

# Foundations of Natural Language Processing at Scale

Chengyin Eng & Tristan Nixon



# Word Embedding

# We usually represent word types with vectors

## Why vectors to encode meaning?

- compute similarity between words (phrases, documents) is very useful!

## Follows the distributional hypothesis:

“Words with similar meaning tend to occur in similar context”

# Summary of tf-idf

- Common baseline model
- Sparse vector representation
- Words are represented by a simple function of the counts of nearby words



We need dense vector representations

# Word2Vec

- Dense vectors to model the meaning of a word as an embedding
- Trains a classifier to distinguish nearby and far-away words
  - Important Insight:  
Use the text as implicitly supervised training data
  - Example: Is “blue” likely to show up near “sky”?
- We take the learned classifier weights as the embeddings

Core idea: **Predict** rather than **count**

# Word2vec is neural-network inspired

- Consists of Continuous Bag of Words (CBOW) and skip gram models
- CBOW
  - Computes the  $p(\text{target word})$  given the context across a window of size  $k$
  - Example: Nobody reads \_\_\_\_ anymore in the 21st century.
- Skip gram with negative sampling
  - Computes  $p(\text{negative context words})$  given the target
  - Example: Nobody \_\_\_\_ classics \_\_\_\_ in the 21st century

# Skip-gram with negative sampling

1. Generate + examples
2. Create -ve examples for context words
3. Train a logistic regression to classify whether a given pair is +ve or -ve
4. Use the weights of this model as the embeddings





What are the limitations of CBOW and Skip-gram?

# Limitations of Word2Vec

CBOW and Skip-gram are predictive models based on local context.

We are losing global context.

# Global Vectors for Word Representation (GloVe)

*Intuition:* derive semantic relationships from co-occurrence matrix

Example:

“The cat sat on the mat”

window size = 1

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

# Global Vectors for Word Representation (GloVe)

*Goal:*

Generate embeddings by building word vectors to show how every pair of words co-occurs

Probability and Ratio	$k = \textit{solid}$	$k = \textit{gas}$	$k = \textit{water}$	$k = \textit{fashion}$
$P(k \textit{ice})$	$1.9 \times 10^{-4}$	$6.6 \times 10^{-5}$	$3.0 \times 10^{-3}$	$1.7 \times 10^{-5}$
$P(k \textit{steam})$	$2.2 \times 10^{-5}$	$7.8 \times 10^{-4}$	$2.2 \times 10^{-3}$	$1.8 \times 10^{-5}$
$P(k \textit{ice})/P(k \textit{steam})$	8.9	$8.5 \times 10^{-2}$	1.36	0.96



What are the limitations of GloVe?

# Limitations of GloVe

GloVe does not know how to deal with out-of-vocabulary (OOV) words. It assigns random vectors to OOV words.

It needs lots of memory.

