Language Technologies Institute

Carnegie Mellon University

Algorithms for NLP

CS 11-711 · Fall 2020

Lecture 10: Masked language modeling and Transformers

Emma Strubell

Announcements

- Share your writeup from P1 (bake-off design and paper summary) on Piazza!
- Friday's recitation will be an overview of P2.

Previously, in language model pretraining…

Language models as word representations

- A problem with (most) word2vec-style word embeddings: single representation per word type.

the new-look **play** area is due to be completed by early spring 2020

gerrymandered congressional districts favor representatives who **play** to the party base

the freshman then completed the three-point **play** for a 66-63 lead

- Instead, want **contextualized** word embeddings: token representations that differ depending on the context.
  - But, still want to leverage self-supervised training on massive data.
  - Solution: use hidden representations from a neural language model.

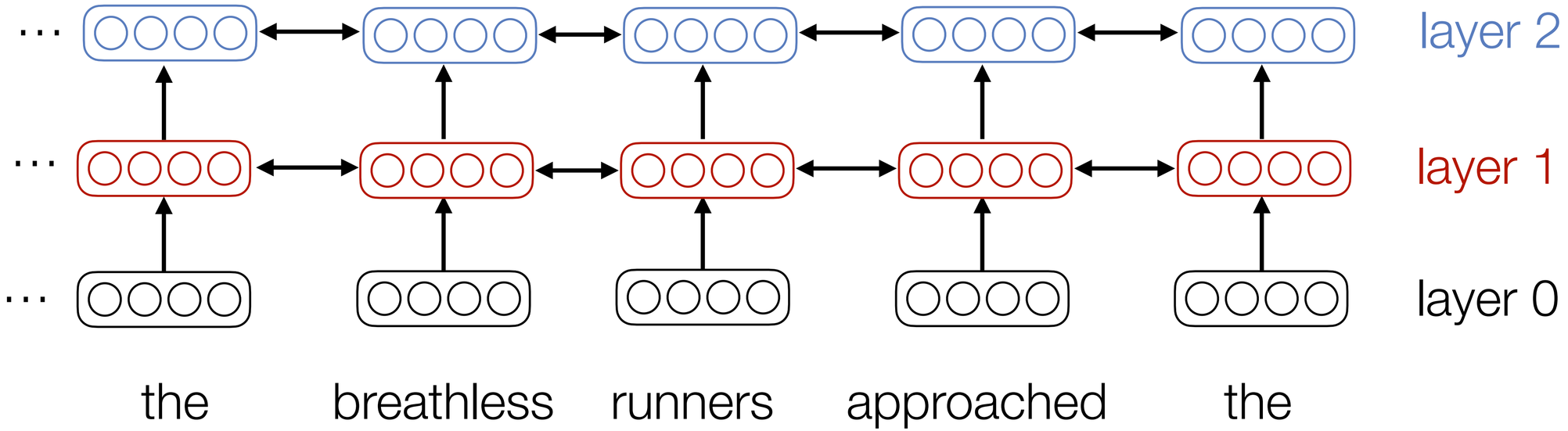
ELMo: Deep contextualized word embeddings

- Key idea: context-dependent embedding for each word interpolates representations for that word from each layer

breathless = γ **(** $s_1 \cdot$ ⬭ **+** $s_2 \cdot$ ⬭ **+** $s_3 \cdot$ ⬭ **)**

$$\sum_{j=1}^{L} s_j = 1$$

- Interpolation weights are task-specific (fine-tuned on supervised data.)



stacked bidirectional LSTM LM

Peters et al. Deep contextualized word representations. NAACL 2018.

ELMo: Deep contextualized word embeddings

- Adding ELMo to existing state-of-the-art models provides significant improvement on essentially all NLP tasks.

	Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
question answering	SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
natural language inference	SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
semantic role labeling	SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
coreference	Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
named entity recognition	NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
sentiment analysis	SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

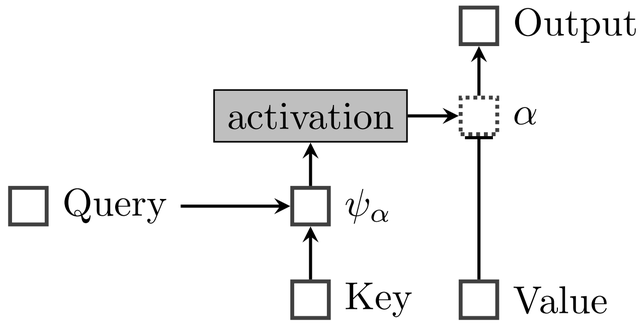
Peters et al. Deep contextualized word representations. NAACL 2018.

Attention in neural networks

Attention in neural networks

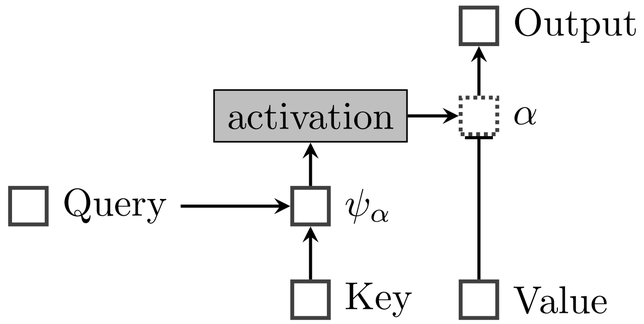
Attention in neural networks

- **Neural attention mechanism**: a weighted average over (word) representations, where the weights are learned, computed as a function of the input.



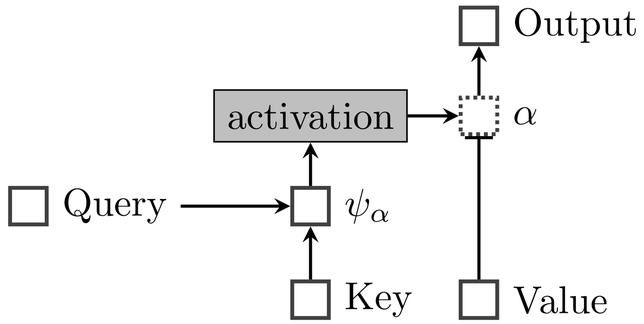
Attention in neural networks

- **Neural attention mechanism**: a weighted average over (word) representations, where the weights are learned, computed as a function of the input.
  - Additive attention: ψ is a feed-forward layer with concatenated [k;q] as input.



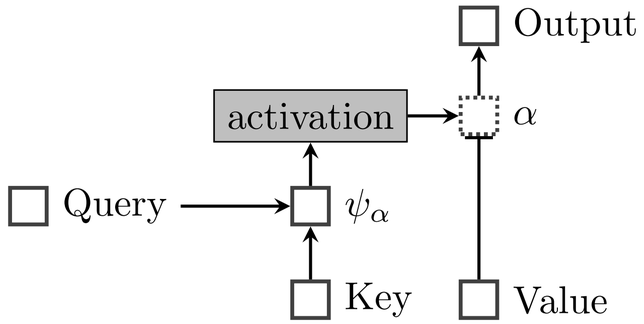
Attention in neural networks

- **Neural attention mechanism**: a weighted average over (word) representations, where the weights are learned, computed as a function of the input.
  - Additive attention: ψ is a feed-forward layer with concatenated [k;q] as input.
  - Multiplicative attention: ψ is a dot product of linear projections of k, q.



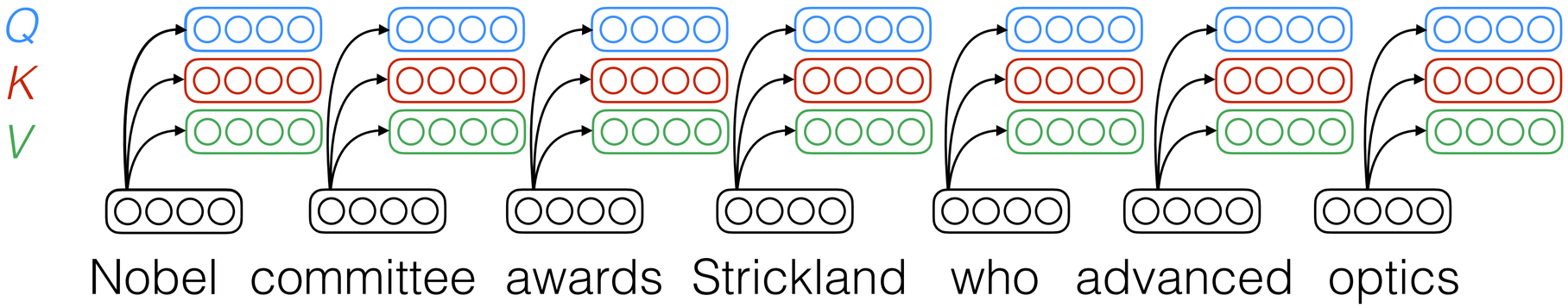
Attention in neural networks

- **Neural attention mechanism**: a weighted average over (word) representations, where the weights are learned, computed as a function of the input.
  - Additive attention: ψ is a feed-forward layer with concatenated [k;q] as input.
  - Multiplicative attention: ψ is a dot product of linear projections of k, q.
- First popularized in neural machine translation, where the network would attend over the source sentence to make each prediction in the target sentence.

