# Transformers

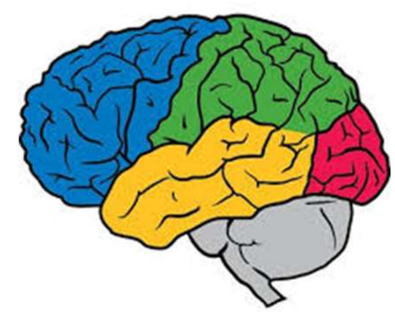
# Attention is All You Need
# (Vaswani et al., 2017)

Debut of the **Transformer** architecture.

The same model used in:

- BERT (Devlin, et al. 2018)
- LISA (Strubell, et al. 2018)
- and others…

Motto paraphrased: *No more RNNs, CNNs, just use Attention!*



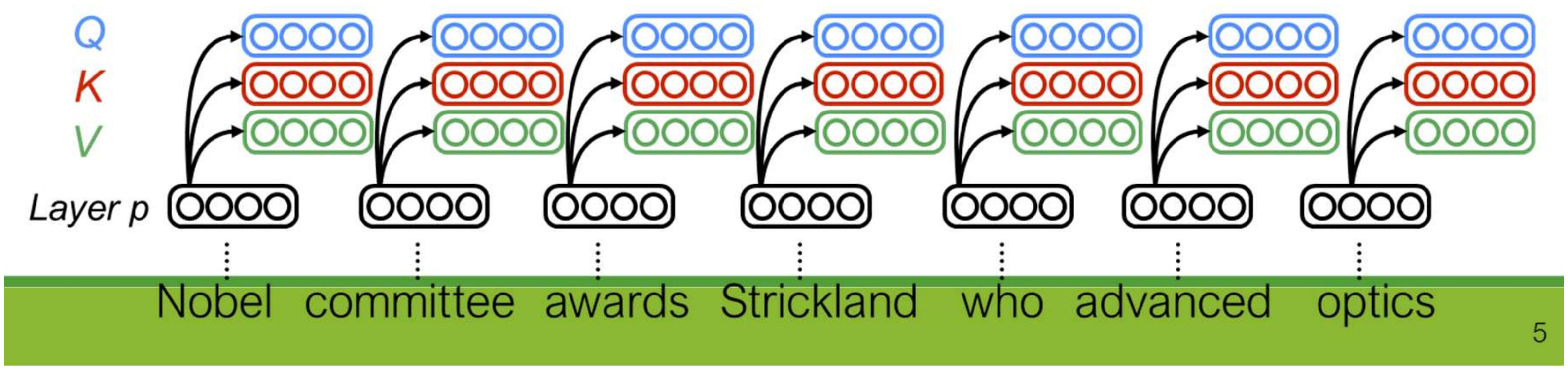
# Transformer: Self-Attention

query $Q$
key $K$
used to calculate attention

value $V$ — base representation of word

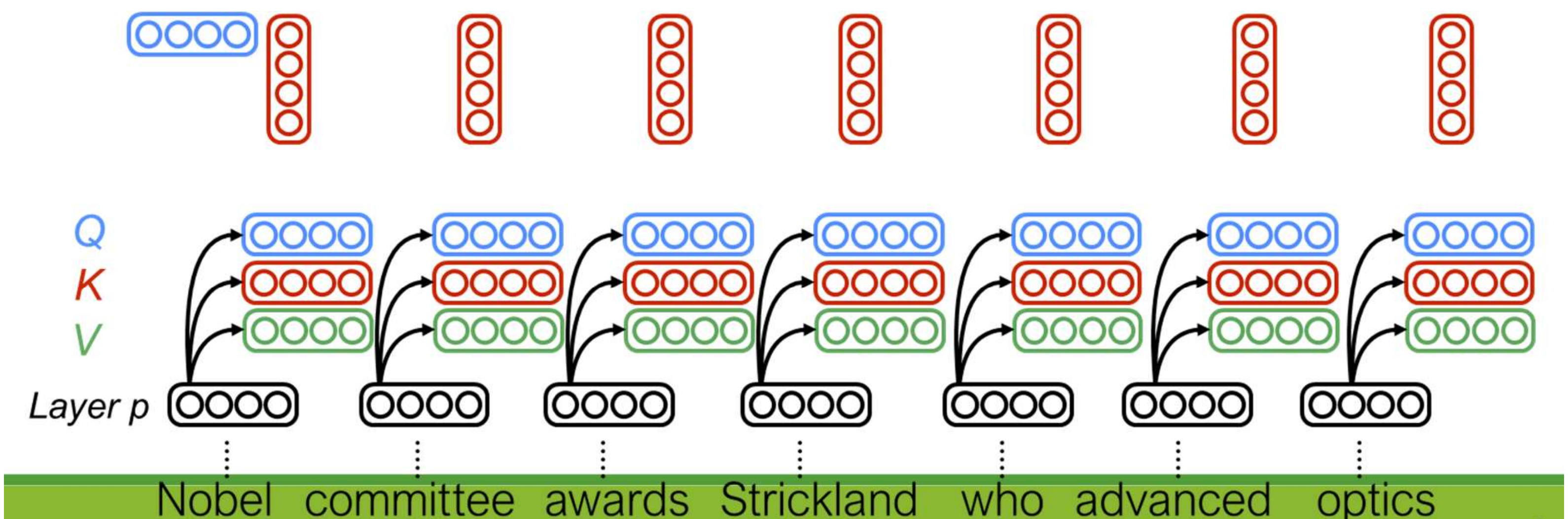
# Transformer: Self-Attention

Slide from Strubell, et al. (2018).



