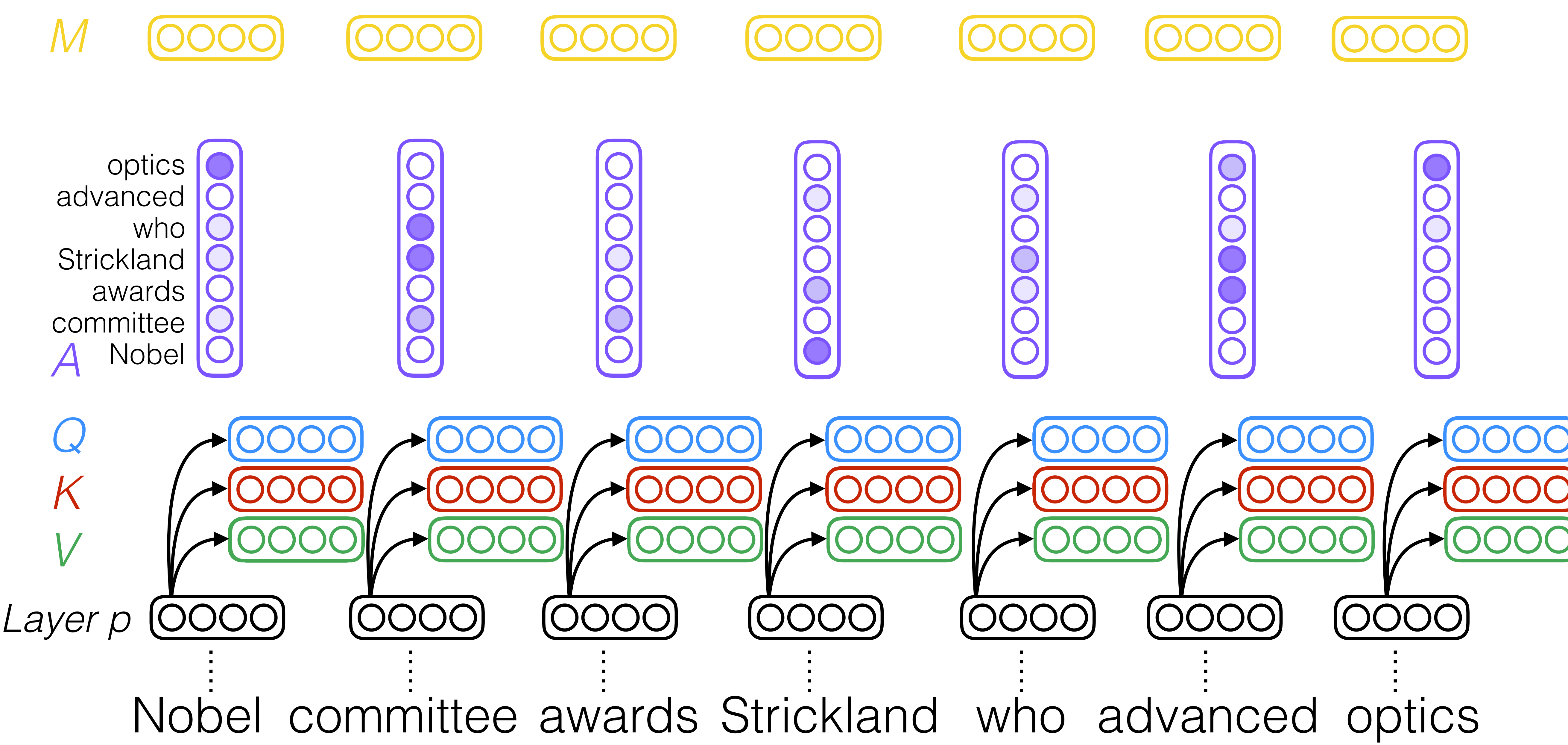
Self-attention

[Vaswani et al. 2017]

