A Recent History of Language Models

Daniel Khashabi

CSCI 601.771: Self-supervised Statistical Models

JOHNS HOPKINS UNIVERSITY

Prompt

- `What I remember from last time is ___X___`

Where is X is less than 10 words.

Planning

- **Last time:** course schedule, defining “self-supervised models”
  - If you missed it, a recorded video is available on Canvas.
- **Today:** Historical work on self-supervised language learning.
- Next week’s **“role” assignments**: expect an email after the class.
- **Office hours:** by appointment.

Self-Supervised Models: Breaking News!!

- Released yesterday!

"Outpainting, a new feature which helps users extend their creativity by continuing an image beyond its original borders"

https://openai.com/blog/dall-e-introducing-outpainting/

Last session:

Self-Supervised Models

as predictive models of the world!

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

The cat sat on the mat.

The cat sat on the mat.

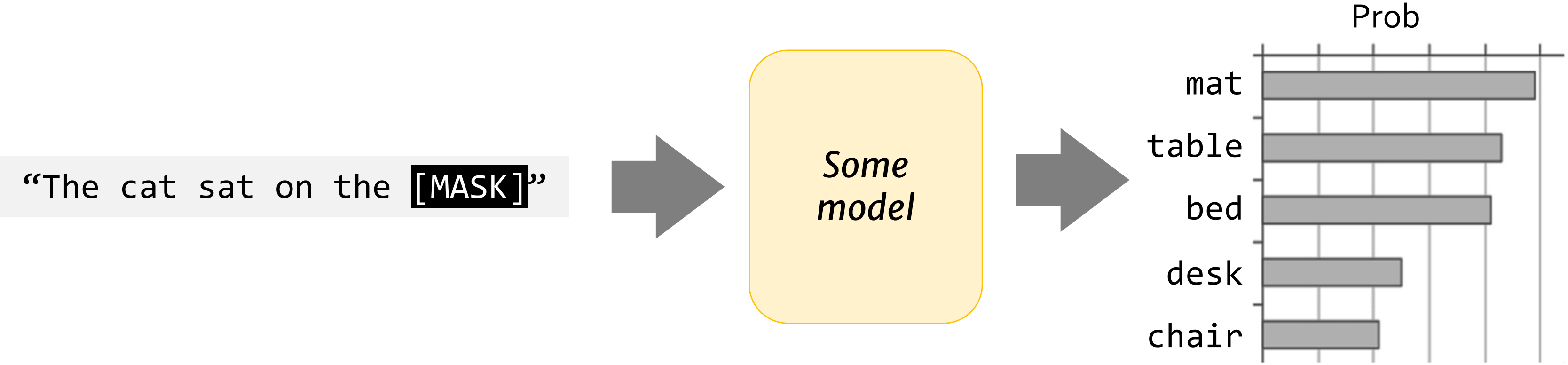
$$\mathbf{P}(\underbrace{\text{mat}}_{\text{next word}} \mid \underbrace{\text{The cat sat on the}}_{\text{context}})$$

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} \mid \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} | \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

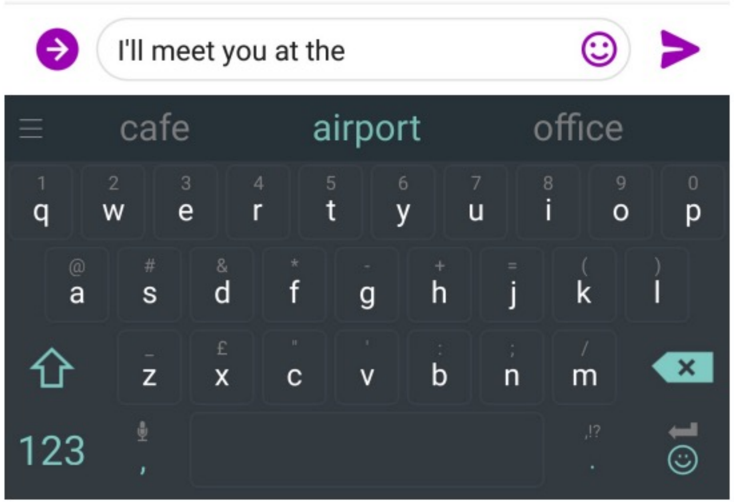
But more broadly,

$$\mathrm{P}(X_1, \ldots, X_t)$$



Language Modeling ≜ learning prob distribution over language sequence

You Use Language Model Every day!



You Use Language Model Every day!

Language Models: A History

- Shannon (1950): The predictive difficulty (entropy) of English.

Prediction and Entropy of Printed English

By C. E. SHANNON

(*Manuscript Received Sept. 15, 1950*)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.

$$\mathbf{P}(X_t | X_1, \ldots, X_{t-1})$$

Andrey Markov

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

1st order approximation: $\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \text{the})$

2nd order approximation: $\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \text{on the})$

Then, approximate these with counts:

$$\text{P}(\text{mat} \mid \text{on the}) \approx \frac{\text{count(“on the mat”)}}{\text{count(“on the”)}}$$

N-gram Language Models

- **Terminology:** *n*-gram is a chunk of *n* consecutive words:
    - unigrams: "cat", "mat", "sat", …
    - bigrams: "the cat", "cat sat", "sat on", …
    - trigrams: "the cat sat", "cat sat on", "sat on the", …
    - four-grams: "the cat sat on", "cat sat on the", "sat on the mat", …
- *n*-gram language model: $$\mathrm{P}(X_t \mid X_1, \ldots, X_{t-1}) \approx \mathrm{P}(X_t \mid \overbrace{X_{t-n+1}, \ldots, X_{t-1}}^{n-1 \text{ elements}})$$

<u>Challenge:</u> Increasing n makes sparsity problems worse.
Typically, we can't have n bigger than 5.

Some partial solutions (e.g., smoothing and backoffs)
though still an open problem.

N-Gram Models in Practice

- You can build a simple **tri**gram Language Model over a 1.7 million words corpus in a few seconds on your laptop*

```
today the ___
```

get probability distribution

```
company   0.153
bank      0.153
price     0.077
italian   0.039
emirate   0.039
...
```

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

`today the ___`

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

<u>Sparsity problem</u>: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

$$\texttt{today } \overbrace{\texttt{the price}}^{\text{condition on this}} \texttt{ \_\_\_\_\_}$$

get probability distribution

of	0.308
for	0.050
it	0.046
to	0.046
is	0.031
...	

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

condition on this

`today the price of ___`

get probability distribution

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
...	

<u>Sparsity problem</u>: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

```
today the price of gold per ton , while production of shoe
lasts and shoe industry , the bank intervened just after it
considered and rejected an imf demand to rebuild depleted
european stocks , sept 30 end primary 76 cts a share .
```

Surprisingly grammatical!

But quite incoherent! To improve coherence, one may consider increasing larger than 3-grams, but that would worsen the sparsity problem!

* Try for yourself: https://nlpforhackers.io/language-models/

[adopted from Chris Manning]

Language Models: A History

- Probabilistic n-gram models of text generation [Jelinek+ 1980's, ...]
    - Applications: Speech Recognition, Machine Translation

532 PROCEEDINGS OF THE IEEE, VOL. 64, NO. 4, APRIL 1976

Continuous Speech Recognition by Statistical Methods

FREDERICK JELINEK, FELLOW, IEEE

Abstract—Statistical methods useful in automatic recognition of continuous speech are described. They concern modeling of a speaker and of an acoustic processor, extraction of the models' statistical parameters, and hypothesis search procedures and likelihood computations of linguistic decoding. Experimental results are presented that indicate *the power of the methods.*

utterance models used will incorporate more grammatical features, and statistics will have been grafted onto grammatical models. Most methods presented here concern modeling of the speaker's and acoustic processor's performance and should, therefore, be universally useful.