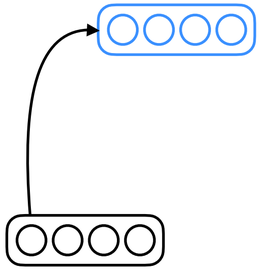


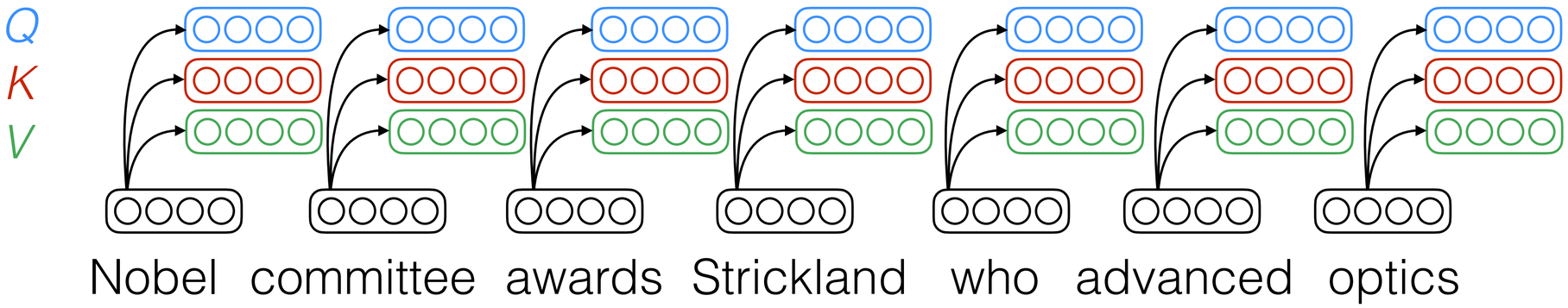
Transformer self-attention

Transformer self-attention

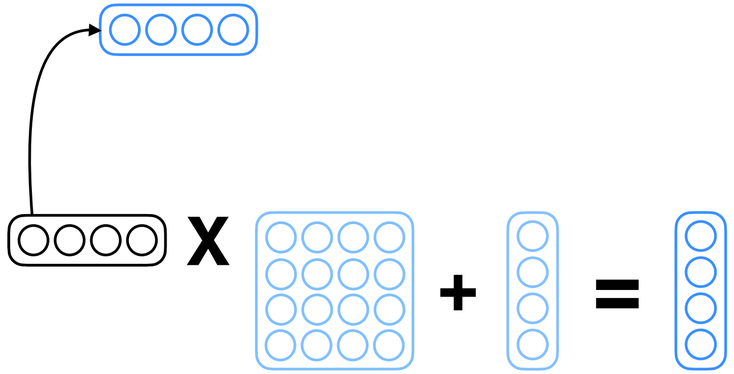


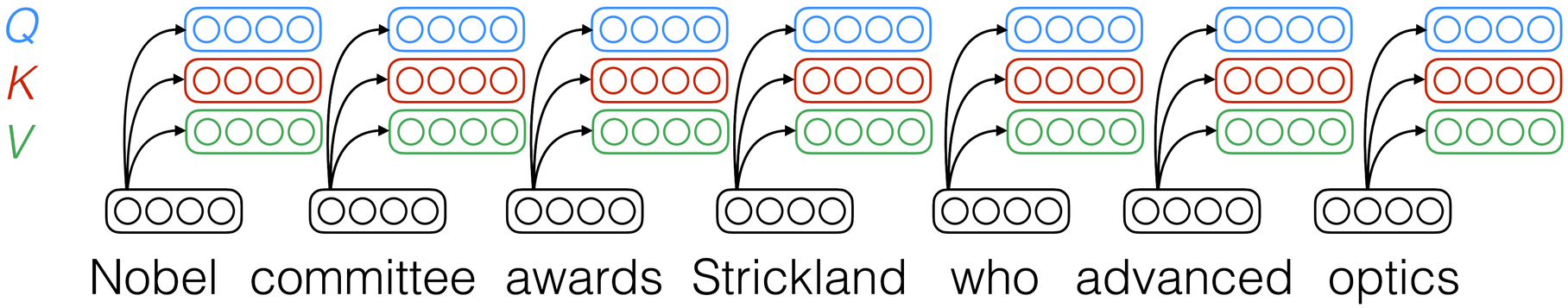
Transformer self-attention



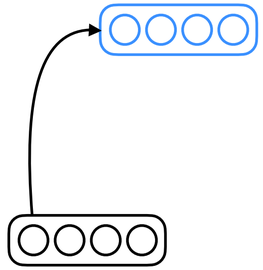


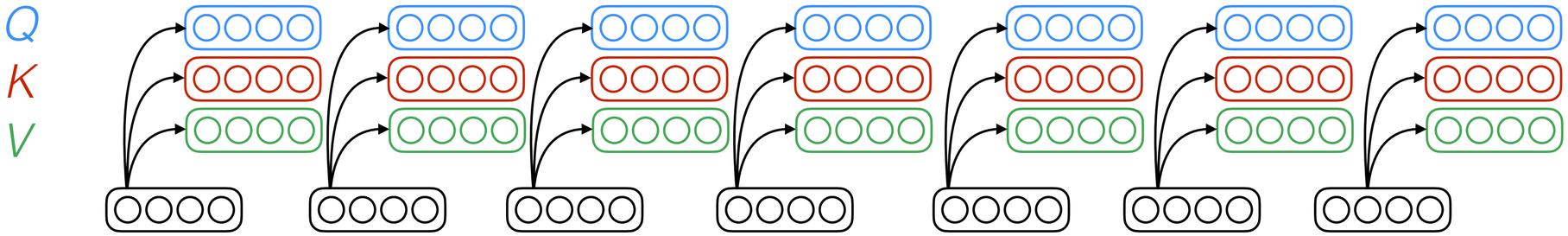
Transformer self-attention



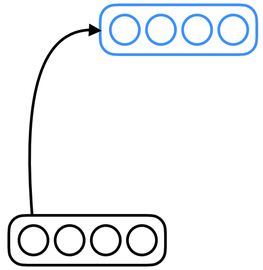


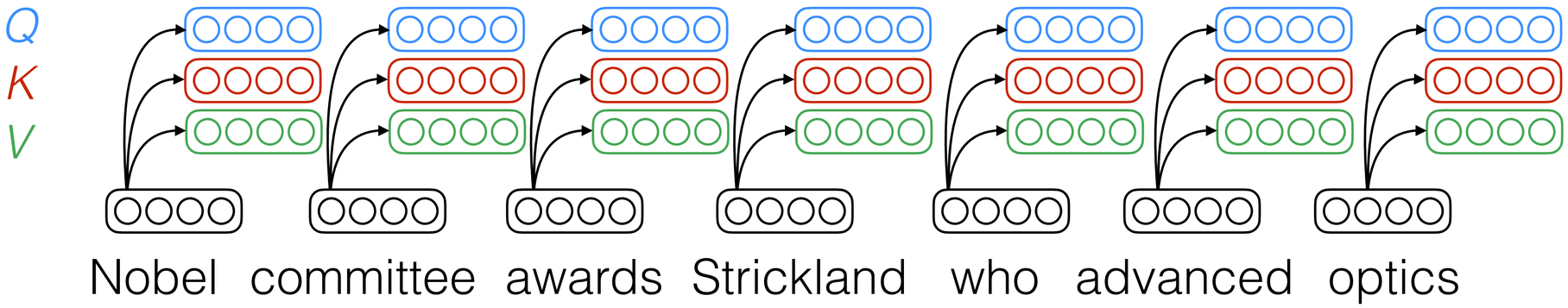
Transformer self-attention



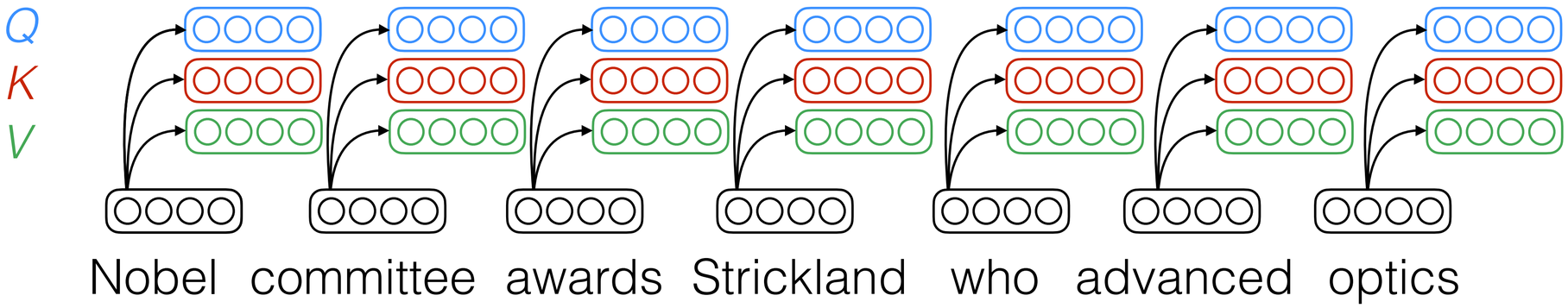


Transformer self-attention

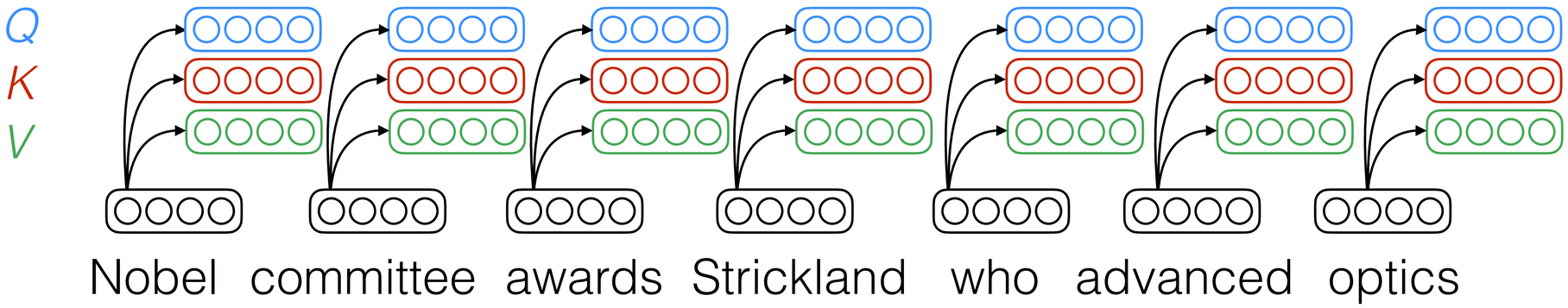




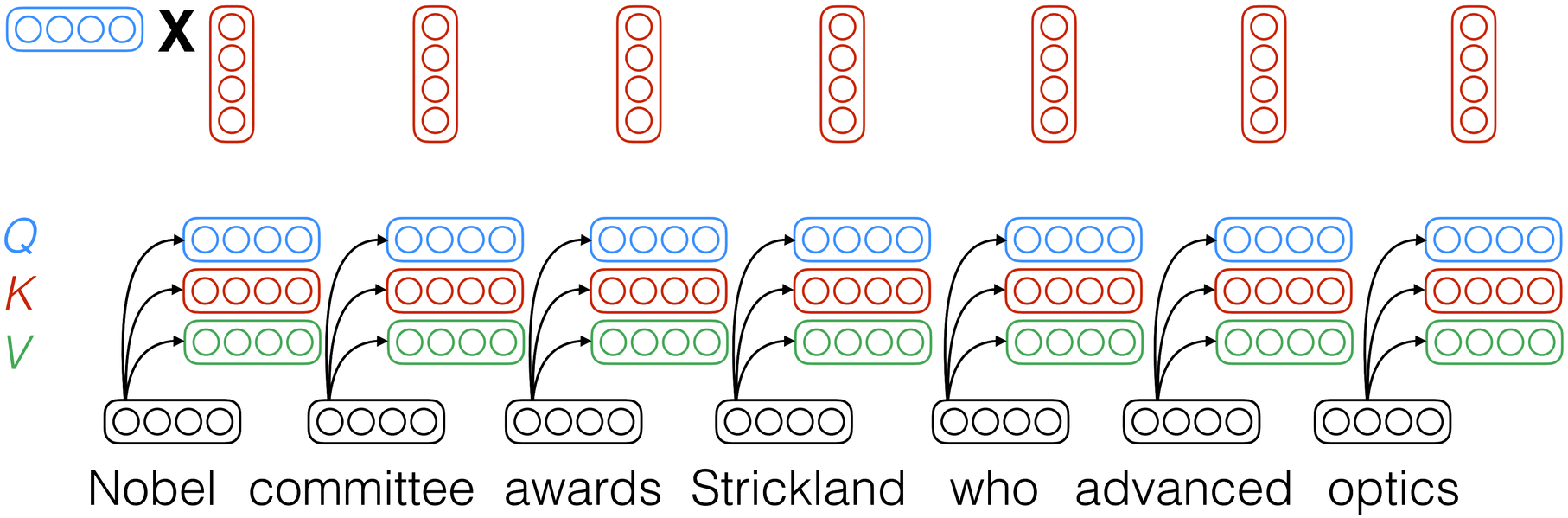
Transformer self-attention



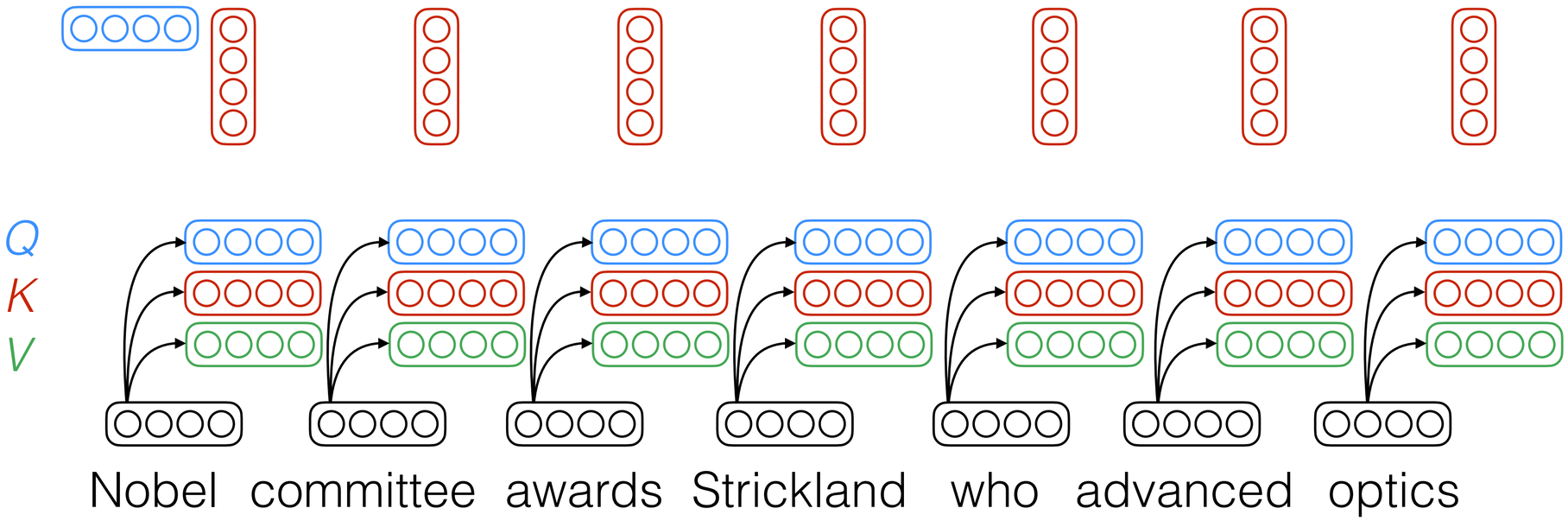
Transformer self-attention



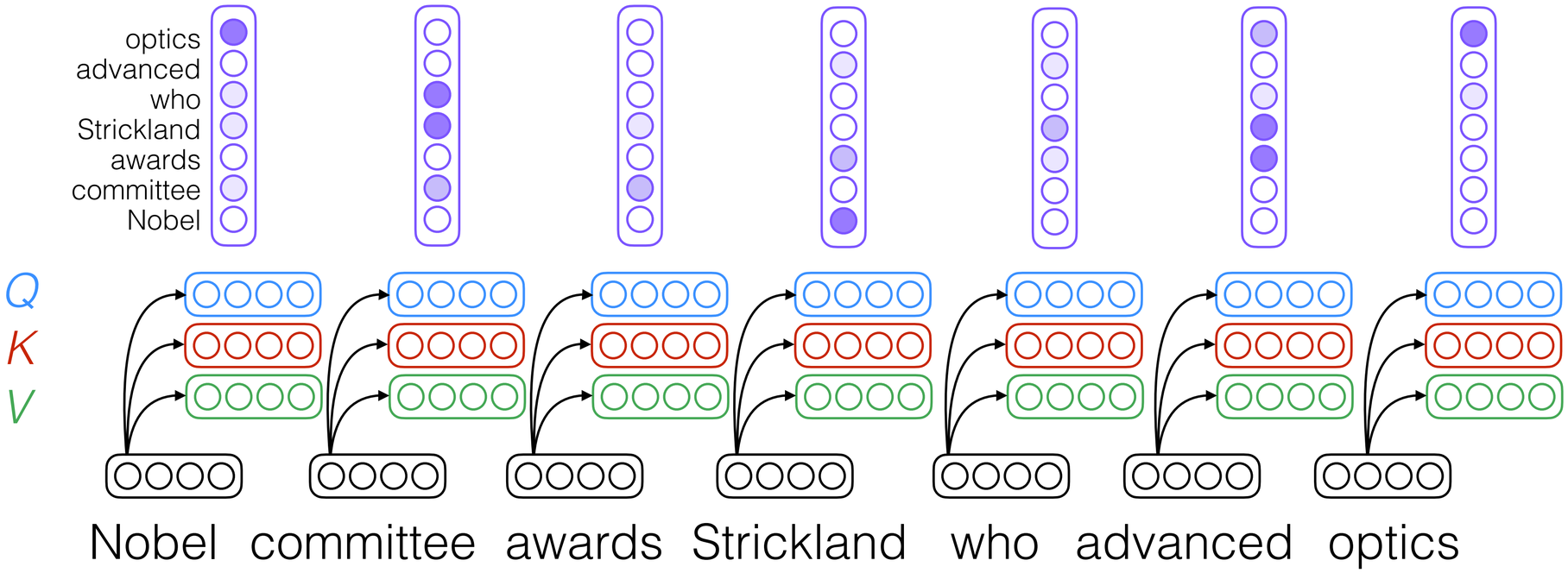
Transformer self-attention



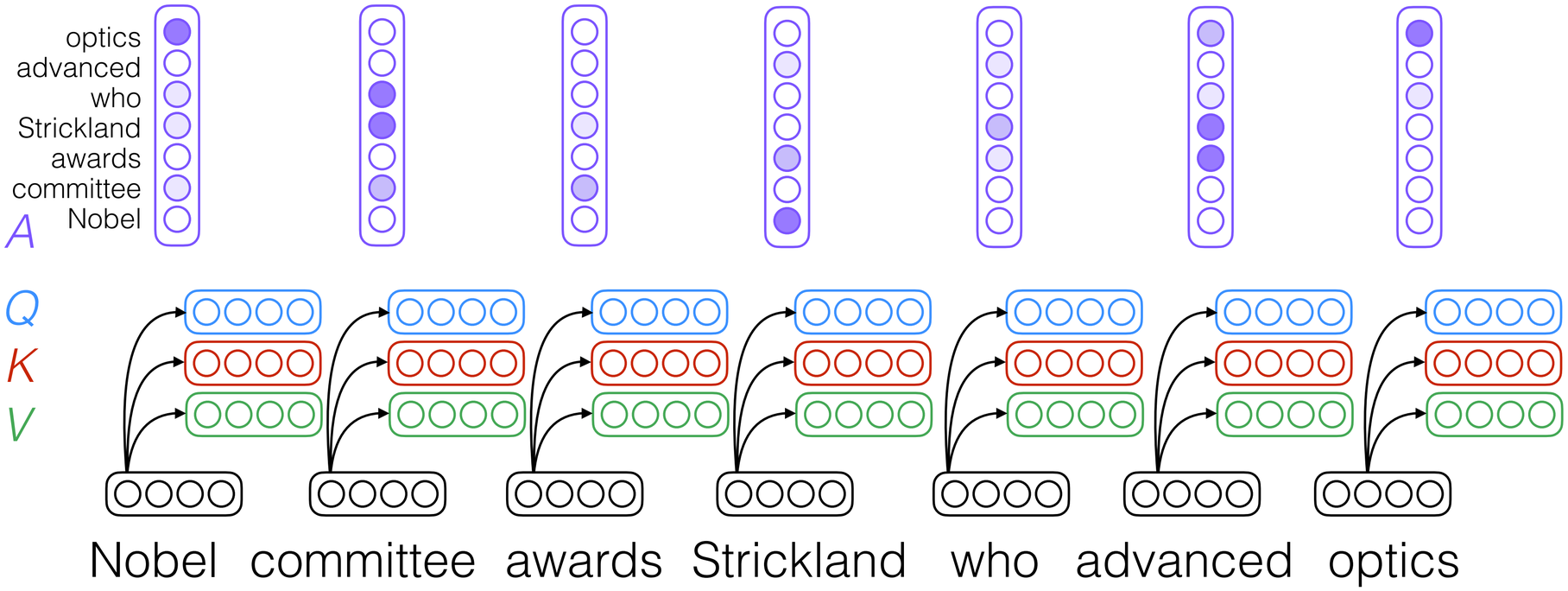
Transformer self-attention



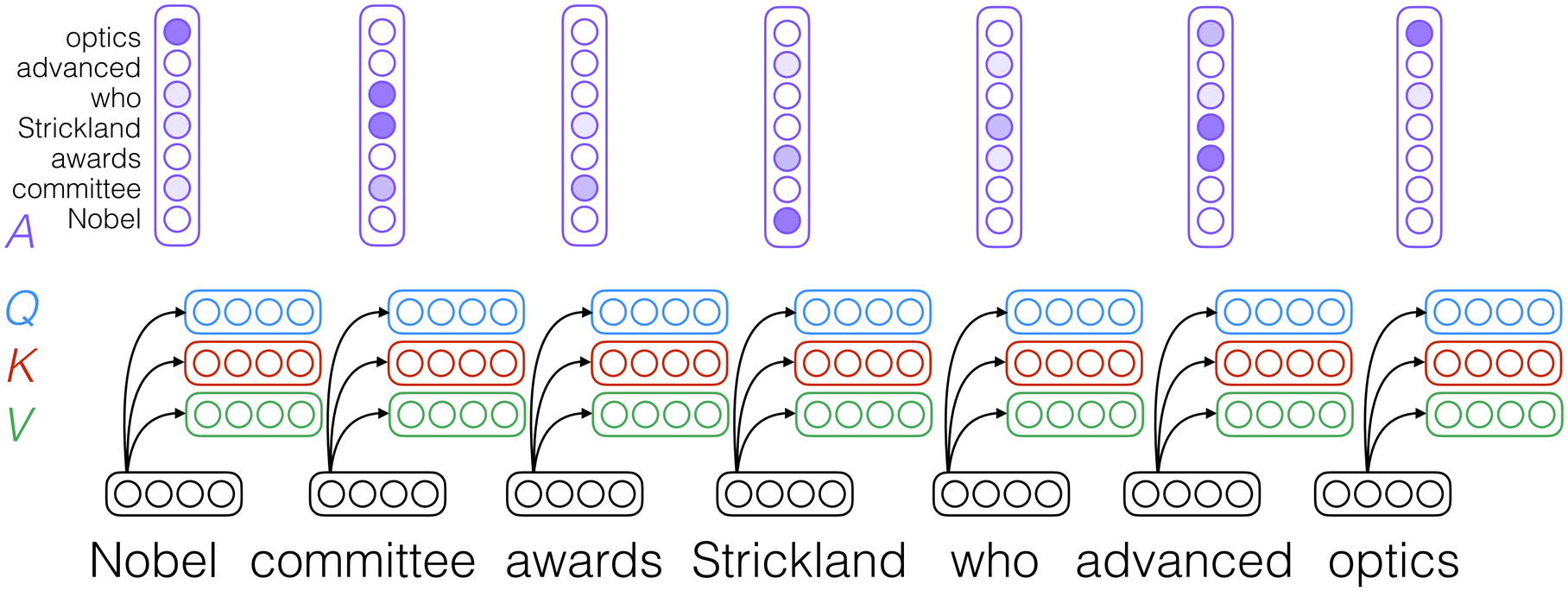
Transformer self-attention



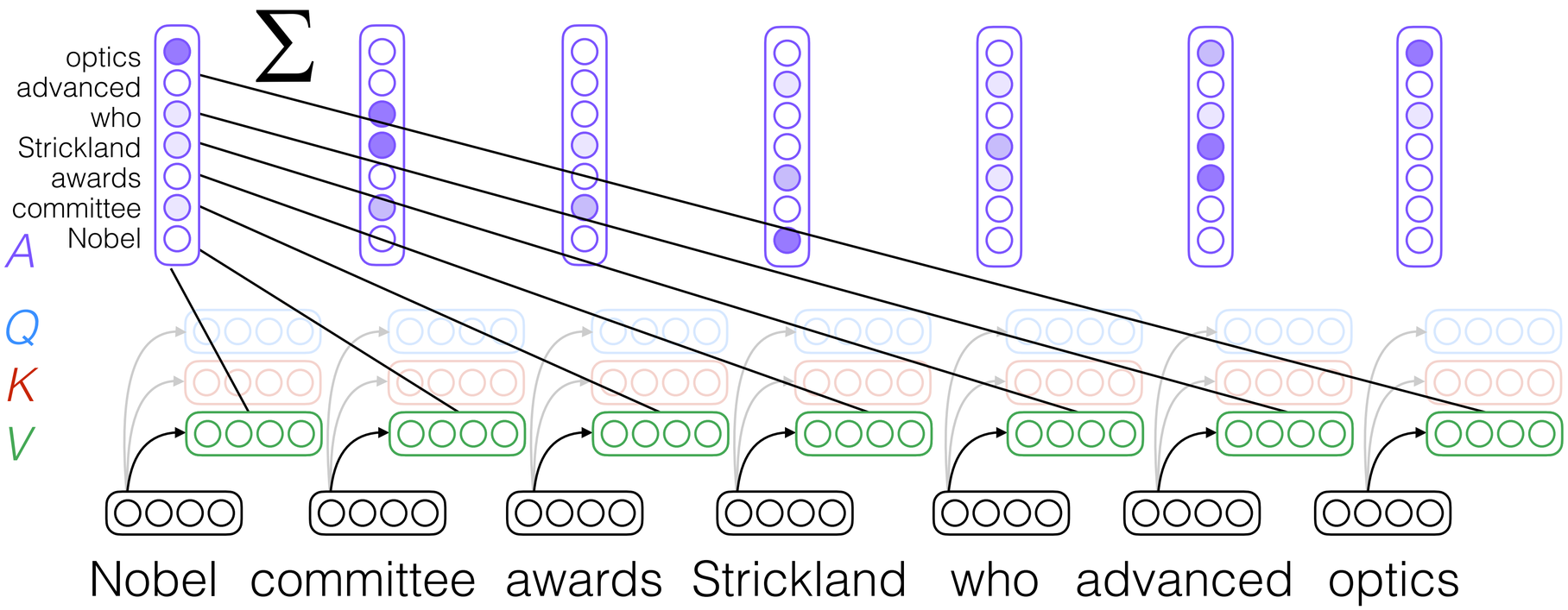
Transformer self-attention



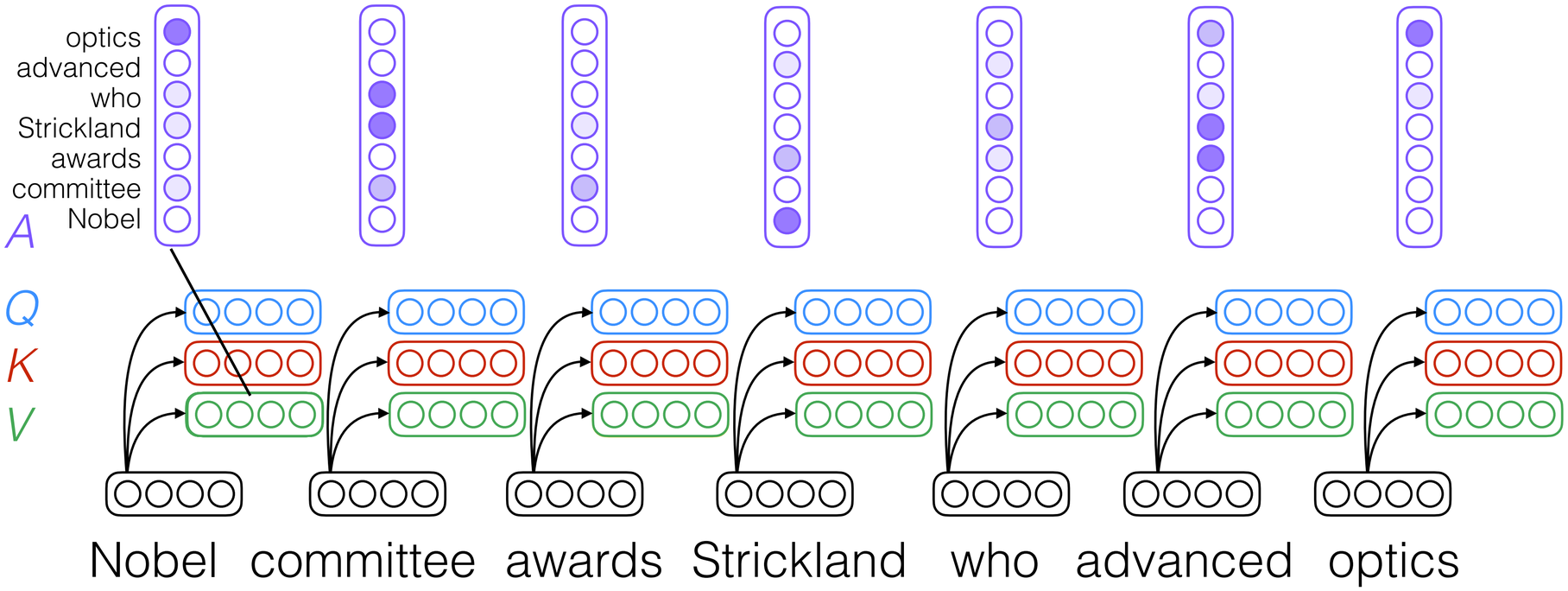
Transformer self-attention



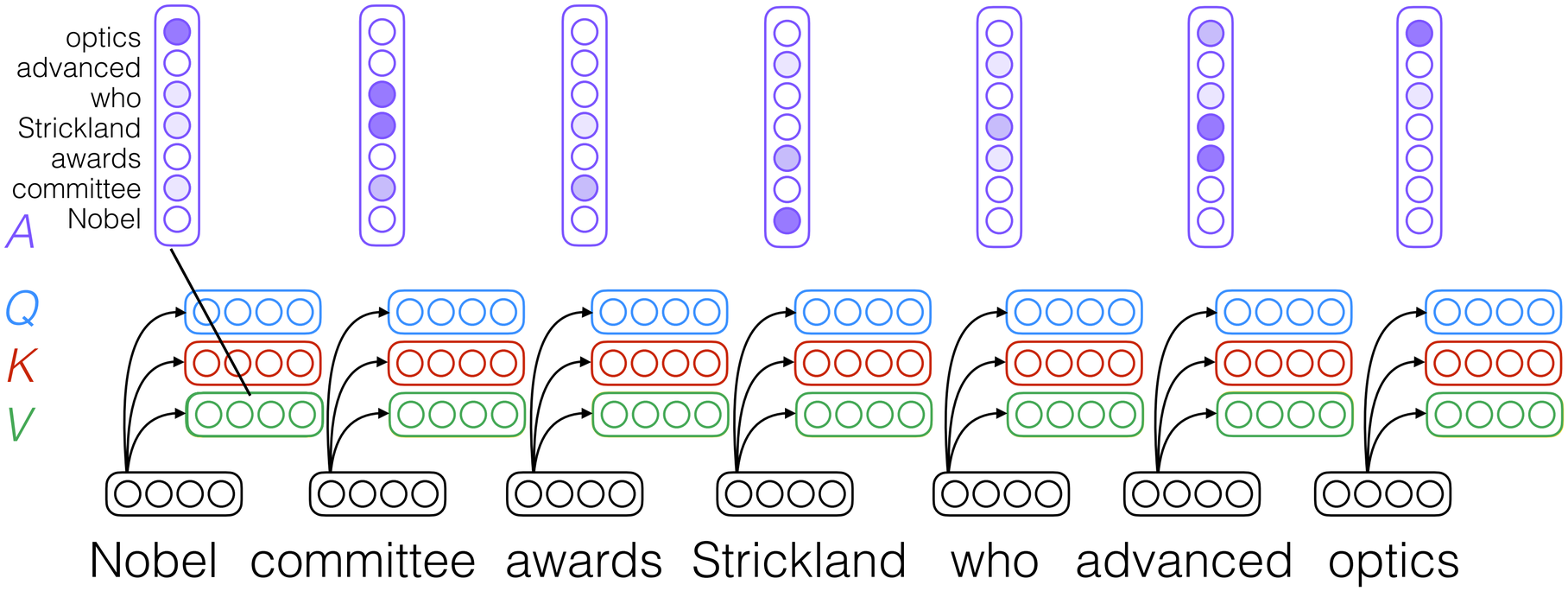
Transformer self-attention



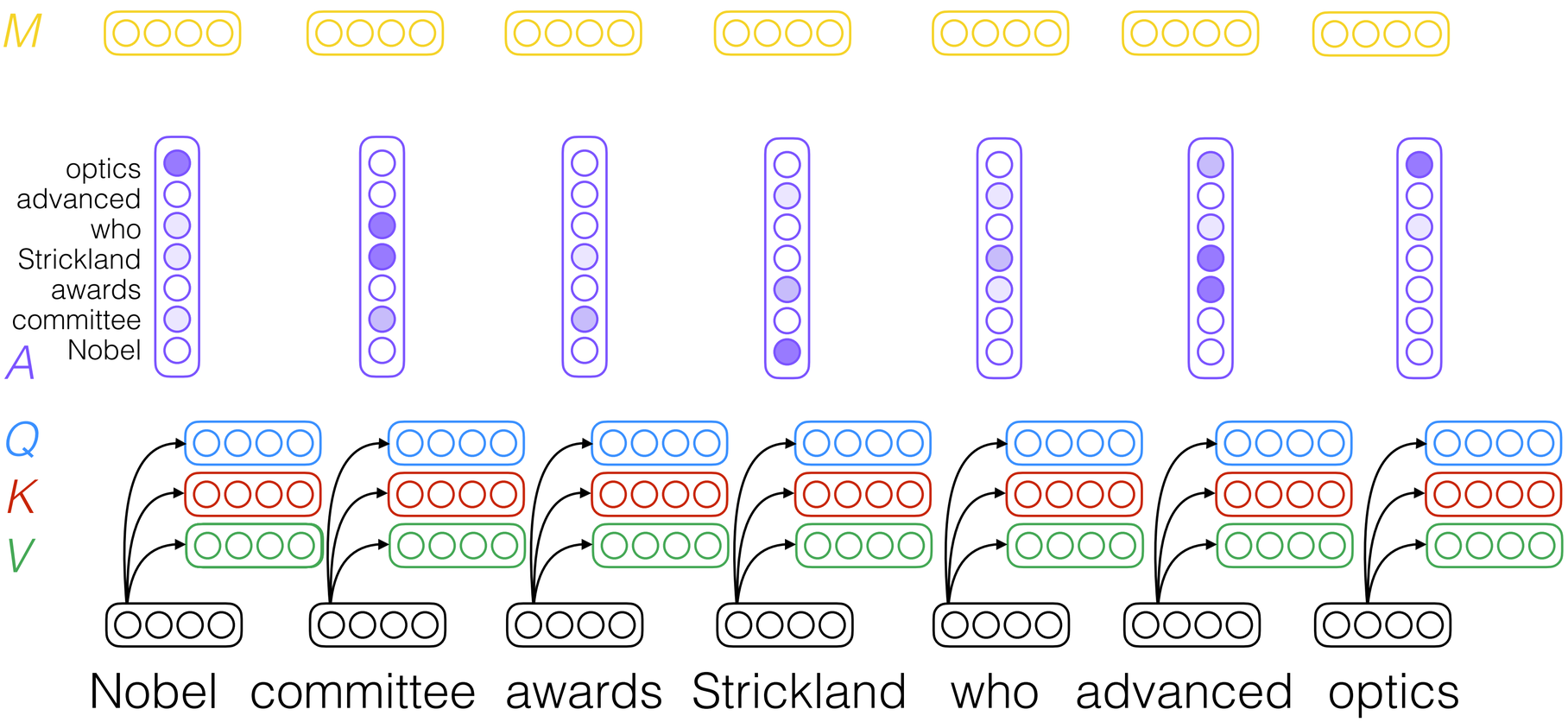
Transformer self-attention



