Attention Mechanism and Transformers

Resources

Article:

- Google Brain, Attention is All You Need
- Bahdanau, Neural Machine Translation by Jointly Learning to Align and Translate

Web:

- Transformer: A Novel Neural Network Architecture for Language Understanding
- Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)
- Jeremy Jordan
  - Attention Mechanism
  - Transformer Architecture
- 3Blue1Brown:
  - Deep Learning Chapter 05: *What is a GPT?*
  - Deep Learning Chapter 06: *Attention in Transformers*
- The AI Hacker, *Illustrated Guide to Transformers*
- StatQuest, *Transformer Neural Networks*
- Campus X, *100 Days of Deep Learning* *Video 68 - 77*

Introduction

Recurrent Neural Networks

- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Working and Disadvantages

- Generate sequence of hidden states, $h_{t-1} \to h_t \to \ldots$, for input position t
- Cannot parallelize training, RNNs need to train *in order*. This becomes critical at longer sequence lengths, **memory constraints** limit *batching* across examples.

Improvements

- Conditional computation
- Factorization tricks

Conclusion

Although Improvements have been made to the efficiency of RNNs, the fundamental problems discussed, remain.

Why Attention Mechanisms?

- Become an essential part of **sequence modelling**
- They allow the model to consider how **different parts of the input or output relate to each other**, *regardless of how far apart* they are. This is a big improvement over RNNs, which process things step-by-step.
- Used to be implemented in conjunction with a recurrent network

Transformers

Attention is All You Need introduces the **Transformer** architecture, which

- Rejects recurrence
- Relies entirely on the **Attention** mechanism to figure out *how much the output depends on the input*
- Allows for parallelization

> The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

Attention-Is-All-You-Need_Google-Brain, page 2

Attention Mechanism

Analogy with Human Brain

- Let's say we have a *translation* task at hand - Translate the text on the first page of *Harry Potter and the Sorcerer's Stone* to Hindi.
- **Human:**
  - Read the whole page once.
  - Now close your eyes and try speaking out the entire Hindi translation in one go.
  - **SEEMS IMPOSSIBLE, RIGHT?**
  - Humans translate sentences by keeping in mind **what parts of the input text are important for the next word**. We pay **attention** to different parts of the input text

at each time step.

- This is the exact problem our LSTM-based Encoder-Decoder architecture faces - it cannot remember all the conext - *The encoded vector cannot carry enough information about all the context from the input*.

Methodolgy

> **Notations**
>
> - h_t - Hidden state of encoder at time step t
> - x_t - Input vector to the encoder ar time step t
> - s_t - Hidden state of the decoder at time step t
> - c_t - The attention input at time step t
> - y_t - Input to the decoder at time step t and $y_0 =$ `<START>` always

Diferrence in Input

- In a vanilla decoder, at each time step, the inputs required for the LSTM cell are (y_{t-1}, s_{t-1}) and the output is (y_t, s_t).
- The attention mechanism introduces another piece of information, which tells the cell how important each h_t is. We call it c_t here.

Weighted Sum of Inputs Using Attention

$$c_t = \sum_{u=1}^{n} \alpha_{t,u} h_u$$

- At each time step of the decoder, c_t is calculated by a weighted sum of the h_u's, for all time steps of the encoder.
- Each weight $\alpha_{t,u}$ is the weight of h_u at time step t of the decoder.

> Neural-Machine-Translatation_Bahdanau-Cho-Bengio, page 3

Calculating $\alpha_{t,u}$

Parameter	Reasoning
h_u	The weight given to the hidden state depends on *what it is*
s_{t-1}	The weight would depend on the *state of the current translation*

$$\alpha_{t,u} = f_\alpha(h_u, s_{t-1})$$

- This function f_α is **approximated by using an ANN**

- This ANN has the **softmax activation** in its output layer to *produce a probability distribution* over the set of h_u
- This ANN is trained by backpropagation, the upstream gradient flows in through the *backpropagation-through-time* algorithm.

