

# Seq2Seq Models

---

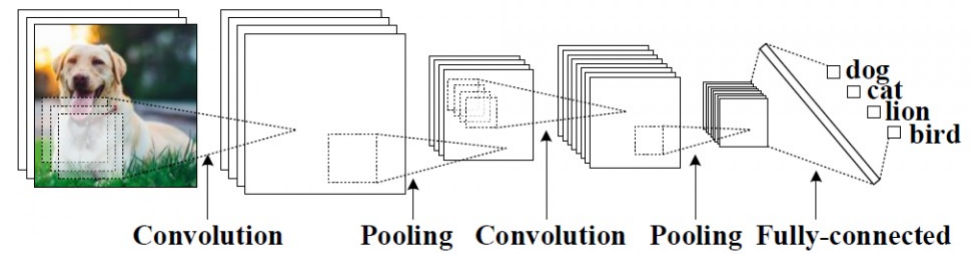
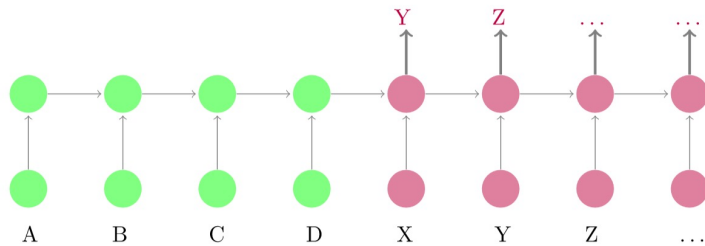
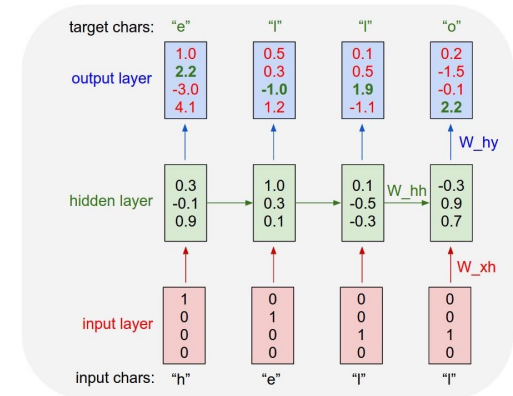
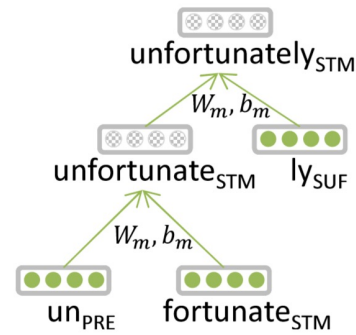
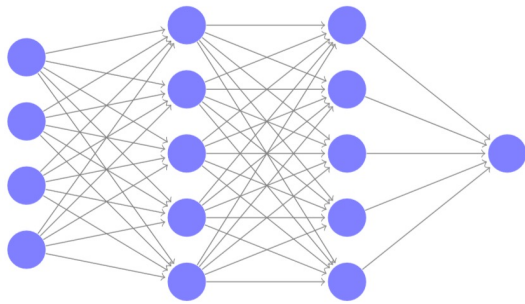
AUSTIN BLODGETT

A solid green horizontal bar at the bottom of the slide.

# Brief Review

---

# Neural Algorithms



NN Task	Example Input	Example Output
Binary Classification	features	+/-
Multiclass Classification	features	decl, imper, inter, ...
Sequence Classification	sentence	POS tags
Sequence to Sequence	(English) sentence	(Spanish) sentence
Structured Prediction	sentence	dependency tree or AMR parse

NN Task	Example Input	Example Output
Binary classification	features	+/-
Multiclass classification	features	decl, imper, inter, ...
Sequence	sentence	POS tags
Sequence to Sequence	(English) sentence	(Spanish) sentence
Tree/Graph Parsing	sentence	dependency tree or AMR parse

# Seq2Seq Tasks

---

- Tasks
  - Machine Translation
  - Automatic Dialogue
  - Question Answering
  - Document Summarization
  - (Some) Semantic Parsing