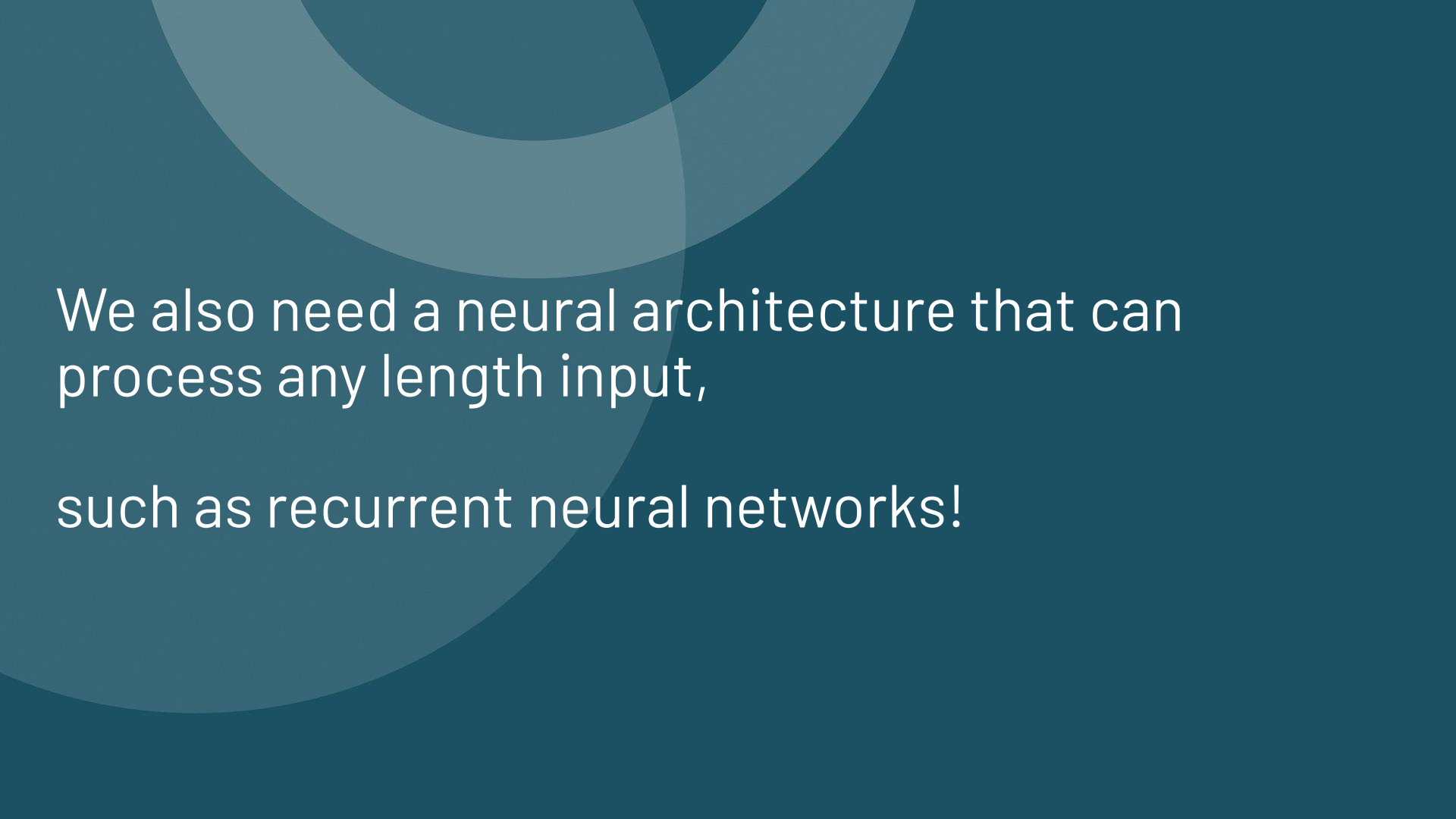
# Summary

	Word2Vec	GloVe
Training Data	Uses texts directly	Focuses on word co-occurrences over the whole corpus
Embedding	Captures whether words appear in similar context	Relates to the probabilities that two words appear together

We need better ways to address OOV and context problem.

That's why we need deep contextualized word embeddings like ELMo, OpenAI-GPT, and BERT!



The background is a solid dark teal color. Overlaid on this are several large, semi-transparent circles in a lighter shade of teal. These circles overlap each other, creating a layered effect. One large circle is positioned in the upper left, another in the upper right, and a third, larger one in the lower left, partially covering the others.

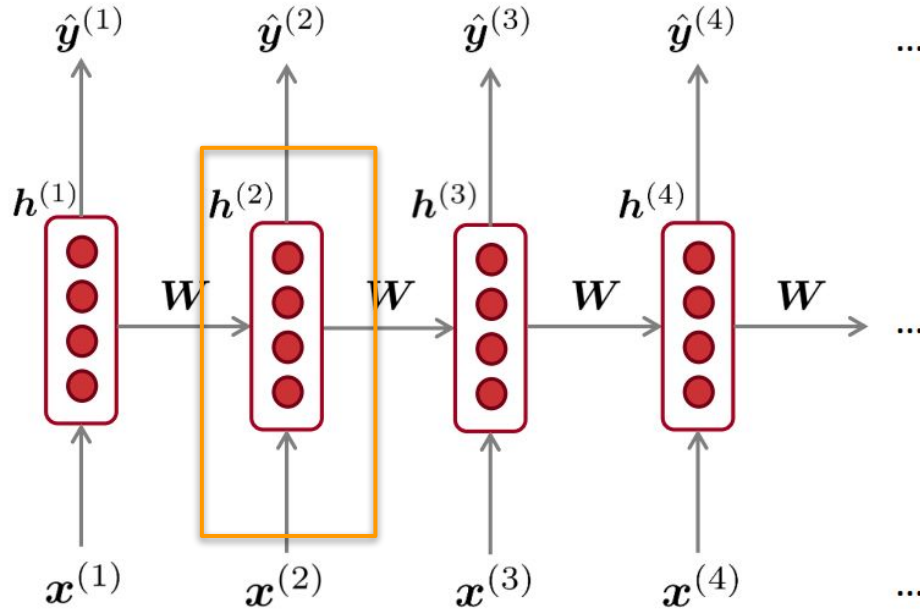
We also need a neural architecture that can  
process any length input,  
such as recurrent neural networks!