Automatic recognition of continuous (English) speech is an

Language Models: A History

- Probabilistic n-gram models of text generation [Jelinek+ 1980’s, …]
  - Applications: Speech Recognition, Machine Translation

- “Shallow” statistical language models (2000’s) [Bengio+ 1999 & 2001, …]

NeurIPS 2000

A Neural Probabilistic Language Model

Yoshua Bengio,* Réjean Ducharme and Pascal Vincent
Département d’Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal
Montréal, Québec, Canada, H3C 3J7
{*bengioy,ducharme,vincentp*}@*iro.umontreal.ca*

LMs w/ Recursive Neural Nets

- Core idea: apply a model repeatedly

outputs

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

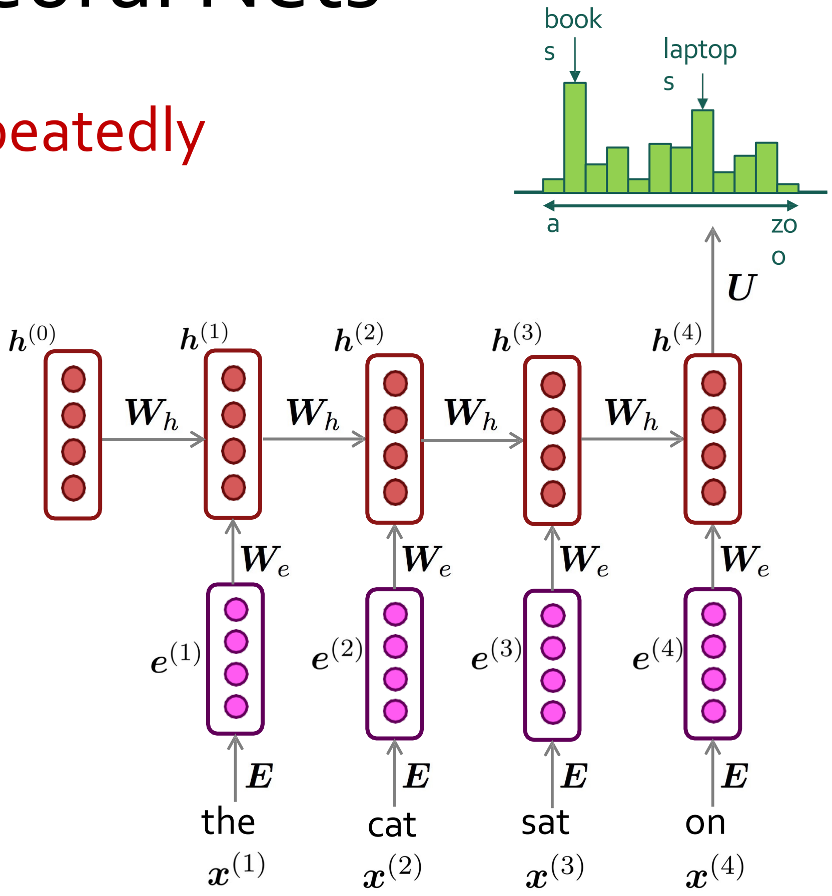
Input embedding

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



RNNs in Practice

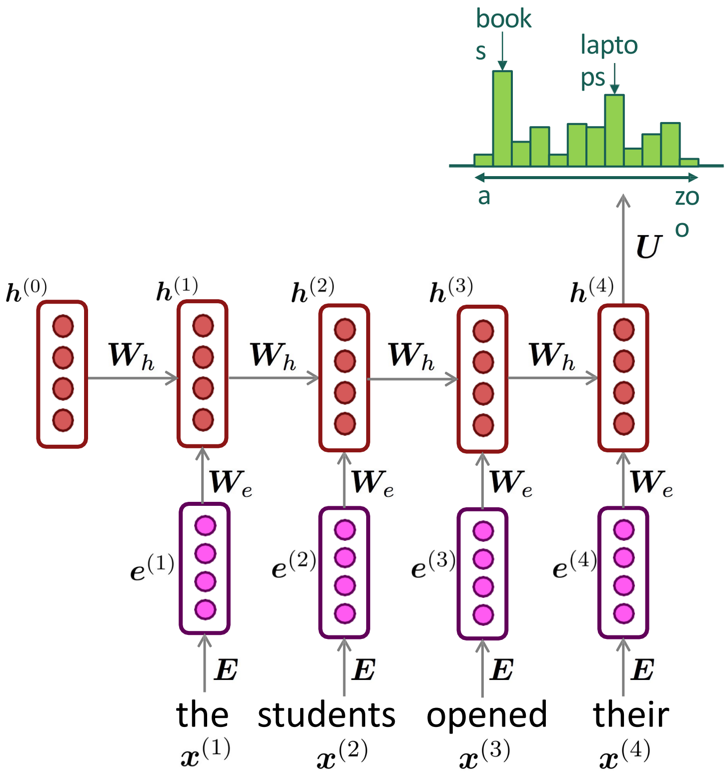
- RNN-LM trained on Obama speeches:

```
The United States will step up to the cost of a new
challenges of the American people that will share the fact
that we created the problem. They were attacked and so that
they have to say that all the task of the final days of war
that I will not be able to get this done.
```

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

RNNs: Pros and Cons

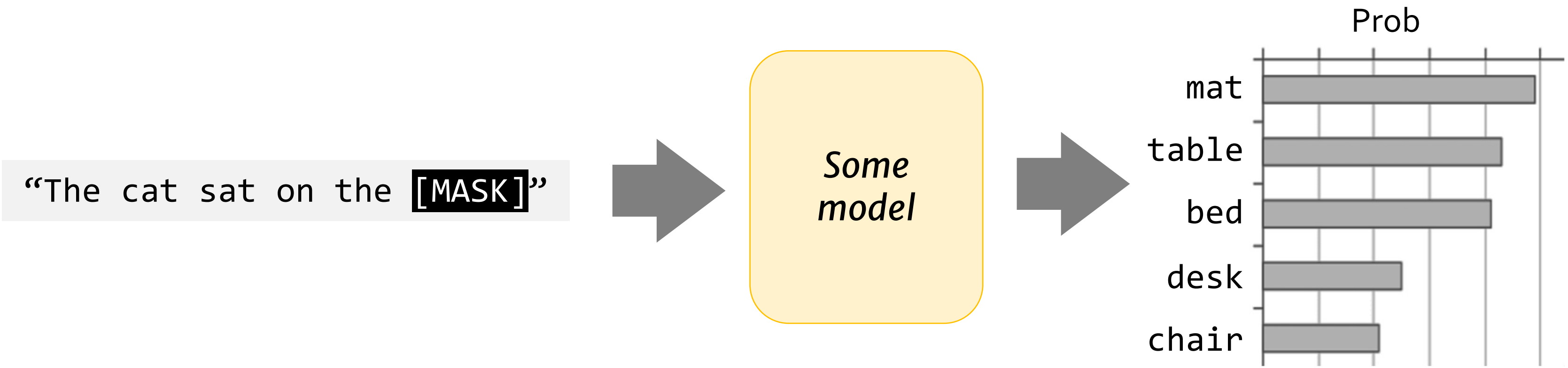
- **Advantages**:
  - Model size doesn't increase for longer inputs
  - Computation for step *t* can (in theory) use information from many steps back
- **Disadvantages**:
  - Recurrent computation is slow.
  - While RNNs in theory can represent long sequences, they quickly forget portions of the input.
  - Vanishing/exploding gradients.



[adopted from Chris Manning]

Let’s evaluate these models!