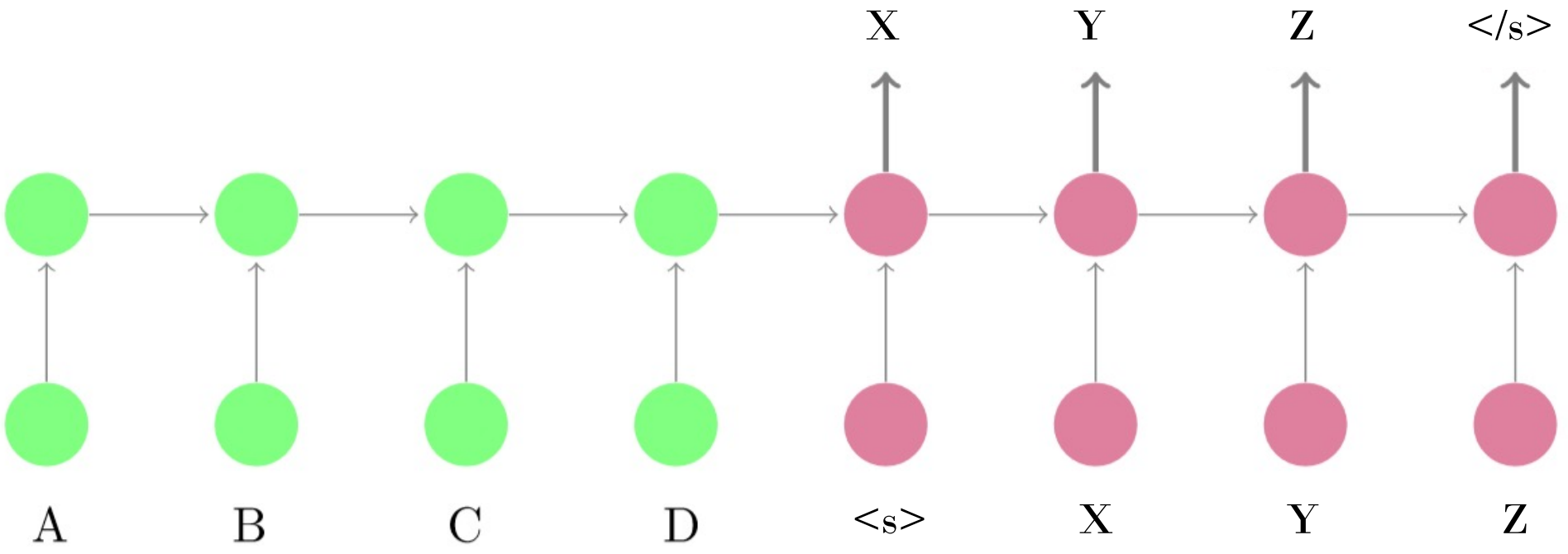
# Encoder-Decoder models

---

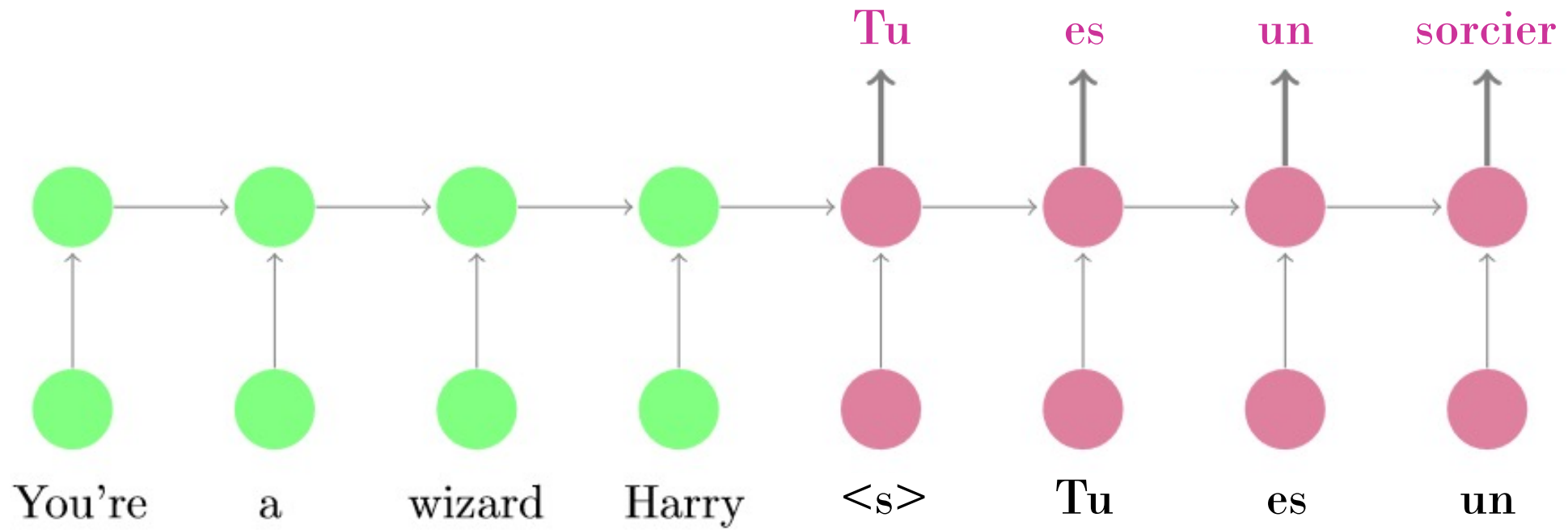
- **Encoder-Decoder model** (also **Seq2Seq**) – Take a sequence as input and predict a sequence as output
- *Input and Output may be different lengths*
- Encoder (*RNN*) models input, Decoder (*RNN*) models output

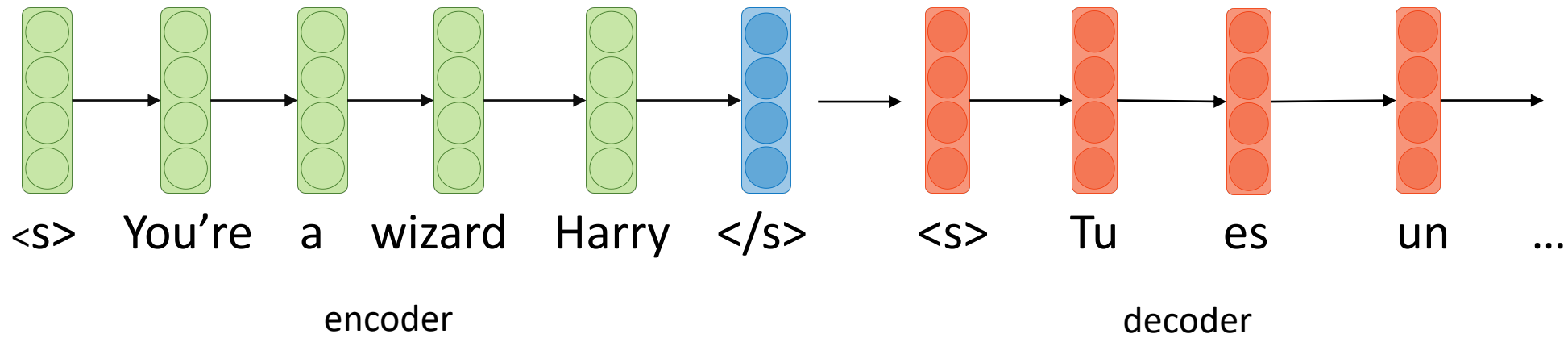
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). **Sequence to sequence learning with neural networks**. In *Advances in neural information processing system*.

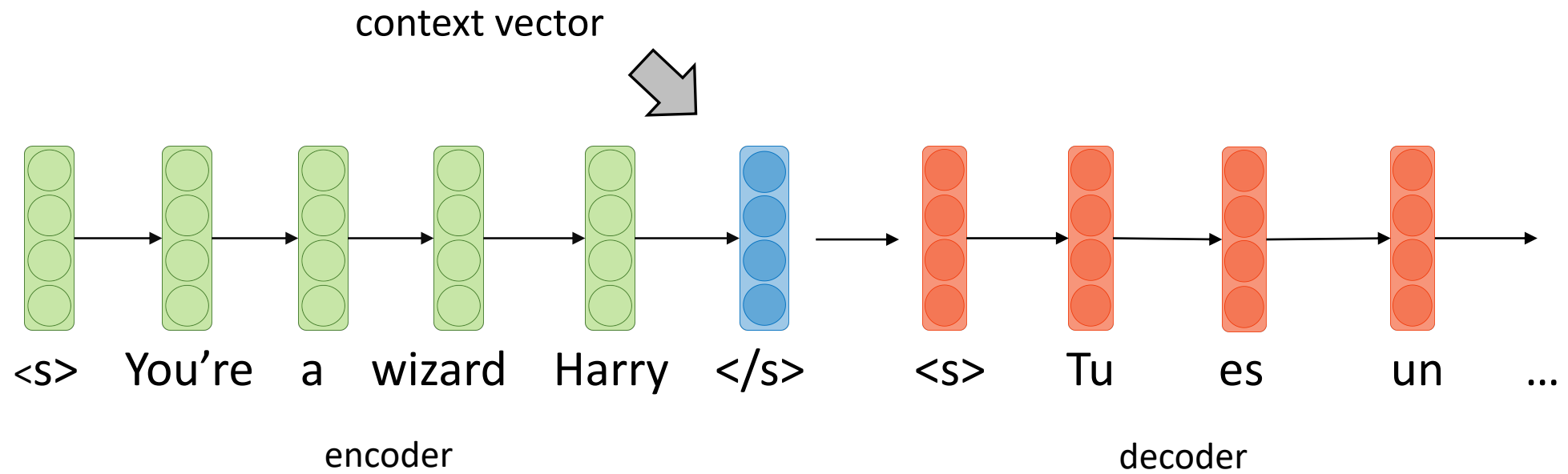
Cho, K., et al. (2014). **Learning phrase representations using RNN encoder-decoder for statistical machine translation**.