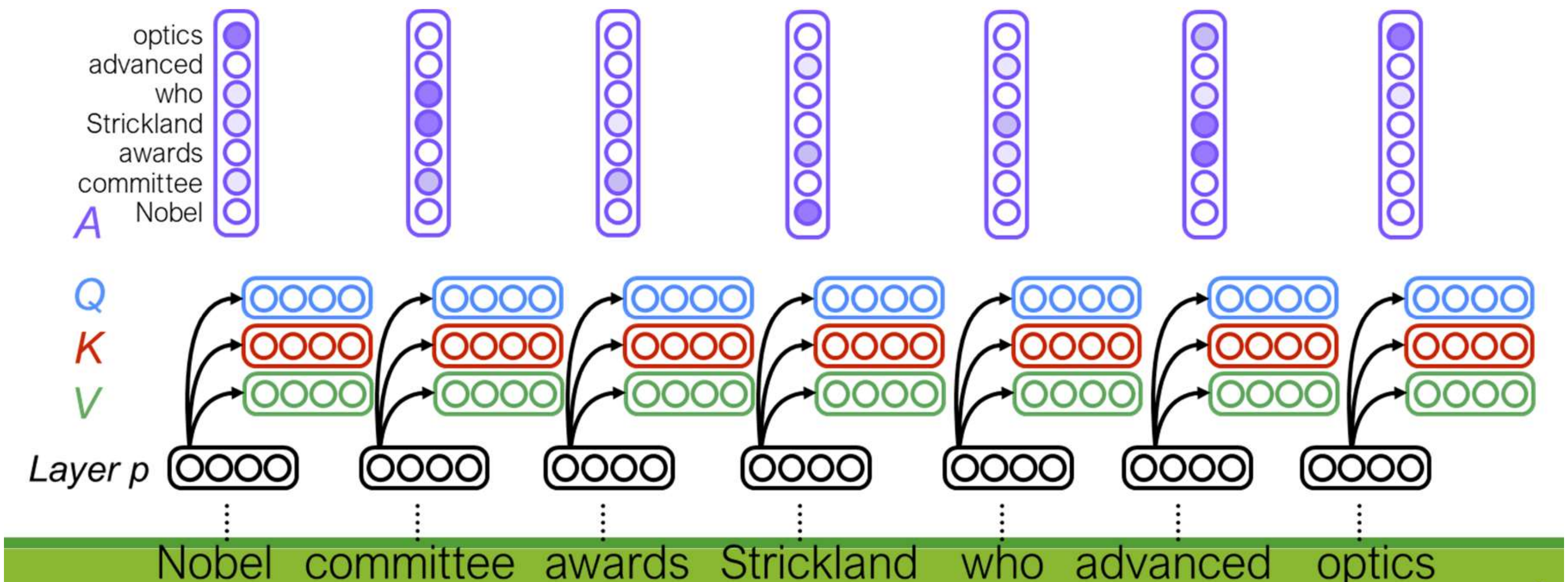
# Transformer: Self-Attention

Slide from Strubell, et al. (2018).



