ECE 696B: Spring 2025
Trustworthy Machine Learning

Lecture 3: *Attention & Transformers*

Instructor: Dr Ravi Tandon

Department of ECE

Lecture Outline

Paper # 1

- First to introduce “attention”
- Published in ICLR 2015
- ”Bahdanau” Attention
- Cited **37,451 times**

(as of Jan 22, 2025)

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Paper # 2

- Introduced “Dot-product” Attention
- Introduced Transformers
- Published in NIPS 2017
- Cited **149,860 times**

(as of Jan 22, 2025)

- *Update*: Cited 150,084

(as of Jan 23, 2025)

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

THE UNIVERSITY OF ARIZONA

Paper # 1: “Bahdanau et. al”

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

THE UNIVERSITY OF ARIZONA

Recap & Shortcomings of Seq2Seq

- Lecture 1: Seq2Seq (Sutskever 2014) proposed an Enc-Dec model
- **Encoder** takes the input/source sequence → converts into a fixed length representation c (denoted as v = hT in Seq2Seq paper)
- **Decoder:** Takes the c and uses it to decode the output sequentially
- Output y(t) is generated conditioned on (c, y(1), y(2),…, y(t-1))
- Typical RNN/LSTM approach: y(t) = Dec (c, s(t), y(t-1)), where
  - c = input representation
  - y(t-1) = previous generated token
  - s(t) = hidden state
- What are some issues/problems with this model ?
  - Source sequence encoded as a single fixed-length vector c.
  - Issue 1: Performance degrades as input sequences increase in length.
  - Issue 2: What if the new sequences (in test set) are longer than those in training ?

New Idea in Bahdanau et. al

- Main Idea: Represent source sequence by a sequence of vectors:

  Source Represented as: {h(1), h(2),…., h(T)}, where T = length of input sequence.

- To generate the ith output token, y(i):
  - We find the relevant information through **an adaptive context vector** c(i)
  - c(i) is **a function of all** {h(1), h(2), …, h(T)}
  - c(i) ”attends” to the most important locations in the source sequence
  - c(i) = a(i,1)h(1) + a(i,2)h(2) + … a(i,T)h(T)
  - If a(i,j) is large → jth source token is useful/important in predicting ith output (otherwise, a(i,j) is small)
  - As before, maintain the hidden state s(i)
  - Decode y(i) = Dec (c(i), s(i), y(i-1))

- Key differences from Seq2Seq:
  - Input influences each output adaptively
  - Long-range dependence can be included

Decoding Flow.. (Bahdanau et. al)

Output y(i) depends on previous output, current state, current context

$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1},s_i,c_i),$$

State Update

$$s_i = f(s_{i-1},y_{i-1},c_i).$$

Context Vector

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

weighted sum of input "annotations"

Weights

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

(weights given to each "annotation" for i-th output)

Alignment Scores

$$e_{ij} = a(s_{i-1},h_j)$$

Alignment Score: i-th output vs j-th input

“Bahdanau et. al. Attention”

Context Vector ← Weights ← Alignment Scores

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

weighted sum of input “annotations”

(weights given to each “annotation” for i-th output)

Alignment Score: i-th output vs j-th input

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i. Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed-length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly.

Bidirectional Encoder

- Traditional RNNs encode sequence from left to right..and compute "hidden" states, h(1), h(2), h(3),…h(T) in L→ R order.
- This paper proposed **Bi-directional RNN (BiRNN)**
- How does it work ? Take two passes at input. **(Quiz: WHY ?)**
- Take a forward pass, i.e., process the input from left to right
  - → obtain h(1), h(2), …, h(T). "forward hidden states"
- Take a backward pass, i.e., process the input from right to left
  - → obtain h'(T), h'(T-1),…, h'(1). "backward hidden states"
- For each input in the source, concatenate the forward/reverse states.

$$h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]^\top$$

Putting everything together..

