Embeddings, Attention, and Transformers

Mathias Unberath, PhD
Assistant Professor
Dept of Computer Science
Johns Hopkins University

Embeddings, Attention, and Transformers

Problems with RNNs for Language

and Otherwise

Challenges When Modeling Sequences with RNNs

Sequences can have arbitrary length

Consider

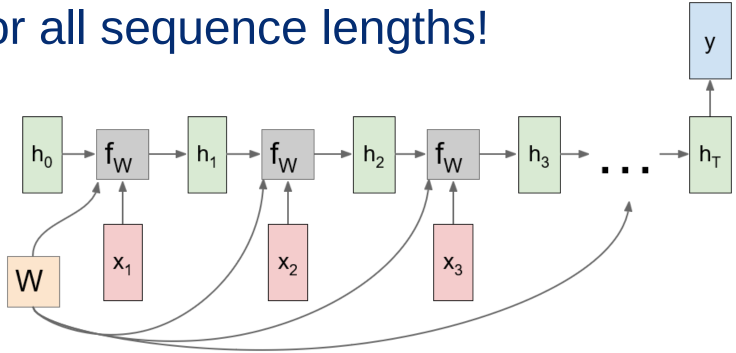
"Reasons were strongly urged both for and against the plan; but Mr. Lincoln finally decided and explained that while he himself was not afraid he would be assassinated, nevertheless, since the possibility of danger had been made known from two entirely independent sources, and officially communicated to him by his future prime minister and the general of the American armies, he was no longer at liberty to disregard it; that it was not the question of his private life, but the regular and orderly transmission of the authority of the government of the United States in the face of threatened revolution, which he had no right to put in the slightest jeopardy. " – A Short Life of Abraham Lincoln

versus

"I went to the garden to pick flowers."

→ In RNNs, same dimensionality of hidden state for all sequence lengths!

→ How to ensure context propagates?

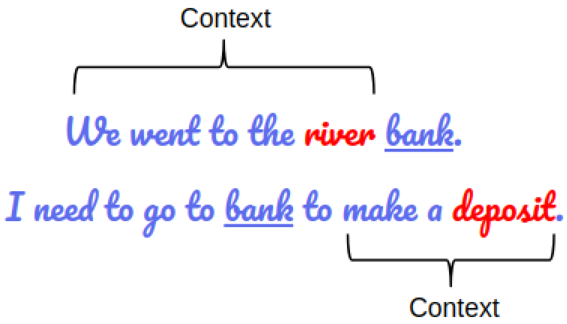


Challenges When Modeling Sequences with RNNs

A related problem: Ambiguity

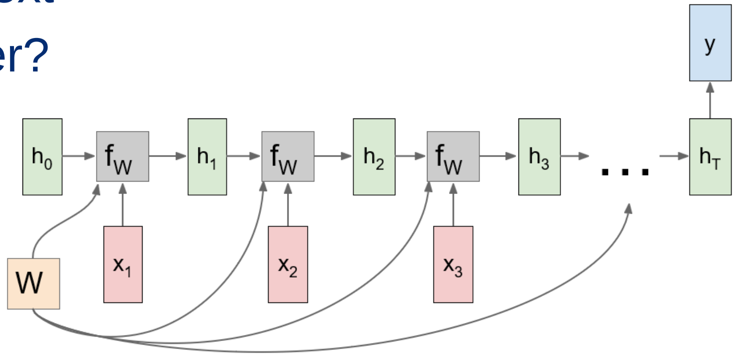
→ In RNNs, same dimensionality of hidden state for all sequence lengths!

→ How to ensure context propagates?



→ "Bank" can only be disambiguated through context

→ What happens to context several sentences later?



Challenges When Modeling Sequences with RNNs

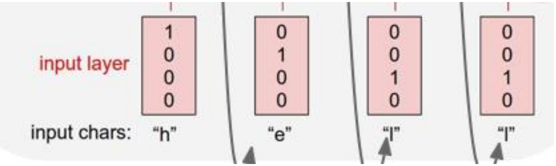
Input representations

One-hot does not scale well

It also does not have nice properties

Let’s start with a better representation!

Vocabulary:
[h,e,l,o]



Embeddings, Attention, and Transformers

The Need for Embeddings

What is embedding?

- Simply put: a fancy lookup table
- Mapping from discrete, categorical variables (e.g. one hot encoded) to continuous vector space
- Sometimes called embedding matrix since it can also be represented as a sparse matrix multiplication