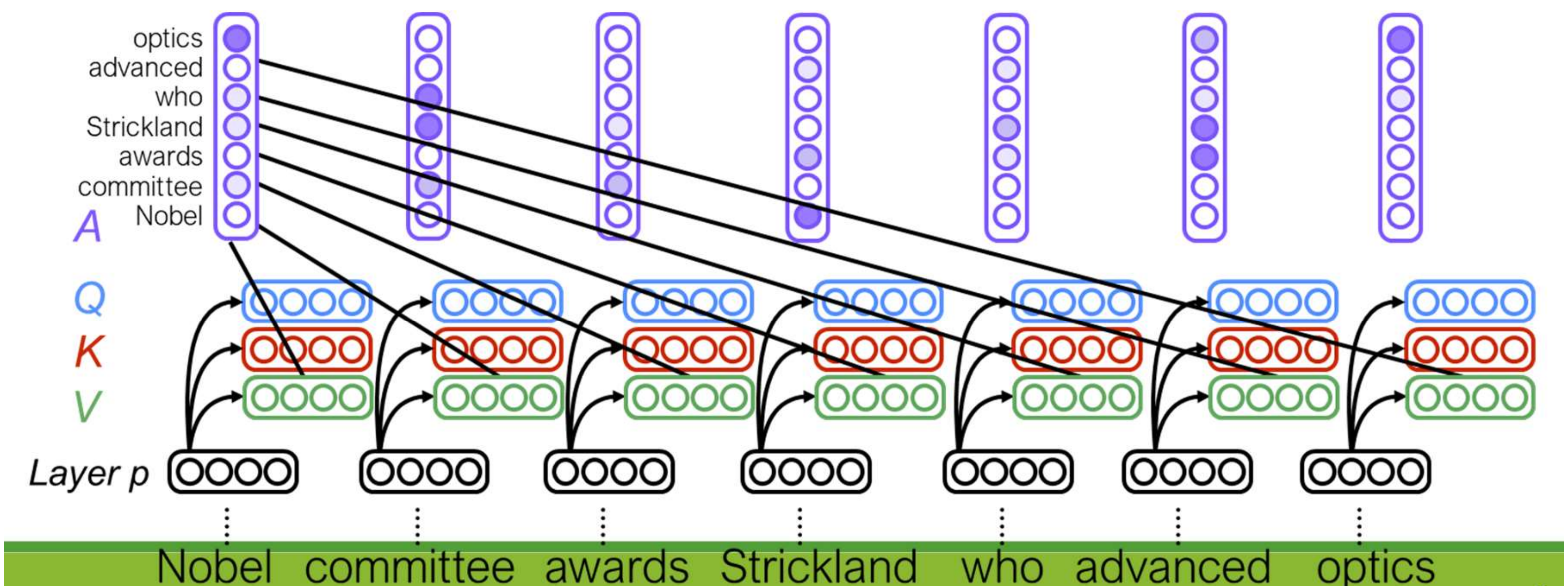
# Transformer: Self-Attention

Slide from Strubell, et al. (2018).



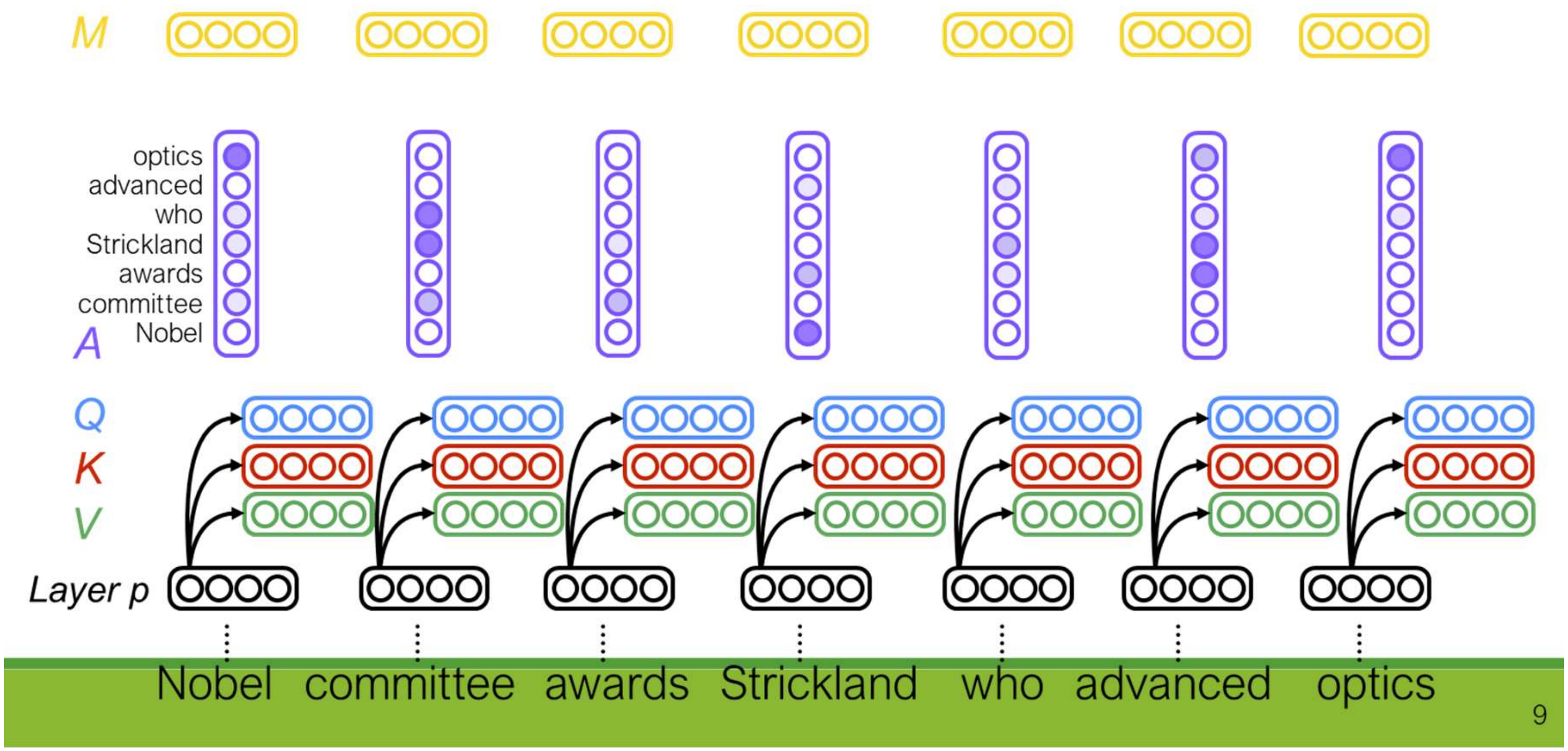
# Transformer: Self-Attention

Slide from Strubell, et al. (2018).