1. Train it on a suitable training documents.
2. Evaluate their predictions on different, unseen documents.



Evaluating Predictions via "Perplexity"

- A measure of how well a probability distribution predicts a sample.
- **Definition:** for a document D with words $w_1, \ldots, w_n$:

$$\text{ppl}(D) = 2^E, \text{ where } E = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

cross entropy

- In our earlier example:

$$E = -\frac{1}{6}\begin{bmatrix} \log_2 \mathbf{P}(\text{mat} \mid \text{the cat sat on the}) + \\ \log_2 \mathbf{P}(\text{the} \mid \text{the cat sat on}) + \\ \log_2 \mathbf{P}(\text{on} \mid \text{the cat sat}) + \\ \log_2 \mathbf{P}(\text{sat} \mid \text{the cat}) + \\ \log_2 \mathbf{P}(\text{cat} | \text{the}) + \\ \log_2 \mathbf{P}(\text{the}) \end{bmatrix}$$

Perplexity: Edge Cases

- **Definition:** for a document D with words $w_1, \ldots, w_n$:

$$\text{ppl}(D) = 2^x, \text{ where}$$
$$x = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- If $P(.)$ uninformative: $\forall w \in V$: $\mathbf{P}(w|w_{1:i-1}) = \frac{1}{|V|} \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{2} n \log_2 \frac{1}{|V|}} = |V|$

- If $P(.)$ is exact: $\mathbf{P}(w_i|w_{1:i-1}) = 1 \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{2} n \log_2 1} = 1$

Perplexity ranges between 1 and |V|.

Lower perplexity is good!

Perplexity is a measure of model's uncertainty about next word (aka "average branching factor")

Evaluation LMs with Perplexity (2016)

n-gram model →

Increasingly complex RNNs

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Source: https://engineering.fb.com/2016/10/25/ml-applications/building-an-efficient-neural-language-model-over-a-billion-words/

Summary So Far

- Language Model (LM), a predictive model for language
- N-gram models, early instances of LMs (until mid 2000's)
- Recurrent Neural Network: A family of neural networks that can be recursively applied to a given context.
- RNN-LMs were shown to be effective LMs (2000's - 2010's)

RNNs, Back to the Cons

- While RNNs in theory can represent long sequences, they quickly forget portions of the input.

Some suggested solutions:

- Changes to the **architecture** makes it **easier** for the RNN to preserve information over many timesteps
  - Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997, Gers+ 2000]
  - Gated Recurrent Units (GRU) [Cho+ 2014]

Many of these variants were the dominant architecture of In 2013–2015.

RNNs, Back to the Cons

- While RNNs in theory can represent long sequences, they quickly forget portions of the input.
- Vanishing/exploding gradients

Some suggested solutions:

- Changes to the **architecture:**
  - lots of new **deep architectures** (RNN or otherwise) add more **direct connections**, thus allowing the gradient to flow)
- Changes to **training**: gradient clipping.

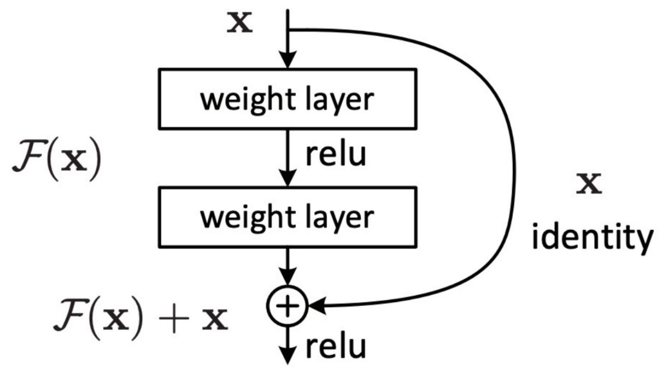


Figure 2. Residual learning: a building block.

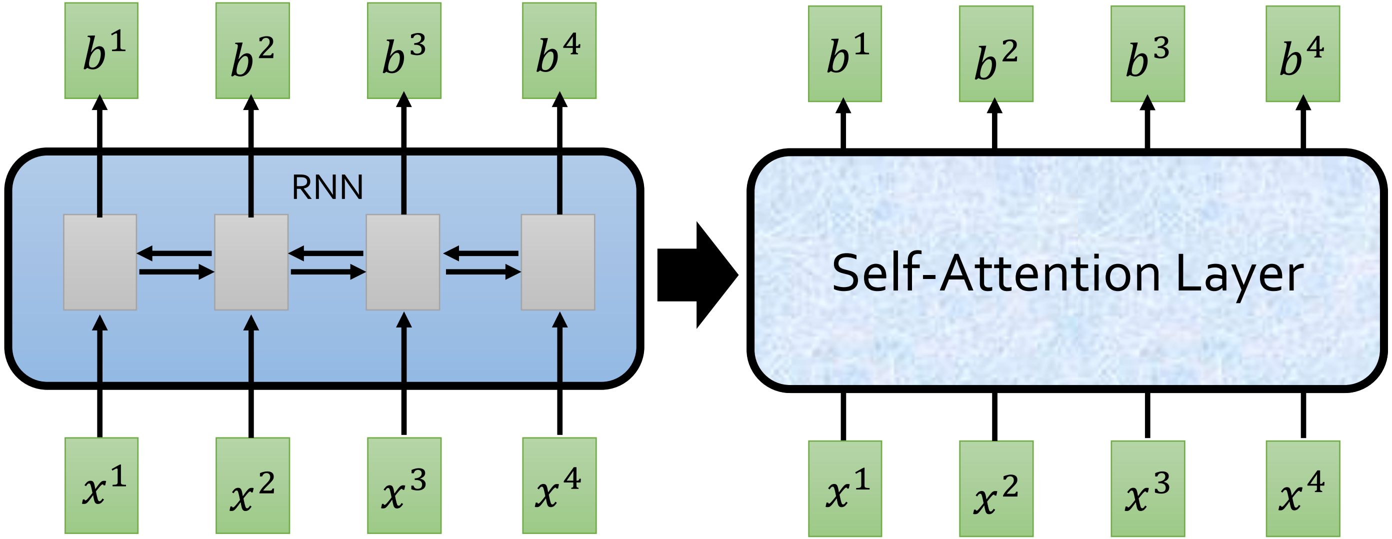
"Deep Residual Learning for Image Recognition", He et al, 2015. https://arxiv.org/pdf/1512.03385.pdf

RNNs, Back to the Cons

- While RNNs in theory can represent long sequences, they quickly forget portions of the input.
- Vanishing/exploding gradients
- Difficult to parallelize

Self-Attention

- b^i is obtained based on the whole input sequence.
- can be parallelly computed.



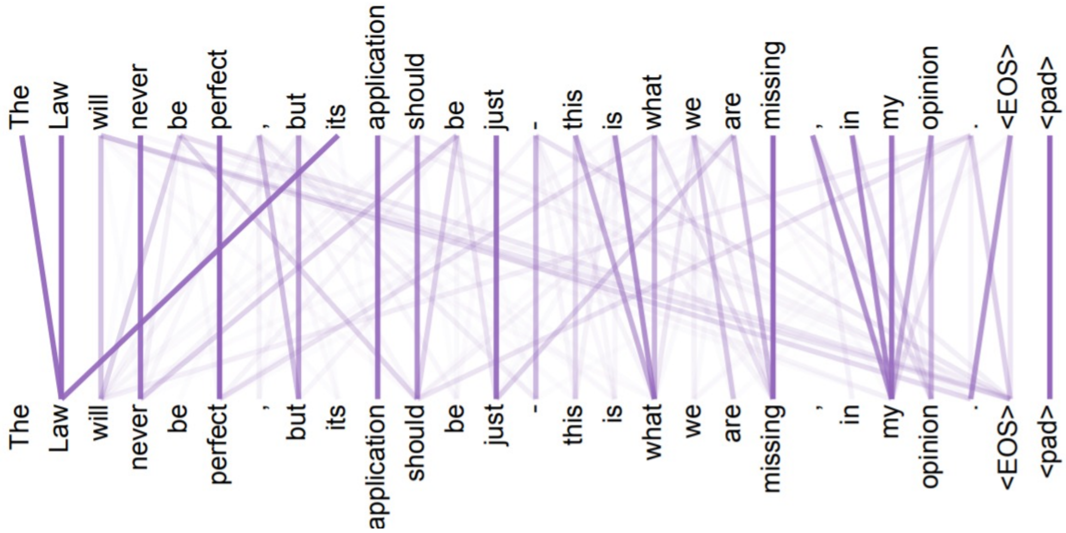
Idea: replace any thing done by RNN with self-attention.