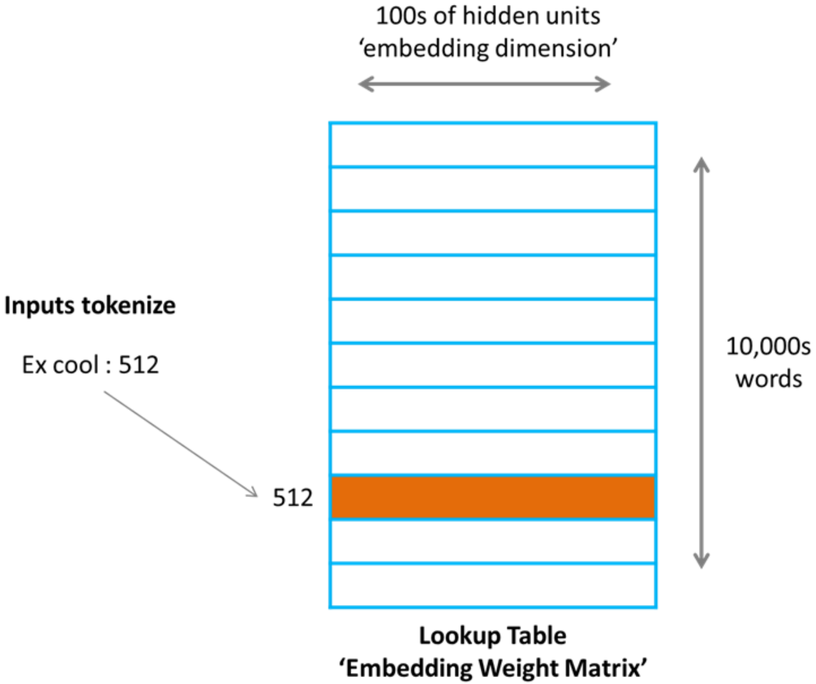


Image from: https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81

Limitation of one-hot encoding

- High dimensional sparse vectors take up a lot of space
- Categories that are similar are not closer to each other
- Want meaningful dimensionality reduction
- Higher dot product = more similar

One-hot encoding

		cat	mat	on	sat	the
the	=>	0	0	0	0	1
cat	=>	1	0	0	0	0
sat	=>	0	0	0	1	0
...		...				

A 4-dimensional embedding

cat	=>	1.2	-0.1	4.3	3.2
mat	=>	0.4	2.5	-0.9	0.5
on	=>	2.1	0.3	0.1	0.4
...		...			

Why is embedding better?

- A way to represent words so it's not just what it is, but also what it means

- More compact representation
  - 50 length embedding on 37 000 books
  - Clear clusters by genre, learned semantic meaning

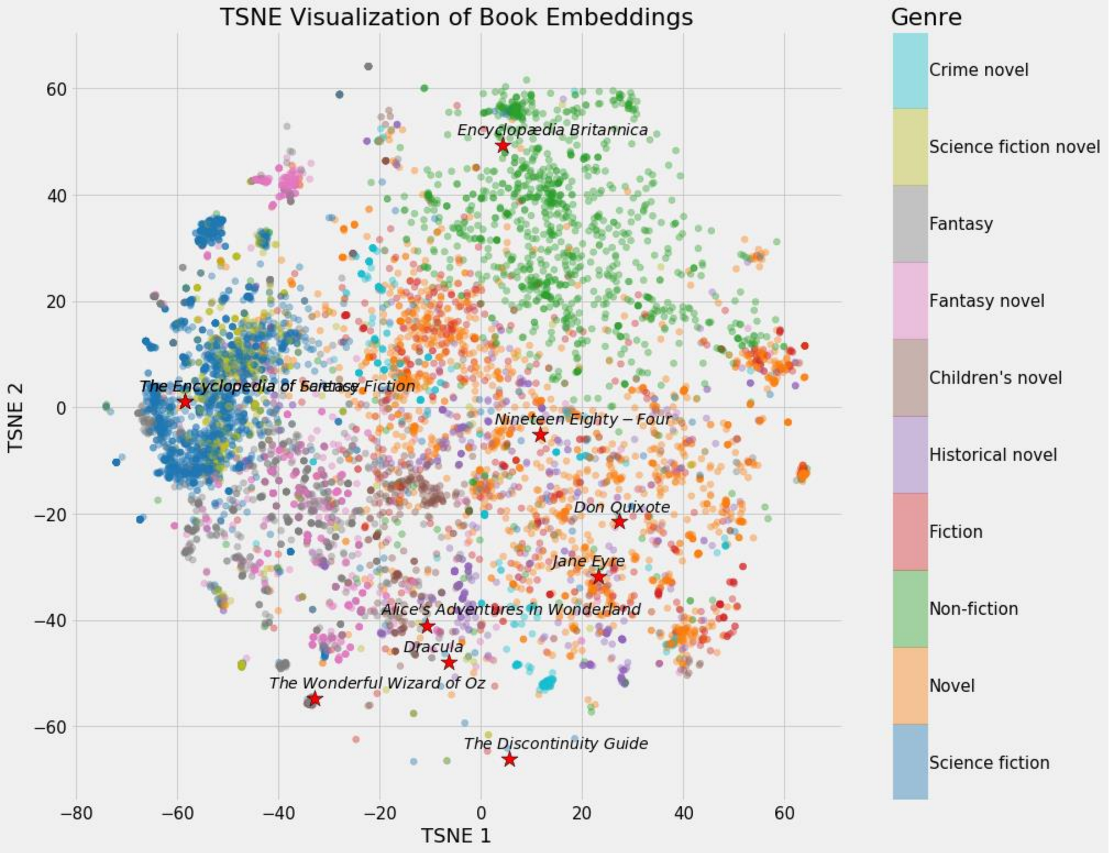


Image from: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526

Embeddings, Attention, and Transformers

How do we find good embeddings?