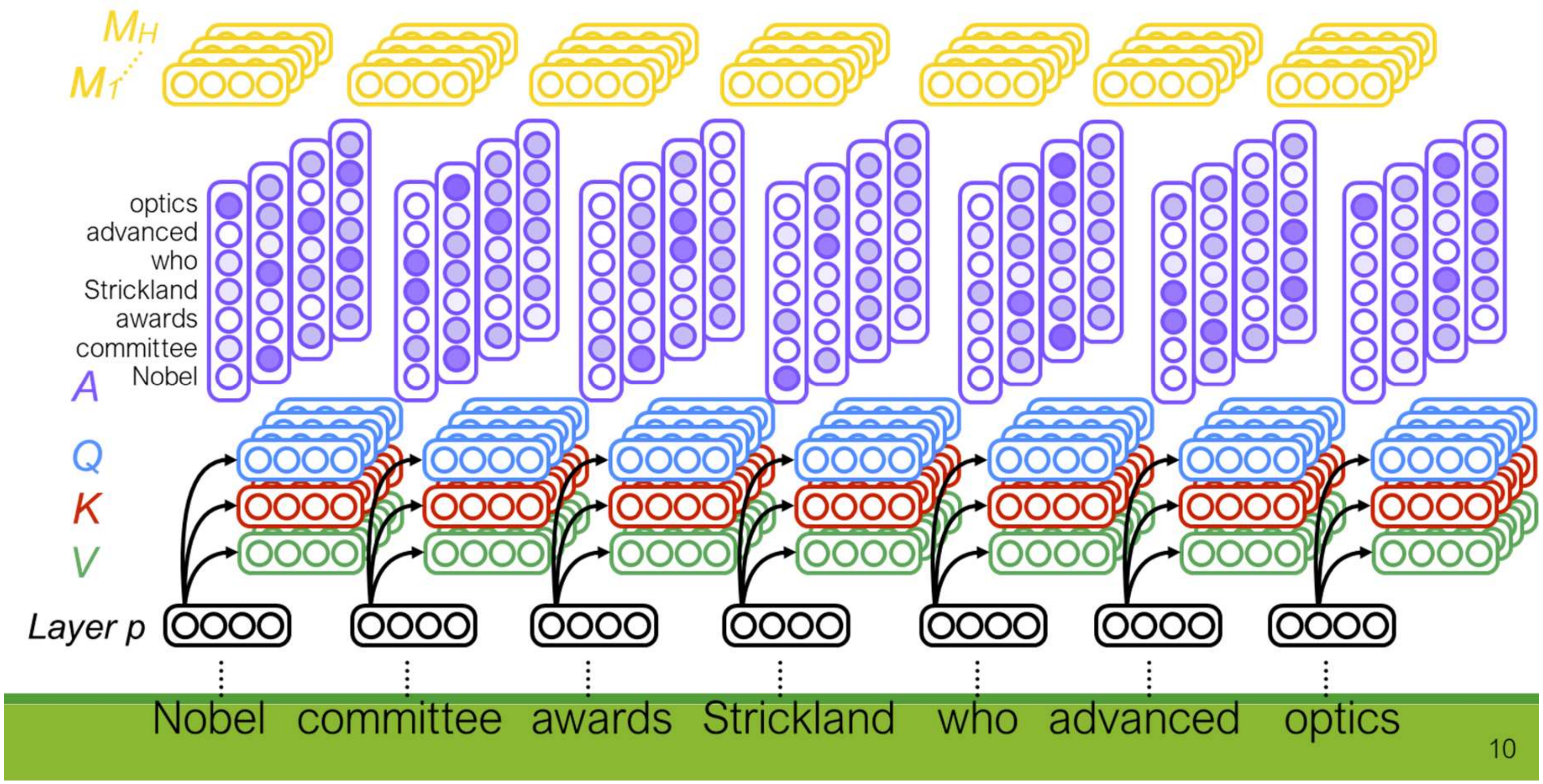
Slide from Strubell, et al. (2018).