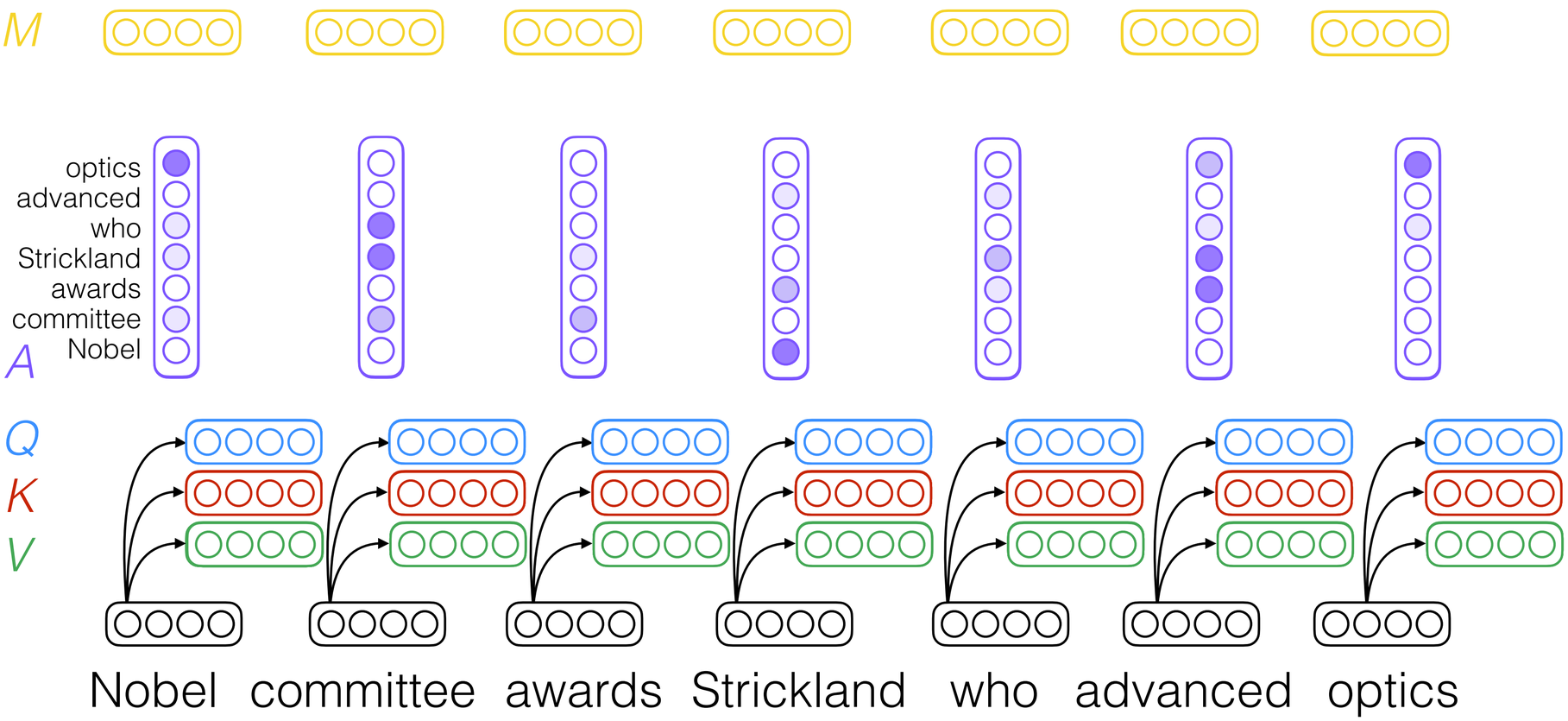
Transformer self-attention



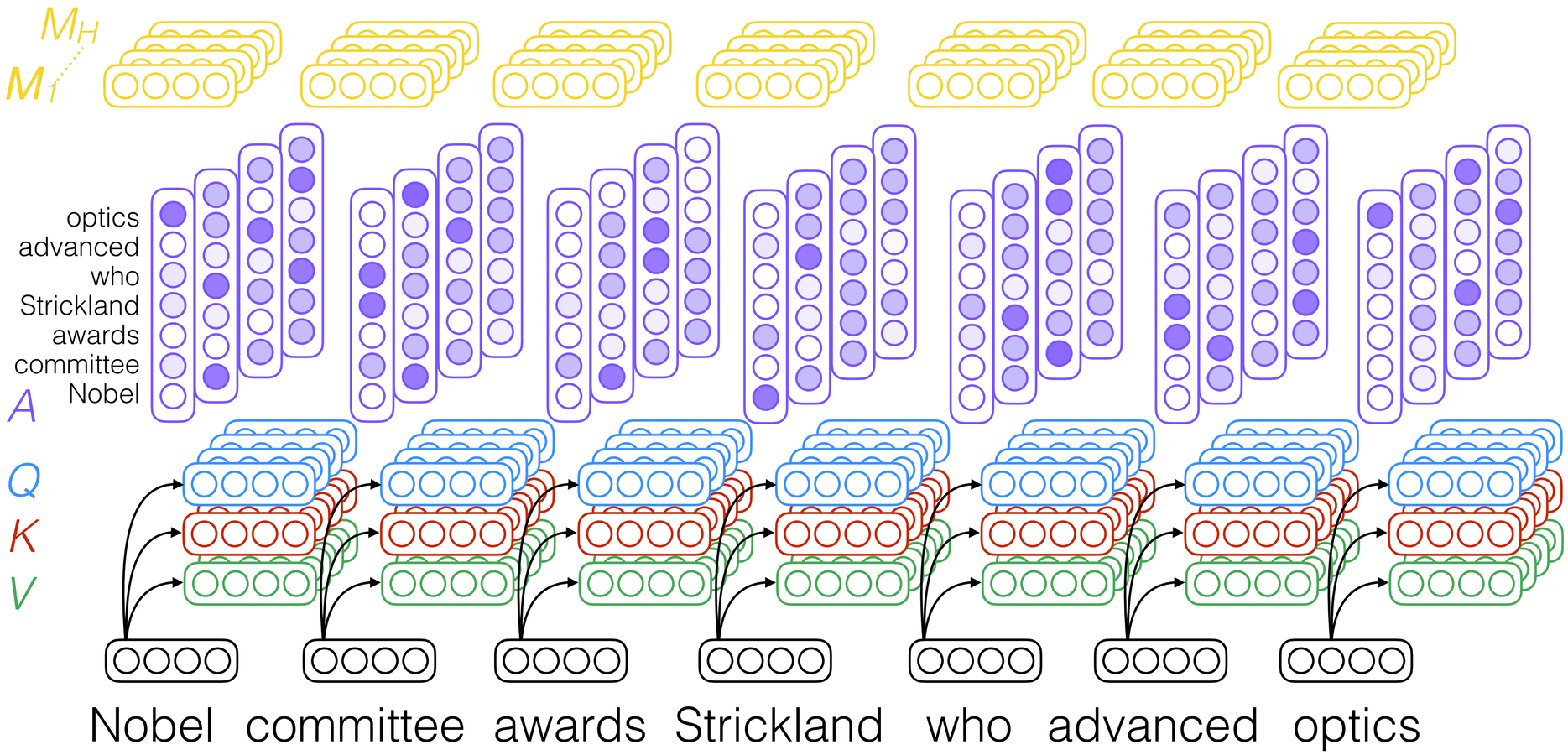
Transformer self-attention



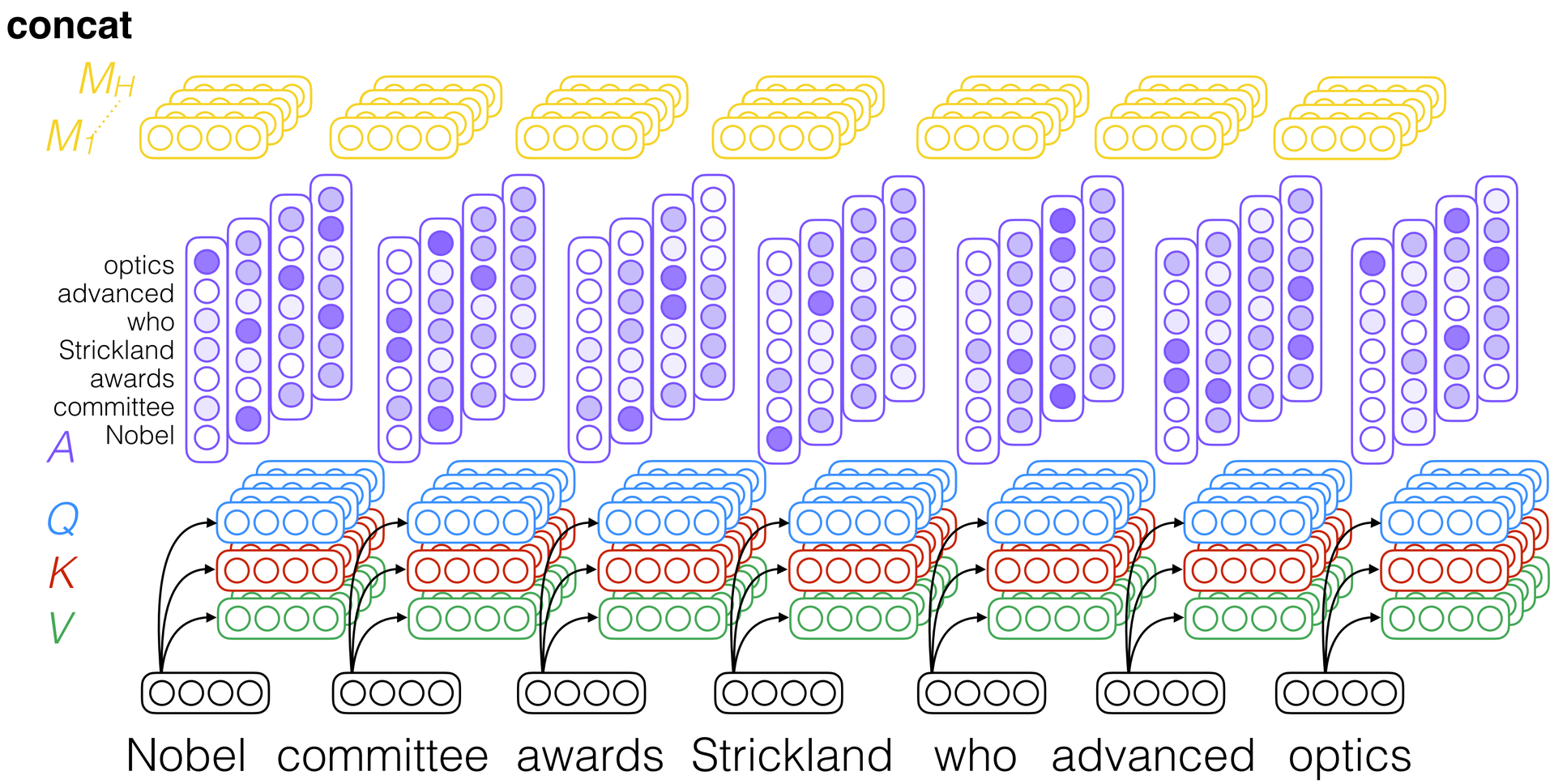
Transformer self-attention



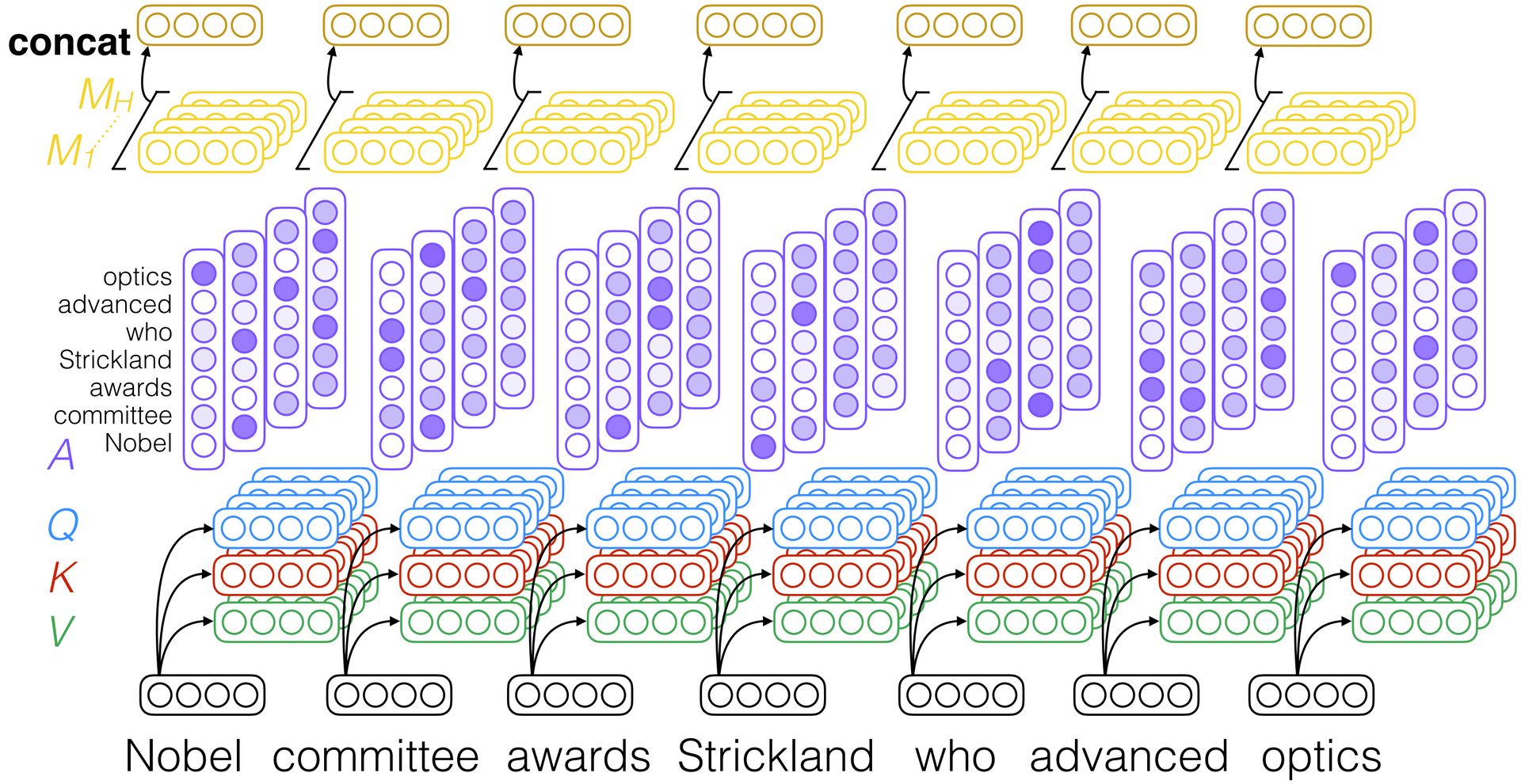
Multi-head self-attention



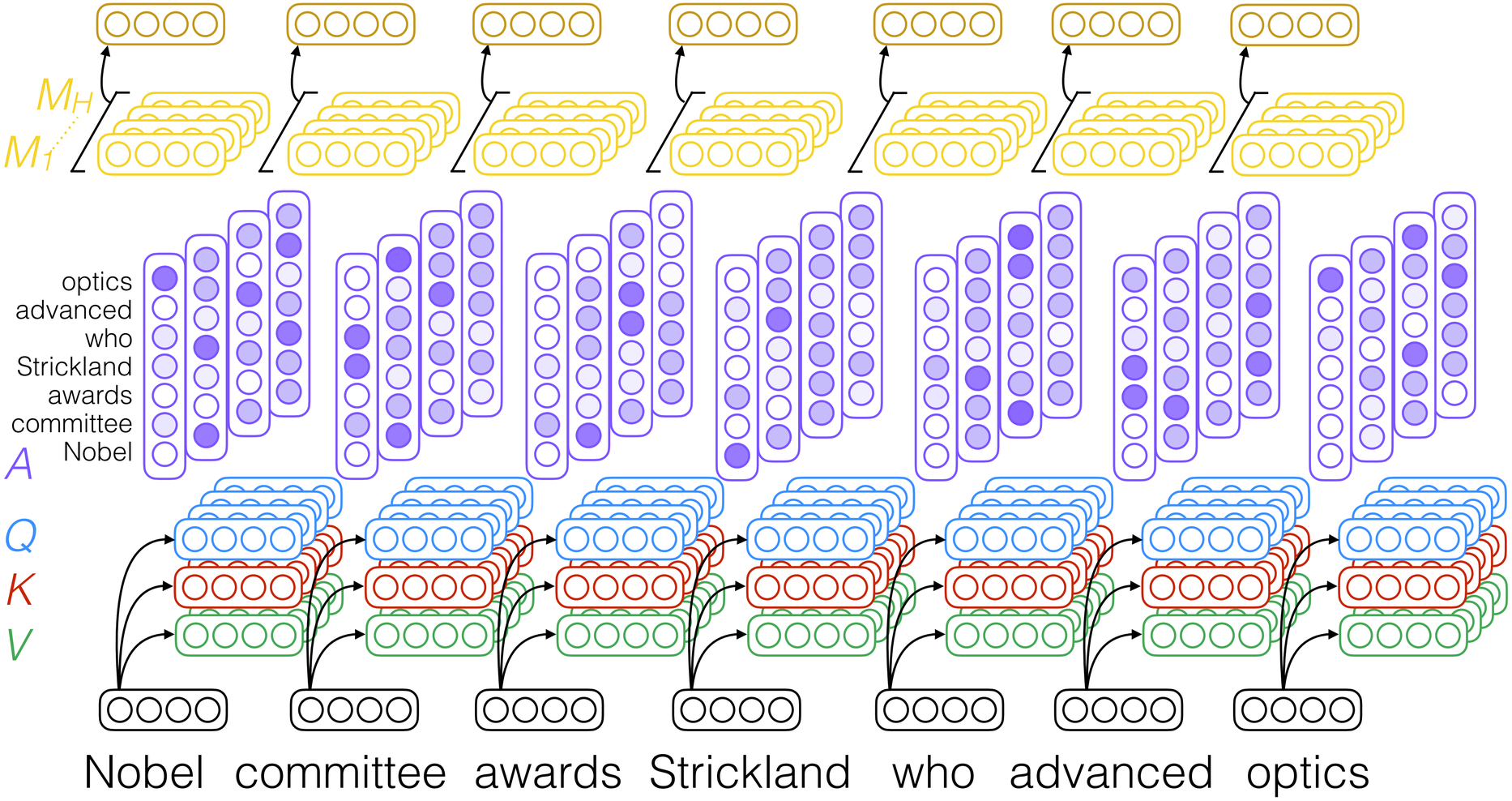
Multi-head self-attention



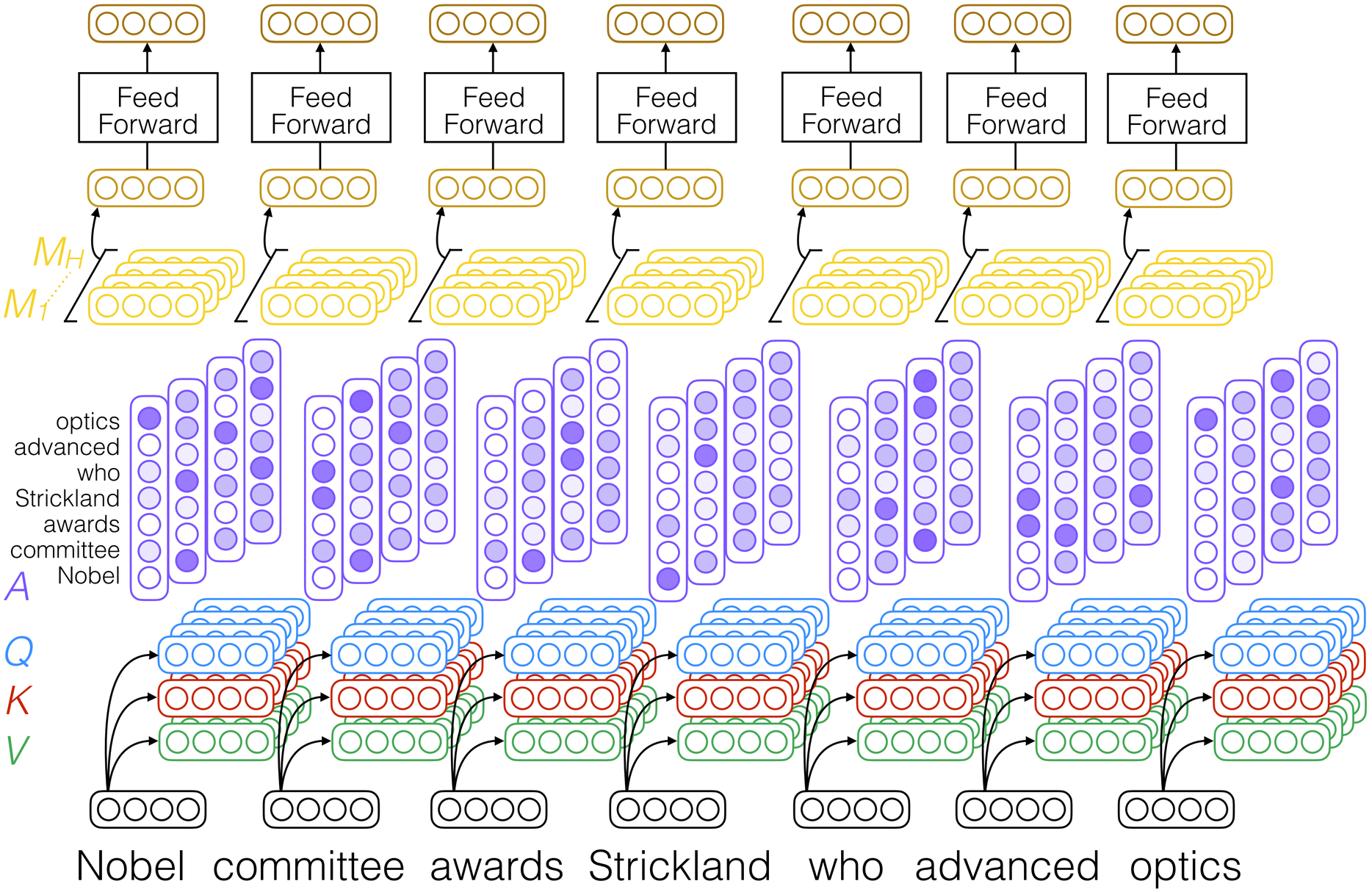
Multi-head self-attention



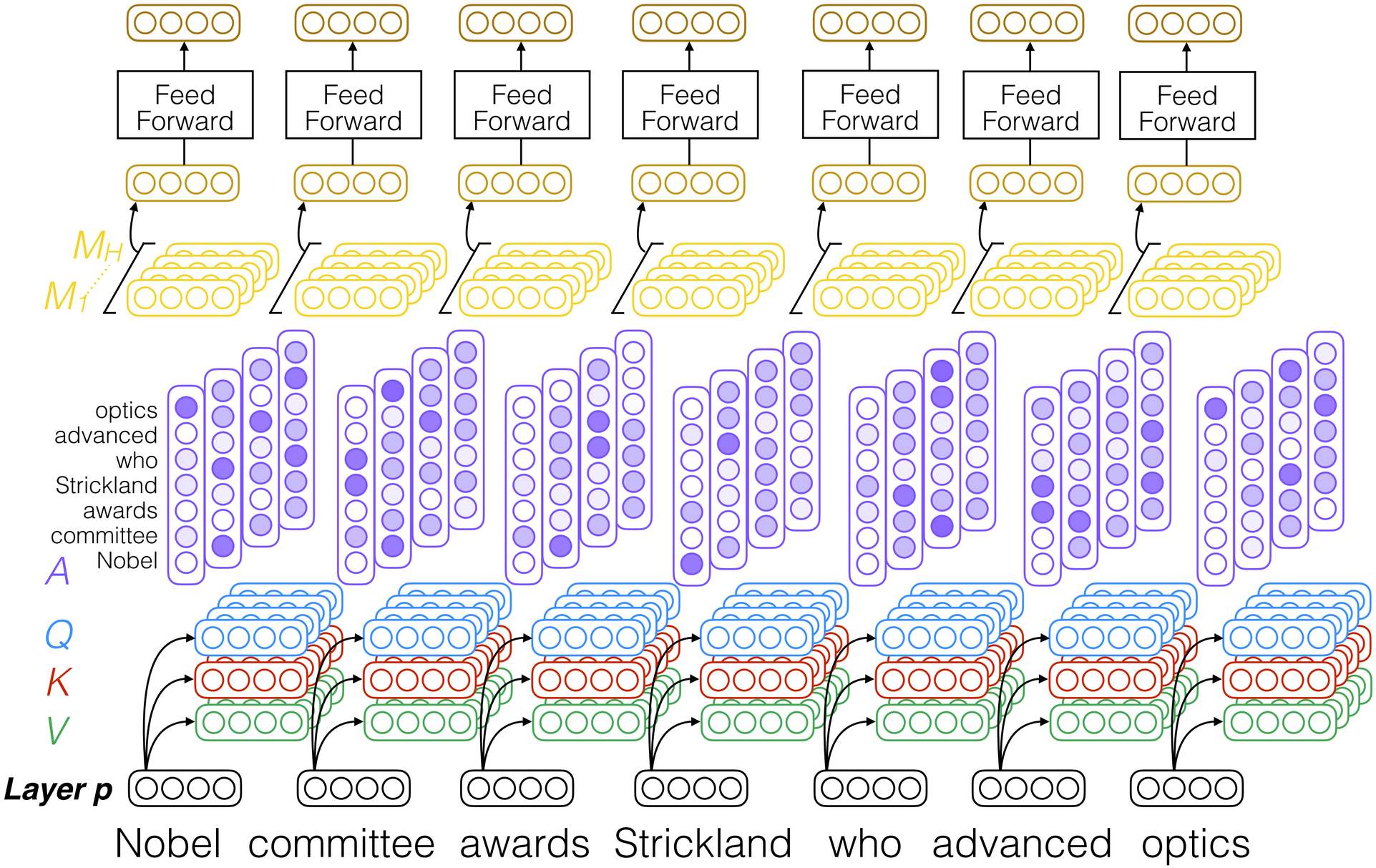
Transformer layer



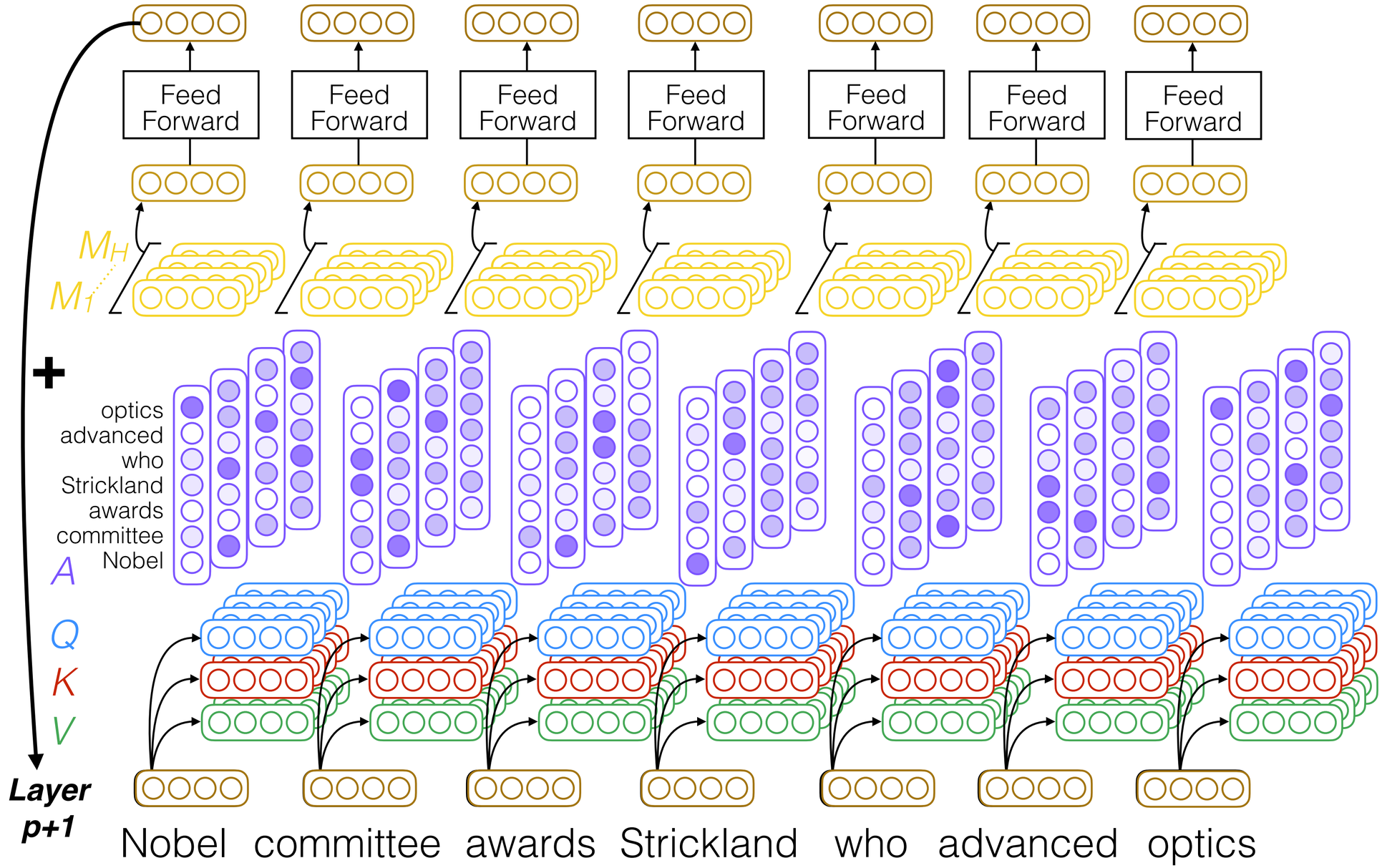
Transformer layer



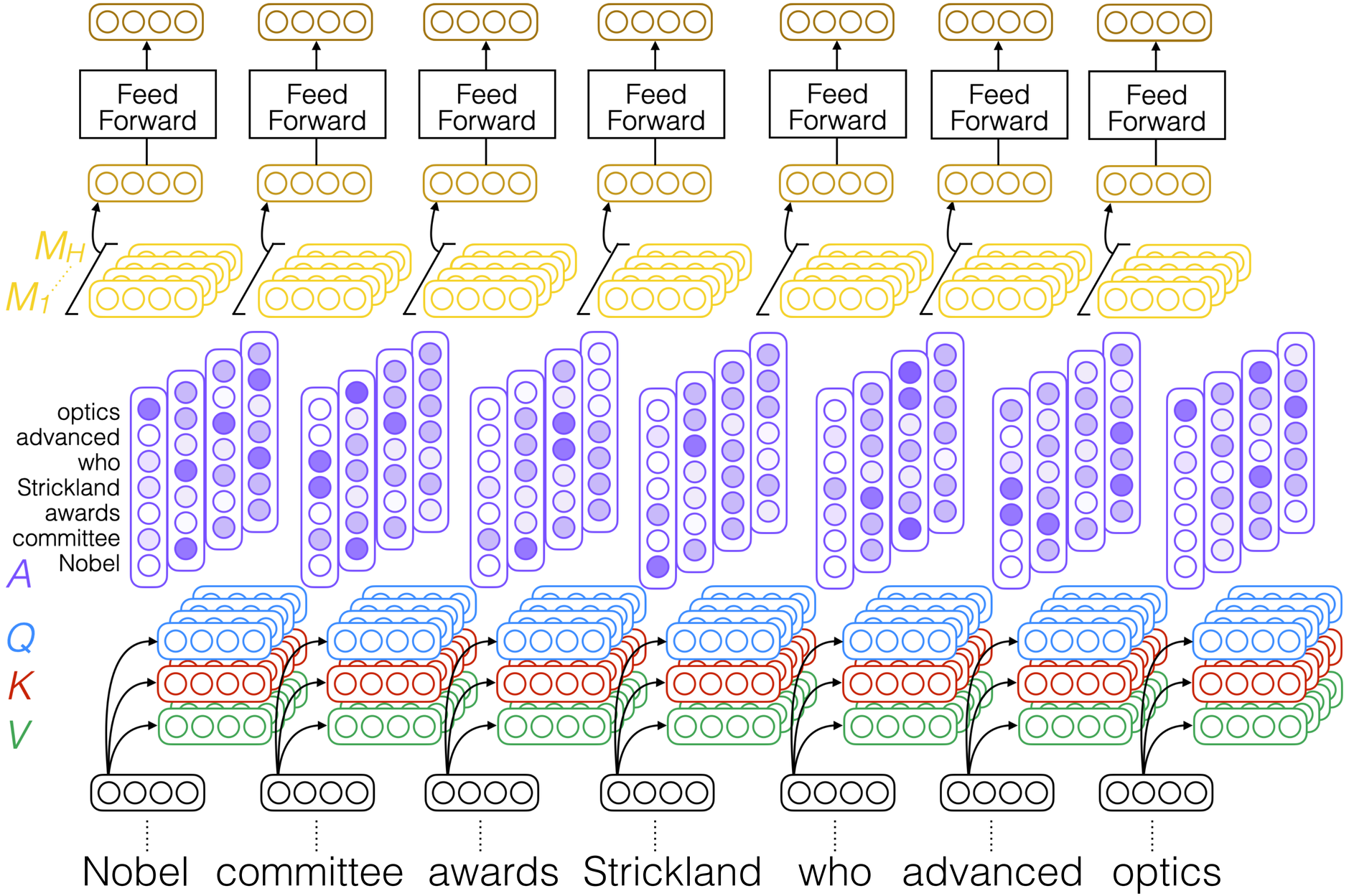
Transformer layer



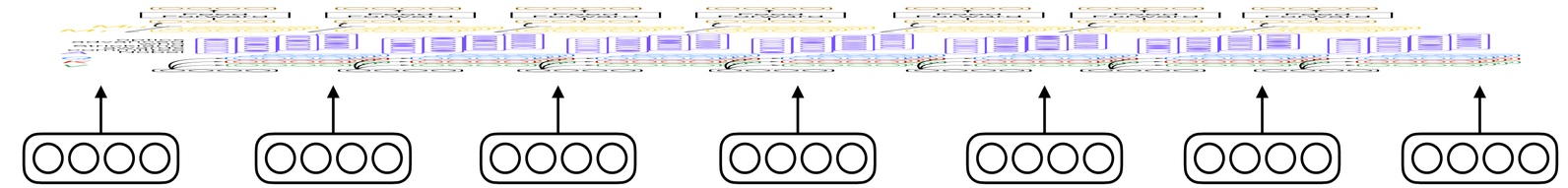
Transformer layer



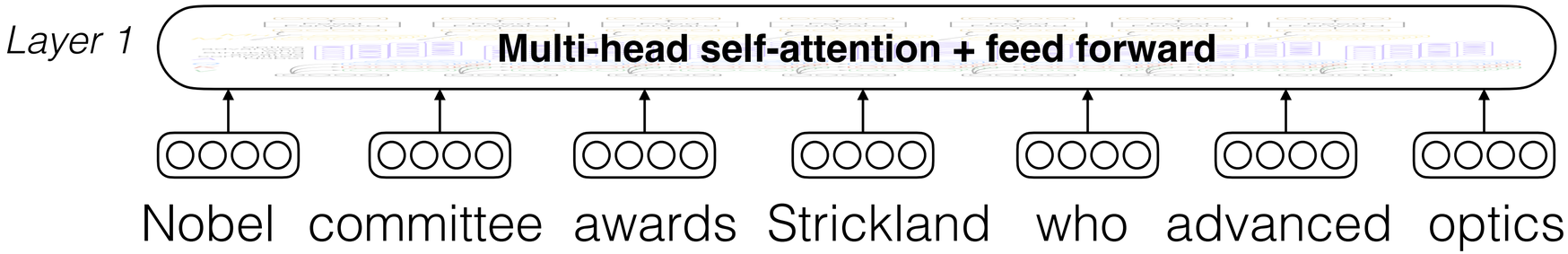
Transformer layer



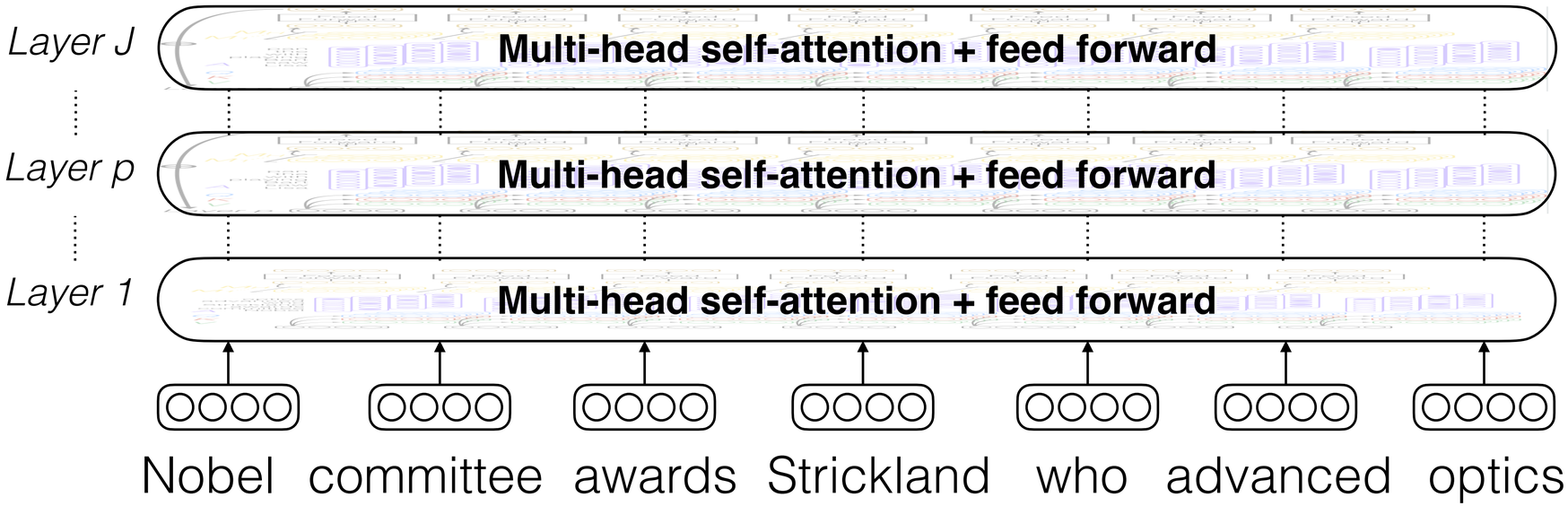
Transformer architecture



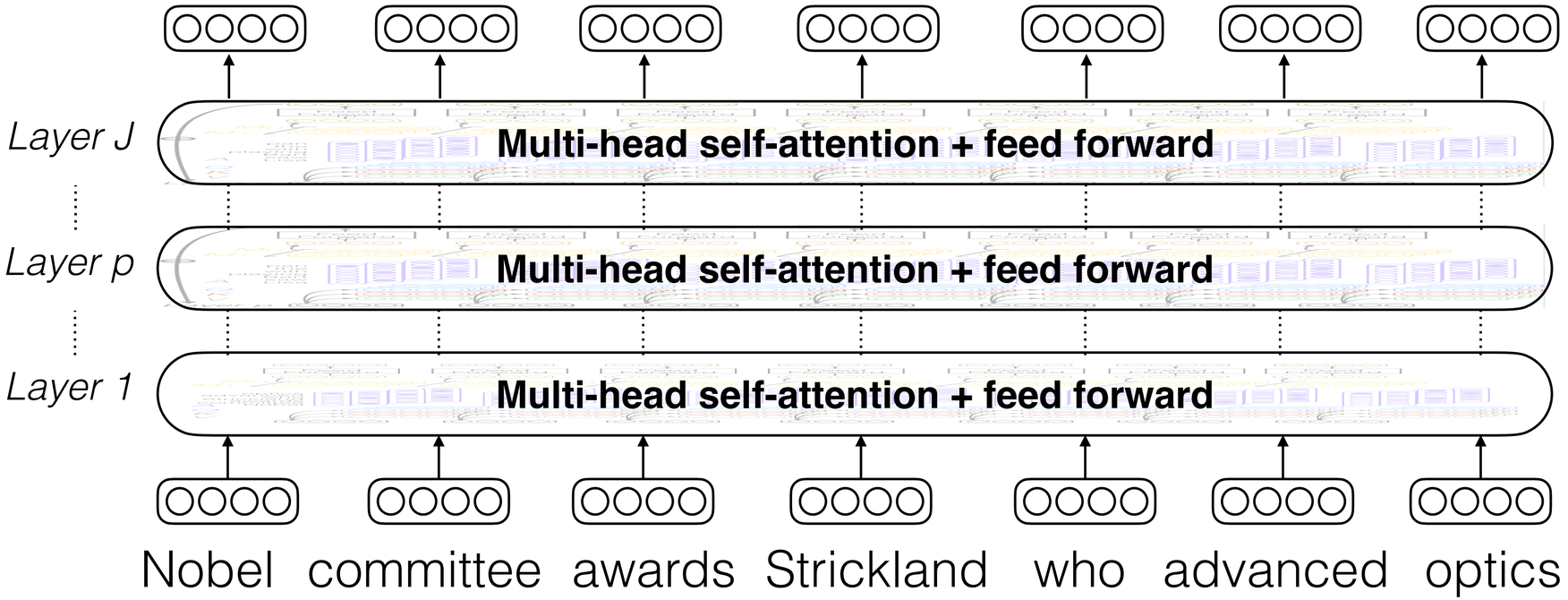
Transformer architecture



Transformer architecture



Transformer architecture



Figures from: Transformer Architecture: The Positional Encoding

Modeling context

Figures from: Transformer Architecture: The Positional Encoding

Modeling context

- Like RNNs, each token observes context from the *entire sequence* at each layer.

Figures from: Transformer Architecture: The Positional Encoding