Word2Vec

- Words that are used in the same context are similar

→ Even if we've only seen dogs, we should know something about cats
"[] are furry, four-legged, and people love to have them as pets!"

- What is a word's context?
  – Ex: Let's say we have a window size of 2 and the following sentence
    - "the *professor* is scribbling on the board and the *student* is writing down the scribbles"
  – Context of the *professor:* "the", "is", and "scribbling"
  – Context of *student*: "and", "the", "is", and "writing"

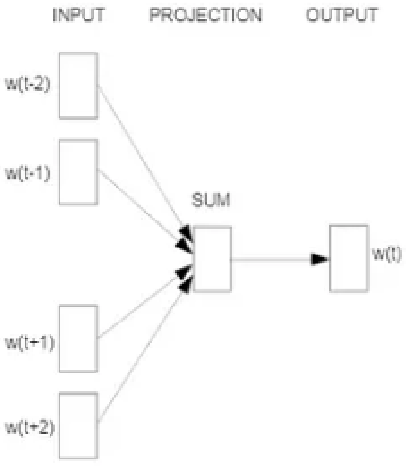
Word2Vec

Continuous bag of words (CBOW)

- Idea: Predict the word from its context!

- Rely on simple few-layer network with cross-entropy loss
  - Embed all context words
  - Then average all embeddings
  - Predict word from context

- Then, use weights to generate embeddings



CBOW

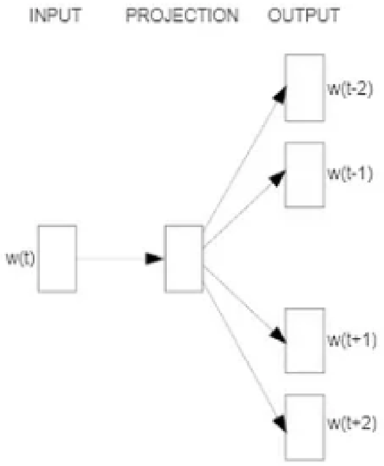
Word2Vec

Skip-gram

- Idea: Predict the context from the word!

- Rely on simple few-layer network with cross-entropy loss
  - Embed the target word
  - Evaluate loss over context words
  - Average gradients over context

- Then, use weights to generate embeddings



Skip-gram

Global Vectors for Word Representation (GloVe)

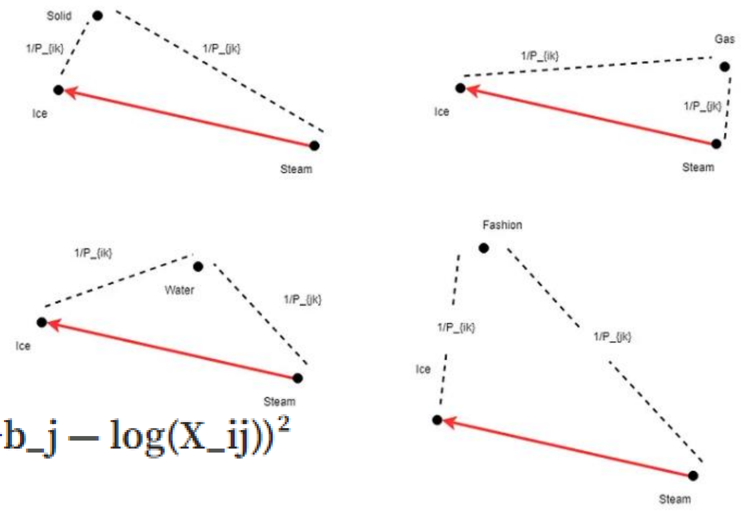
- Idea: Word-word co-occurrence probabilities carry potential for embedding

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

- Embedding function $F(w_i, w_j, u_k) = P_{ik}/P_{jk}$,
  and restructure to $F(w_i - w_j, u_k) = P_{ik}/P_{jk}$