# Transformer: Self-Attention



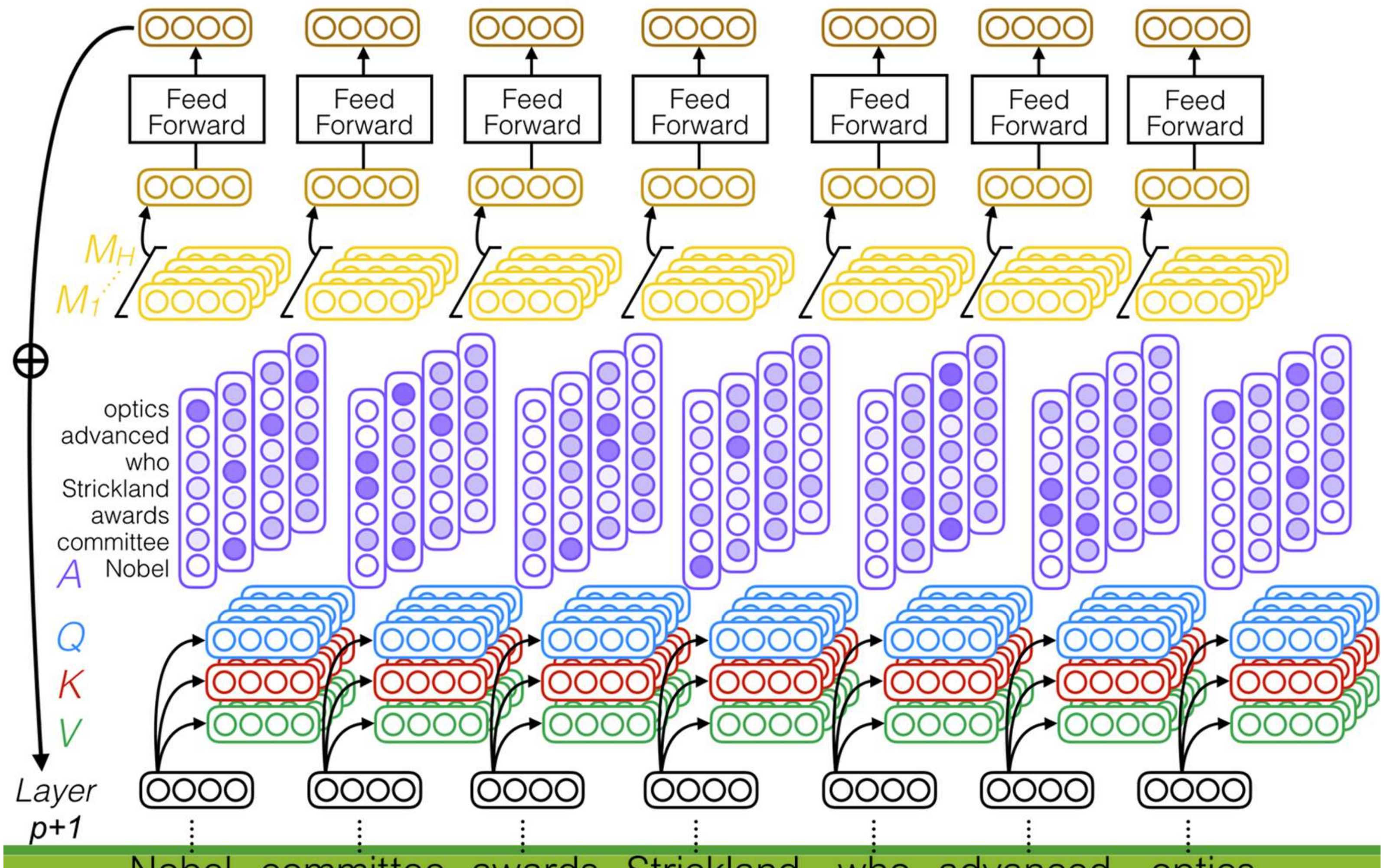
# Transformer: (Multi-head) Self-Attention

Slide from Strubell, et al. (2018).



# Transformer: (Multi-head) Self-Attention

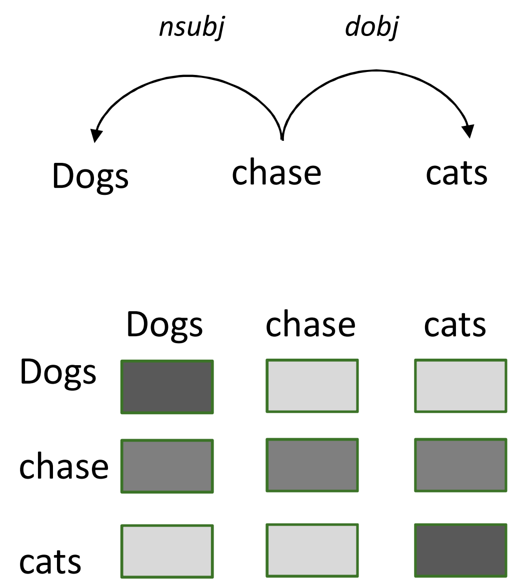
Slide from Strubell, et al. (2018).



# Linguistically Motivated

## Strengths

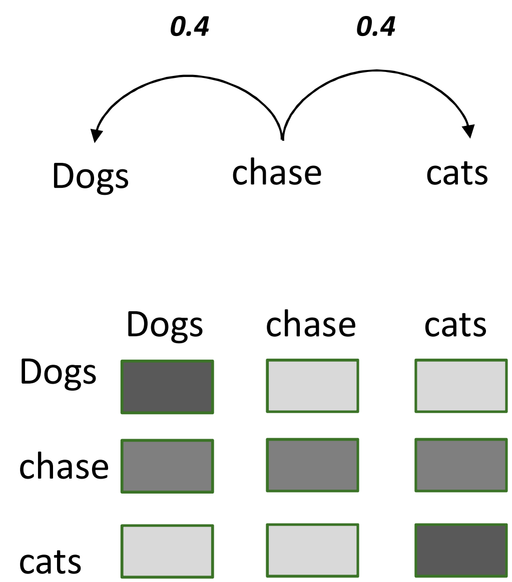
- Captures Long-distant dependencies!
- Intuitively: learns *weighted unlabeled dependencies*