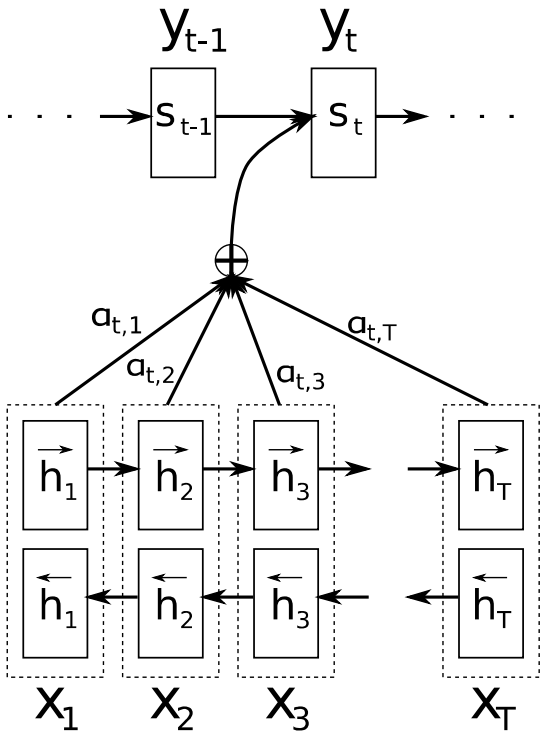


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Key architectural takeaways

- Introduced "attention"
- Adaptively using context from input
- Bi-directional RNNs for encoding input
- Also referred to as "additive attention"
- Why ? (see Appendix of the paper; Sec. A)
  - Attention scores are additive in nature
  - Linearly combine j-th input's annotation
  - Along with (i-1)-output state

$$e_{ij} = a(s_{i-1}, h_j) = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right),$$

Results (1): Improvement in BLEU Scores

Results reported on the WPT 14 dataset (English-to-French translation)

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50⋆	28.45	36.15
Moses	33.30	35.63

Table 1: BLEU scores of the trained models computed on the test set. The second and third columns show respectively the scores on all the sentences and, on the sentences without any unknown word in themselves and in the reference translations. Note that RNNsearch-50⋆ was trained much longer until the performance on the development set stopped improving. (○) We disallowed the models to generate [UNK] tokens when only the sentences having no unknown words were evaluated (last column).

- Moses: conventional phrase based translation (considered "high-quality")
- Basic encoder-decoder architectures:
  - RNNencdec-30 or RNNencdec-50 (30/50 denote the length of training sequences)
- Proposed "**RNNsearch**" attention model:
  - RNNsearch-30 or RNNsearch-50 (30/50 denote the length of training sequences)

Results (2): Impact of Sentence Length

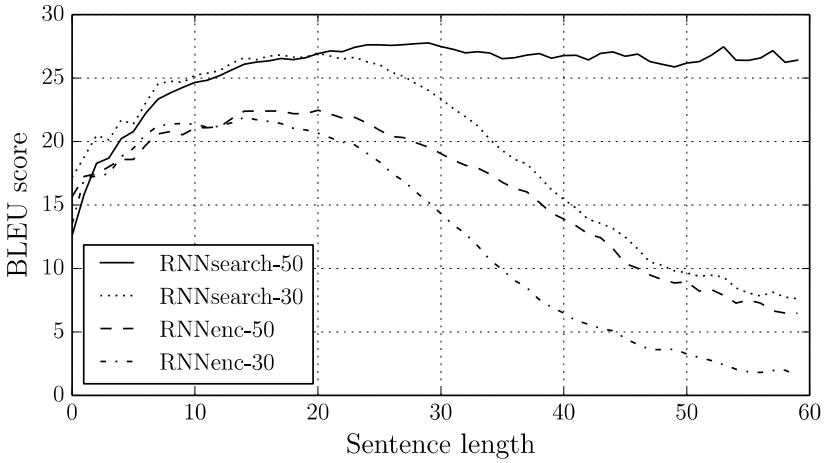
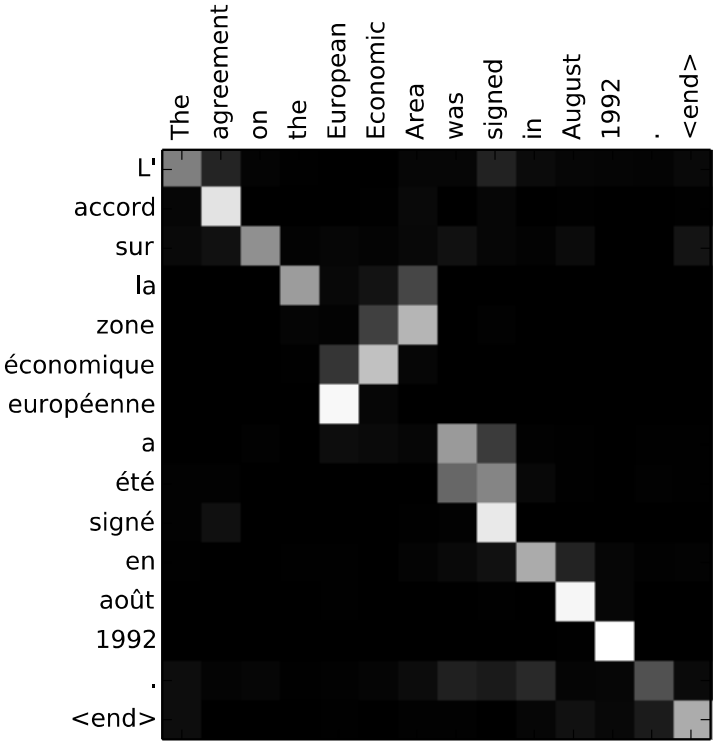