- Distance (dashed line) changes with words k
- Always normalize w.r.t. red line

- After some massaging: $J(w_i, w_j) = (w_i^* u_j + b_i + b_j - \log(X_{ij}))^2$



Behaviour of vector distances to a probe word w.r.t. $w_i - w_j$

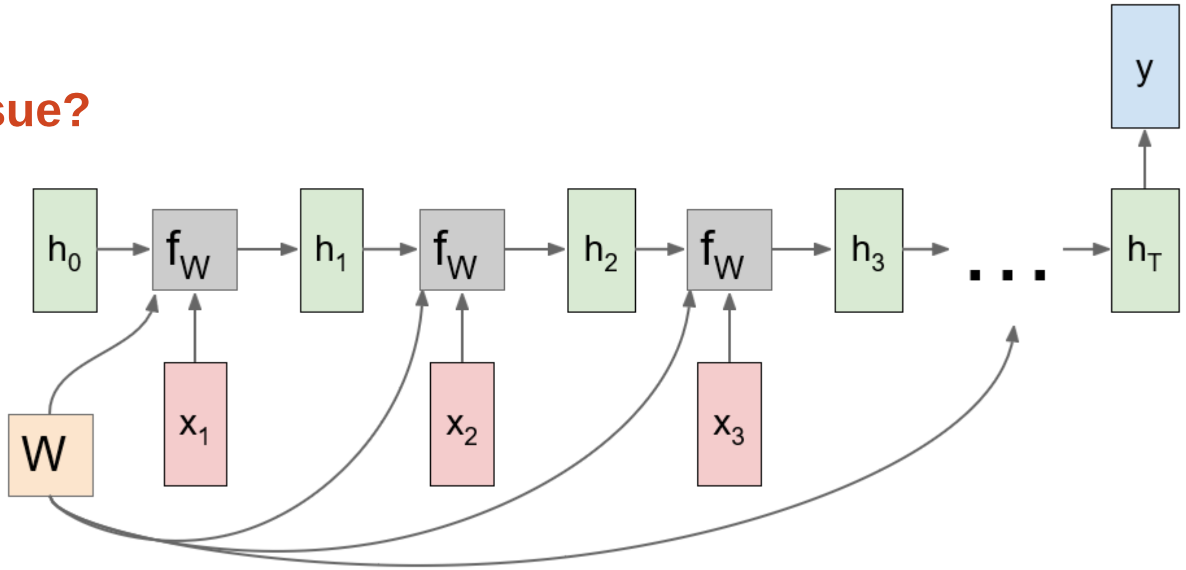
Embeddings, Attention, and Transformers

From RNNs to Transformers

The Big Picture Idea

- RNNs can only consider “context” that’s available in h
- Long sequences: Missed context!

- **How to address this issue?**



The Big Picture Idea

- RNNs can only consider “context” that’s available in h
- Long sequences: Missed context!

- **How to address this issue?**
- **Rather than memorizing everything….**
- **Would be convenient to have a mechanism to “revisit” previous values**
  - **Context can remain large**
  - **Context and importance of specific words in context can change**

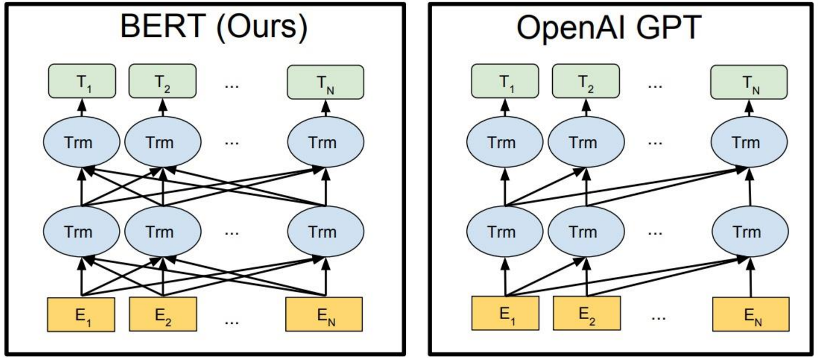
The Big Picture Idea

An overview of the transformer architecture

- Representations are computed “per token”
- Similar to RNNs, the parameters are the same at every time point
- Different from RNNs, however,
  - Parameters act not only on hidden state
  - Architecture uses all/previous tokens to compute representation

- Key ingredient of this architecture?