"Neural machine translation by jointly learning to align and translate" Bahdanau etl. 2014;
"Attention is All You Need" Vaswani et al. 2017

[adopted from Hung-yi Lee]

Attention

- <u>Core idea</u>: on each step of the decoder, *use direct connection* to *focus ("attend") on a particular part* of the context.



[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

Defining Self-Attention

- **Terminology:**
  - **Query**: to match others
  - **Key**: to be matched
  - **Value**: information to be extracted

- **Definition:** Given a set of vector **values**, and a vector **query**, *attention* is a technique to compute a weighted sum of the **value**, dependent on the **query**.

[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

q: query (to match others)

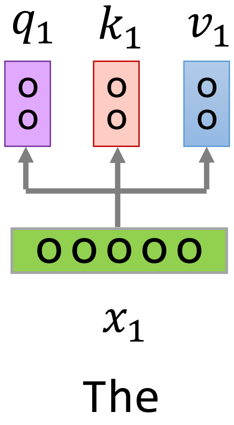
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

$$v_i = W^v x_i$$



q: query (to match others)

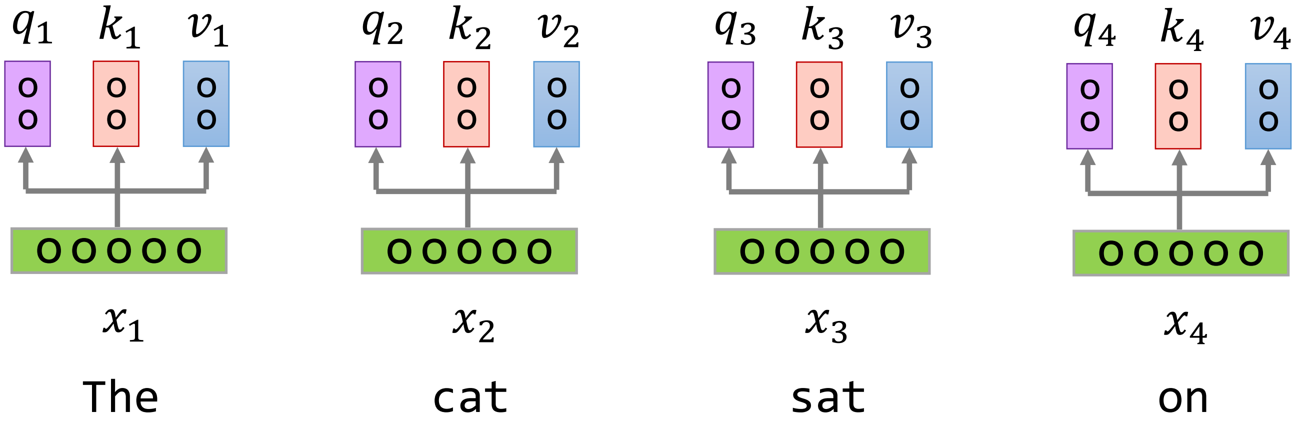
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

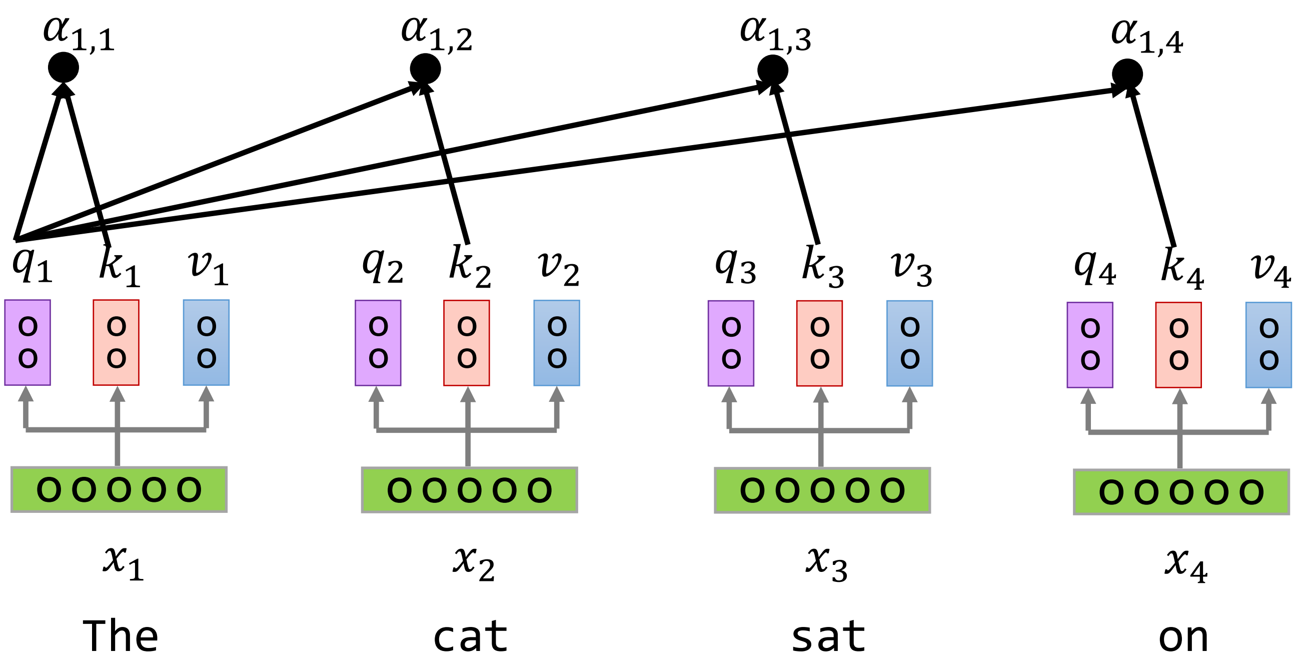
$$v_i = W^v x_i$$



$$\alpha_{1,i} = \underbrace{q^1 \cdot k^i / \alpha}_{\text{Scaled dot product}}$$