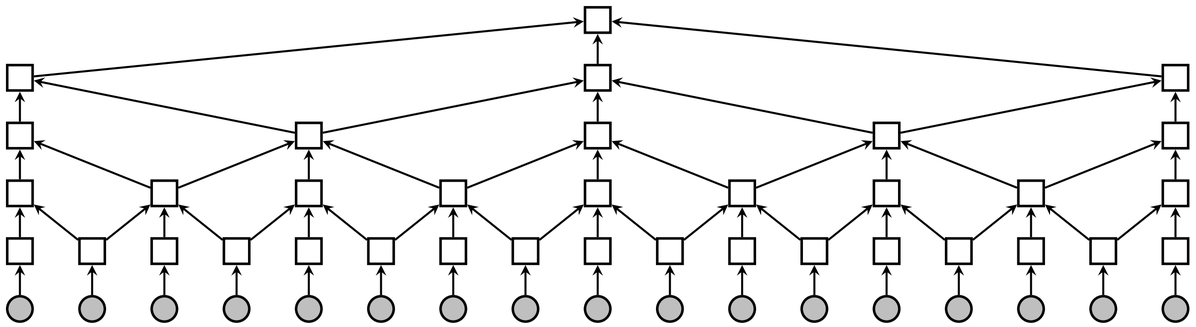
Modeling context

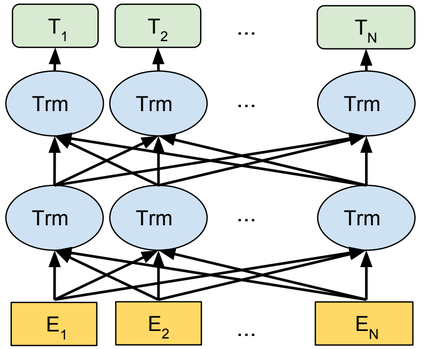
- Like RNNs, each token observes context from the *entire sequence* at each layer.
- Like CNNs, representations at each layer can be computed *entirely in parallel*.

Figures from: Transformer Architecture: The Positional Encoding

Modeling context

- Like RNNs, each token observes context from the *entire sequence* at each layer.
- Like CNNs, representations at each layer can be computed *entirely in parallel*.
  - Dilated CNNs obtain wide (enough) context and are faster than Transformers, but at the cost of expressivity: impose a regular pattern of connectivity between tokens, whereas in Transformer connectivity is learned / each token always has access to all other tokens.





Figures from: Transformer Architecture: The Positional Encoding

Positional encoding

Figures from: Transformer Architecture: The Positional Encoding

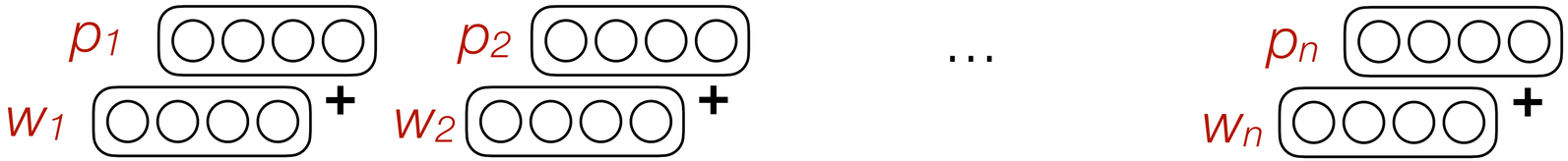
Positional encoding

- **Positional embeddings** encode a token's position in the sentence

Figures from: Transformer Architecture: The Positional Encoding

Positional encoding

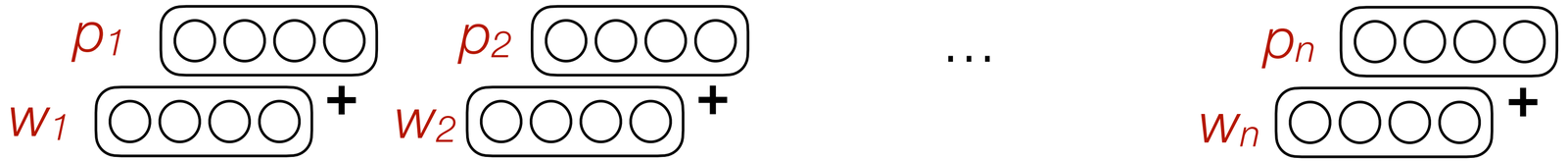
- **Positional embeddings** encode a token's position in the sentence