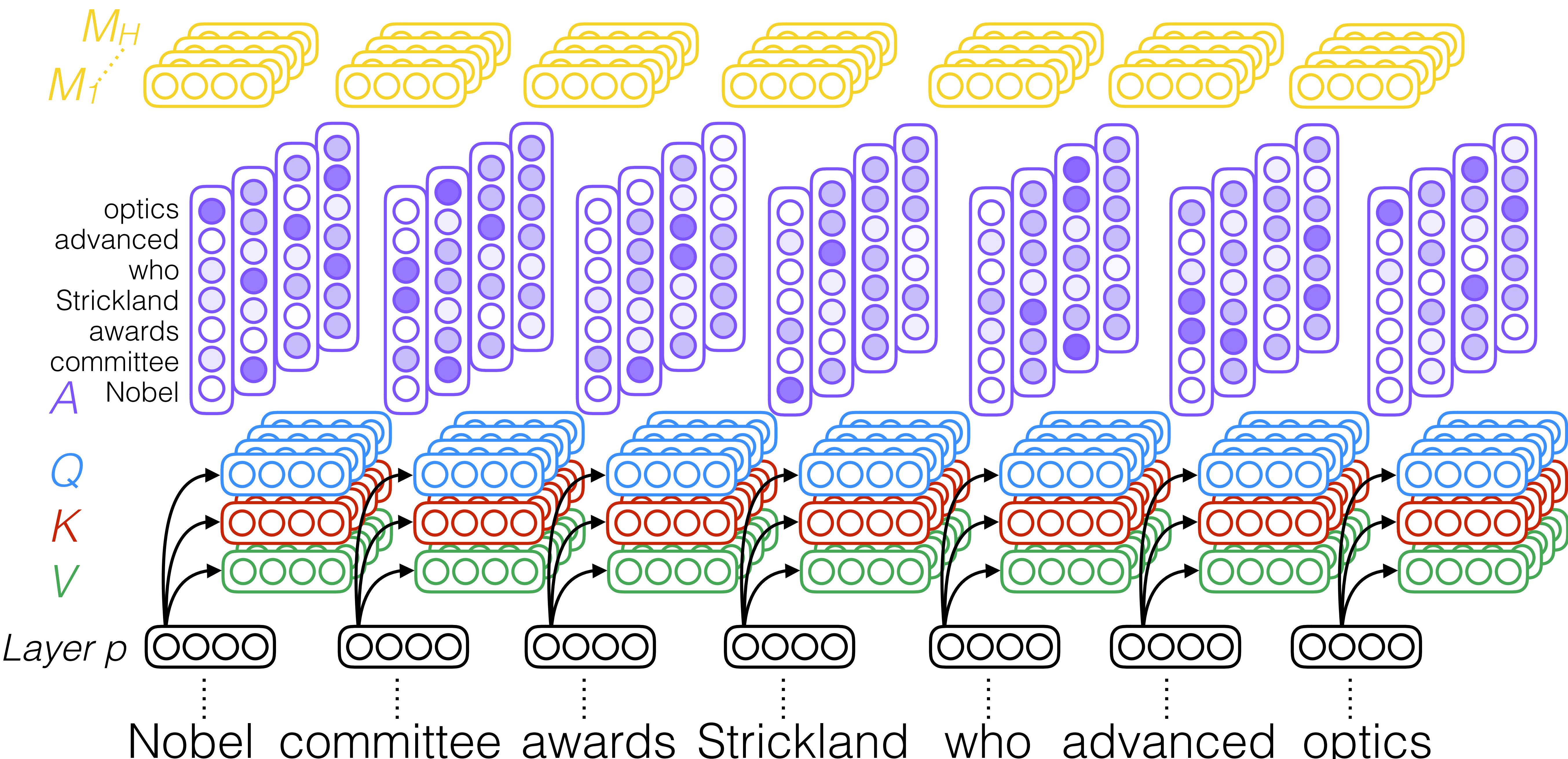


[Vaswani et al. 2017]