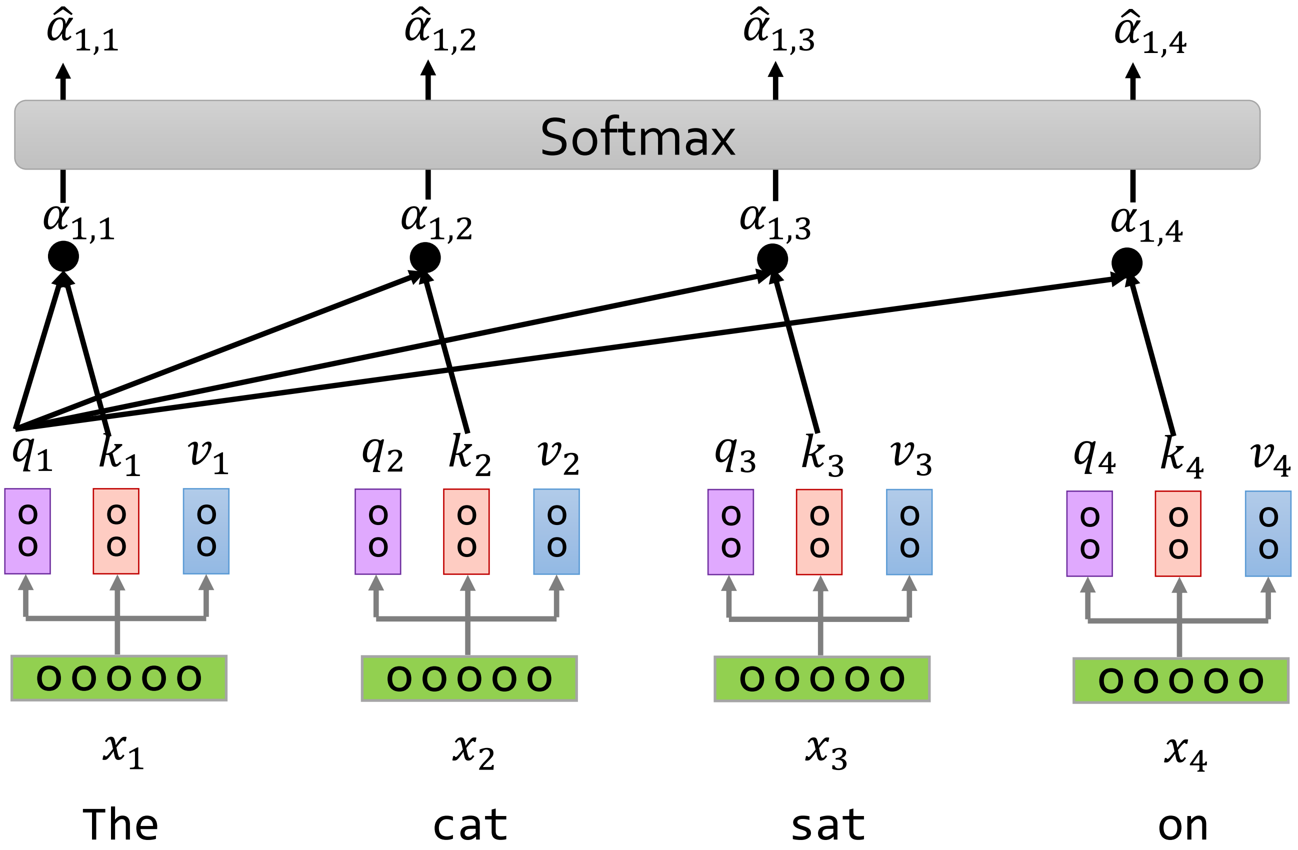
q: query (to match others)
k: key (to be matched)
v: value (information to be extracted)

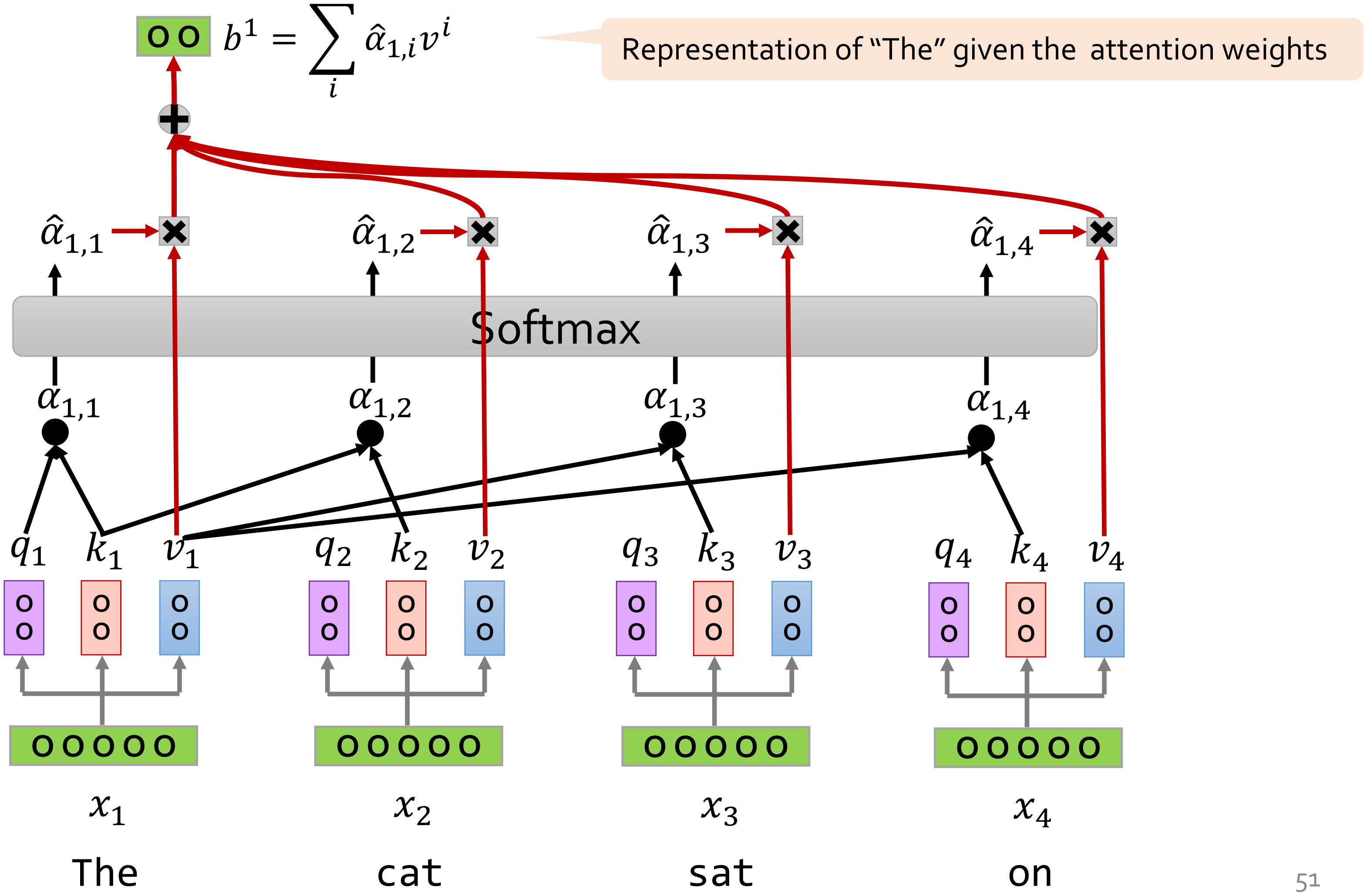
How much should "The" attend to other positions?