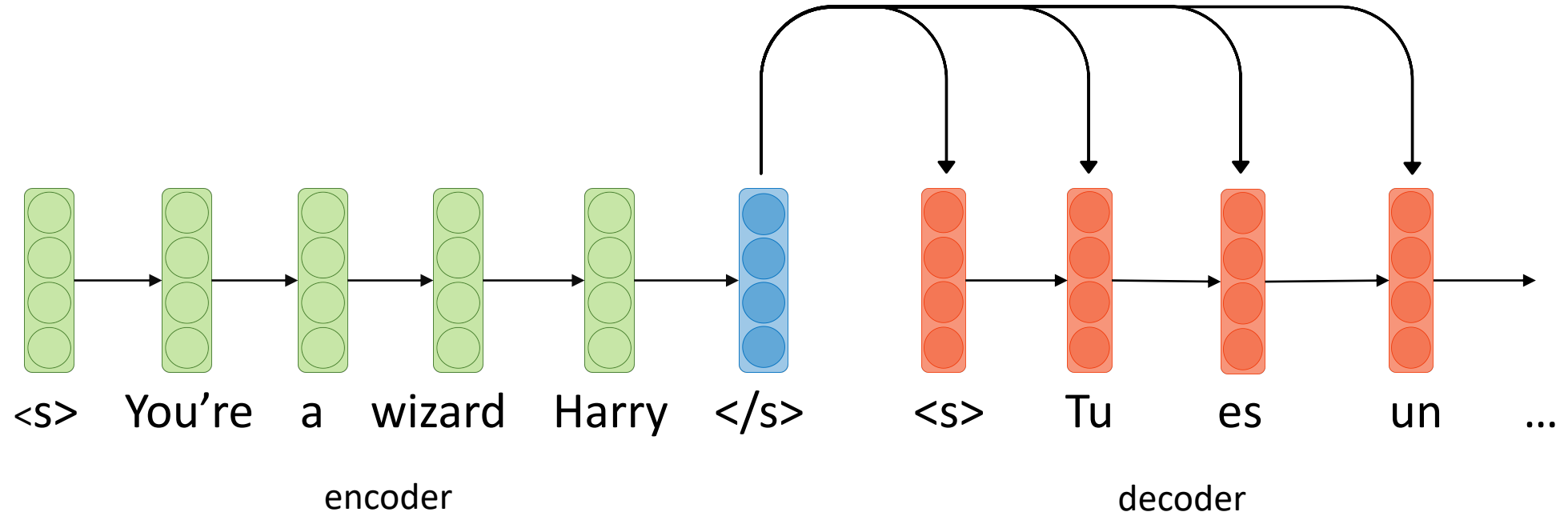








Cho et al., (2014): connect to every state in decoder



# Seq2Seq with RNNs

- RNNs are slow: each timestep needs the first to be completed
- RNNs struggle to capture long-distance dependencies:
  - The **keys** to the door **are/is** gray
  - Caused by issues in managing hidden state: when is it OK to forget that “keys” is the subject and is plural?

# Attention

---

- A way of allowing tokens to “explicitly” represent their dependent words
  - Intuition: a continuous version of word alignments
- For a given word in the output, represents the importance of each word in the input
- We’ll talk about how much the network “attends” to each word.
- First used in MT, improved BLEU score by 10 pts (huge!)

# Activity

---

Question

Answer

What is the preferred weapon of the Jedi ?

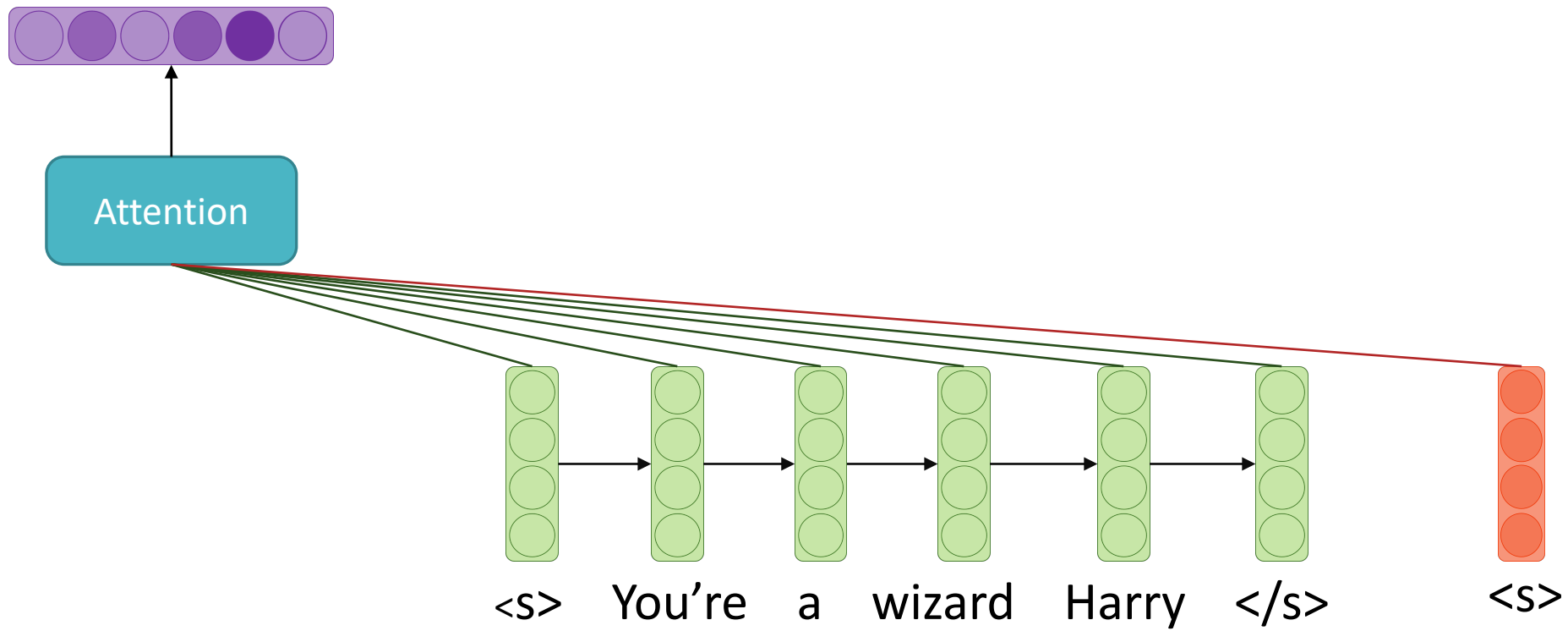
a light saber

PRON AUX DET ADJ NOUN ADP DET PROPN

1 2 3

# Attention

---





# Attention

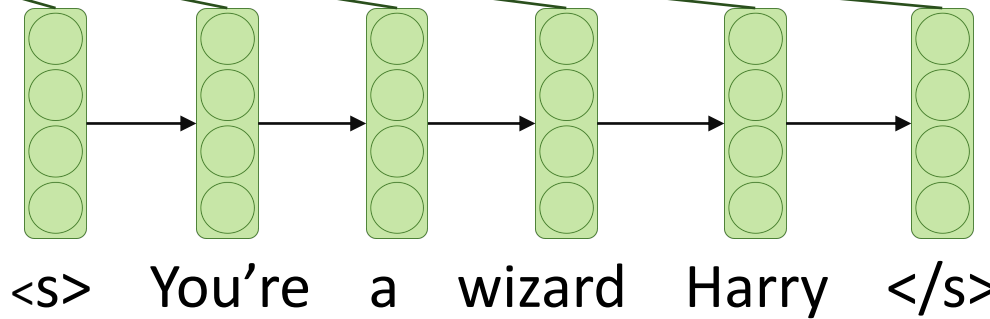
---

(softmax normalized)