Figures from: Transformer Architecture: The Positional Encoding

Positional encoding

- **Positional embeddings** encode a token's position in the sentence

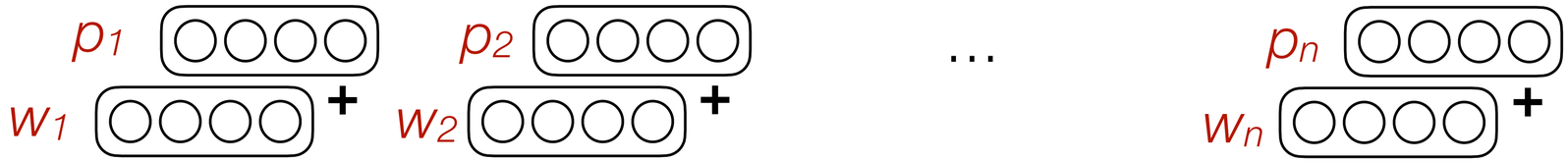


- **Sinusoidal position encodings** generalize to any sequence length.

Figures from: Transformer Architecture: The Positional Encoding

Positional encoding

- **Positional embeddings** encode a token's position in the sentence

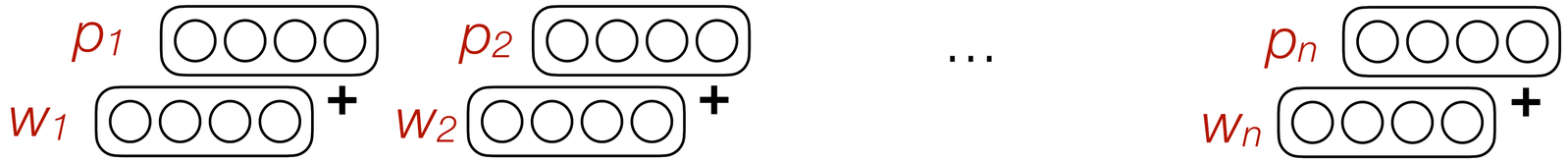


- **Sinusoidal position encodings** generalize to any sequence length.

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \quad \text{with} \quad \omega_k = \frac{1}{10000^{2k/d}}$$

Positional encoding

- **Positional embeddings** encode a token's position in the sentence



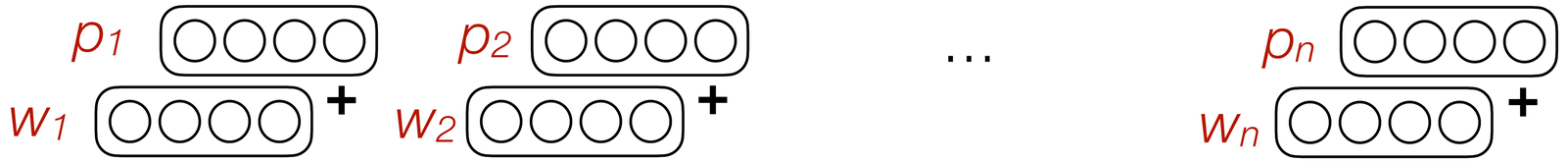
- **Sinusoidal position encodings** generalize to any sequence length.

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \quad \text{with} \quad \omega_k = \frac{1}{10000^{2k/d}}$$

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

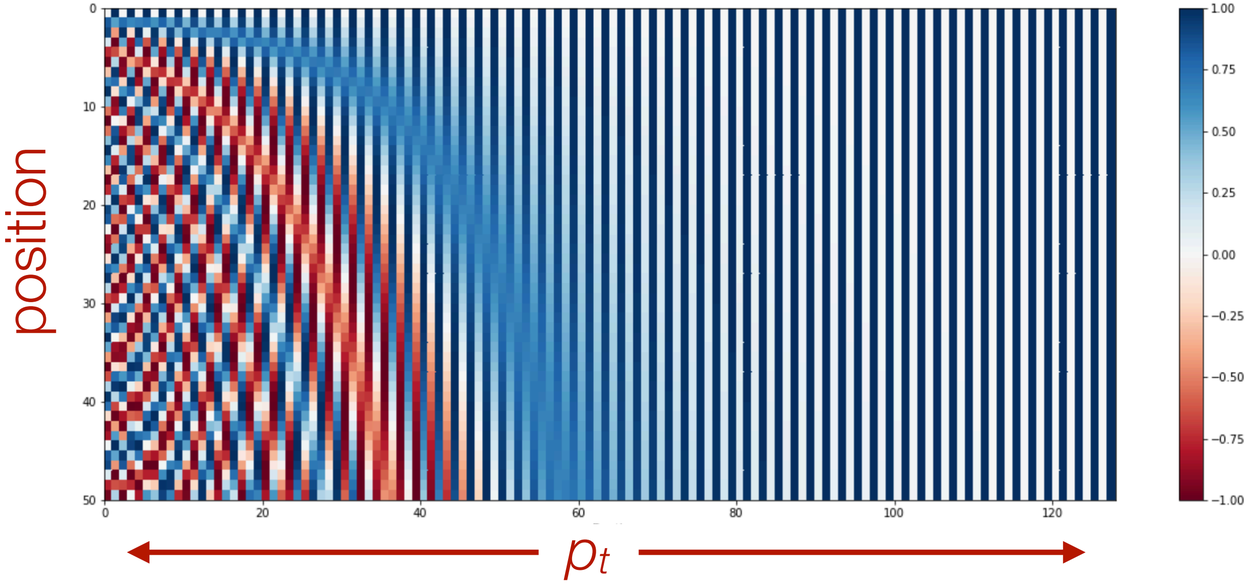
Positional encoding

- **Positional embeddings** encode a token's position in the sentence



- **Sinusoidal position encodings** generalize to any sequence length.