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Key Takeaways

- Enc-Dec architecture performance degrades as length increases
- "Attention" (RNNsearch-50) does not degrade with length

Results (3): Visualizing “Annotations”/Attention



Key Takeaways

- Generally observe alignment across English & French words (monotonic)
- Non-monotonic trends as well (off diagonal entries)
- E.g.: adjectives & nouns could be ordered differently across languages
- For instance [European economic area] → [zone economique europeen]

Paper # 2: “Attention is all you need”

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

THE UNIVERSITY OF ARIZONA

Motivation behind *Transformers*

- All papers until this point used Attention with a Recurrent Architecture
- Even though context information from input was used adaptively
- Relied on RNN architectures (sequential philosophy)

- Proposal of this paper:
  - Introduced **Transformers**
  - Give up on recurrence, entirely rely on attention
  - Attention should be able to learn global dependencies
  - Positional Encodings (to account for lack of recurrence)
  - Significantly more Parallelizable than recurrent architectures

Self-Attention—Key Building Block (1)

- Goal: given a sequence of token embeddings (x(1), x(2),…, x(N)), produce a new set of embeddings (x'(1), x'(2),…,x'(N)).

- Both input and output embeddings d-dimensional, i.e. x(i) is a 1xd vector.

- **Self Attention block has three weight matrices:**
  - **Query weight matrix:** Wq (size = Nxd)
  - **Key weight matrix:** Wq (size = Nxd)
  - **Value weight matrix:** Wv (size = Nxd)

- **Step 1:** Let us start with the i-th token embedding x(i)
  - First compute value vector = v(i) = x(i)Wv for the i-th token (d-dimensional)
  - Think of this as a new "intermediate" embedding we have obtained from the original embedding x(i).
  - One can then "stack" all the N value vectors in a matrix = V (Nxd)
  - This is called as the **Value Matrix = V = X . Wv**

Self-Attention—Key Building Block (2)

Self Attention block has three weight matrices:

- **Query weight matrix:** Wq (size = Nxd)
- **Key weight matrix:** Wq (size = Nxd)
- **Value weight matrix:** Wv (size = Nxd)

- **Step 2:** Let us take the i-th token embedding x(i)
  - Each token now creates a query vector
    - q(i) = x(i)Wq (d-dimensional)
    - We can stack all the N query vectors in a Query matrix = Q
    - Think of "query" as each token's question..
    - We now have the **Query Matrix = Q = X Wq**
- **Step 3:** Let us take the i-th token embedding x(i)
  - Each token now creates a key vector
    - k(i) = x(i)Wk (d-dimensional)
    - We can stack all the N key vectors in a Key matrix = K
    - Think of "key" as each token's response..
    - We now have the **Key Matrix = K = X Wk**

Self-Attention—Key Building Block (3)

Self Attention block has three weight matrices:

- **Query weight matrix:** Wq (size = Nxd)
- **Key weight matrix:** Wq (size = Nxd)
- **Value weight matrix:** Wv (size = Nxd)

- **Step 4: Dot-Product Attention:**
  - We would like to see the "similarity" between i-th query and j-th key
  - How similar is the question raised by the i-th token's query to the response (key) provided by the j-th token.
  - Similarity(i, j) = s(i,j) = $q(i)^T k(j)$
  - s(i,j) represents how much does i-th token "attend" to j-th token.
  - Stack all the N^2 pairs of s(i,j) values in a matrix.
  - We call these as the S matrix (raw-attention scores)
  - **Raw Attention Matrix: $S = QK^T$**