# RNN architecture is a feedback loop

outputs

hidden states

input sequence  
(any length)



Core idea: RNN use the same weights

output distribution

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

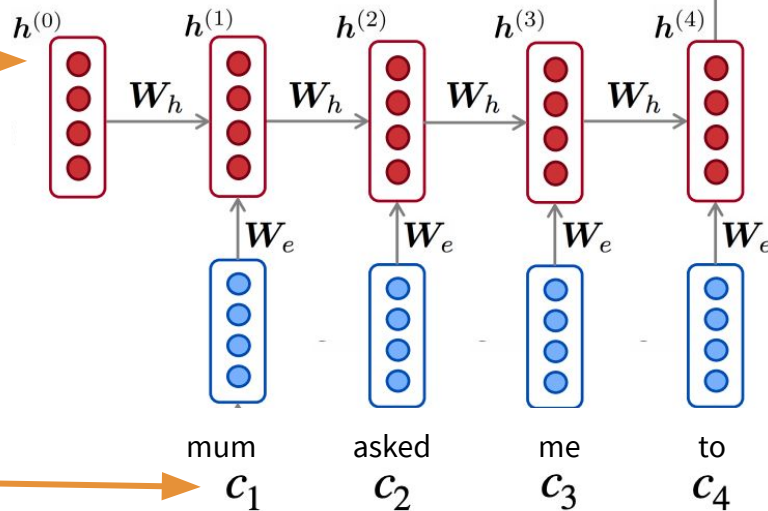


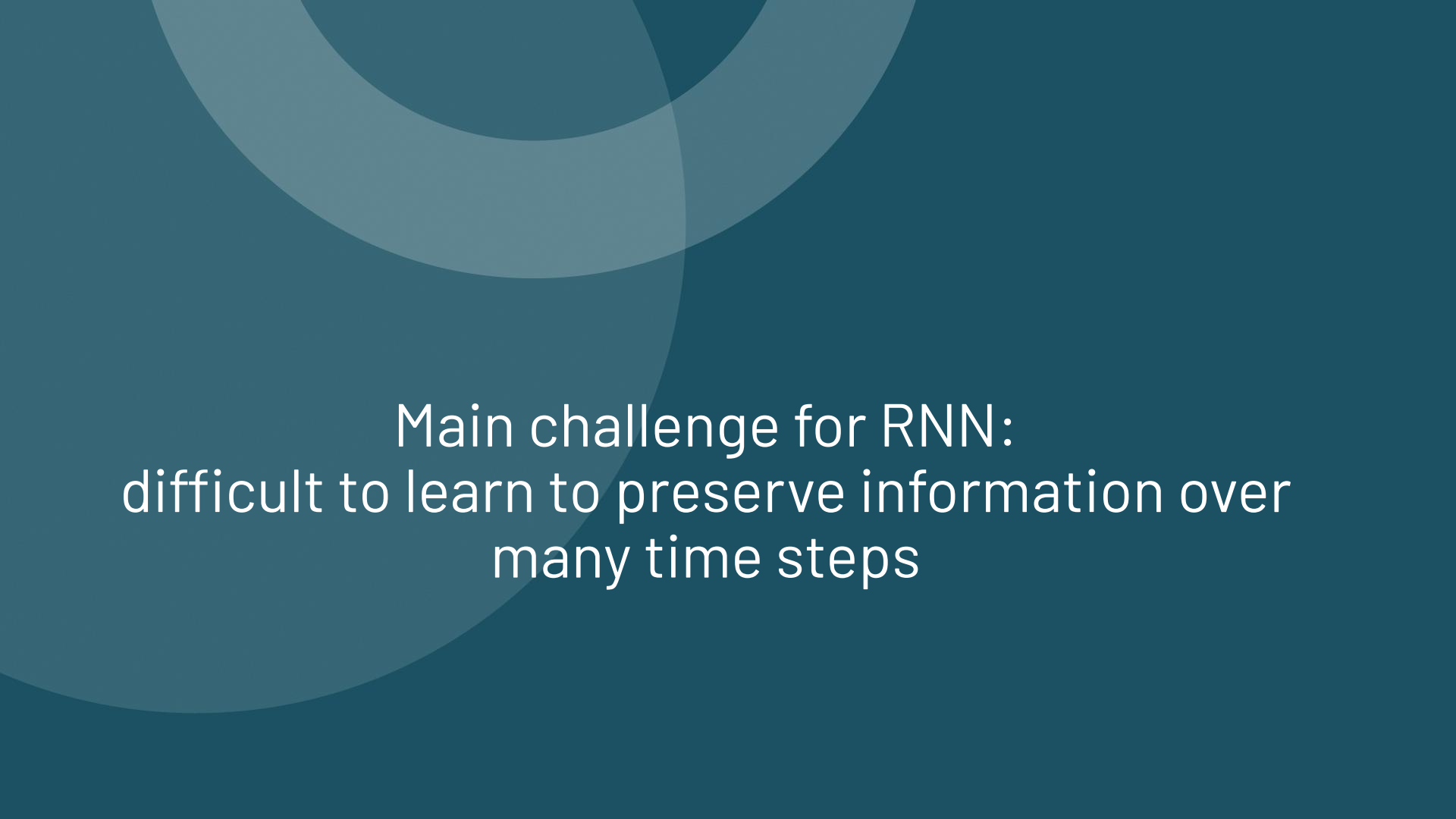
hidden states

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

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

word embeddings



The background of the slide features a dark teal color with several overlapping, lighter teal circular shapes that create a layered, abstract effect.

