Intro to Neural Nets

Attention & Transformers

Today's Agenda

Attention Mechanism in RNNs for Translation (2014)

- Sequence to sequence models with attention in translation tasks.

More General, Approach (No RNNs Required)

- We can hard-code a measure of semantic similarity between a focal token, and all other tokens in the sequence

QKV Framework & Multi-head Attention

- We can incorporate trainable weights around the attention mechanism.

TF-IDF is “Static Attention”

Not all phrases are of equal importance…

- E.g., David less important than Beckham
- If a term occurs all the time, observing its presence is less informative

Inverse-document frequency (IDF) helps address this.

$$\mathrm{IDF} = \log(N/n_j)$$

- Term ‘weighting’ is then calculated as Term Frequency (TF) x IDF
- n_j= # of docs containing the term, N = total # of docs
- A term is deemed important if it has a high TF and/or a high IDF.
- As TF goes up, the word is more common generally. As IDF goes up, it means very few documents contain this term.

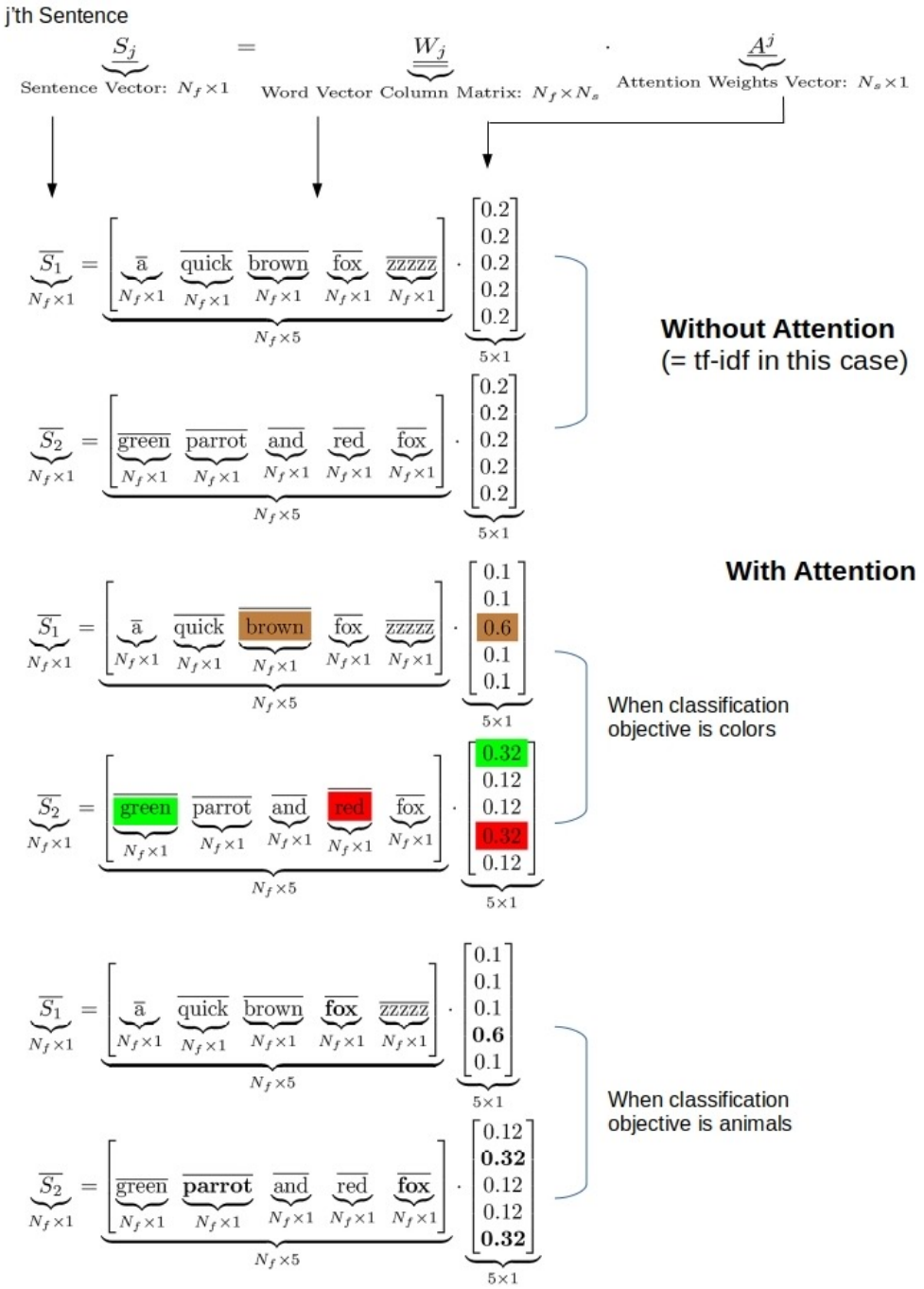
What Would Adaptive Weights Look Like?