Attention

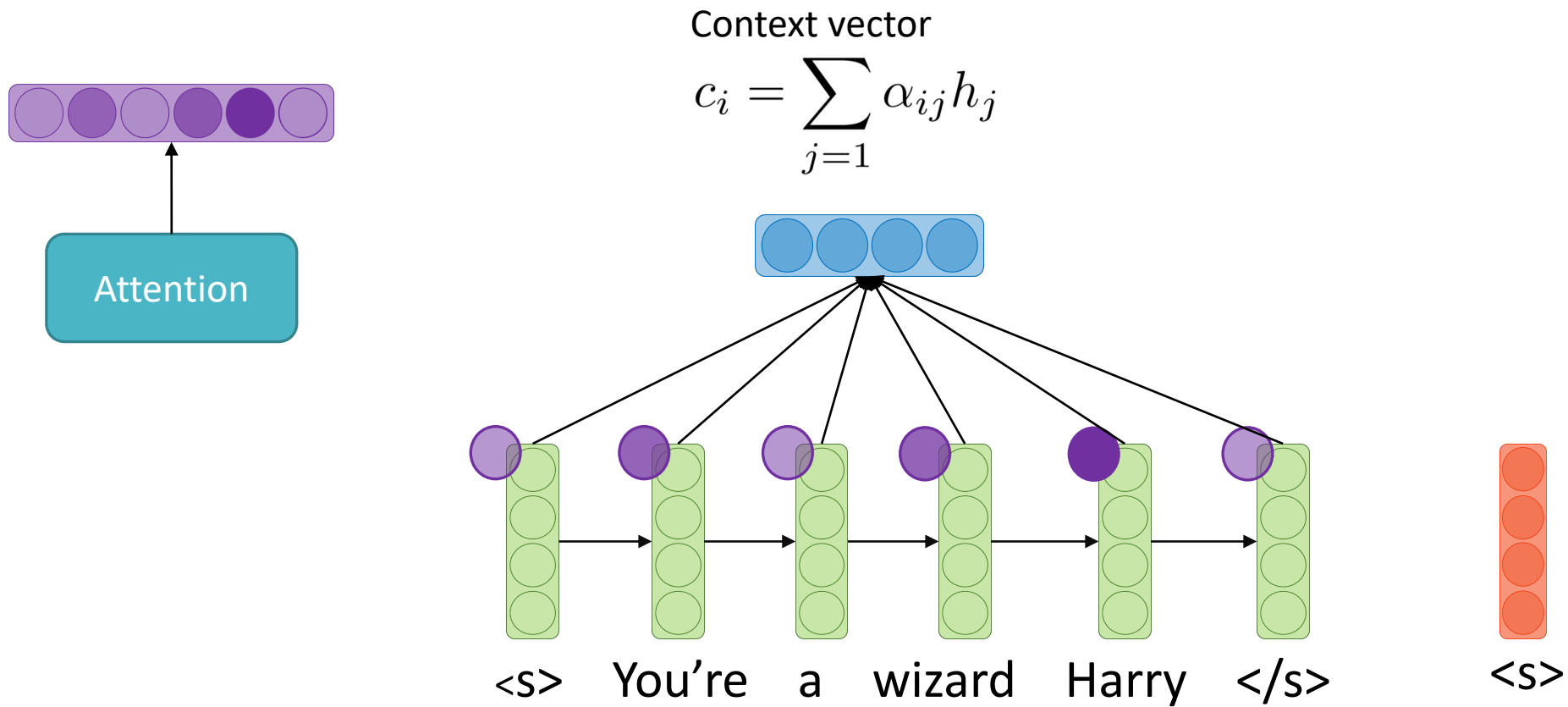
Encoder States



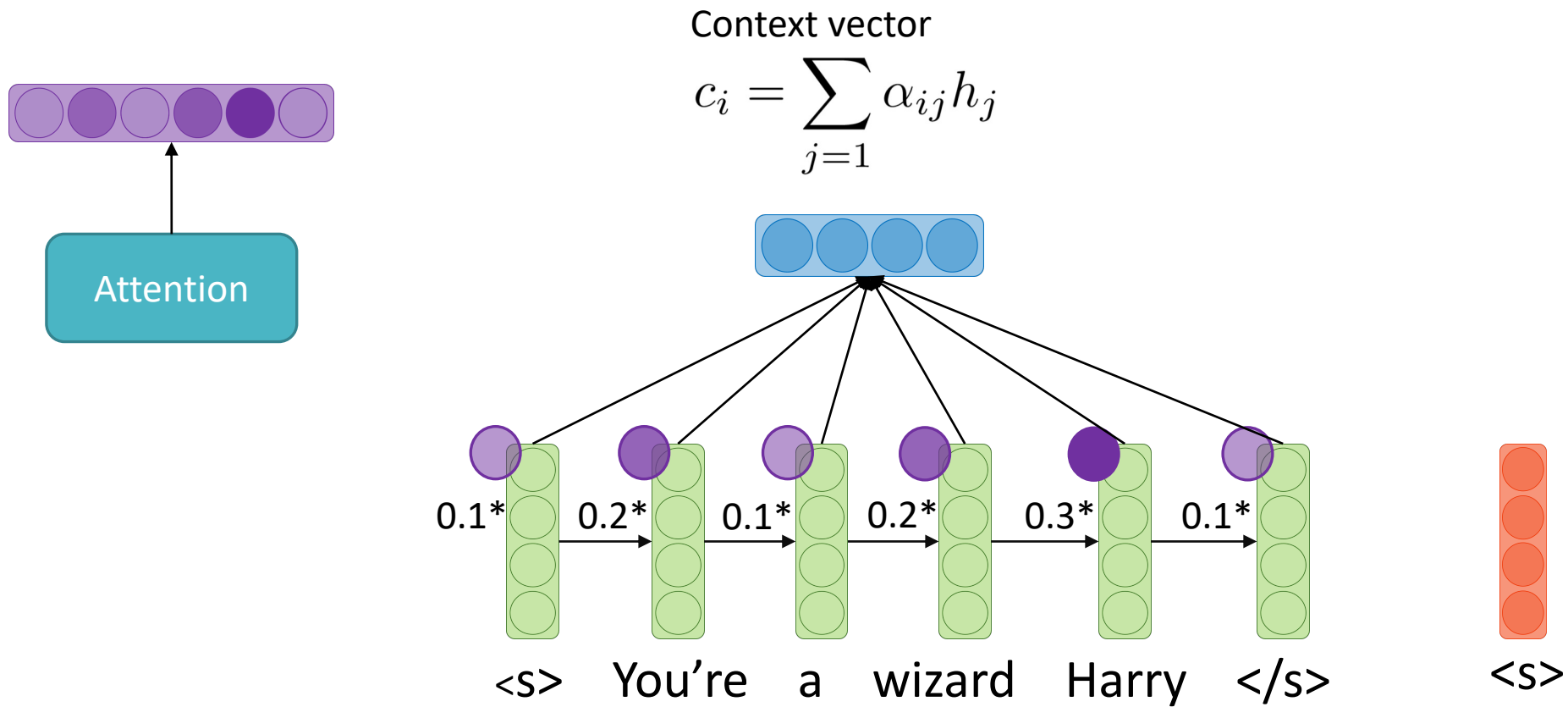
Decoder State



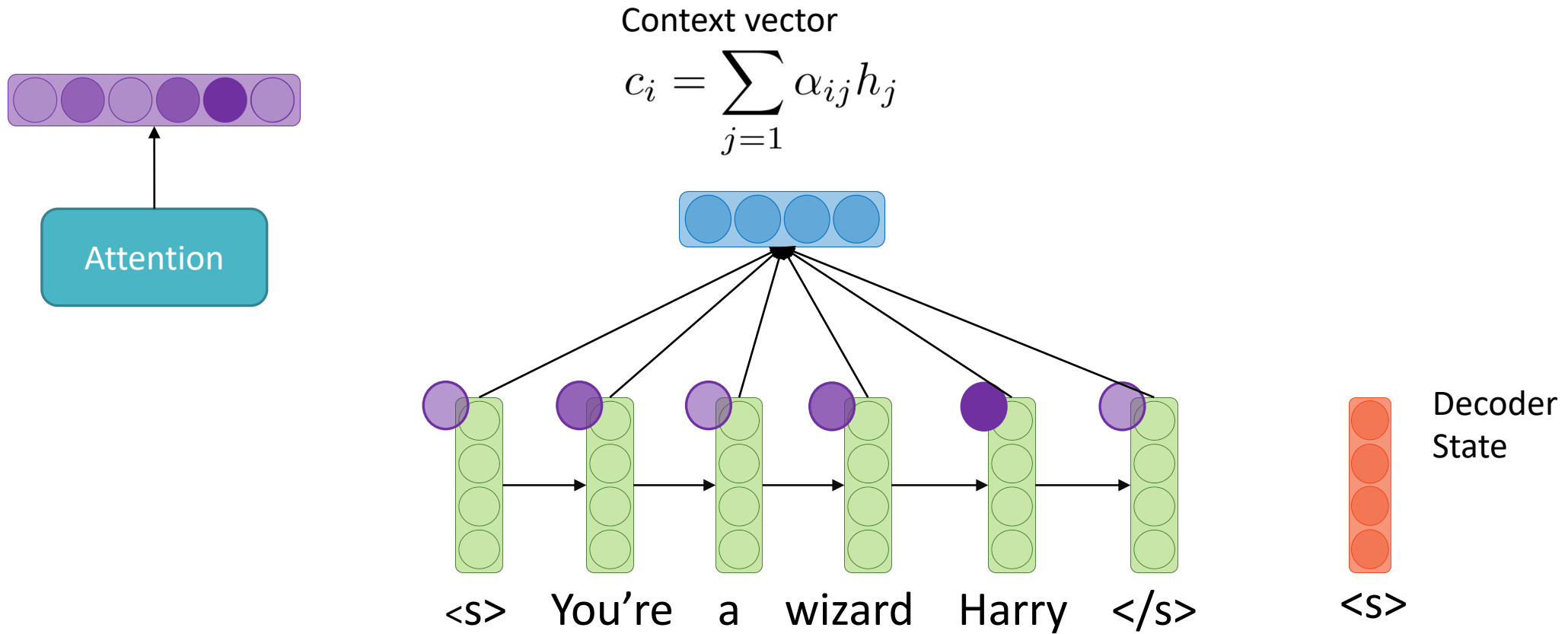
# Attention



# Attention



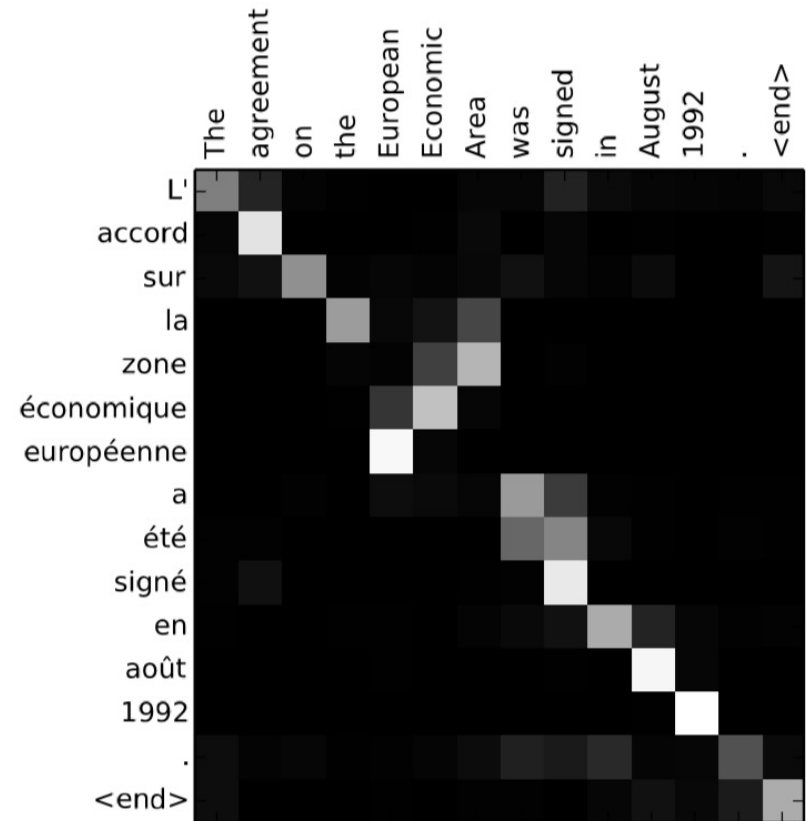
# Attention



# Attention

Every attention value depends on one word in the source and one in the target.

Attention matrix tells us how “important” a source word is for each target word (much like alignment).



# Attention vs. RNN

---

- Shift “context management” into its own module (attention)
- Allow decoder state to handle everything else (e.g. grammar of the target language)
- Benefits
  - More parallelizable
  - No more “remembering” problem: each token gets its own context vector that is more independent of other tokens’ context vectors