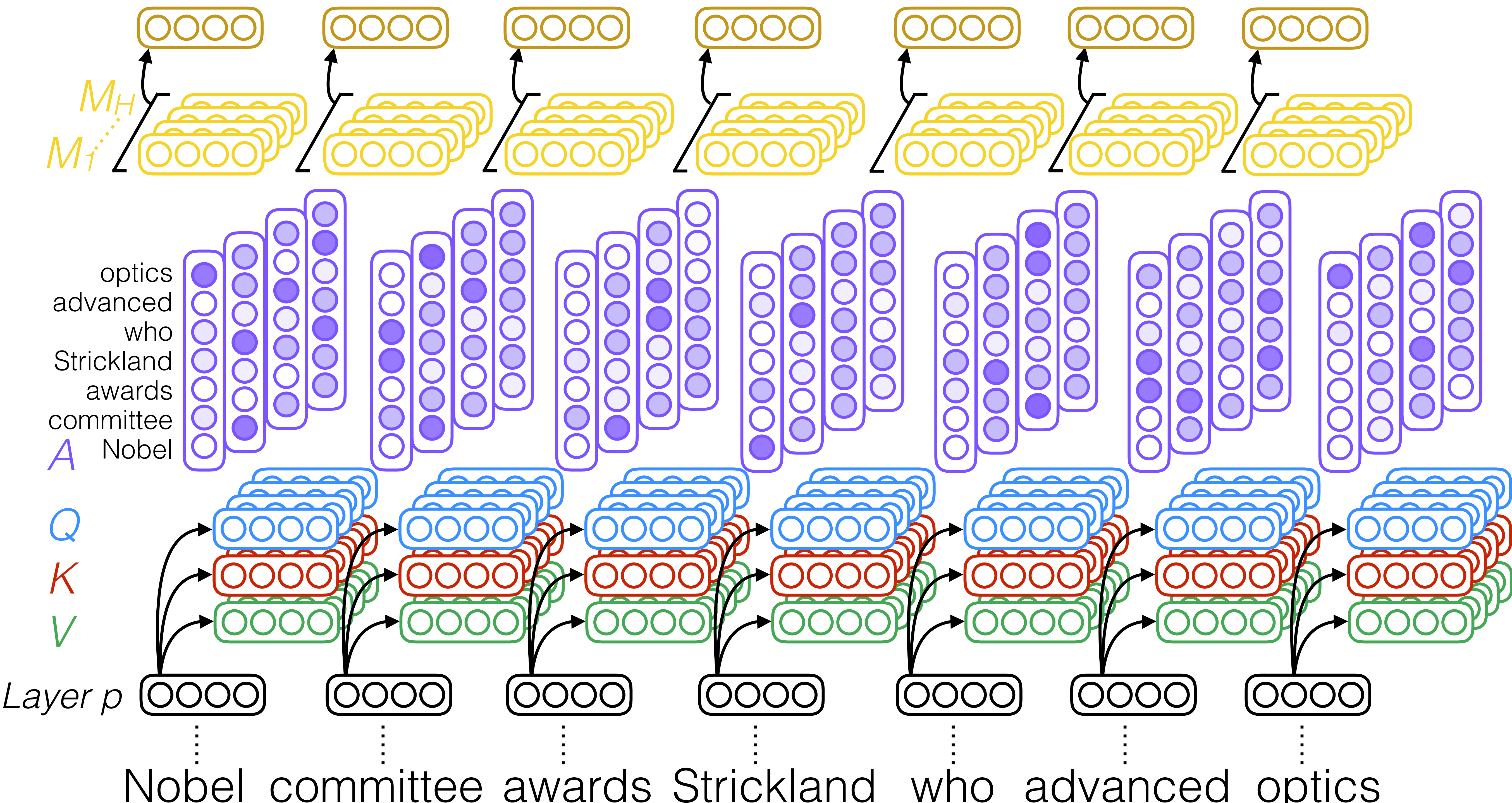
Self-attention



Multi-head self-attention

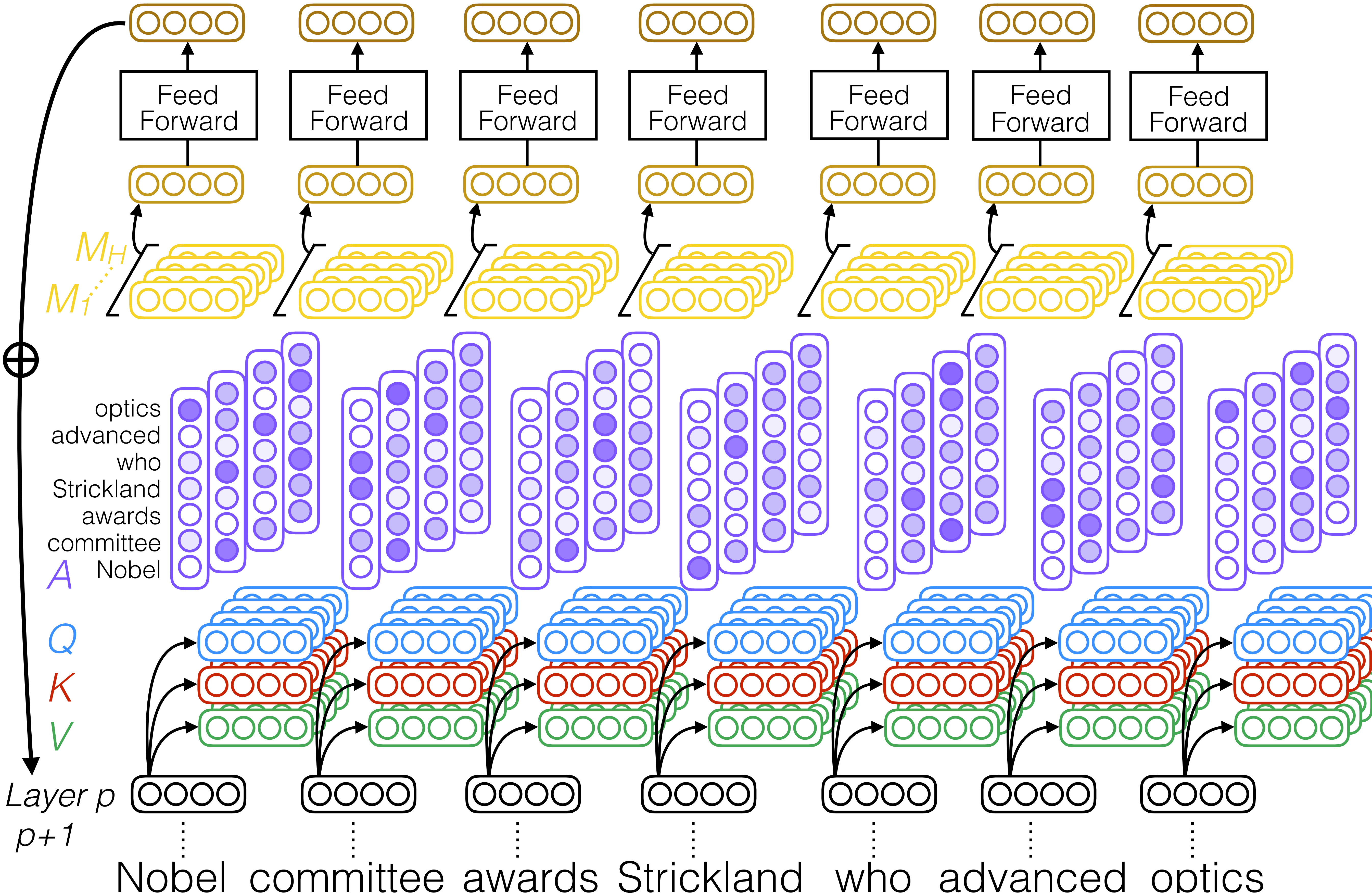