> A more detailed formula is given in Neural-Machine-Translatation_Bahdanau-Cho-Bengio, page 3

Performance

BLEU Score

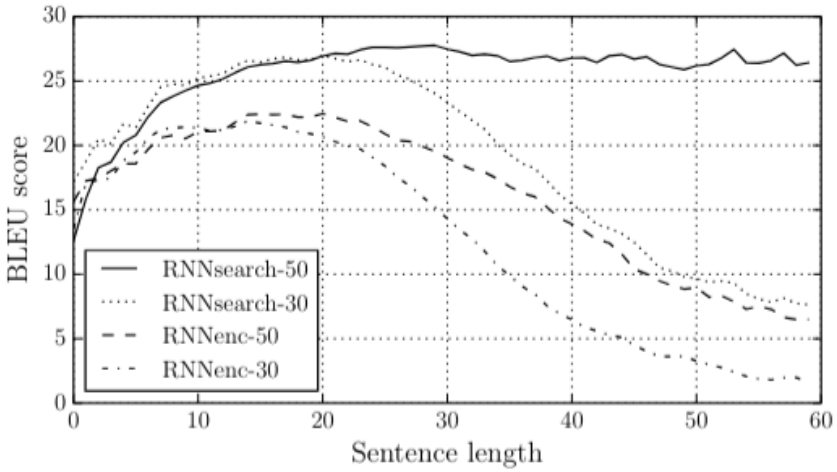
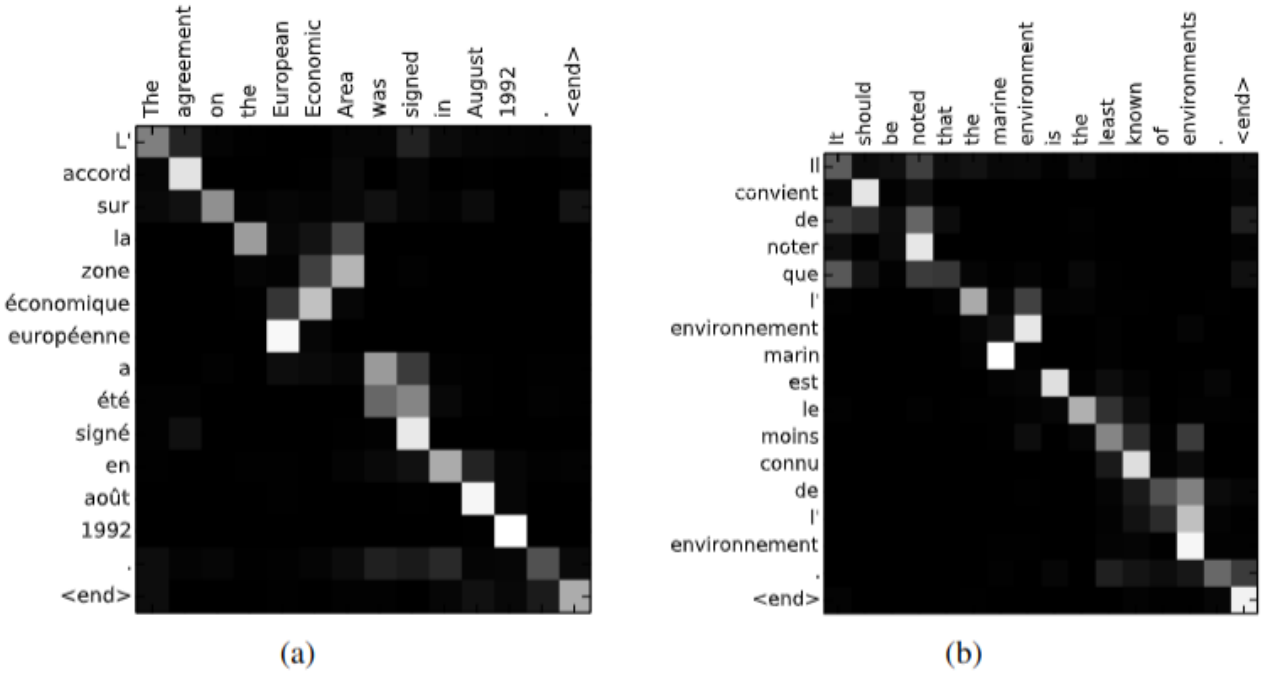


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

- The BLEU score is a performance metric for machine translation algorithms.
- For non-attention based models, the BLEU score drops significantly after the sentence length exceeds 30 words.
- With attention based models, this is avoided and the score remains constant.