→ **Attention**



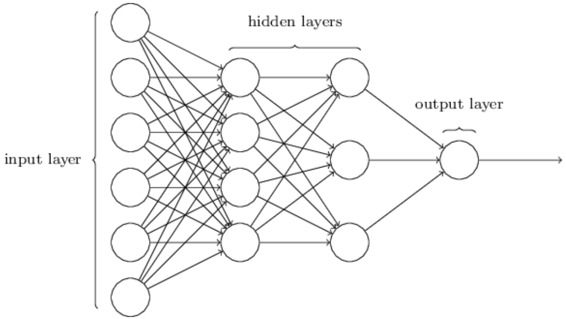
Embeddings, Attention, and Transformers

The Attention Mechanism

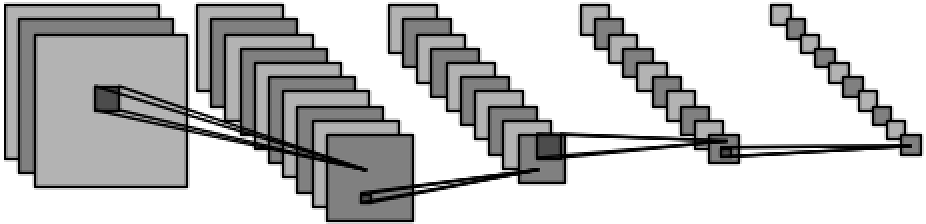
Updating Representations

Until now

- Perceptrons
  - Don’t scale well



- Convolutions
  - Build receptive field slowly

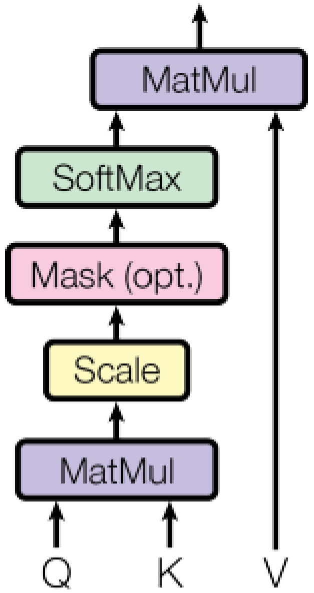


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *NeurIPS*

Updating Representations

Now: (Self-)Attention

- Representations x_i
- From those, we compute (using parameters)
  - Query q_i
  - Key k_i
  - Value v_i
- Then: *attention (q, k, v) = Softmax (q * k^T) * v*