Main challenge for RNN:  
difficult to learn to preserve information over  
many time steps

In a vanilla RNN, the hidden state is constantly  
being rewritten

$$h(t) = \sigma(W_h h(t-1) + W_x x(t) + b)$$

The background is a solid reddish-orange color. On the right side, there are several overlapping circles of varying shades of the same color, creating a layered, abstract effect.

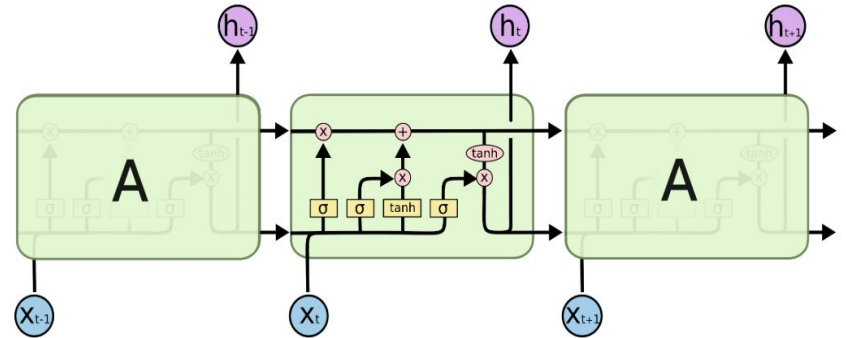
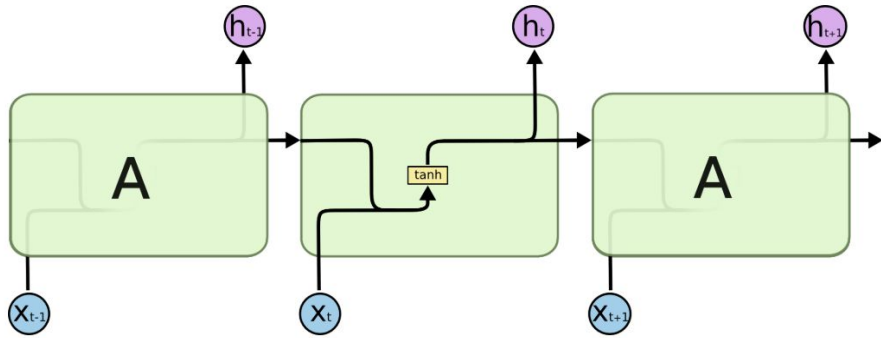
How about building a RNN with separate memory?