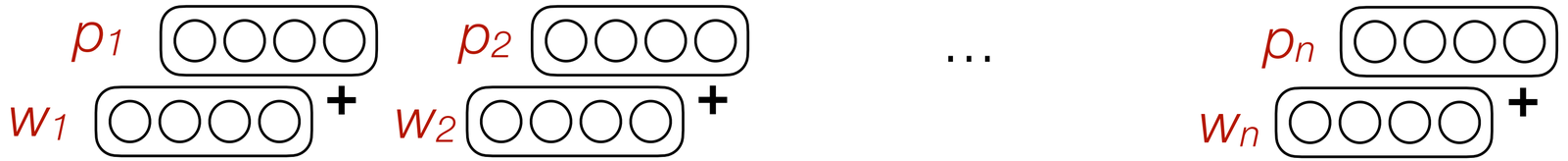
$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \quad \text{with} \quad \omega_k = \frac{1}{10000^{2k/d}}$$



$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

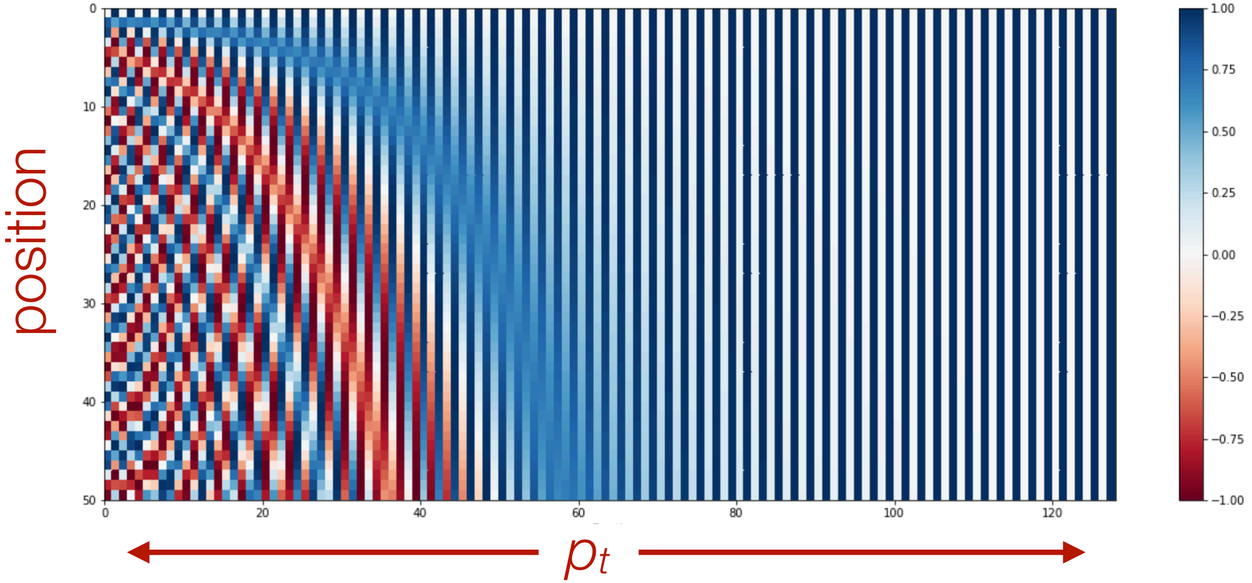
Positional encoding

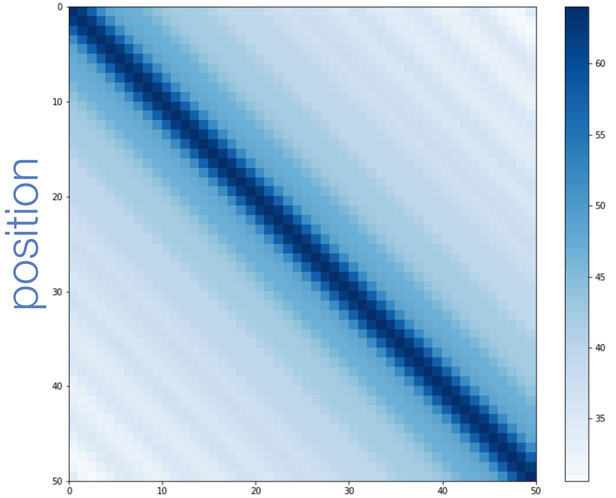
- **Positional embeddings** encode a token's position in the sentence



- **Sinusoidal position encodings** generalize to any sequence length.

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases} \quad \text{with} \quad \omega_k = \frac{1}{10000^{2k/d}}$$





$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Sub-word representations

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

BPE [Gage, 1994; Sennrich et al., 2016]

Algorithm 1 Byte-pair encoding

```
Input: set of strings D, target vocab size k
procedure BPE(D, k)
    V ← all unique characters in D
          (about 4,000 in English Wikipedia)
    while |V| < k do                    ▷ Merge tokens
        t_L, t_R ← Most frequent bigram in D
        t_NEW ← t_L + t_R               ▷ Make new token
        V ← V + [t_NEW]
        Replace each occurrence of t_L, t_R in
            D with t_NEW
    end while
    return V
end procedure
```

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

BPE [Gage, 1994; Sennrich et al., 2016]

Algorithm 1 Byte-pair encoding

Input: set of strings D, target vocab size k
procedure BPE(D, k)
  $V \leftarrow$ all unique characters in D
    (about 4,000 in English Wikipedia)
  while $|V| < k$ **do** ▷ Merge tokens
    $t_L, t_R \leftarrow$ Most frequent bigram in D
    $t_{\text{NEW}} \leftarrow t_L + t_R$ ▷ Make new token
    $V \leftarrow V + [t_{\text{NEW}}]$
    Replace each occurrence of t_L, t_R in
      D with t_{NEW}
  end while
  return V
end procedure

unigram LM [Kudo, 2018]

Algorithm 2 Unigram language modeling

Input: set of strings D, target vocab size k
procedure UNIGRAMLM(D, k)
  $V \leftarrow$ all substrings occurring more than
    once in D (not crossing words)
  while $|V| > k$ **do** ▷ Prune tokens
    Fit unigram LM θ to D
    for $t \in V$ **do** ▷ Estimate token 'loss'
      $L_t \leftarrow p_\theta(D) - p_{\theta'}(D)$
      where θ' is the LM without token t
    end for
    Remove $\min(|V| - k, \lfloor \alpha |V| \rfloor)$ of the
    tokens t with highest L_t from V,
    where $\alpha \in [0, 1]$ is a hyperparameter
  end while
  Fit final unigram LM θ to D
  return V, θ
end procedure

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

Original:	furiously
BPE:	_fur | iously
Unigram LM:	_fur | ious | ly

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

Original:	furiously
BPE:	_fur | iously
Unigram LM:	_fur | ious | ly

Original:	tricycles
BPE:	_t | ric | y | cles
Unigram LM:	_tri | cycle | s

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.

Original:	furiously
BPE:	_fur | iously
Unigram LM:	_fur | ious | ly

Original:	tricycles
BPE:	_t | ric | y | cles
Unigram LM:	_tri | cycle | s

Original:	Completely preposterous suggestions
BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
Unigram LM:	_Complete | ly | _pre | post | er | ous | _suggestion | s

Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

Sub-word representations

- Word embeddings suffer from UNK/OOV, but character embeddings are slow.
- Solution: **sub-word units**.
- BPE and unigram LM both implemented in the SentencePiece library

Original:	furiously
BPE:	_fur | iously
Unigram LM:	_fur | ious | ly

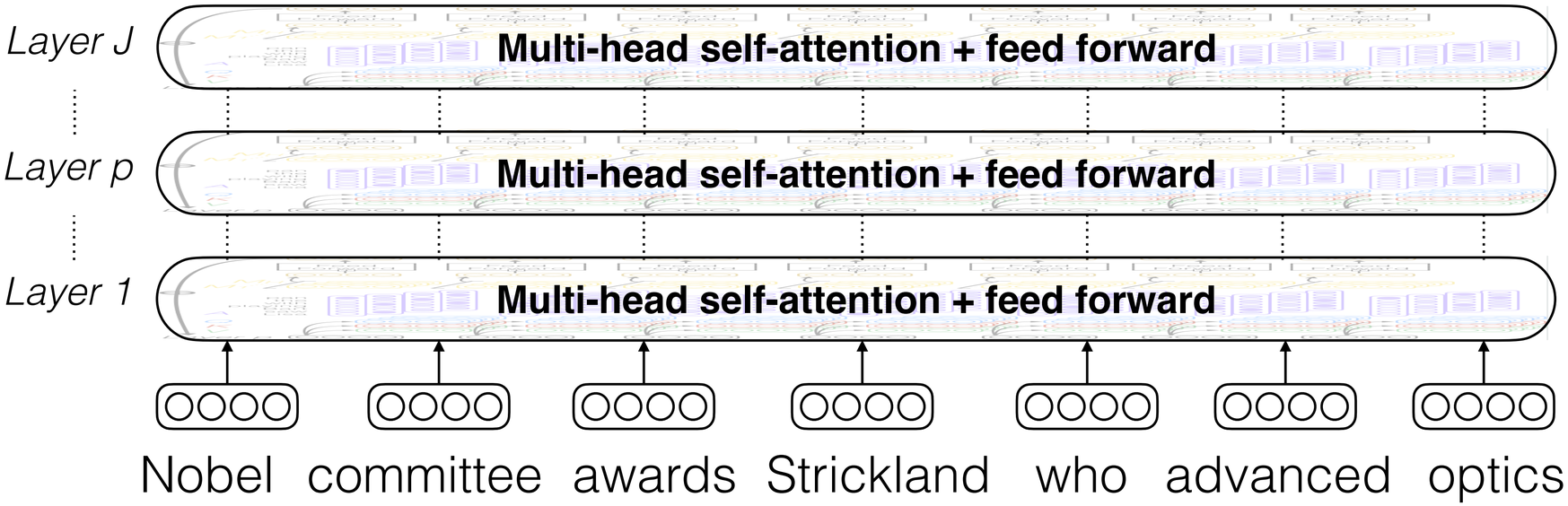
Original:	tricycles
BPE:	_t | ric | y | cles
Unigram LM:	_tri | cycle | s

Original:	Completely preposterous suggestions
BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
Unigram LM:	_Complete | ly | _pre | post | er | ous | _suggestion | s

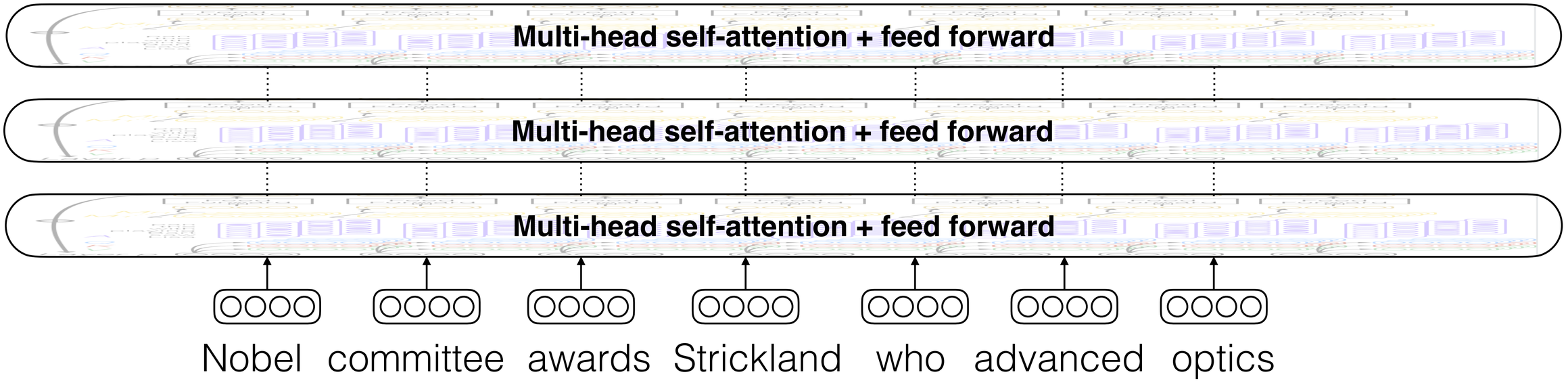
Figures from: Bostrom and Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. arXiv preprint 2004.03720. 2020.

BERT

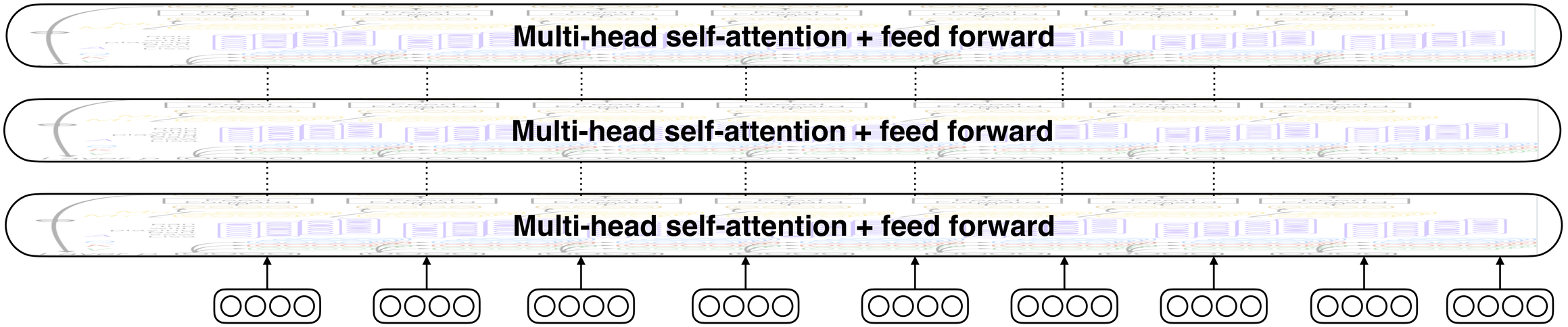
- Transformer language model trained with **non-autoregressive** masked language modeling (MLM) objective.



BERT

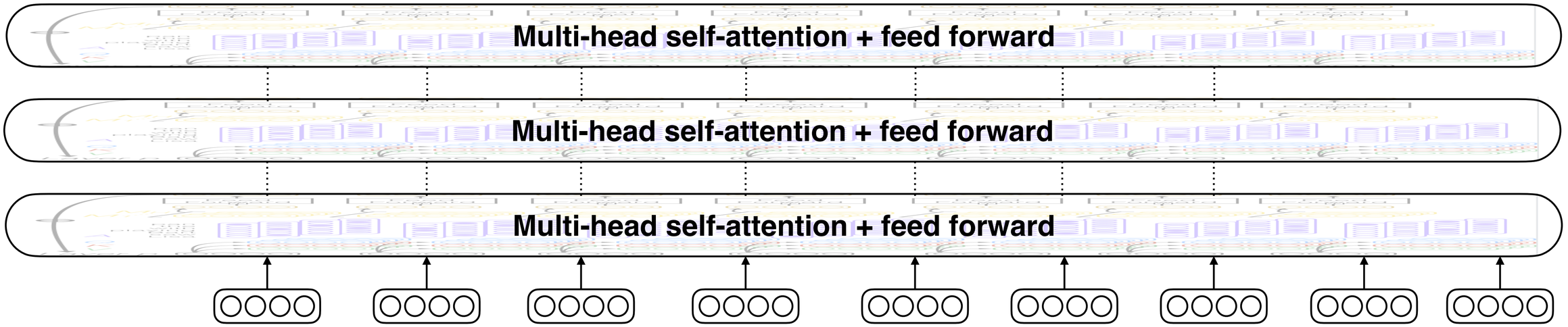

Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
Nobel
committee
awards
Strickland
who
advanced
optics