$$\sigma(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

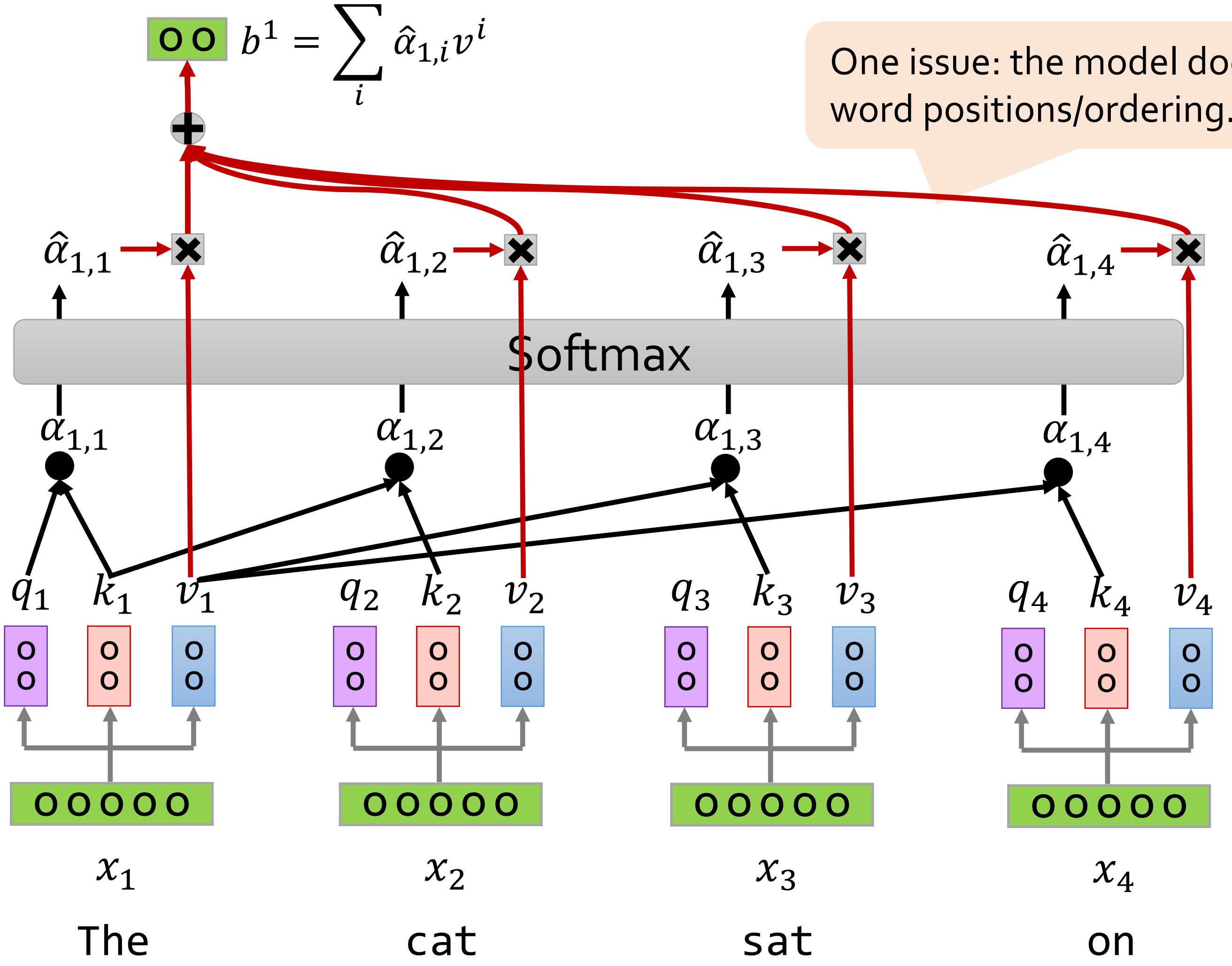
How much should "The" attend to other positions?





$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
Representation of "The" given the attention weights
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
x_1
x_2
x_3
x_4
The
cat
sat
on

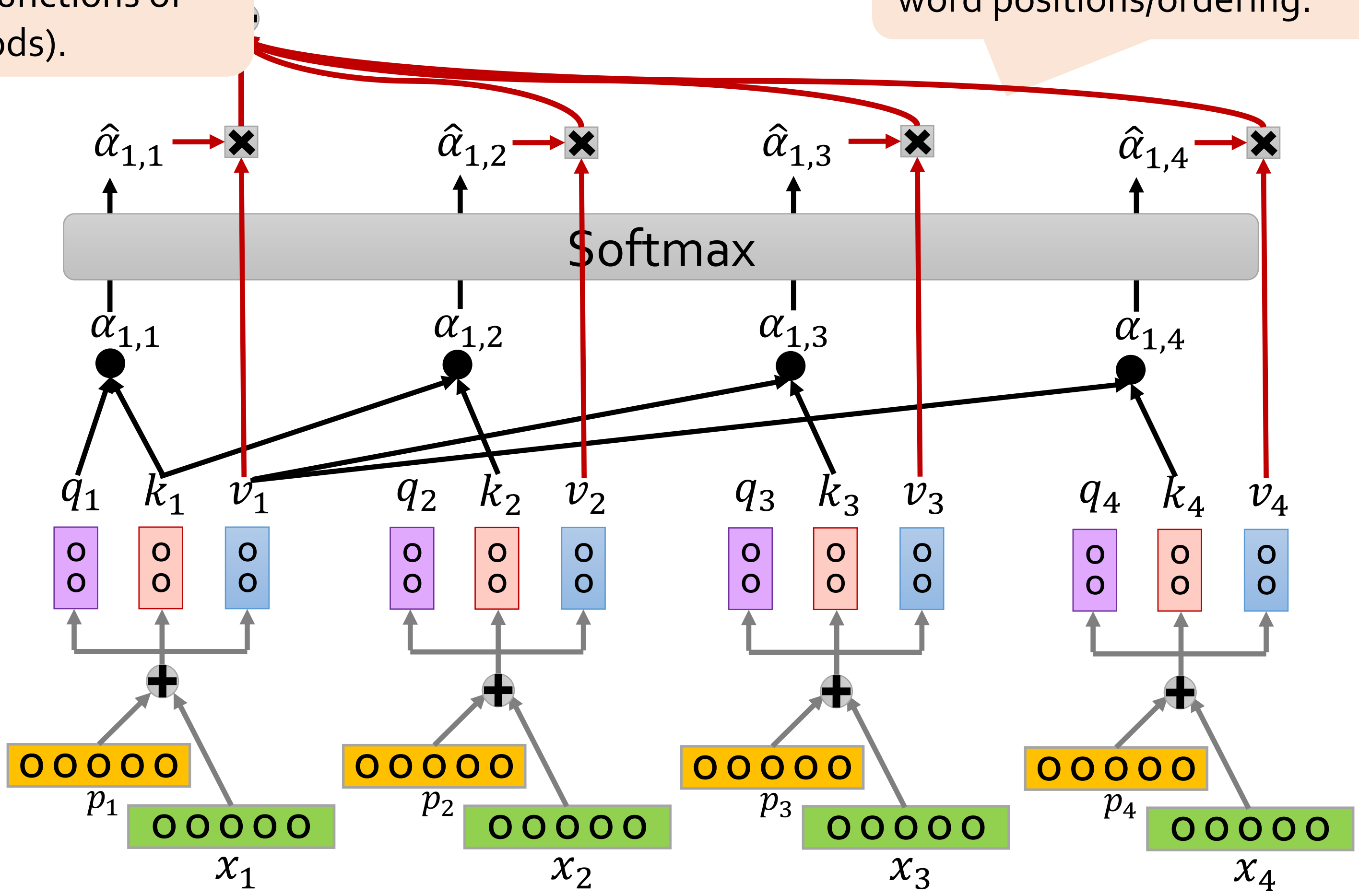
One issue: the model doesn't know word positions/ordering.



p_i are unique fixed vectors (sinusoidal functions of varying periods).

One issue: the model doesn't know word positions/ordering.