# Linguistically Motivated

## Strengths

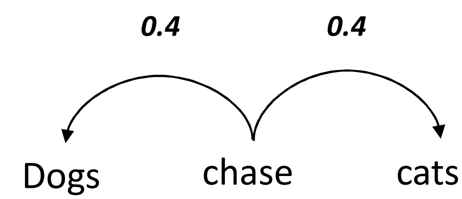
- Captures Long-distant dependencies!
- Intuitively: learns *weighted unlabeled dependencies*



# Linguistically Motivated

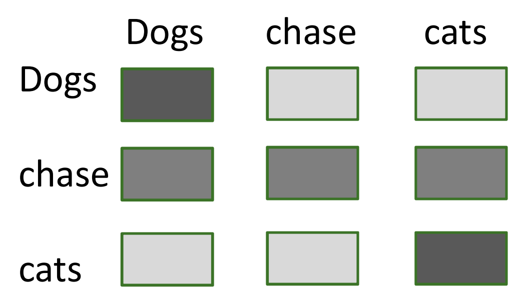
## Strengths

- Captures Long-distant dependencies!
- Intuitively: learns *weighted unlabeled dependencies*




## Weaknesses (addressed in next slides)

- Weak model of word order
- One layer can't distinguish dependencies
- No locality bias



# Positional Encoding

How to turn this into a sequence modal:

Add “positional encoding” as extra input.
Weak representation of position.

Possible Improvement: Unlike RNN and CNN,
No locality bias.

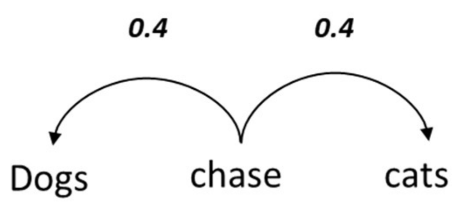
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

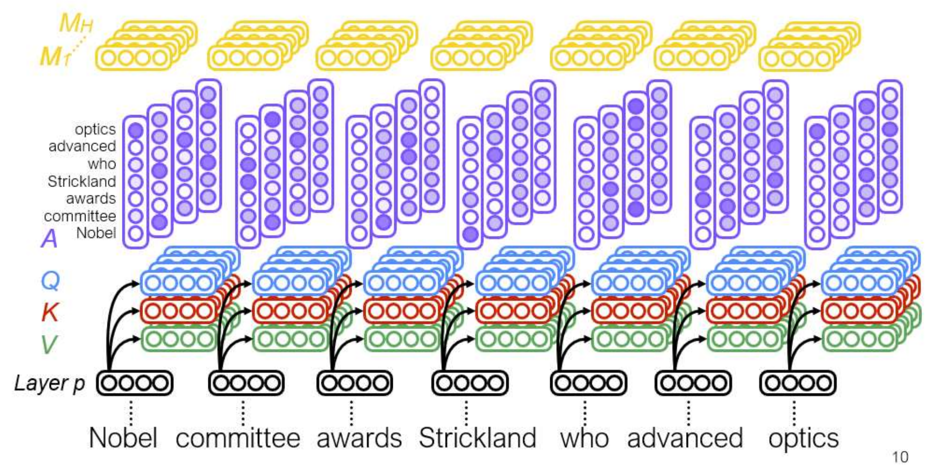
(pos is position, i is dimension)

# Multi-Head Attention

How to distinguish dependencies:

One attention layer can't distinguish two dependencies (subject vs. object).




Use multiple attention layers, hopefully one represents subject, one object, etc.

# Transformer Architecture

- Encode-Decoder with Transformers instead of RNNs
- Large improvement over LSTM encoder-decoder. Why?

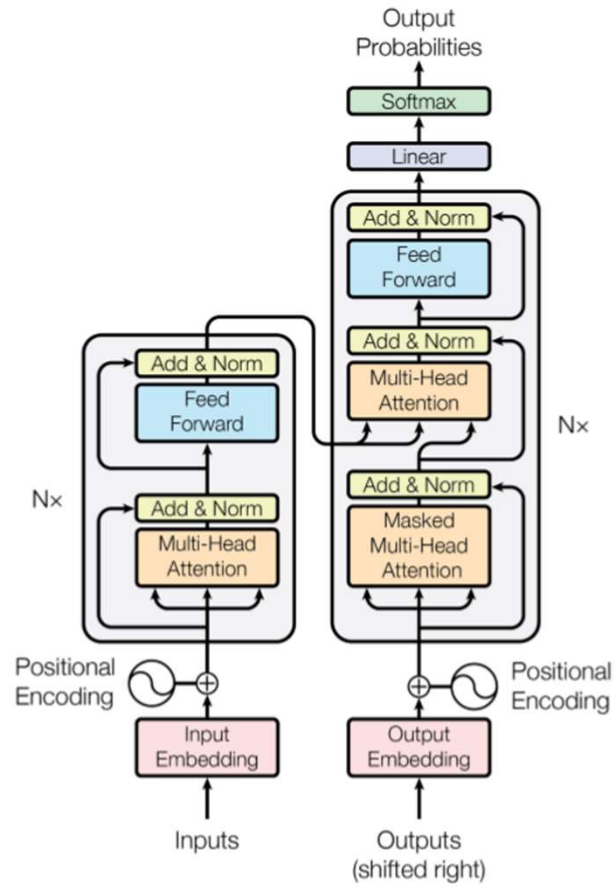



Figure 1: The Transformer - model architecture.