# Long Short-Term Memory (LSTM)

Has cell states

- Has additional “forget” gates
- Can reduce both vanishing and exploding gradient problems
- Allows the error to backprop through unlimited time steps

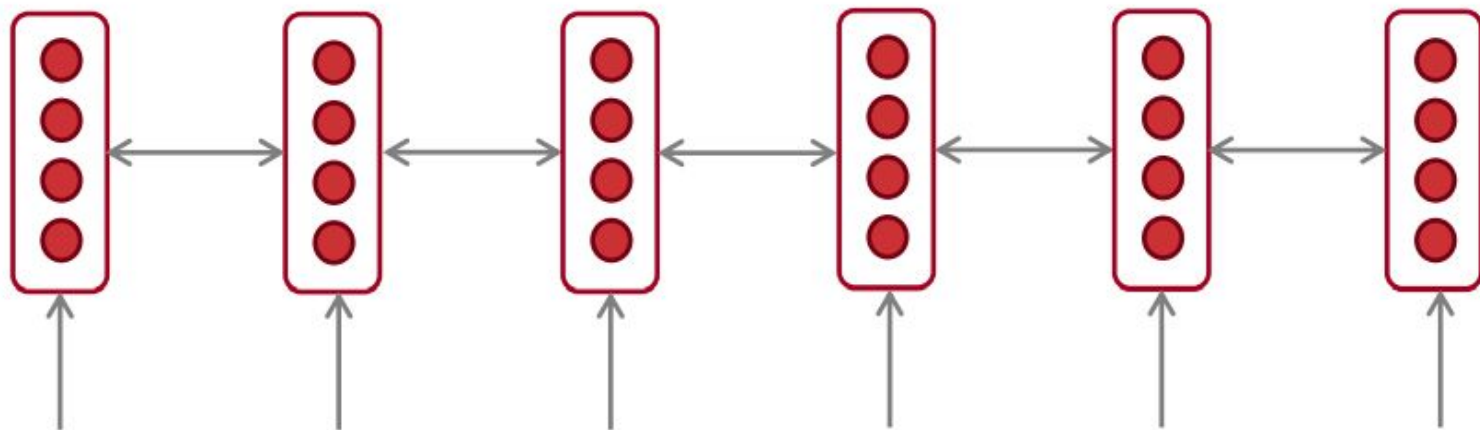
# Single-Layer RNN vs LSTM RNN



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



# Bidirectional RNNs are intuitive for language tasks



LSTM is pretty good.

But is there another big step?

YES, ATTENTION!

# Transformers

Rather than forgetting information like LSTMs, they pay attention to important words