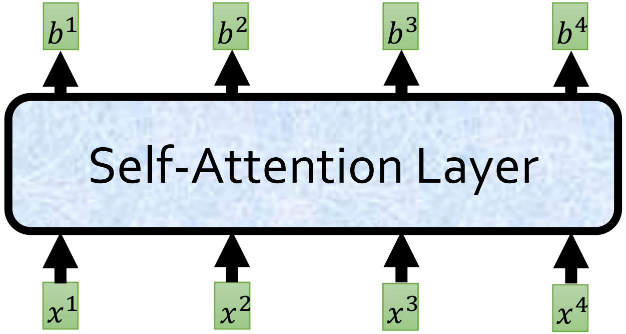
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



Self-Attention: Back to Big Picture

- Attention is a way to focus on particular parts of the input
- Can write it in matrix form:

$$\boldsymbol{b} = \mathrm{softmax}\left(\frac{QK^{\mathrm{T}}}{\alpha}\right)V$$



- Efficient implementations
- Better at maintaining long-distance dependances in the context.

Self-Attention

$$\boldsymbol{b} = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\alpha}\right)V$$

hardmaru
@hardmaru

The most important formula in deep learning after 2018

Self-Attention

What is self-attention? Self-attention calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations. Formally, an input sequence of n tokens of dimensions d, $X \in \mathbf{R}^{n\times d}$, is projected using three matrices $W_Q \in \mathbf{R}^{d\times d_q}$, $W_K \in \mathbf{R}^{d\times d_k}$, and $W_V \in \mathbf{R}^{d\times d_v}$ to extract feature representations Q, K, and V, referred to as query, key, and value respectively with $d_k = d_q$. The outputs Q, K, V are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{1}$$

So, self-attention can be written as,

$$S = D(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \tag{2}$$

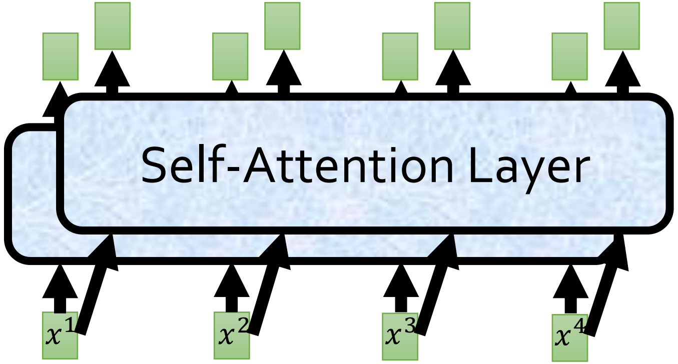
where softmax denotes a *row-wise* softmax normalization function. Thus, each element in S depends on all other elements in the same row.

9:08 PM · Feb 9, 2021 · Twitter Web App

553 Retweets **42** Quote Tweets **3,338** Likes

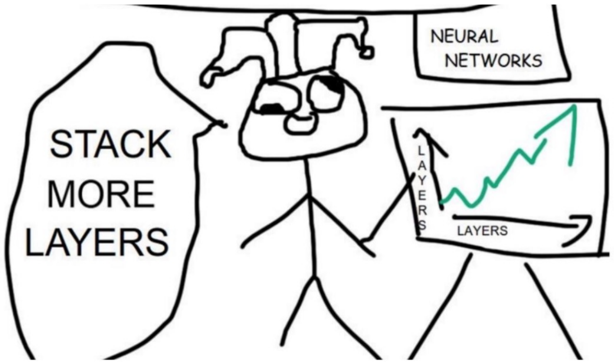
Multi-Headed Self-Attention

- Multiple parallel attention layers is quite common.
  - Each attention layer has its own parameters.

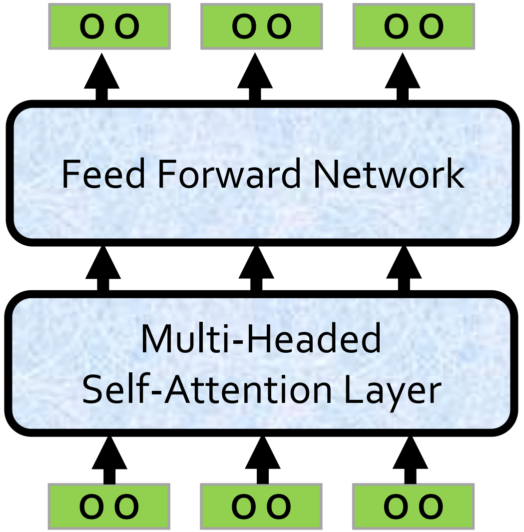


[Vaswani et al. 2017]

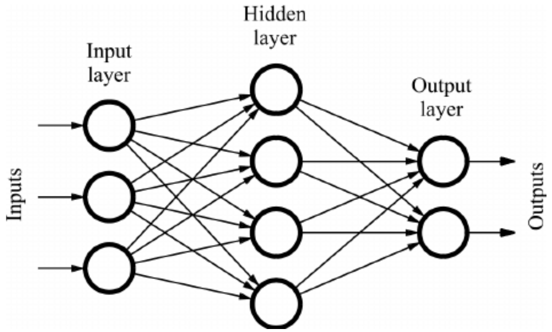
How Do We Make it Deep?



- Add a feed-forward network on top it to add more capacity/expressivity.
- Repeat!



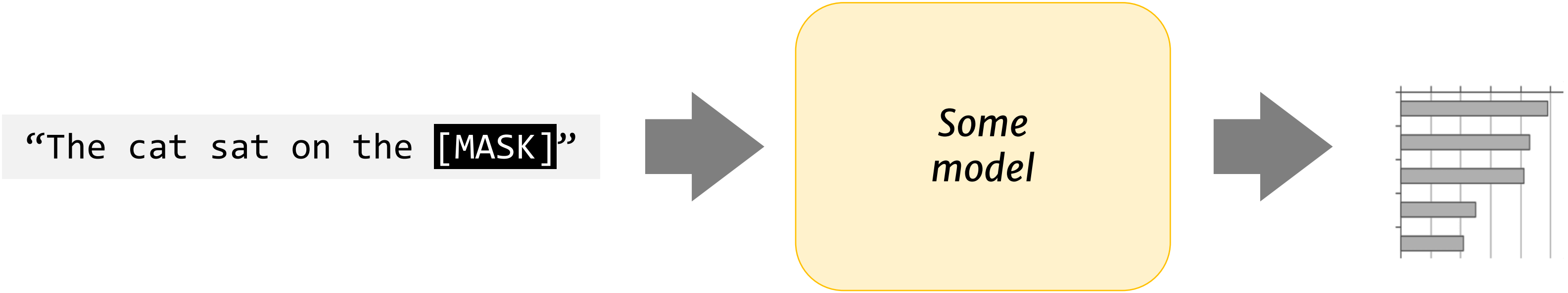
Feedforward Net: Refresher



A fully-connected network of nodes and weights.