Figure Out Weights Useful for Prediction

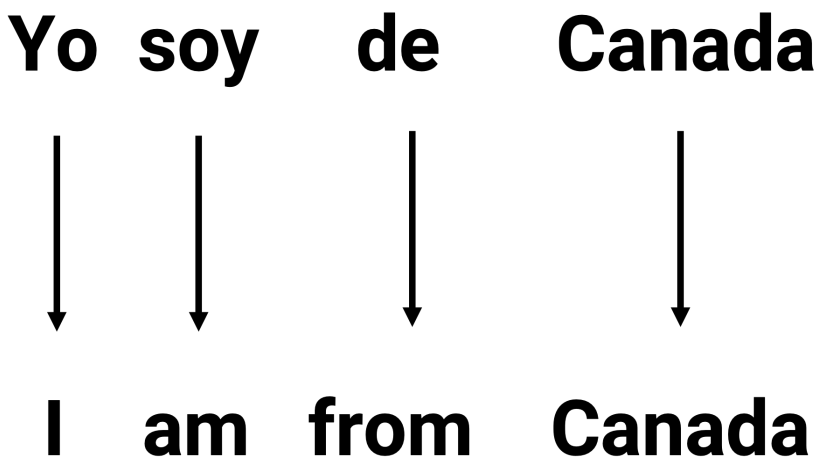
- Rather than telling the network the weights, which is what TF-IDF effectively does, we provide the network a weight matrix to update!

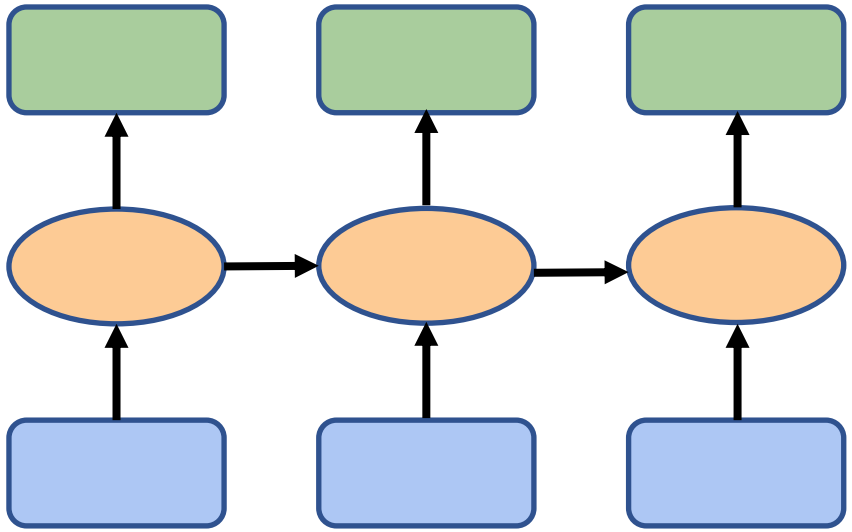
This Yields Task-specific Token Weights

- See some simple examples on the right.
- Our network may learn that certain tokens are more important for predicting a given label.