# Transformers

---

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

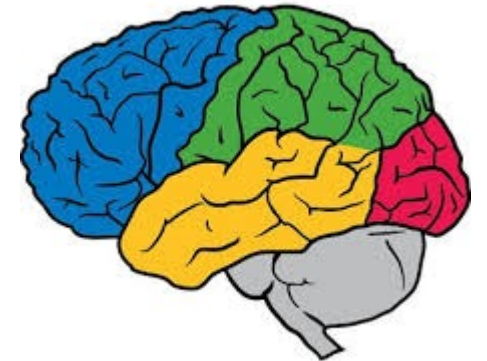
---

Debut of the **Transformer** architecture

The same model used in:

- BERT (Devlin, et al. 2018)
- LISA (Strubell, et al. 2018)
- RoBERTa (Liu et al., 2019)
- and others...

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





# Introducing Self-Attention

---

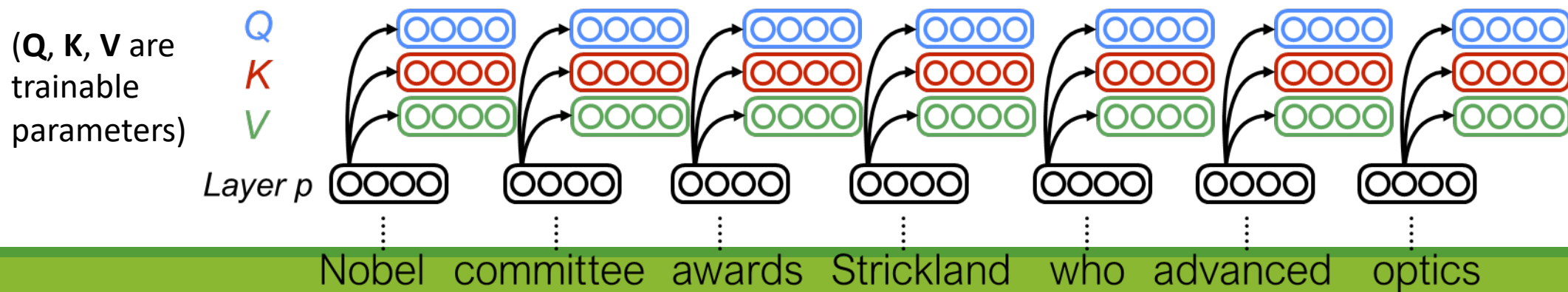
- Shift “context management” into its own module (attention)
- Allow decoder state to handle everything else (e.g. grammar of the target language)
- Benefits
  - More parallelizable
  - No more “remembering” problem: each token gets its own context vector that is more independent of other tokens’ context vectors