# Transformer Architecture

- Encode-Decoder with Transformers instead of RNNs
- Large improvement over LSTM encoder-decoder. Why?
  - Long-distance relations
  - better representation of syntax
  - faster to train (when using TPUs)

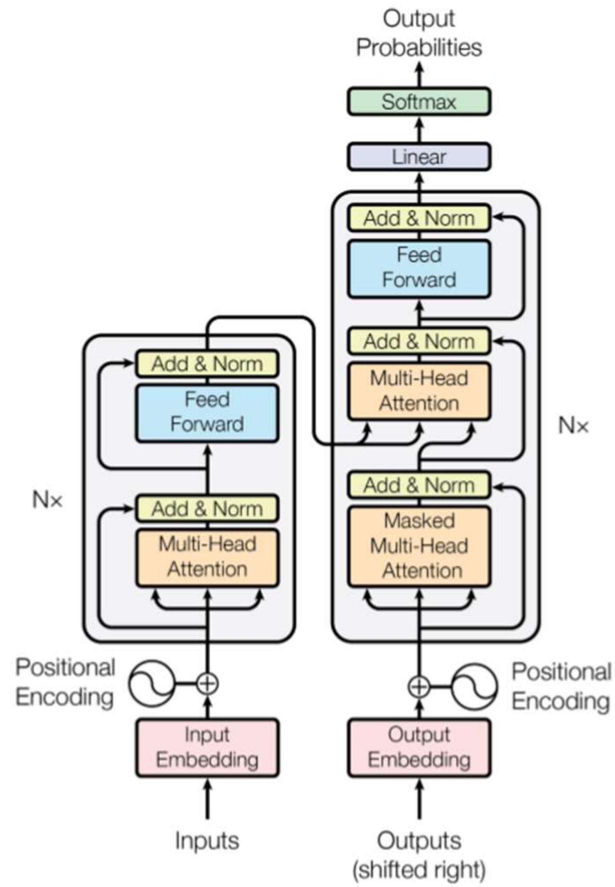



Figure 1: The Transformer - model architecture.

# Replicate, Extend, etc.

Tensorflow

https://github.com/tensorflow/tensor2tensor

Pytorch

https://github.com/jadore801120/attention-is-all-you-need-pytorch

Annotated Code

http://nlp.seas.harvard.edu/2018/04/03/attention.html

Illustrated Explanation

http://jalammar.github.io/illustrated-transformer/

# Transfer Learning: ELMo & BERT

# Why Transfer Learning

- “Free” increases in accuracy
- For some tasks, data is sparse or expensive (AMR, low resource languages, etc.)
- May capture information that is useful but not present in labelled data.
- Possibly closer to genuine linguistic representations.

# Review: Word Embeddings

**GloVe** is a collection of pretrained (static) word embeddings that can be plugged into your models.

Approximate semantic features: King - Man + Woman = Queen

Trained on millions of sentences

Can be “tuned”: your model can adjust GloVe features to be more useful for your task.

Pennington, J., Socher, R., & Manning, C. (2014). **Glove: Global vectors for word representation**. In *Proceedings of the 2014 conference on EMNLP.*

# ELMo: Deep contextualized word representations (Peters et al., 2018)

**E**mbeddings from **L**anguage **Mo**dels (ELMo)

Word vector representations that is a function of input sentence.

Based on biLSTM language model.