Multi-head self-attention



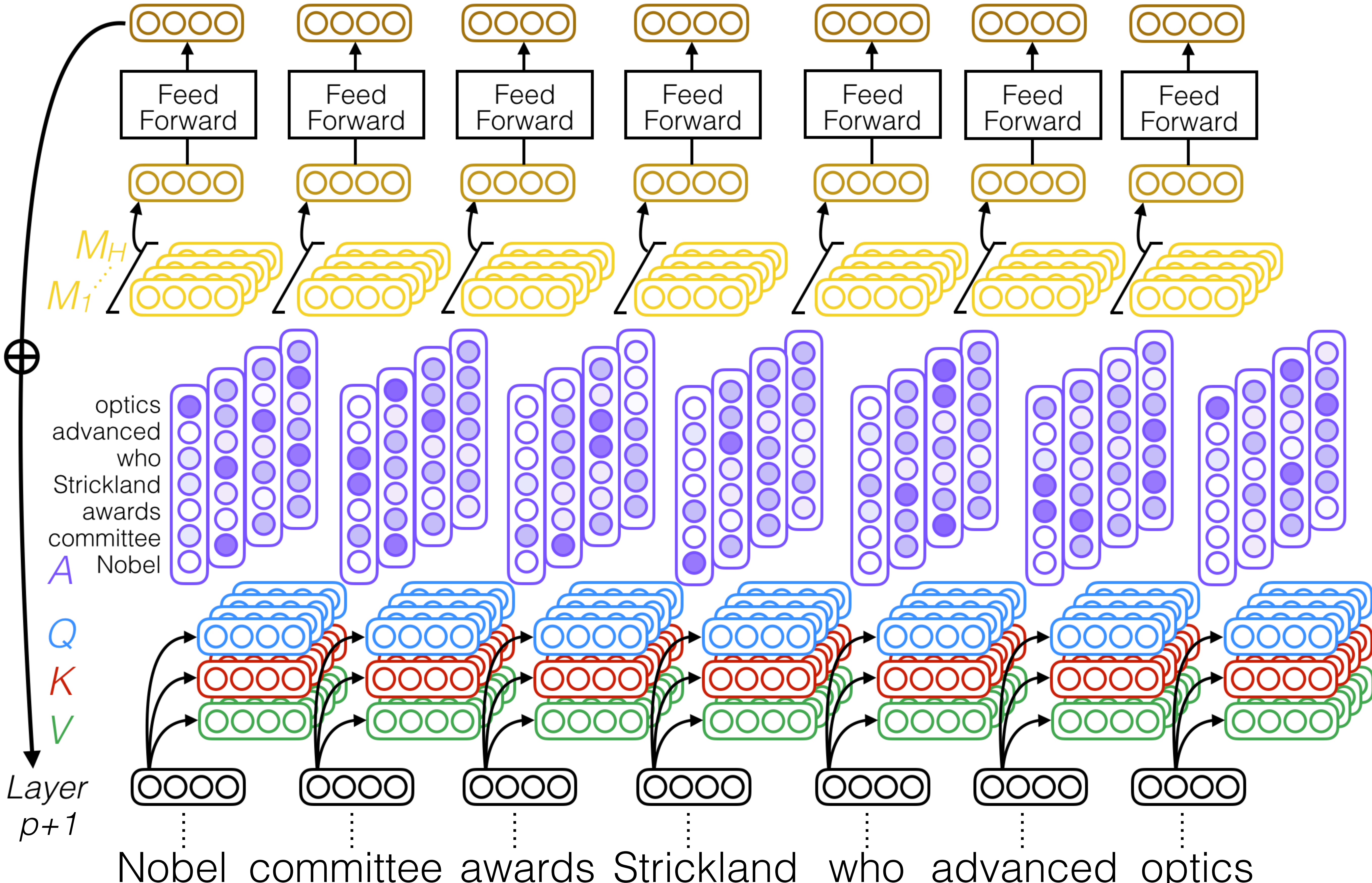
[Vaswani et al. 2017]