Original implementation in 2017:  
<https://arxiv.org/pdf/1706.03762.pdf>

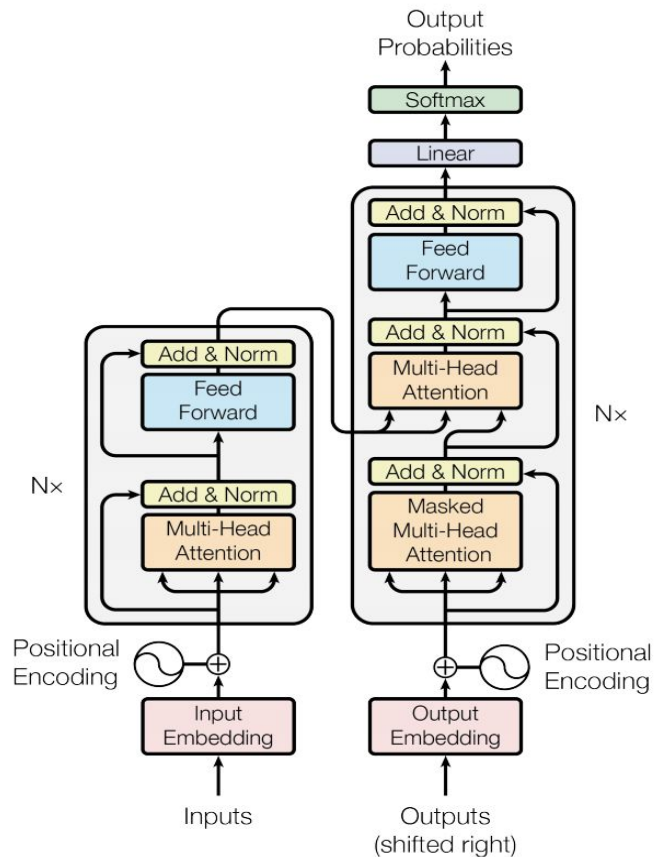


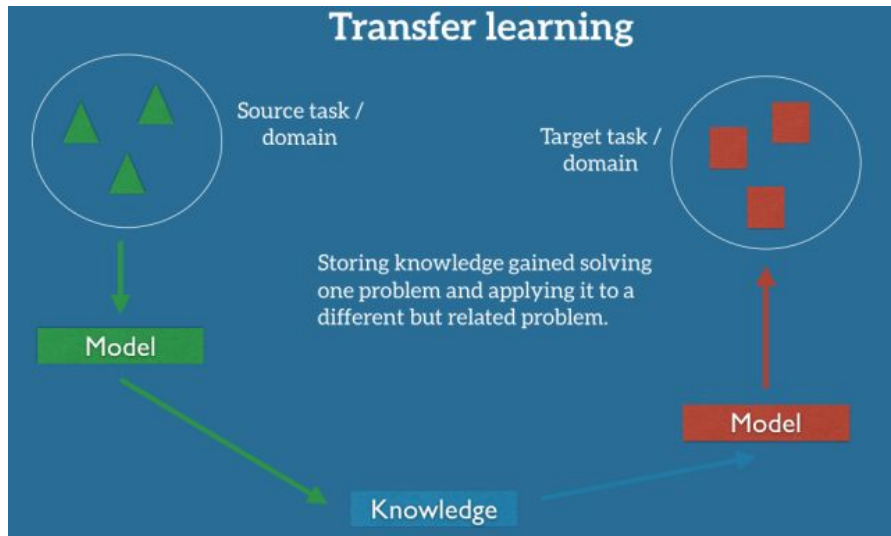
Figure 1: The Transformer - model architecture.



# Transfer Learning

# Transfer Learning

IDEA: Intermediate representations learned for one task may be useful for other related tasks



# When to use Transfer Learning?

	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*



# Size and Impact of NLP models

# Recap of NLP use case variety

- Sentiment analysis (stock news)
- Question answering (chatbots)
- Translation
- Semantic similarity (literature search)
- Summarization (clinical decision support)
- Named entity recognition (content recommendation)
- Speech recognition (Siri)



# Language models have become increasingly large

Year	Model	# of Parameters	Dataset Size
2019	BERT [39]	3.4E+08	16GB
2019	DistilBERT [113]	6.60E+07	16GB
2019	ALBERT [70]	2.23E+08	16GB
2019	XLNet (Large) [150]	3.40E+08	126GB
2020	ERNIE-GEN (Large) [145]	3.40E+08	16GB
2019	RoBERTa (Large) [74]	3.55E+08	161GB
2019	MegatronLM [122]	8.30E+09	174GB
2020	T5-11B [107]	1.10E+10	745GB
2020	T-NLG [112]	1.70E+10	174GB
2020	GPT-3 [25]	1.75E+11	570GB
2020	GShard [73]	6.00E+11	–
2021	Switch-C [43]	1.57E+12	745GB

**Table 1: Overview of recent large language models**

Source: [Bender and Gebru et al. 2021](#)

# NLP models have far-ranging consequences

- Environmental cost
- Financial cost
- Societal cost

# Training NLP models causes enormous carbon footprint

- Training a Transformer emits 284 tonnes of CO<sub>2</sub>
  - Global average per person: 4.8 tonnes
  - US average: 16 tonnes
- Training a single BERT base model:
  - Without hyperparameter tuning: requires as much energy as trans-American flight
  - With neural architecture search: emits CO<sub>2</sub> equivalent to 4 average cars in their whole lifetime

Sources: [Bender and Gebru et al. 2021](#), [Strubell et al 2019](#), <https://ourworldindata.org/co2-emissions>,

# Training NLP models is costly

- Training a BERT model:
  - 110 million parameters: \$2.5k - \$50k
  - 340 million parameters: \$10k - \$200k
  - 1.5 billion parameters: \$80k - \$1.6mil
- Increase in 0.1 BLEU score for English to German translation = \$150k
  - BLEU score is a benchmark metric used to compare the performance of translation

# NLP models contain downstream societal impacts

- Size doesn't guarantee diversity
- Data doesn't capture changing social views
- Data encode bias, which translates to models in production
  - without malicious intent: tweet flagging, healthcare applications, etc.
  - with malicious intent: generate automated abusive text, conspiracy theories, etc.
- Coherence  $\neq$  understanding

# Resources

- [A visual guide to using BERT for the first time](#)
- [NLP Progress](#)