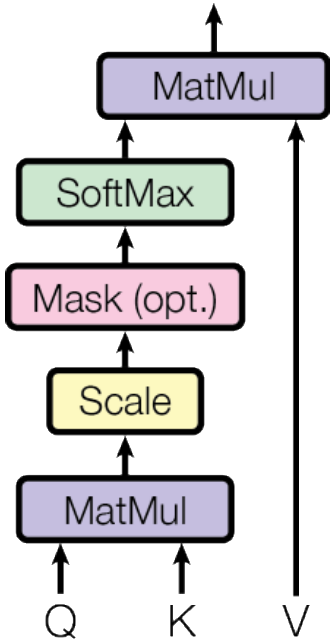
Self-Attention—Key Building Block (1)

- **Step 5: Updated Values via Attention**
  - Let us see what we have for the i-th token
  - We have the intermediate embedding: v(i) = x(i)Wv
  - Attention scores: s(i,1), s(i,2),…, s(i,N)
  - Scaled attention scores: $\frac{s(i,1)}{\sqrt{d}}, \frac{s(i,2)}{\sqrt{d}}, \ldots, \frac{s(i,N)}{\sqrt{d}}$
  - Convert these scores to probability scores:
    - p(i,1), p(i,2),…p(i,N) = Softmax($\frac{s(i,1)}{\sqrt{d}}, \frac{s(i,2)}{\sqrt{d}}, \ldots, \frac{s(i,N)}{\sqrt{d}}$)
  - Update v(i) to obtain the new embedding of i-th token

    x’(i) = p(i,1)v(1) + p(i,2)v(2) + .. p(i,i)v(i) + …..p(I,N)v(N)

Scaled Dot-Product Attention (Summary)

Scaled Dot-Product Attention



1. **Compute Queries, Keys, and Values:**

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V.$$

2. **Compute Raw Attention Scores:**

$$S = QK^\top.$$

3. **Scale the Attention Scores:**

$$S_{\text{scaled}} = \frac{S}{\sqrt{d}}.$$

4. **Compute Attention Weights:**

$$P = \text{softmax}(S_{\text{scaled}}).$$

5. **Compute Weighted Sum of Values:**

$$X' = P \cdot V.$$

6. **Apply Output Projection (Optional):**

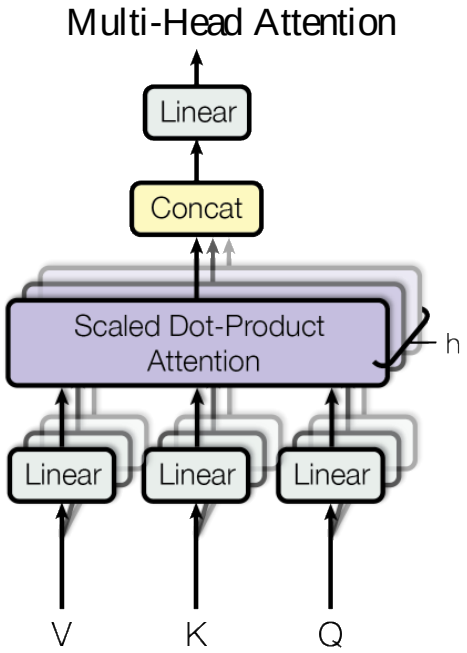
$$X'' = X'W_O.$$

7. **Add Residual Connection and Normalize:**

$$\text{Output} = \text{LayerNorm}(X'' + X).$$

Multi-head Attention

- **Rinse & Repeat attention block multiple independent times**
- h times/ h attention "heads"
- Each head has it's own triple of key, query and value weight matrices.
- Concatenate the outputs of heads→ Combine them back to size d
- Final embeddings after multi-head attention block.



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Positional Encoding

- Since we gave up on recurrence, we need to encode positions of the tokens in the sequence.
- Idea: **Positional Encodings.** For each token i, create a positional encoding (of dimension d).
- Final input: add token embeddings with Positional encodings

$$\mathrm{P}(i, 2j) = \sin\left(\frac{i}{10000^{\frac{2j}{d}}}\right),$$

$$\mathrm{P}(i, 2j+1) = \cos\left(\frac{i}{10000^{\frac{2j}{d}}}\right),$$

where:

- i: The position of the token in the sequence (0-based index),
- j: The index of the embedding dimension,
- d: The dimensionality of the embedding.

$$X_{\text{input}} = X + P,$$

where:

- $X \in \mathbb{R}^{N \times d}$: The token embedding matrix, with N tokens and embedding dimensionality d,
- $P \in \mathbb{R}^{N \times d}$: The positional embedding matrix, where each row encodes the position of a token in the sequence,
- $X_{\text{input}} \in \mathbb{R}^{N \times d}$: The combined input embedding matrix, which includes both token and positional information.