BERT



input: two 128-512 token sequences

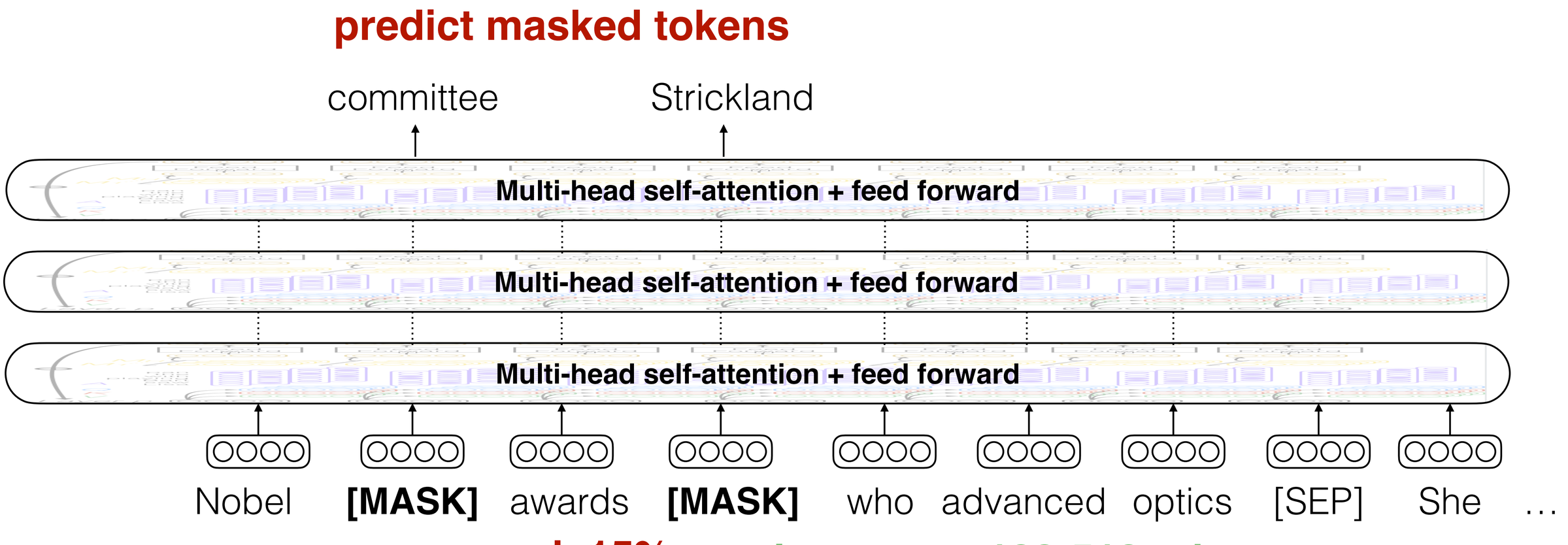
BERT



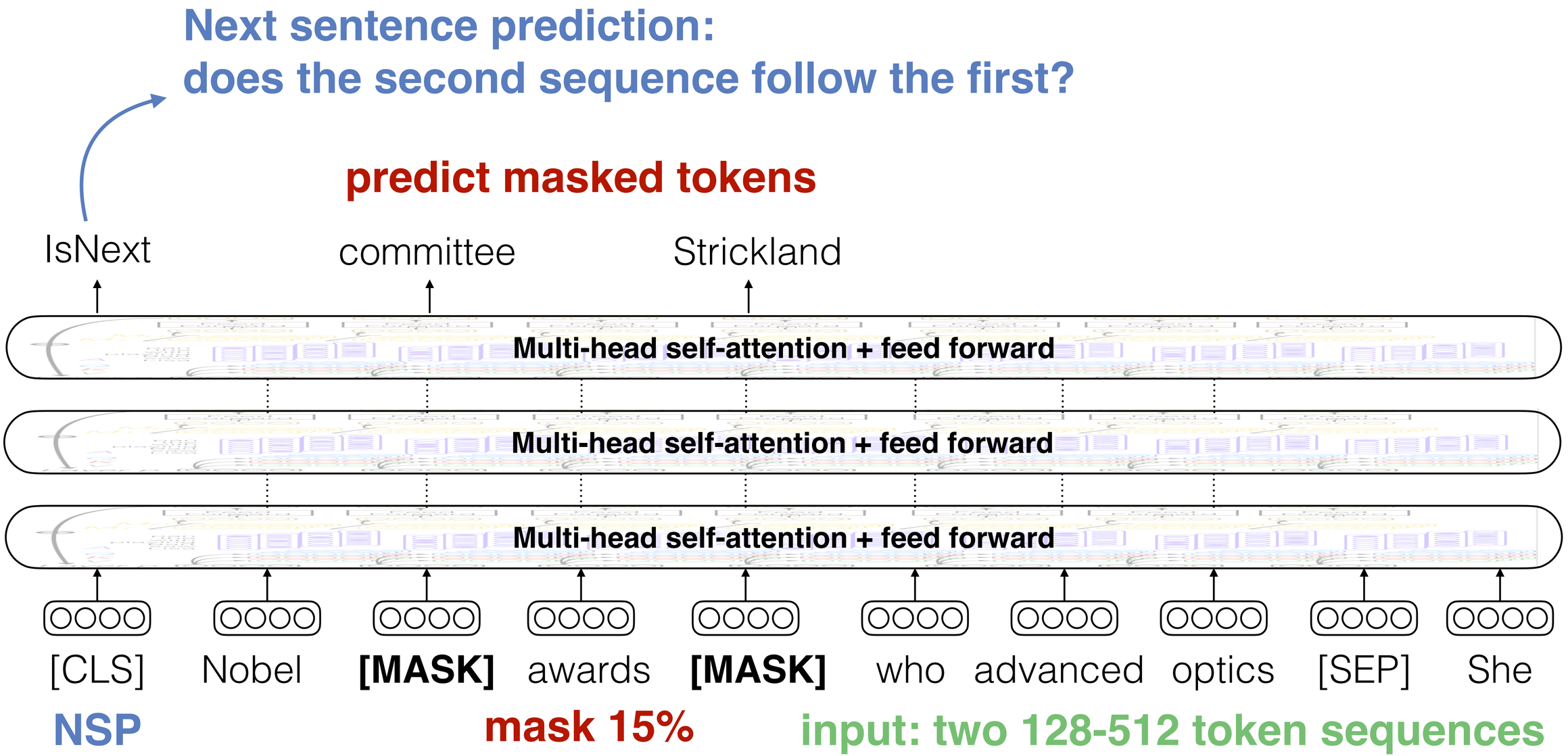
mask 15%

input: two 128-512 token sequences

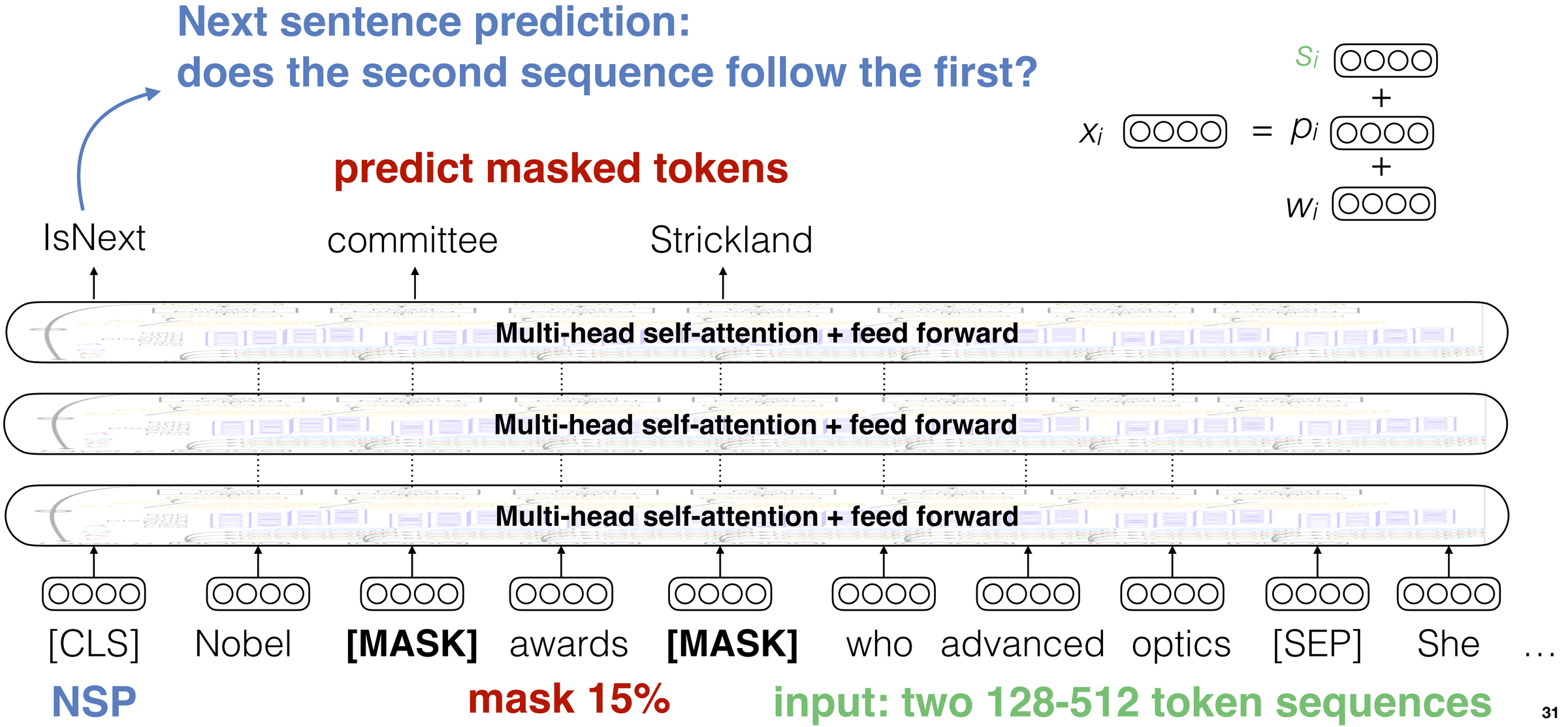
BERT


predict masked tokens
committee
Strickland
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
Nobel
[MASK]
awards
[MASK]
who
advanced
optics
[SEP]
She
…
mask 15%
input: two 128-512 token sequences

BERT

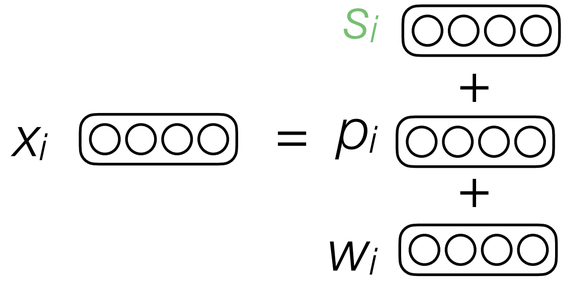

Next sentence prediction:
does the second sequence follow the first?
predict masked tokens
IsNext
committee
Strickland
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
[CLS]
Nobel
[MASK]
awards
[MASK]
who
advanced
optics
[SEP]
She
…
NSP
mask 15%
input: two 128-512 token sequences

BERT


Next sentence prediction:
does the second sequence follow the first?
predict masked tokens
x_i = p_i + w_i + s_i
IsNext
committee
Strickland
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
Multi-head self-attention + feed forward
[CLS]
Nobel
[MASK]
awards
[MASK]
who
advanced
optics
[SEP]
She
…
NSP
mask 15%
input: two 128-512 token sequences

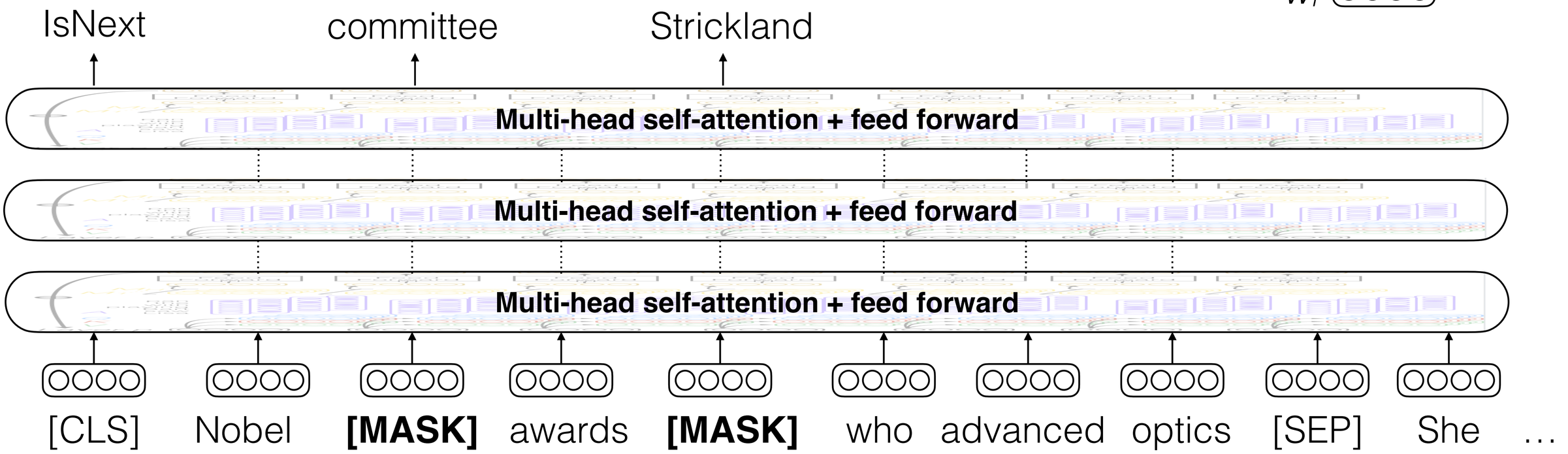
BERT

s_i: which sequence is this token from?



Next sentence prediction: does the second sequence follow the first?

predict masked tokens



NSP

mask 15%

input: two 128-512 token sequences

Evaluating pretrained LMs

Evaluating pretrained LMs

- Intrinsic (perplexity) and extrinsic (fine-tuning on downstream task) evaluations.

Evaluating pretrained LMs

- Intrinsic (perplexity) and extrinsic (fine-tuning on downstream task) evaluations.
- Standard extrinsic benchmarks for LM pretraining: **(Super)GLUE** and **SQuAD**

Evaluating pretrained LMs

- Intrinsic (perplexity) and extrinsic (fine-tuning on downstream task) evaluations.
- Standard extrinsic benchmarks for LM pretraining: **(Super)GLUE** and **SQuAD**

GLUE: https://gluebenchmark.com/
General Language Understanding Evaluation

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

Evaluating pretrained LMs

- Intrinsic (perplexity) and extrinsic (fine-tuning on downstream task) evaluations.
- Standard extrinsic benchmarks for LM pretraining: **(Super)GLUE** and **SQuAD**

GLUE: https://gluebenchmark.com/
General Language Understanding Evaluation

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

SQuAD: https://rajpurkar.github.io/SQuAD-explorer/
Stanford Question Answering Dataset

The Yuan dynasty (Chinese: 元朝; pinyin: Yuán Cháo), officially the Great Yuan (Chinese: 大元; pinyin: Dà Yuán; Mongolian: Yehe Yuan Ulus[a]), was the empire or ruling dynasty of China established by Kublai Khan, leader of the Mongolian Borjigin clan. Although the Mongols had ruled territories including today's North China for decades, it was not until 1271 that Kublai Khan officially proclaimed the dynasty in the traditional Chinese style. His realm was, by this point, isolated from the other khanates and controlled most of present-day China and its surrounding areas, including modern Mongolia and Korea. It was the first foreign dynasty to rule all of China and lasted until 1368, after which its Genghisid rulers returned to their Mongolian homeland and continued to rule the Northern Yuan dynasty. Some of the Mongolian Emperors of the Yuan mastered the Chinese language, while others only used their native language (i.e. Mongolian) and the 'Phags-pa script.

What is the Chinese name for the Yuan dynasty?
Ground Truth Answers: Yuán Cháo | Yuán Cháo | 元朝

When did Khan formally declare the Yuan dynasty?
Ground Truth Answers: 1271 | 1271 | 1271

Fine-tuning pretrained LMs

- Unlike in ELMo, all model parameters are fine-tuned (receive gradient updates w.r.t. supervised downstream task.)

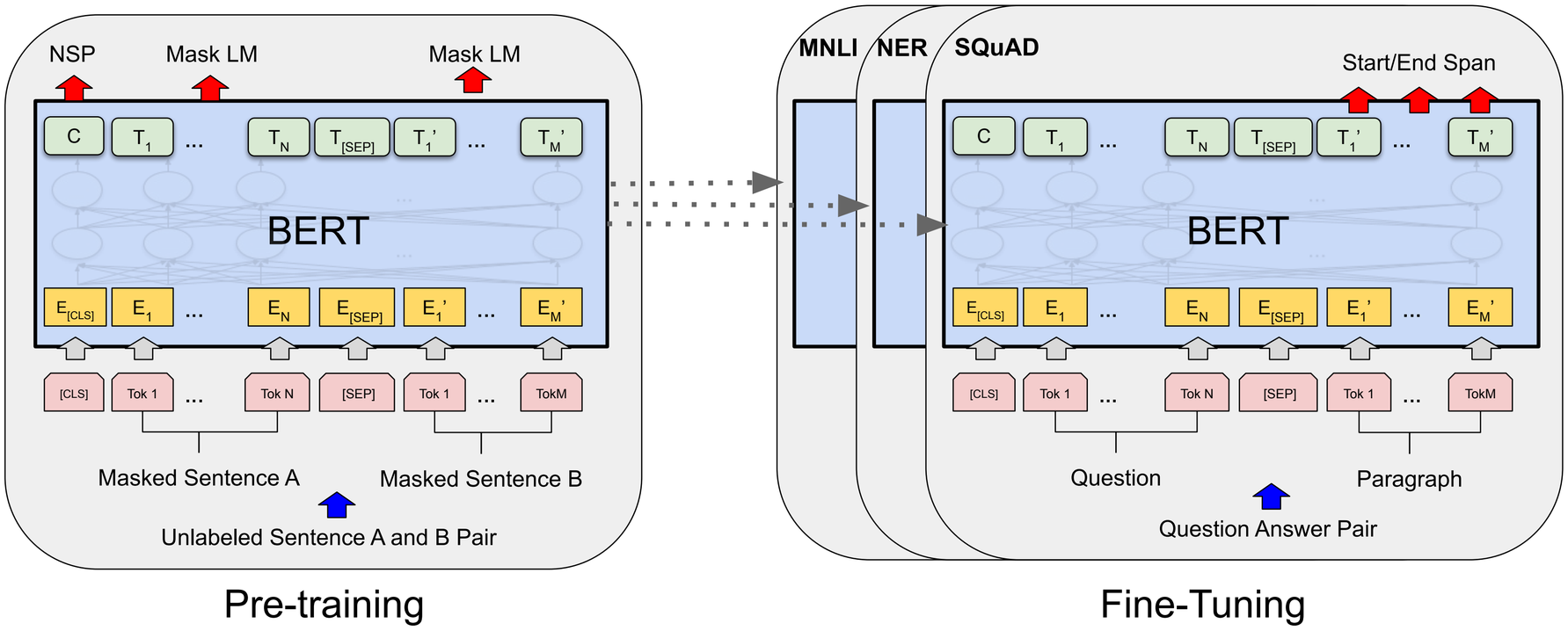


Figure from: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL 2019.

Fine-tuning pretrained LMs

- **Sentence classification:**
  - Provide single sentence as input
  - Use final representation at CLS position as features for a linear classifier (e.g. logistic regression).

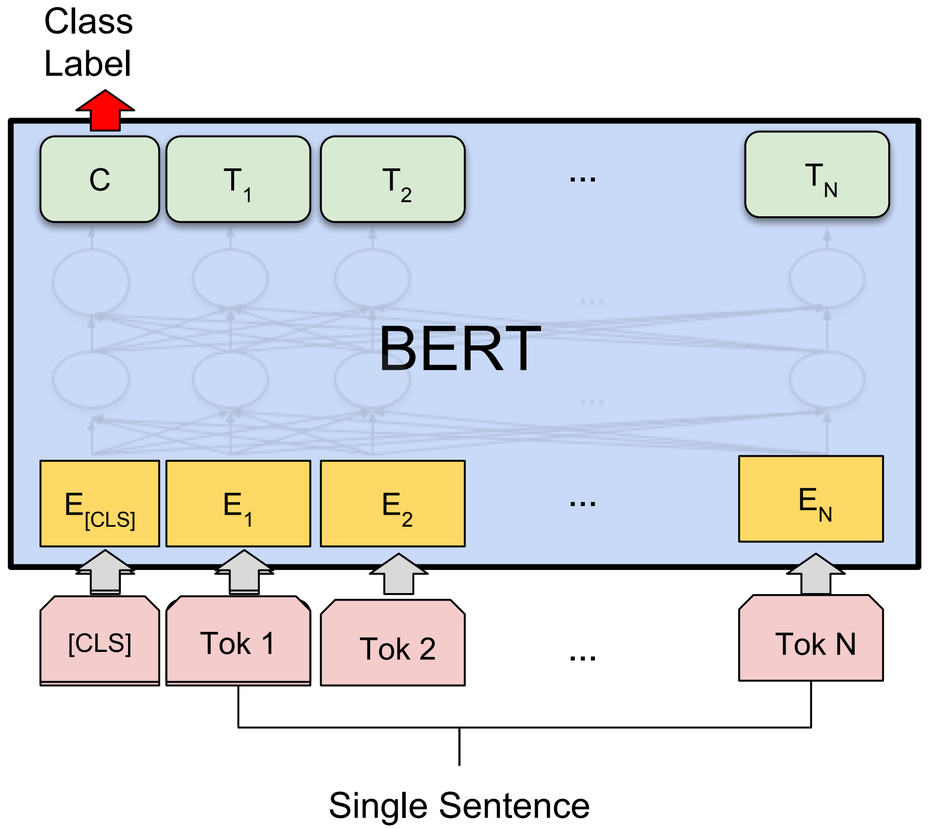


Figure from: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL 2019.

Fine-tuning pretrained LMs

- **Sentence pair classification:**
  - Provide both sentences as input, separated by SEP token.
  - Input embeddings indicate sentence identity.
  - Use final representation at CLS position as features for a linear classifier (e.g. logistic regression).

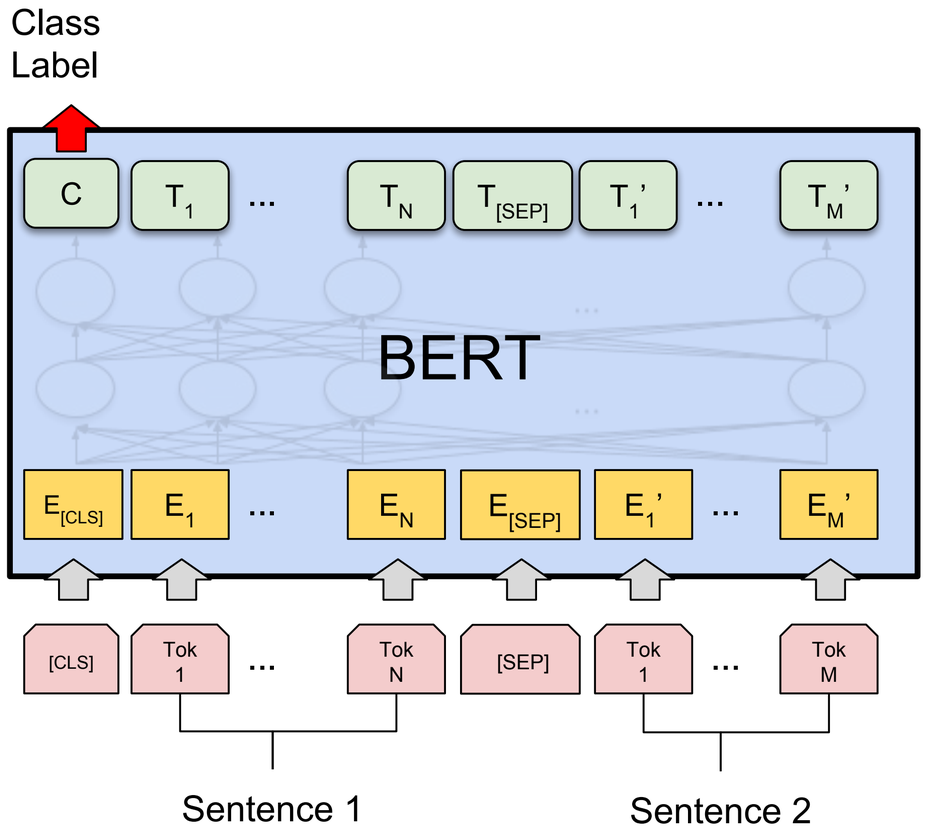


Figure from: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL 2019.