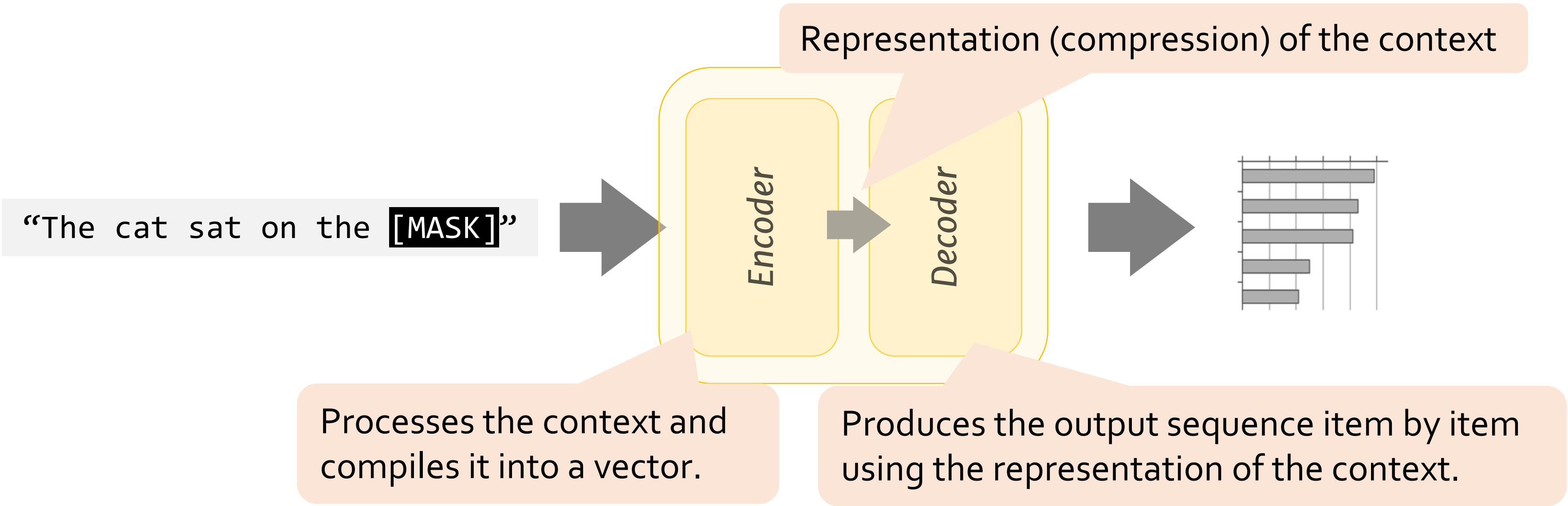
Encoder-Decoder Architectures

- It is useful to think of generative models as two sub-models.

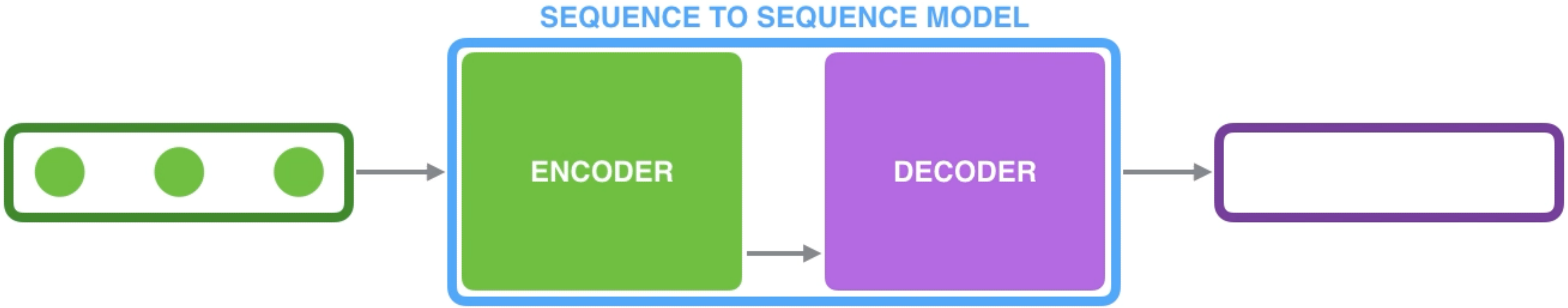


Encoder-Decoder Architectures

- It is useful to think of generative models as two sub-models.



Encoder-Decoder Architectures



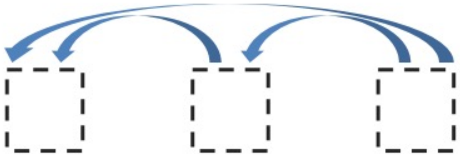
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Transformer [Vaswani et al. 2017]

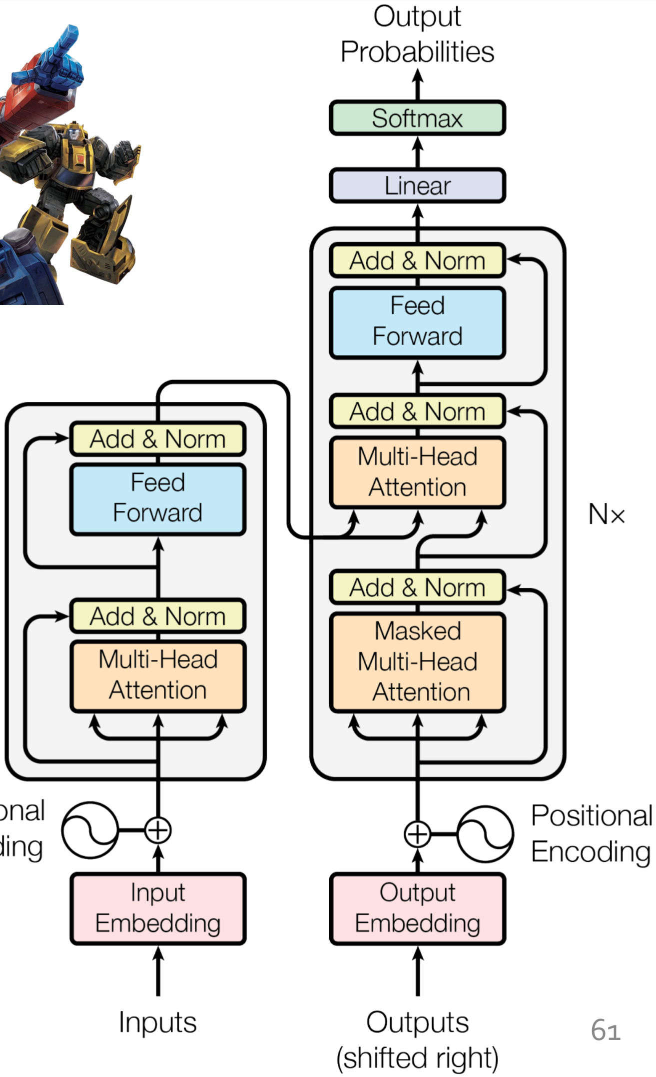
- An encoder-decoder architecture built with attention modules.
- 3 forms of attention

Encoder-Decoder Attention

Encoder Self-Attention



MaskedDecoder Self-Attention



Impact of Transformers

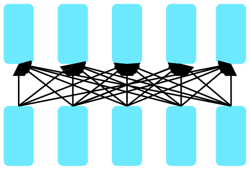
- Let to better predictive models of language!

Model	Layers	Heads	Perplexity
LSTMs (Grave et al., 2016)	-	-	40.8
QRNNs (Merity et al., 2018)	-	-	33.0
Transformer	16	16	19.8

["Efficient Content-Based Sparse Attention with Routing Transformers" Roy et al. 2020]

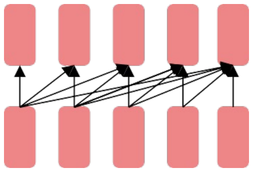
Impact of Transformers

- A building block for a variety of LMs



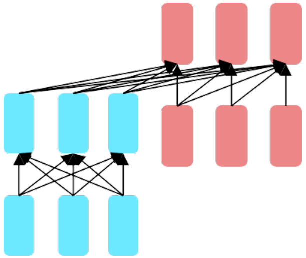
Encoders

- **Examples:** BERT, RoBERTa, SciBERT.
- Captures bidirectional context. Wait, how do we pretrain them?



Decoders

- **Examples:** GPT-2, GPT-3, LaMDA
- Other name: **causal or auto-regressive language model**
- Nice to generate from; can't condition on future words



Encoder-Decoders

- **Examples:** Transformer, T5, Meena
- What's the best way to pretrain them?

Wrapping it up

- Yaaay we know Transformers now! 🥳

- Next sessions: several extensions on Transformers.