Final Transformer Architecture

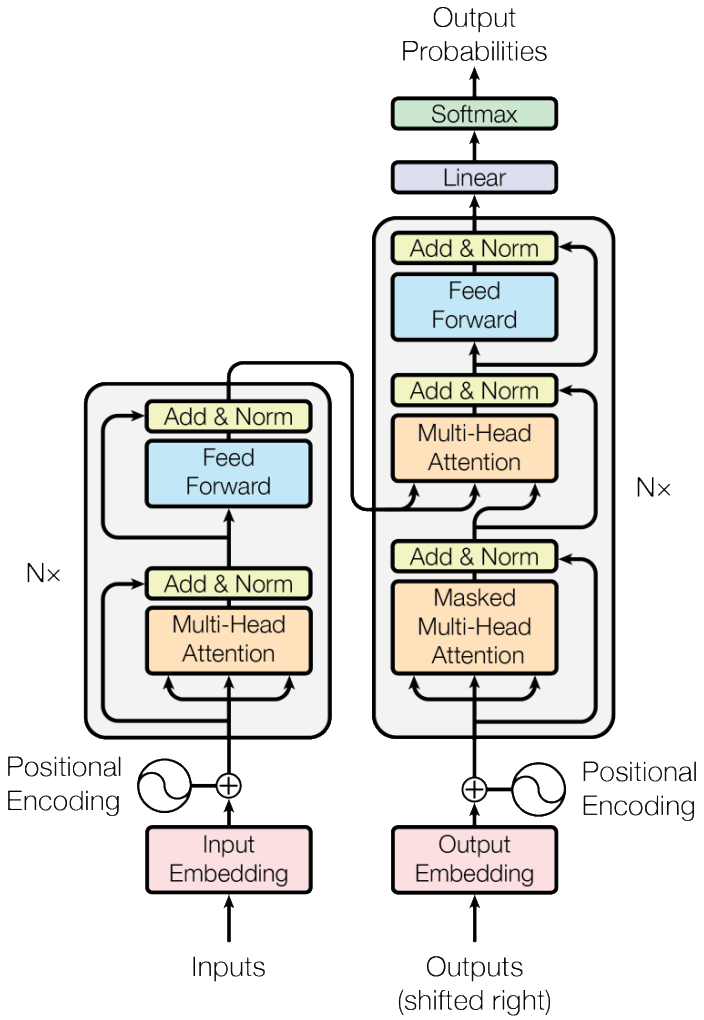


Figure 1: The Transformer - model architecture.

Results (1): State-of-the-art in MT

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 [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Results (2): Visualizing Attention