# Transformer: Self-Attention

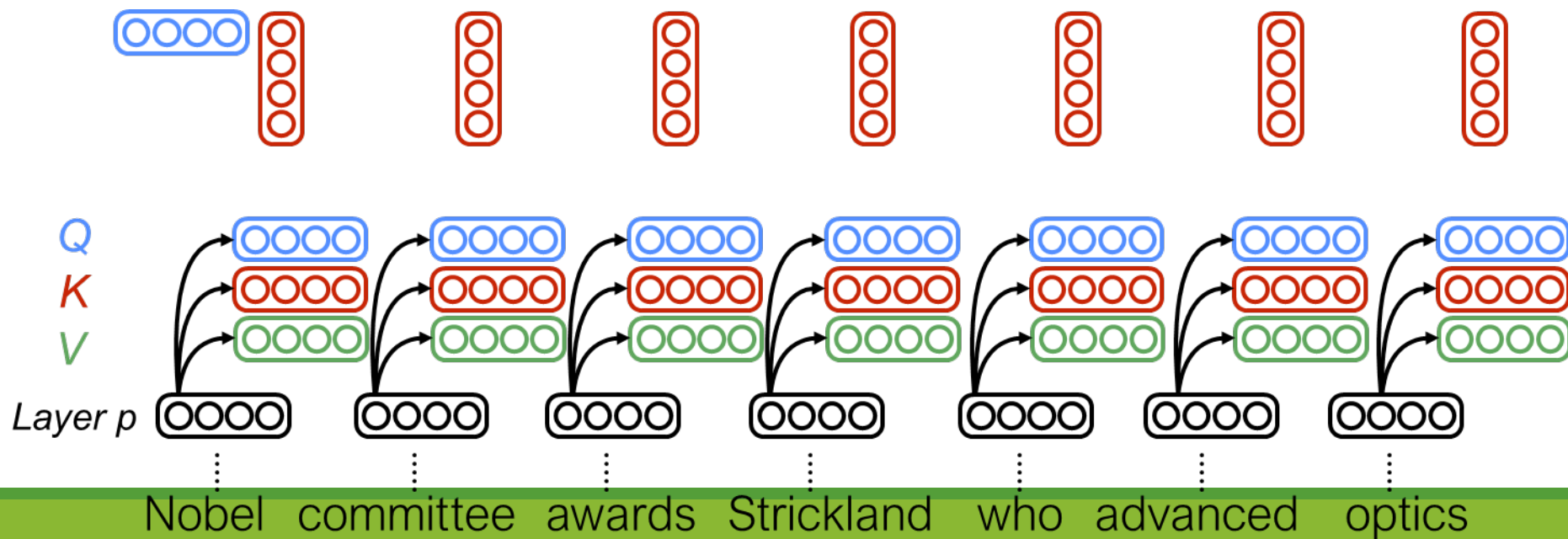
---

query  $Q$  }  
key  $K$  } used to calculate  
value  $V$  — base representation  
of word