# Motivation

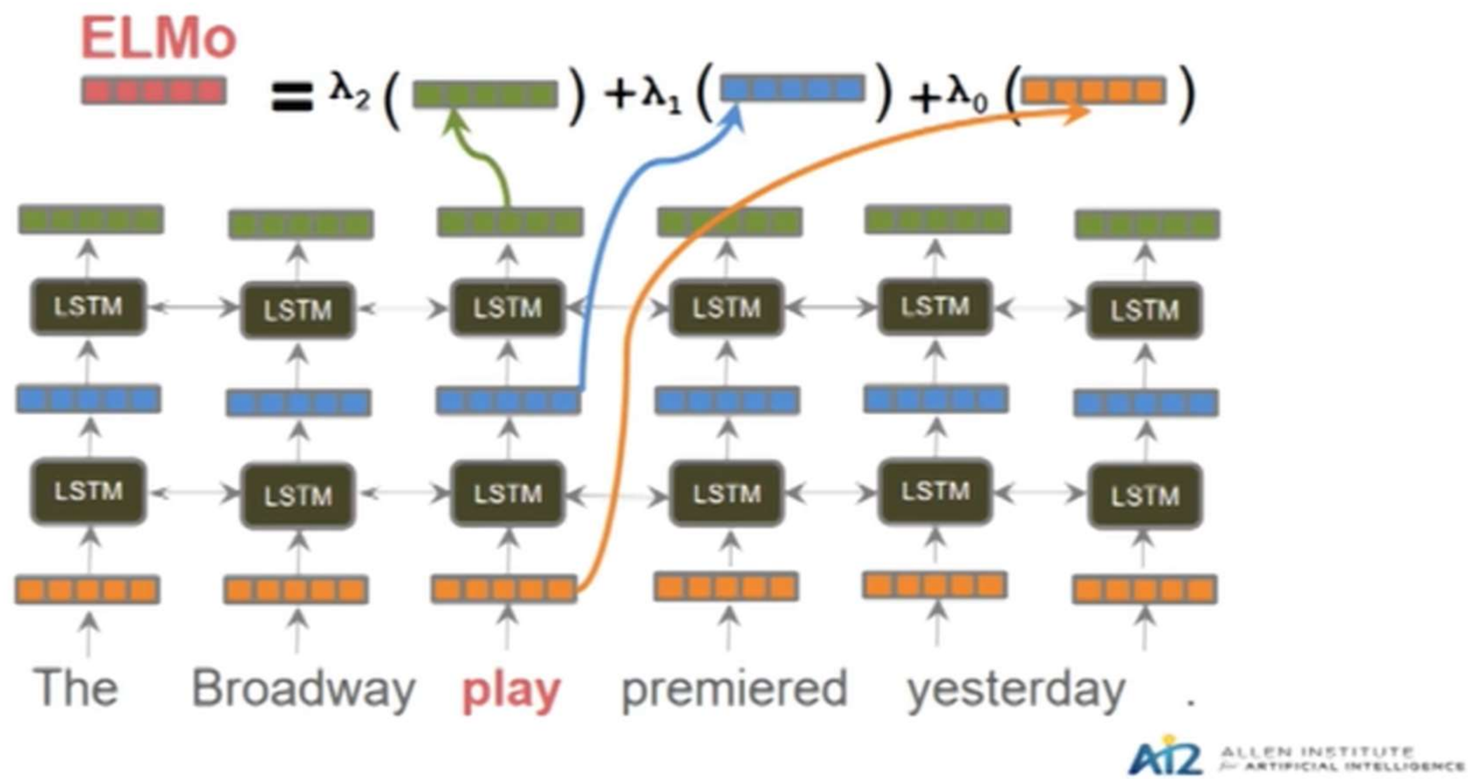
Previous embedding models fail to address *polysemy* or *orthographic variation* (morphology).

Idea:
Build pre-trained word embeddings that are a function of the input sentence.

| | |
|---|---|
| **polysemy** | The Broadway play premiers tomorrow.<br>Let's play outside. |
| **morphology** | play, plays, playing,<br>multiplayer, Play |

slide from
(Peters et al., 2018)

# Embeddings from Language Models



# Replicate, Extend, etc.

https://allennlp.org/elmo

Tensorflow

https://github.com/allenai/bilm-tf

Pytorch

https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md

# BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., 2018)

**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
Transfer learning using transformers and new prediction tasks.

# Masked LM

slides from Devlin et al. (2018)

- **Solution**: Mask out *k*% of the input words, and then predict the masked words
  - We always use *k* = 15%

```
                          store                  gallon
                            ↑                       ↑
the man went to the [MASK] to buy a [MASK] of milk
```

- Too little masking: Too expensive to train
- Too much masking: Not enough context

# Masked LM

slides from Devlin et al. (2018)

- Problem: Mask token never seen at fine-tuning
- Solution: 15% of the words to predict, but don't replace with `[MASK]` 100% of the time. Instead:
- 80% of the time, replace with `[MASK]`

  `went to the store → went to the [MASK]`
- 10% of the time, replace random word

  `went to the store → went to the running`
- 10% of the time, keep same

  `went to the store → went to the store`

# Next Sentence Prediction

slides from Devlin et al. (2018)

- To learn *relationships* between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

**Sentence A** = `The man went to the store.`
**Sentence B** = `He bought a gallon of milk.`
**Label** = `IsNextSentence`

**Sentence A** = `The man went to the store.`
**Sentence B** = `Penguins are flightless.`
**Label** = `NotNextSentence`

# Experiments

- GLUE: **Textual Inference** (MNLI, RTE, WNLI), **Question Similarity** (QQP), **Question Answering** (QNLI), **Sentiment Analysis** (SST-2), **Grammaticality** (CoLa), **Semantic Similarity** (STS-B, MRPC)
- SQuAD (Question Answering)
- Named Entity Recognition
- SWAG (Adverserial Sentence Prediction)

# Experiments: GLUE

| System | MNLI-(m/mm) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.9 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 88.1 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.2 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.1 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **91.1** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **81.9** |

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

# Replicate, Extend, etc.

Includes 104 languages

Tensorflow

https://github.com/google-research/bert

Pytorch

https://github.com/huggingface/pytorch-pretrained-BERT