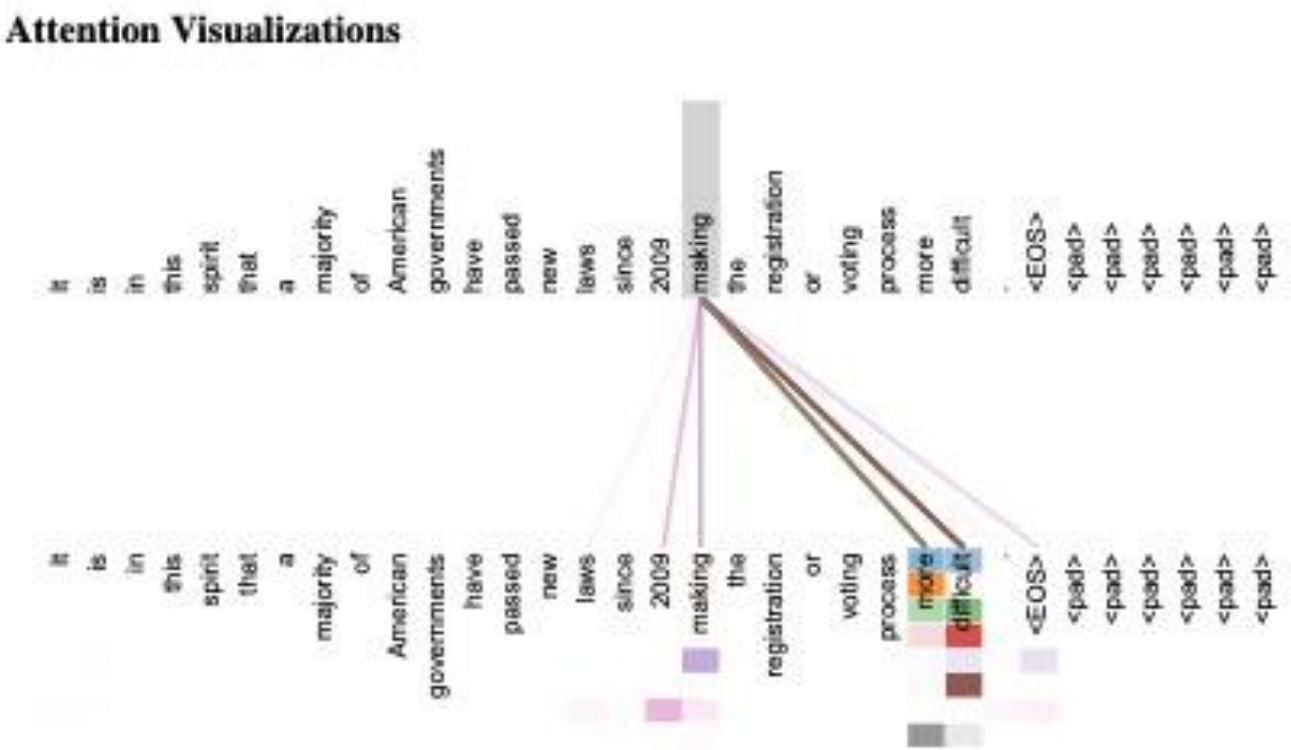


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

Results (3): Role of Multiple Attention Heads

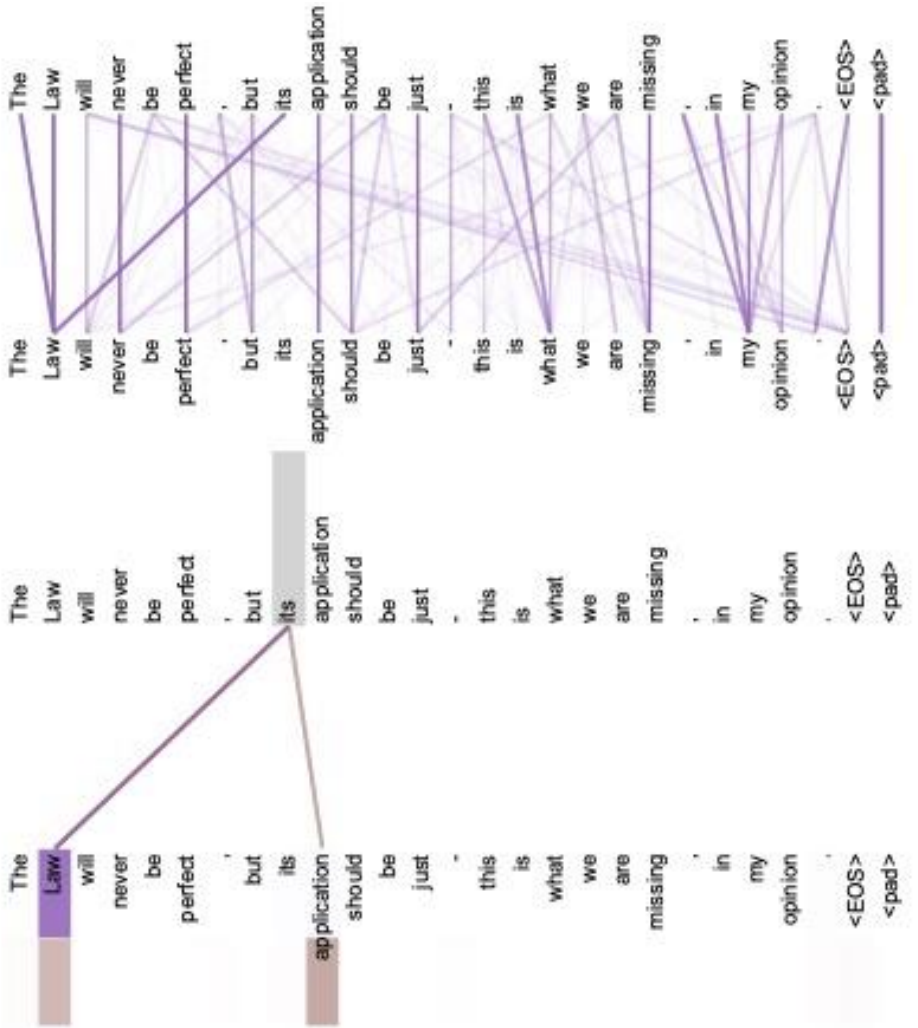


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.