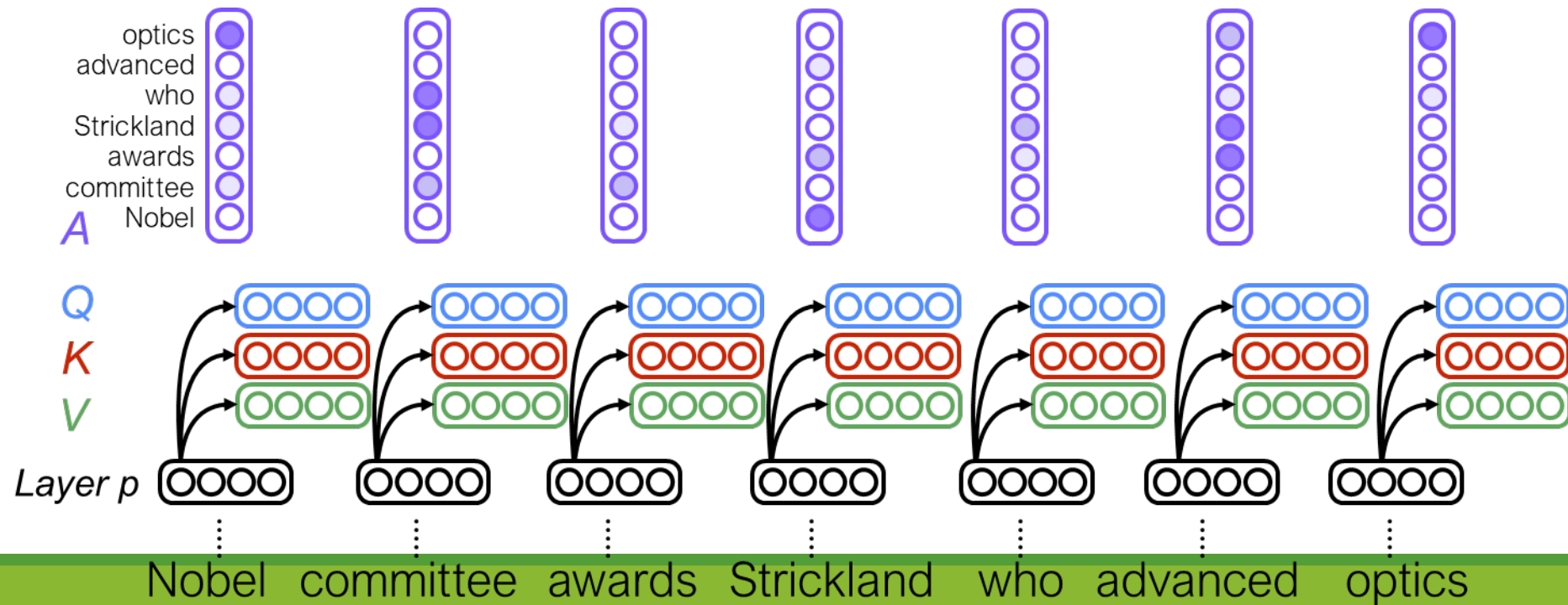
# Transformer: Self-Attention



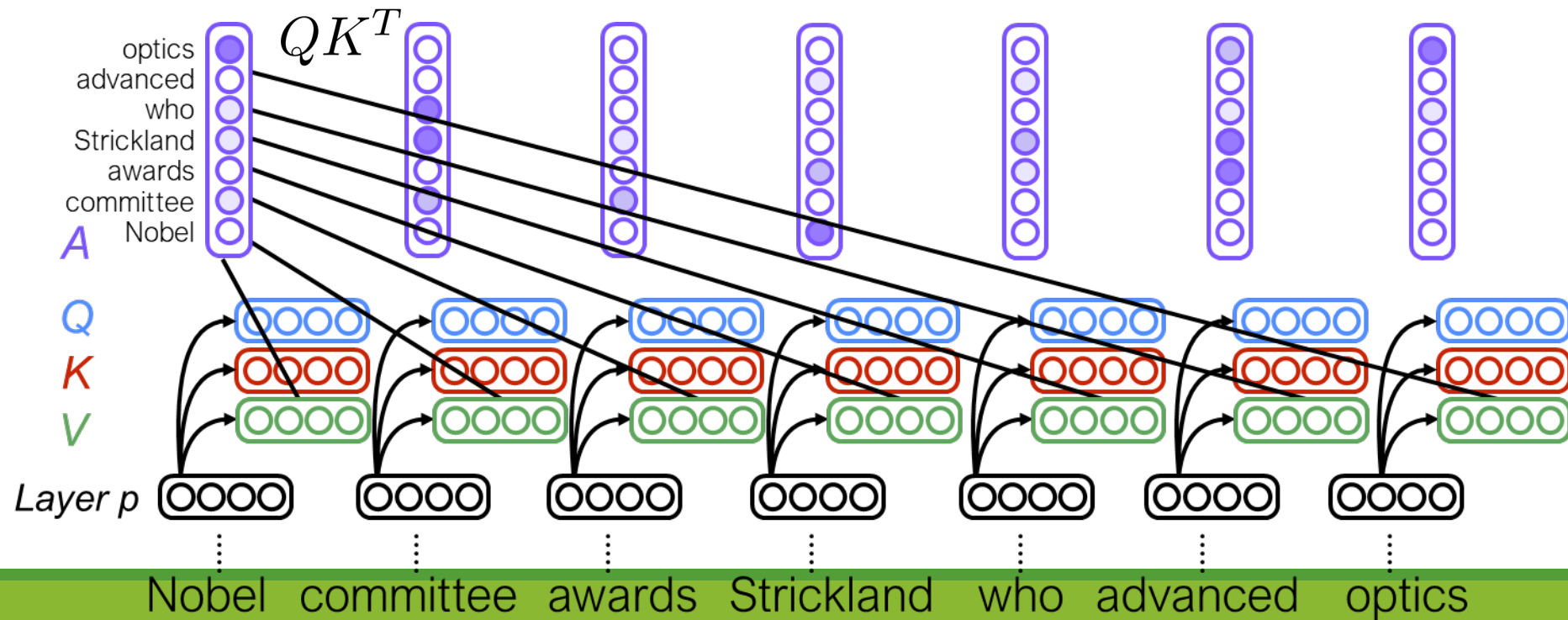
# Transformer: Self-Attention



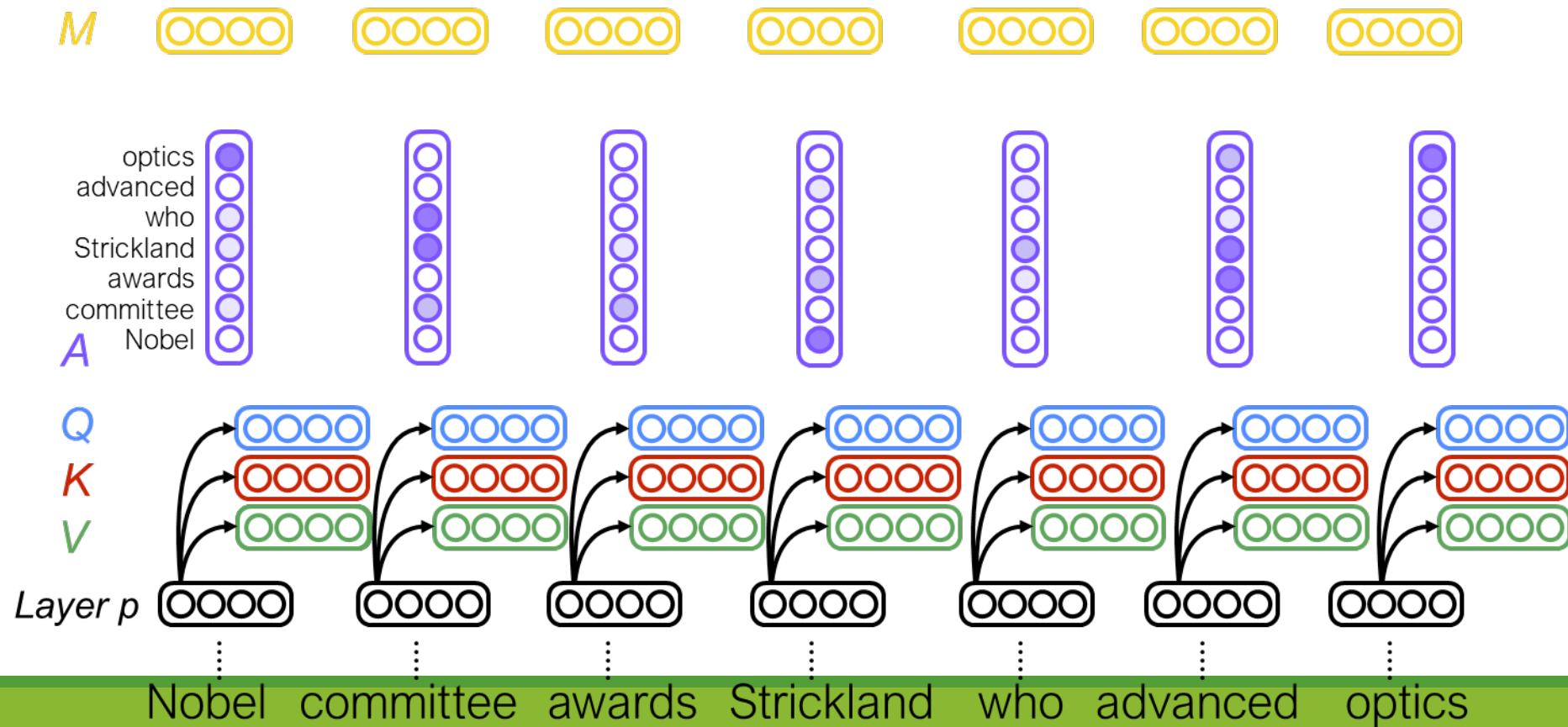
# Transformer: Self-Attention



# Transformer: Self-Attention



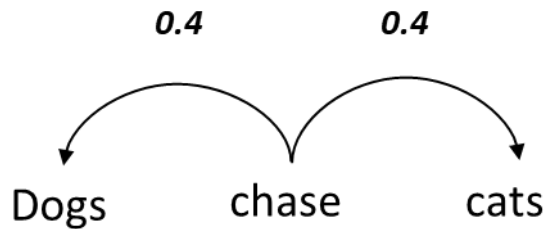
# Transformer: Self-Attention



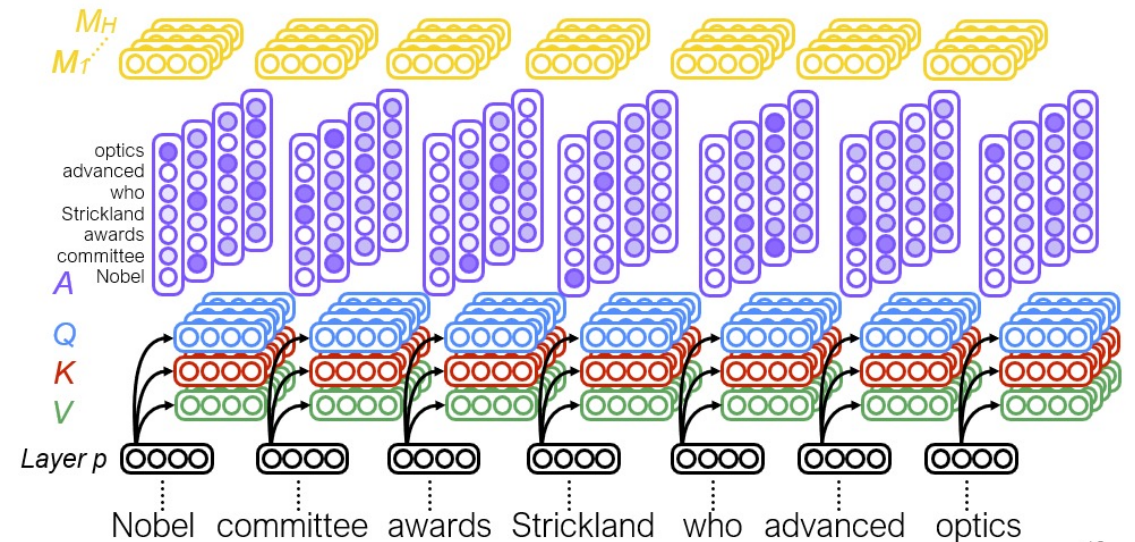
# Multi-Head Attention

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.