Multi-head self-attention



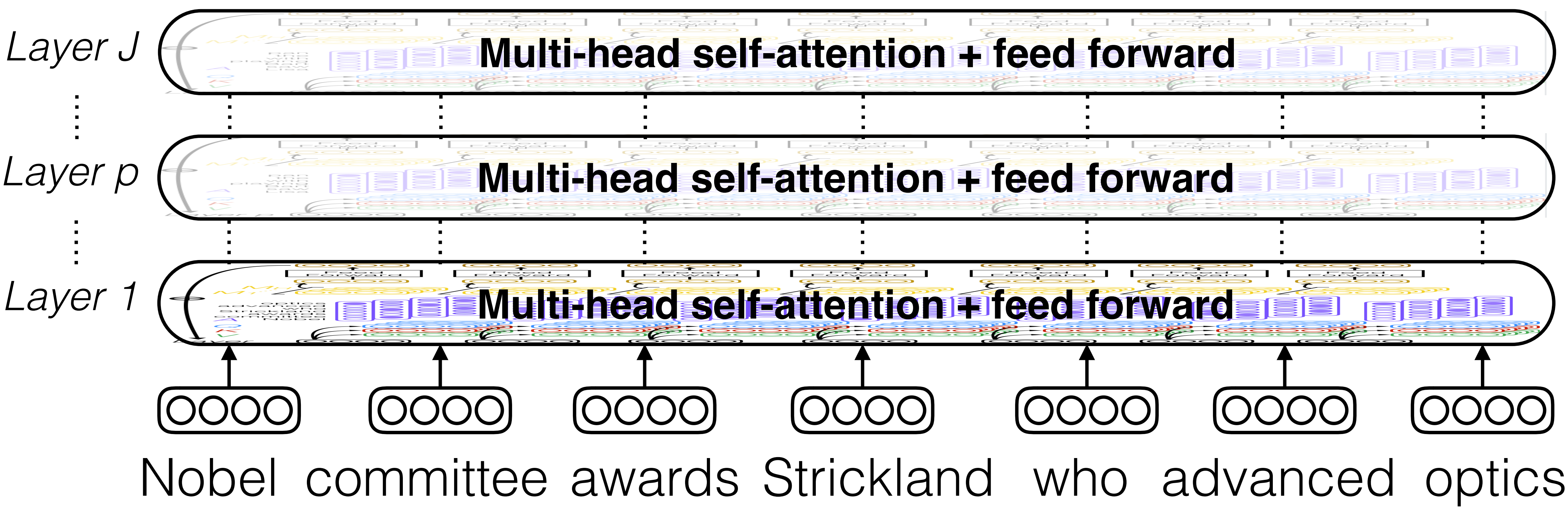
[Vaswani et al. 2017]

Multi-head self-attention



[Vaswani et al. 2017]

Multi-head self-attention



For next week:

- The full Transformer architecture
- The encoder/decoder paradigm
- Using neural language models for transfer learning: ELMo and BERT

UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1BL411t7RV

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/umass-cs685

NLP 语言模型 问答系统 文本生成 BERT 语义解析 GPT-3 知识推理 模型蒸馏 迁移学习 注意力机制 transformer

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]