Heatmap of $\alpha_{t,u}$

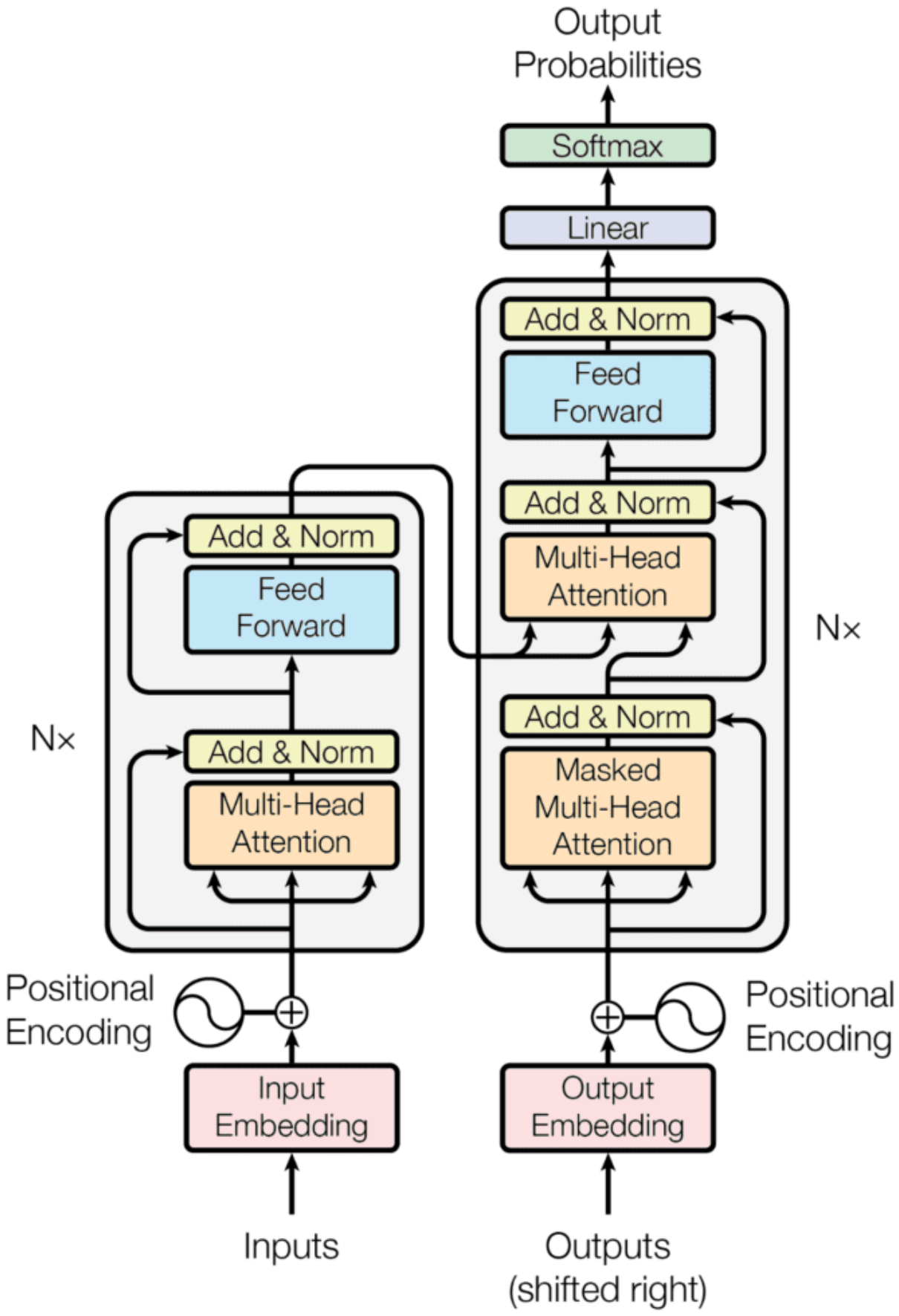


- This heatmap shows the $\alpha_{t,u}$ for an English-to-French translation task implementing the attention mechanism.
- This heatmap shows the *attention* given to each English word while generating each French word.

How to Read

- Lot of attention was given to "European" while generating "europeenne"
- Similarly, "agreement" was highly attended to while generating the word "accord"

Transformer Model Architecture



Encoder-Decoder Stacks

Transformers follow the *Encoder-Decoder* structure as described in the figure.

- The **encoder** maps the input sequence of symbol representations $\vec{x} = (x_1, x_2, \ldots, x_n)$ to a sequence of continuous representations $\vec{z} = (z_1, z_2, \ldots, z_n)$
- Given $\vec{z}$, the **decoder** generates an output sequence $\vec{y} = (y_1, y_2, \ldots, y_m)$ of symbols, *one element at a time*.
- At each time step the model is **auto-regressive** - consuming the **previously generated symbols** as **additional input** when *generating the next*.