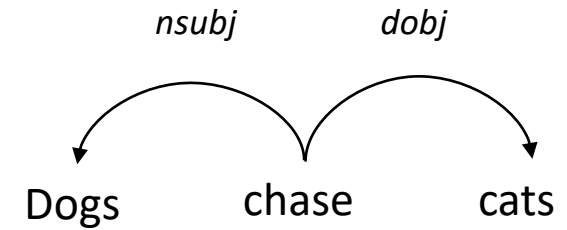


# Linguistically Motivated

---

## Strengths

- Captures long-distance dependencies!
- Intuitively: approximates *weighted unlabeled dependencies*



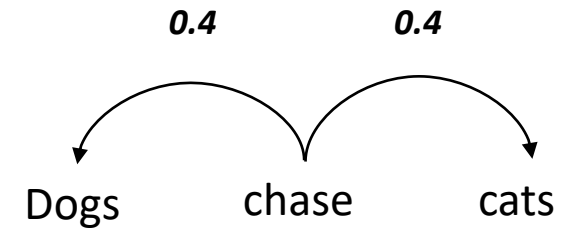
	Dogs	chase	cats
Dogs			
chase			
cats			



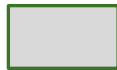




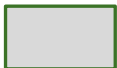

# Linguistically Motivated

---

## Strengths

- Captures long-distance dependencies!
- Intuitively: approximates *weighted unlabeled dependencies*



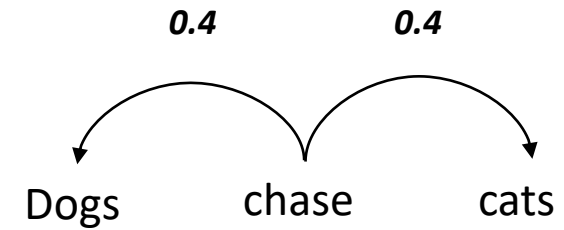
	Dogs	chase	cats
Dogs			
chase			
cats			

# Linguistically Motivated

---

## Strengths

- Captures long-distance dependencies!
- Intuitively: approximates *weighted unlabeled dependencies*



## Weaknesses (addressed in next slides)

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

	Dogs	chase	cats
Dogs			
chase			
cats			

# 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)

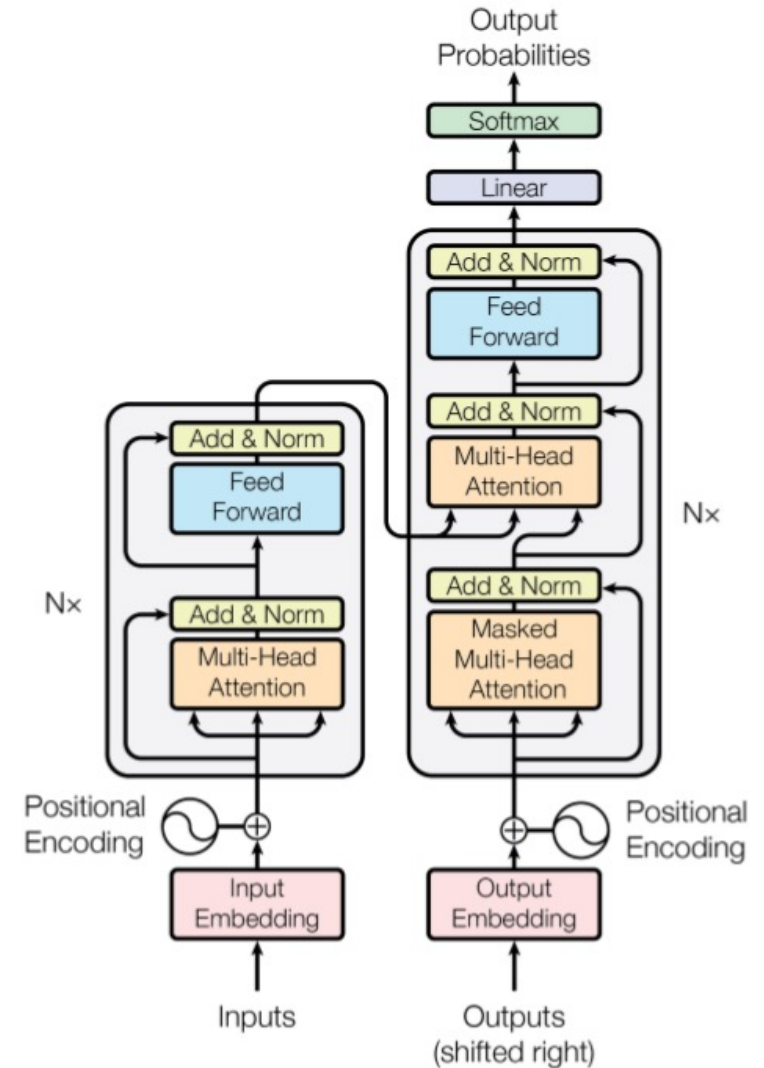


Figure 1: The Transformer - model architecture.

# Positional Encoding

---

Caveat: self-attention on its own doesn't have awareness of word order, needs to be represented another way (position embeddings)

# 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

---

- For some tasks, data is sparse or expensive (AMR, low resource languages, etc.)
- Transfer Learning: taking a pre-trained model and applying it to a new dataset and/or task
- “Free” increases in accuracy
- 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*.

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

---

## Bidirectional Encoder Representations from Transformers

- Idea: if we train a model using a very general task on a **massive** corpus, could we get representations out of that model that are useful for any task?
- Introduced new tasks (masked language modeling, next sentence prediction)



- **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

# 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 (Adversarial Sentence Prediction)

# Experiments: GLUE

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
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
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>

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); BERT<sub>BASE</sub> = (L=12, H=768, A=12); BERT<sub>LARGE</sub> = (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/>.

# Impact

---

- Most successful early **contextualized word embedding** model
  - “contextualized”: a word’s representation depends on its context, differs from use to use
- Provided massive performance gains for most tasks in English and other high-resource languages

# Demo

---

AllenNLP demo of BERT

<https://demo.allennlp.org/masked-lm>



# Replicate, Extend, etc.

---

Includes 104 languages

Tensorflow

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

Pytorch

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

# Recent Transfer Learning Projects

---

RoBERTa (Liu et al., 2019)

<https://github.com/pytorch/fairseq>

XLNet (Yang et al., 2019)

<https://github.com/zihangdai/xlnet>