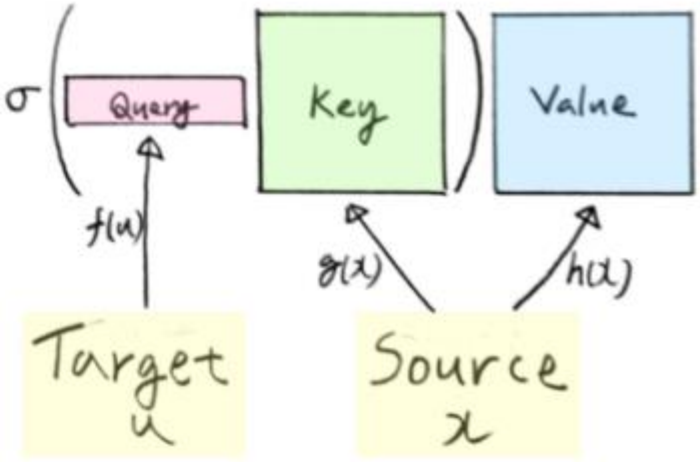


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *NeurIPS*

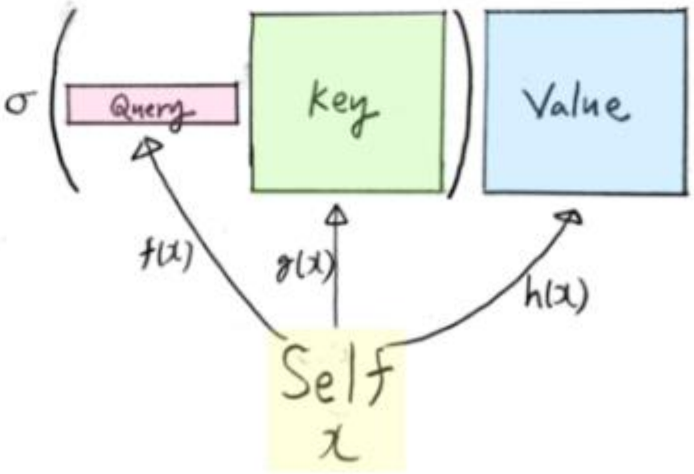
Updating Representations

Now: Self- and Cross-attention

Cross-attention

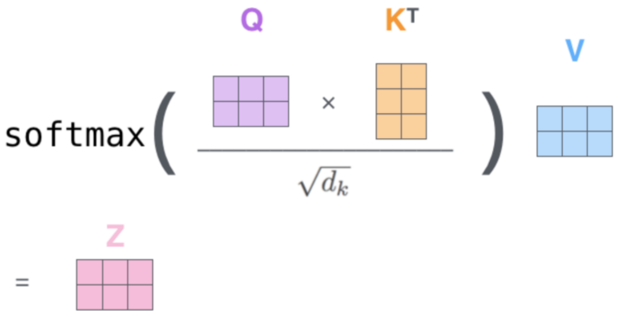


Self-attention

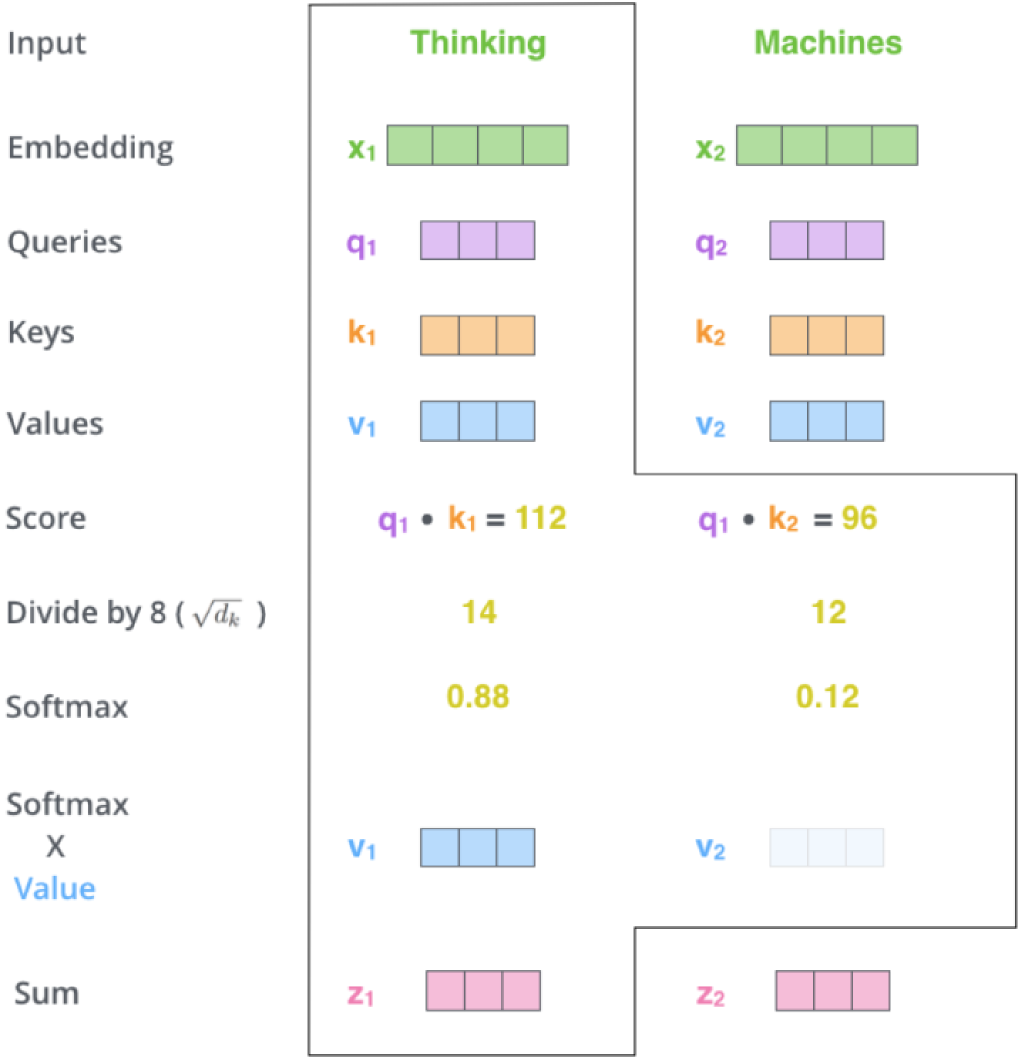


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *NeurIPS*

An Example



Q
K^T
V
softmax
$\sqrt{d_k}$
Z
=

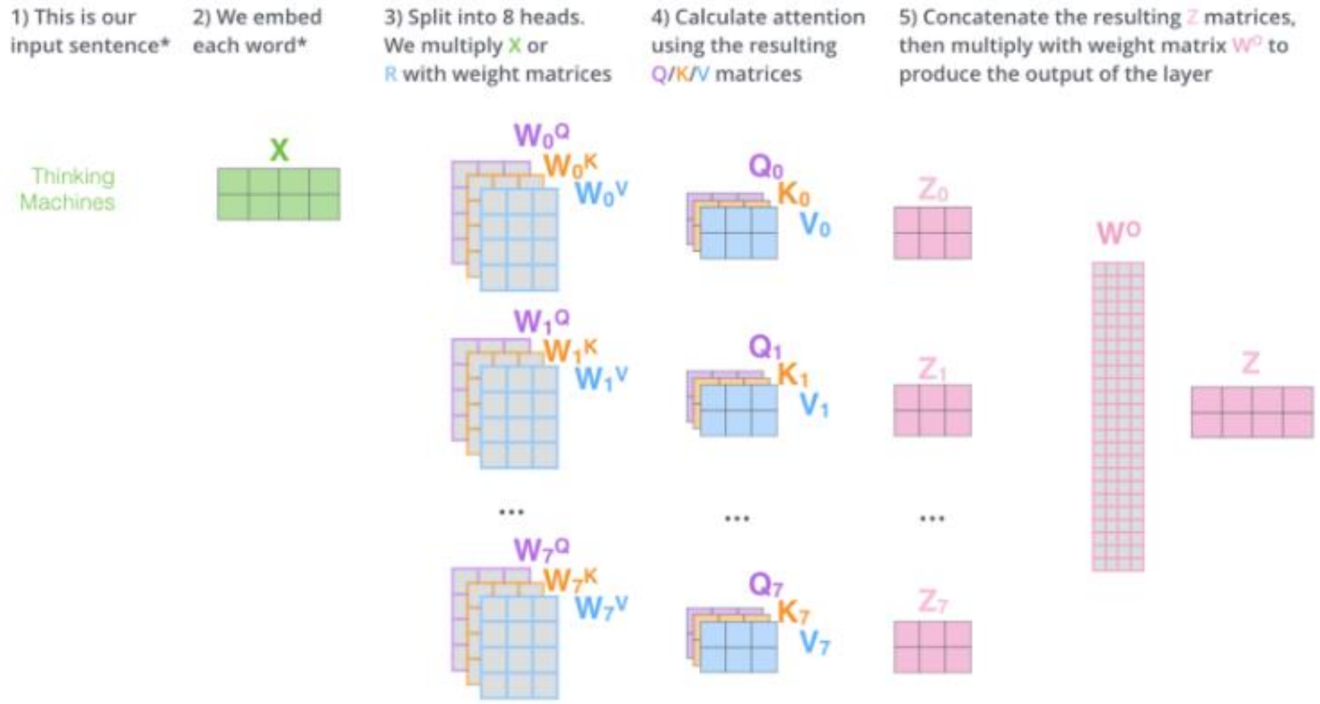


Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \bullet k_1 = 112$
$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v_1
v_2
Sum
z_1
z_2

Multi-head Attention

Can there be too much of a good thing?