Fine-tuning pretrained LMs

- **Sequence labeling:**
  - Use final representations at each timestep as features for a linear classifier (e.g. logistic regression).
  - Need to deal with sub-tokens.

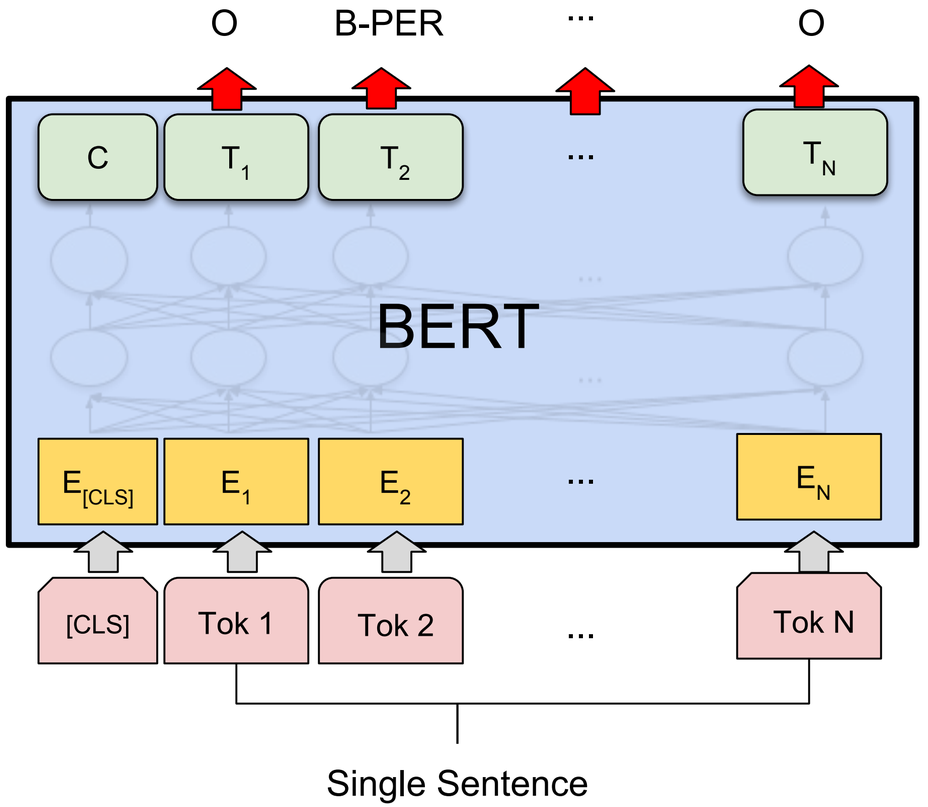


Figure from: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL 2019.

Fine-tuning pretrained LMs

- **Question answering:**
  - Use final representations at each timestep as features to score every token as start or end of a span. Take the highest-scoring pair.

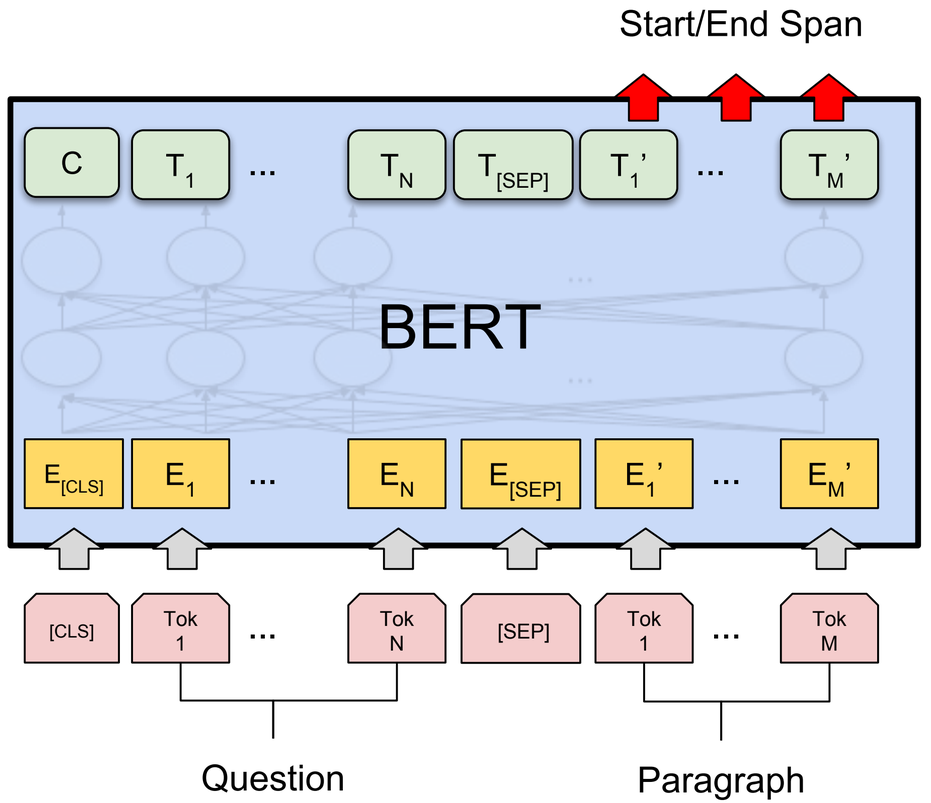
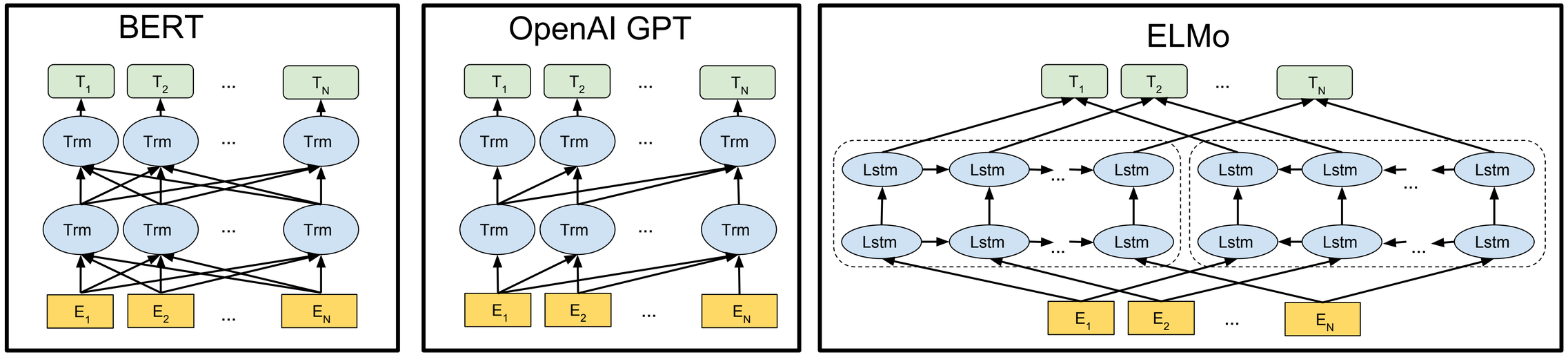


Figure from: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL 2019.

Other popular Transformer LMs



- GPT/XLNet
- RoBERTa
- ELECTRA

ELECTRA [Clark et al. 2020]

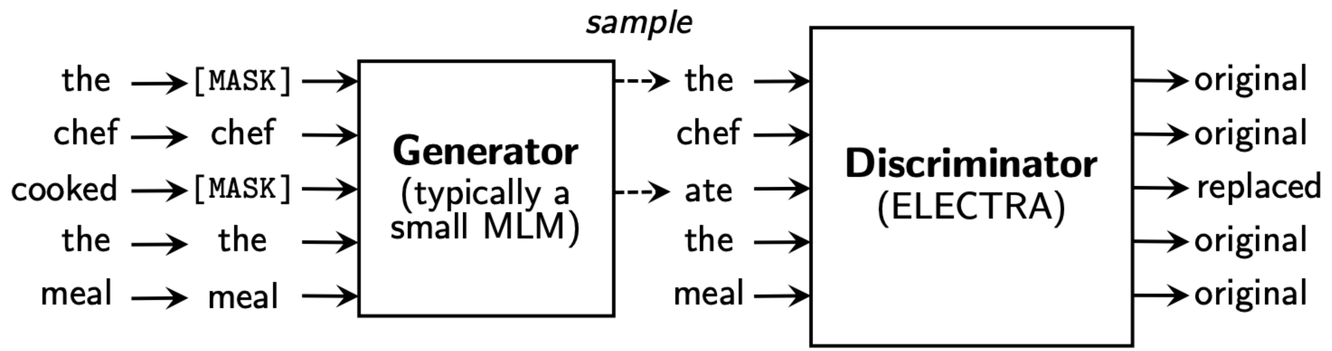


Figure from: Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL 2019.

How does performance scale with data?

How does performance scale with data?

RoBERTa
(Liu et al. 2019)

Training data

Dataset	16GB	160GB	Δ	err reduction
SQuAD	87.3	89.4	2.1	16.5%
MNLI-m	89.0	90.2	1.2	10.9%
SST-2	95.3	96.4	0.9	19.1%

XLNet
(Yang et al. 2019)

~3B

Dataset	13GB	126GB	Δ	err reduction
SQuAD	87.8	88.8	1.0	8.2%
MNLI-m	88.4	89.8	1.4	12.1%
SST-2	94.4	95.6	1.2	21.4%

T5
(Raffel et al. 2020)

~8M **Training tokens** ~34B

Dataset	2^{23}	2^{25}	2^{27}	2^{29}	2^{35}
SQuAD	70.92	76.27	79.78	80.97	80.88
GLUE	76.34	79.55	82.62	82.87	83.28
SGLUE	59.29	64.76	69.97	72.03	71.36

Data *source* effects performance

English

news wikipedia fiction*

European parliament

web crawl

English

books

Dataset	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

Table 8: Performance resulting from pre-training on different datasets. The first four variants are based on our new C4 dataset.

T5 [Raffel et al. 2020]

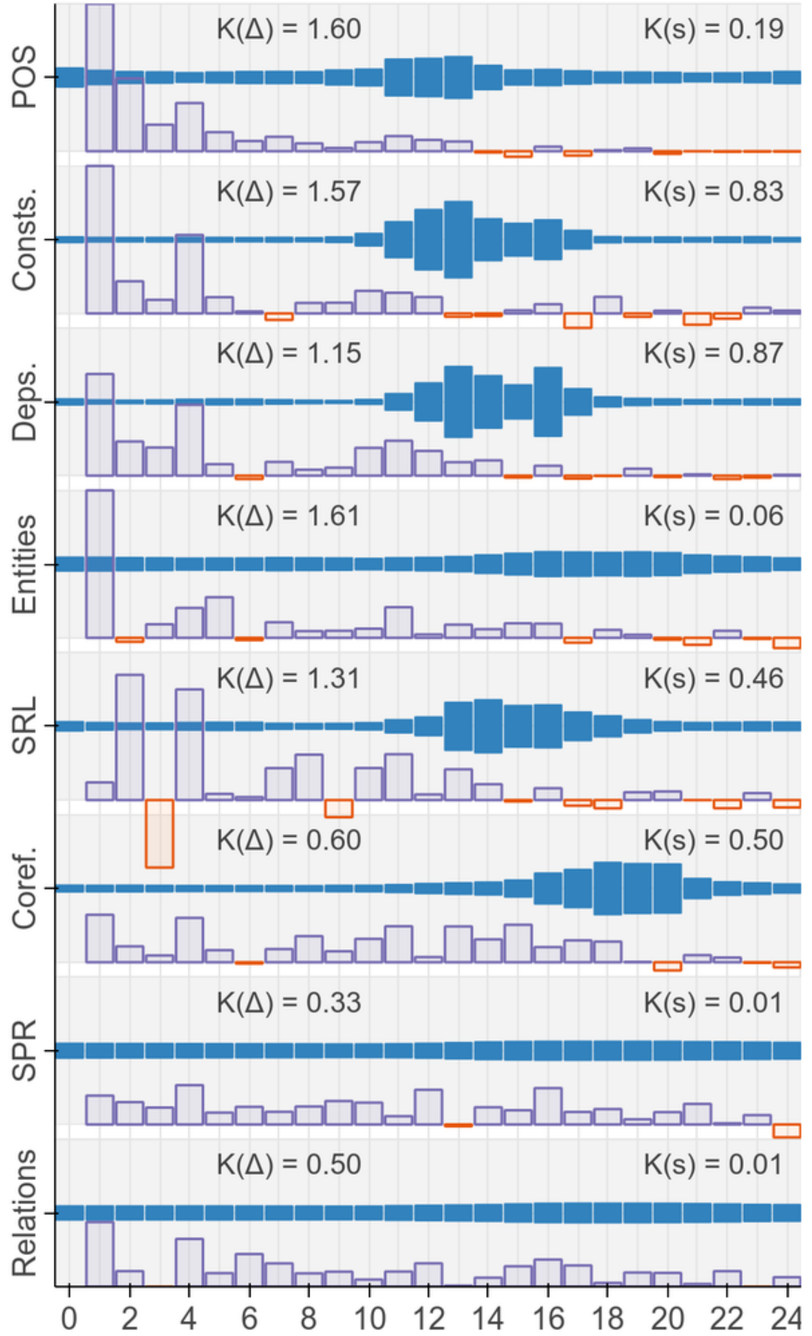
Data *source* effects performance

Field	Task	Dataset	SOTA	BERT-Base		SCIBERT	
				Frozen	Finetune	Frozen	Finetune
Bio	NER	BC5CDR (Li et al., 2016)	88.85[7]	85.08	86.72	88.73	**90.01**
		JNLPBA (Collier and Kim, 2004)	**78.58**	74.05	76.09	75.77	77.28
		NCBI-disease (Dogan et al., 2014)	**89.36**	84.06	86.88	86.39	88.57
	PICO	EBM-NLP (Nye et al., 2018)	66.30	61.44	71.53	68.30	**72.28**
	DEP	GENIA (Kim et al., 2003) - LAS	**91.92**	90.22	90.33	90.36	90.43
		GENIA (Kim et al., 2003) - UAS	**92.84**	91.84	91.89	92.00	91.99
	REL	ChemProt (Kringelum et al., 2016)	76.68	68.21	79.14	75.03	**83.64**
CS	NER	SciERC (Luan et al., 2018)	64.20	63.58	65.24	65.77	**67.57**
	REL	SciERC (Luan et al., 2018)	n/a	72.74	78.71	75.25	**79.97**
	CLS	ACL-ARC (Jurgens et al., 2018)	67.9	62.04	63.91	60.74	**70.98**
Multi	CLS	Paper Field	n/a	63.64	65.37	64.38	**65.71**
		SciCite (Cohan et al., 2019)	84.0	84.31	84.85	**85.42**	**85.49**
Average				73.58	77.16	76.01	79.27

SciBERT [Beltagy et al. 2020]

pre-training LM on in-domain data improves end-task performance

What does BERT know?



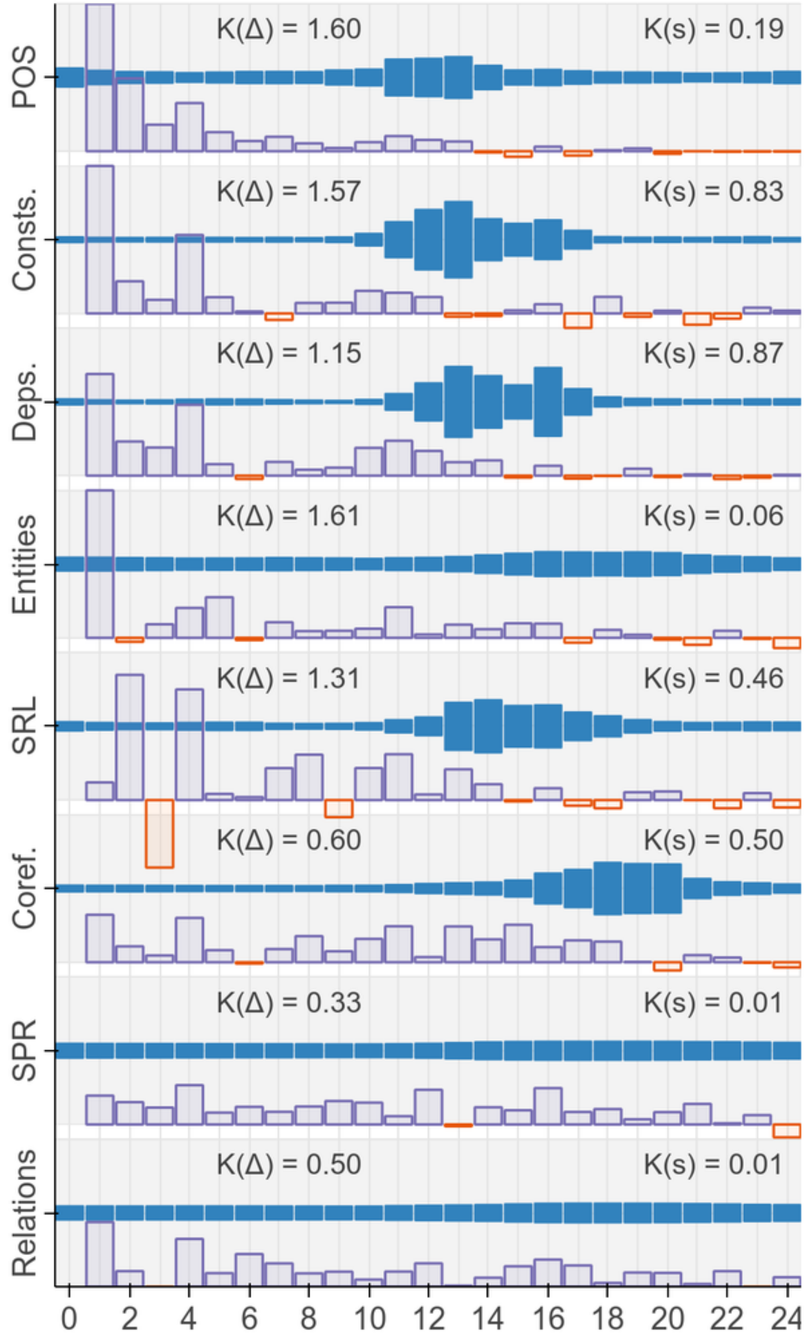
Tenney et al. BERT Rediscovers the Classical NLP Pipeline. ACL 2019.

Ettinger. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. TACL 2020.

What does BERT know?

Context	$\text{BERT}_{\text{LARGE}}$ predictions
A robin is a ____	*bird, robin, person, hunter, pigeon*
A daisy is a ____	*daisy, rose, flower, berry, tree*
A hammer is a ____	*hammer, tool, weapon, nail, device*
A hammer is an ____	*object, instrument, axe, implement, explosive*
A robin is not a ____	*robin, bird, penguin, man, fly*
A daisy is not a ____	*daisy, rose, flower, lily, cherry*
A hammer is not a ____	*hammer, weapon, tool, gun, rock*
A hammer is not an ____	*object, instrument, axe, animal, artifact*

Table 13: NEG-88-SIMP predictions by $\text{BERT}_{\text{LARGE}}$



Tenney et al. BERT Rediscovers the Classical NLP Pipeline. ACL 2019.

Ettinger. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. TACL 2020.

Announcements

- Share your writeup from P1 (bake-off design and paper summary) on Piazza!
- Friday's recitation will be an overview of P2.