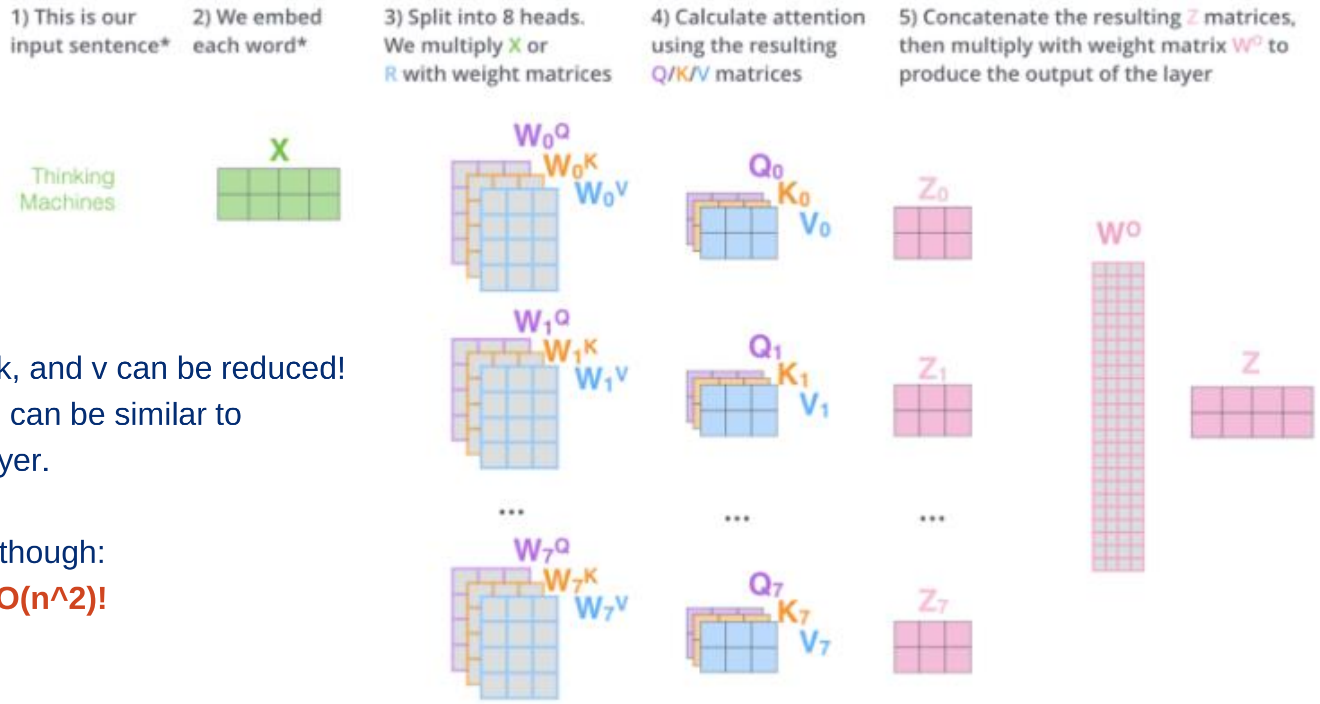
- Calculate multiple attention scores
- Merge them via concatenation and feedforward projection using W^0



Multi-head Attention

Can there be too much of a good thing?

- Calculate multiple attention scores
- Merge them via concatenation and feedforward projection using W^0



Dimensionality of q, k, and v can be reduced!
Overall, computation can be similar to standard attention layer.

Generally speaking, though:
Attention is costly O(n^2)!

Embeddings, Attention, and Transformers

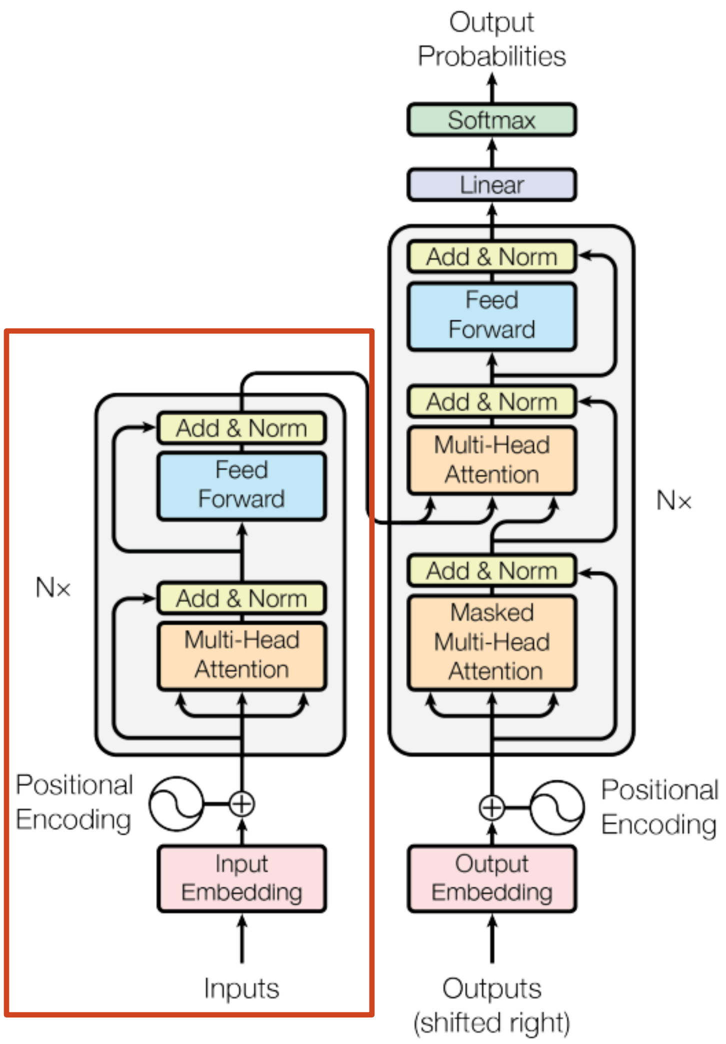
The Transformer Architecture

The Transformer

Encoding

- Input encoding
  - Tokenization
  - Token embedding
  - Positional encoding (added to embedding)

- N x Encoder block (N=6 here)
  - Multi-head attention
  - Norm and residual addition
  - Feed forward (MLP); applied to each token separately
  - Norm and residual addition



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attenti

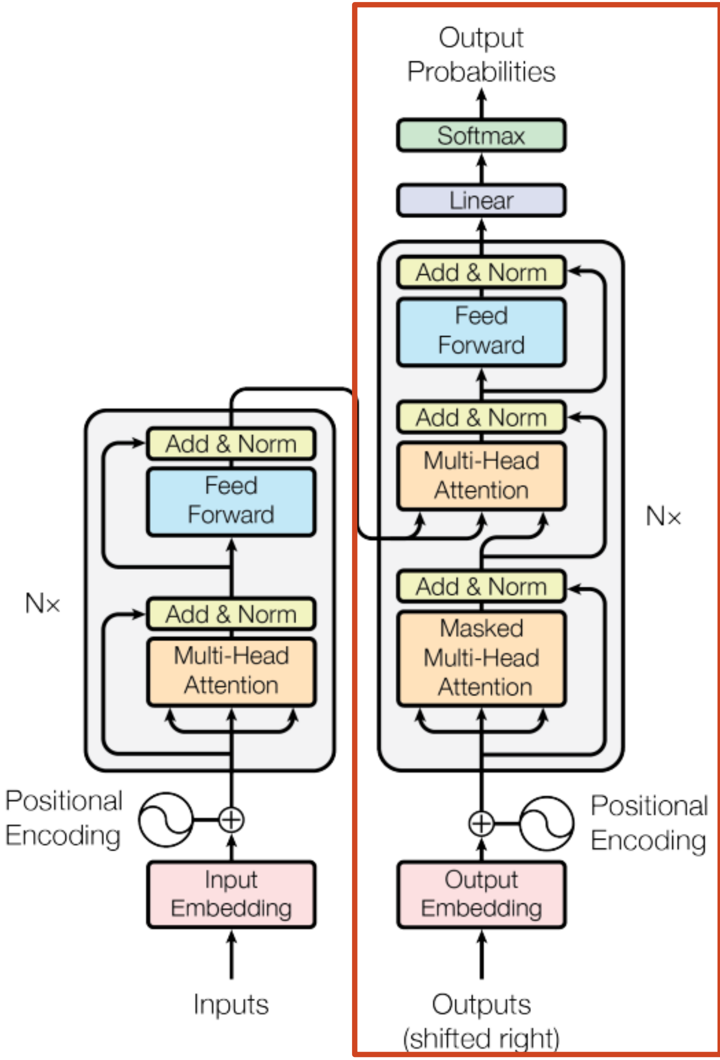
The Transformer

Encoding

- Output encoding
  - Token embedding
  - Positional encoding (added to embedding)

- N x Decoder block (N=6 here)
  - Multi-head attention, norm and residual addition
  - **Cross attention** (multi-head), norm and residual
  - Feed forward (MLP), norm and residual

- Use linear embedding transformation to map outputs to next-token probabilities



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attenti

The Transformer

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Embeddings, Attention, and Transformers

Questions?