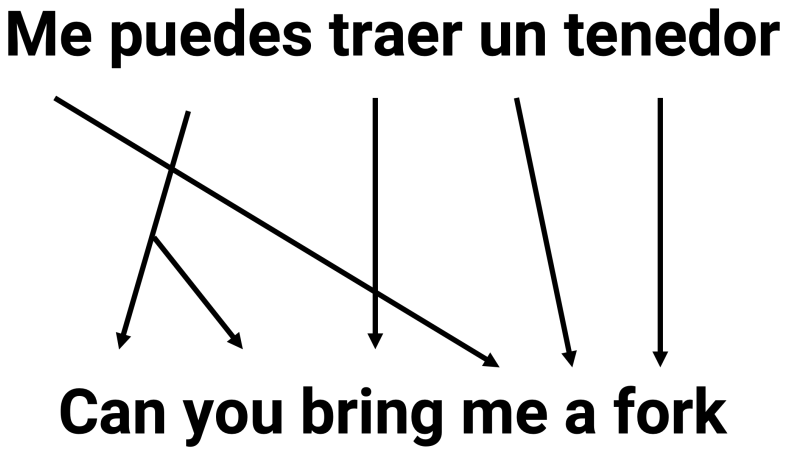


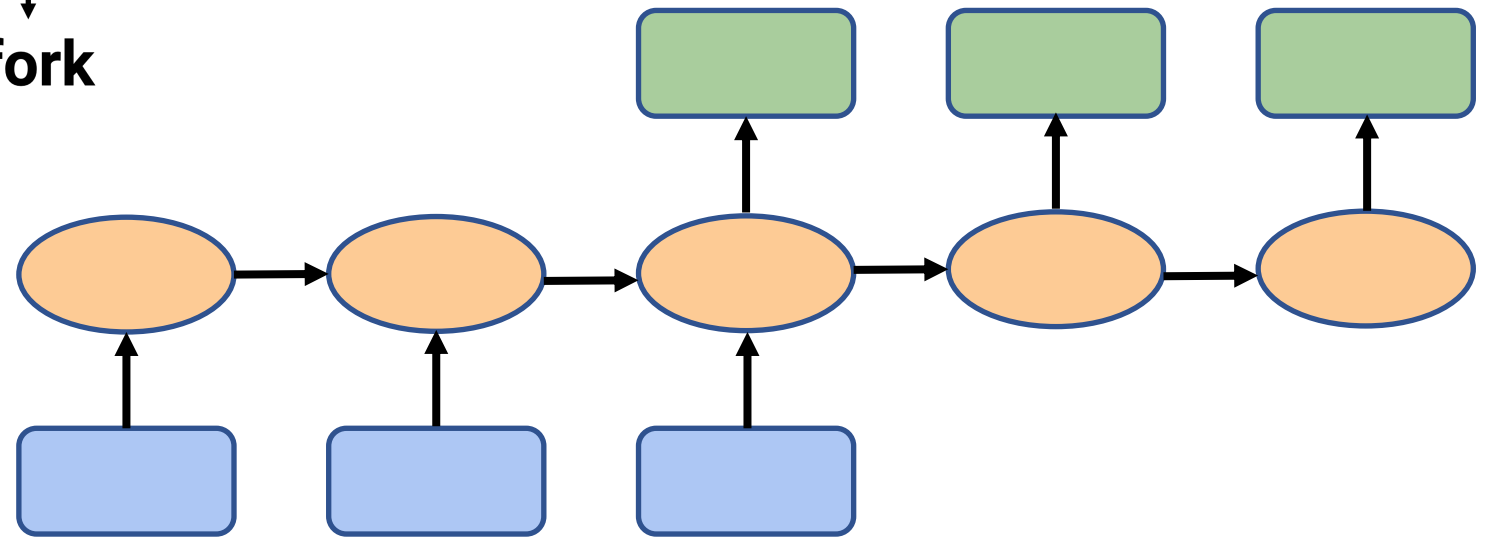
Where Attention Came From: Sequence-to-Sequence Models





Struggling to Remember Long Sequences





The First (RNN) Attention Model

Neural Machine Translation by Jointly Learning to Align and Translate

Dzmitry Bahdanau
Jacobs University Bremen, Germany

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

" … allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word …"

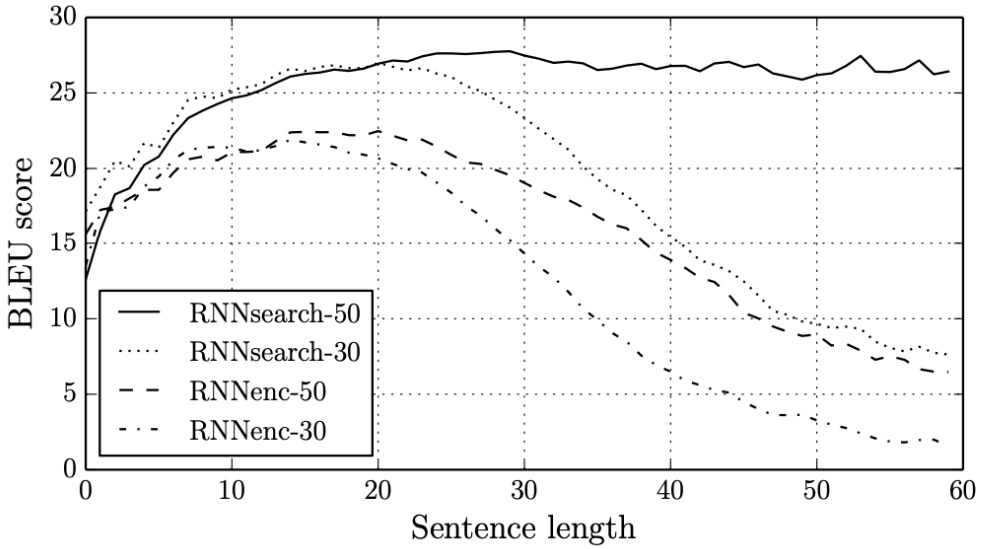


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

The First (RNN) Attention Model

Three RNNs

- Bidirectional RNNs handle the input sentence (encoder).
- A single RNN handles the output sequence (decoder).

Process

- Bidirectional RNN reads sentences in both directions. Produces a pair of hidden state outputs for each token of input.
- Unidirectional RNN reads output sentence forwards and produces a single hidden state, s.
- Attention weights at a particular token position are calculated across all input tokens comparing s_t-1 with each pair of h_i hidden states. This yields a vector of attention weights. We softmax the attention weights make them positive and add up to 1 (easier to differentiate in backprop).
- Attention weights down- or up-weight each positional hidden state, h.
- Finally, prediction, y_t, is based on the attention-reweighted hidden states of the input series and the last hidden state from token prediction y_t-1 (i.e., s_t-1).

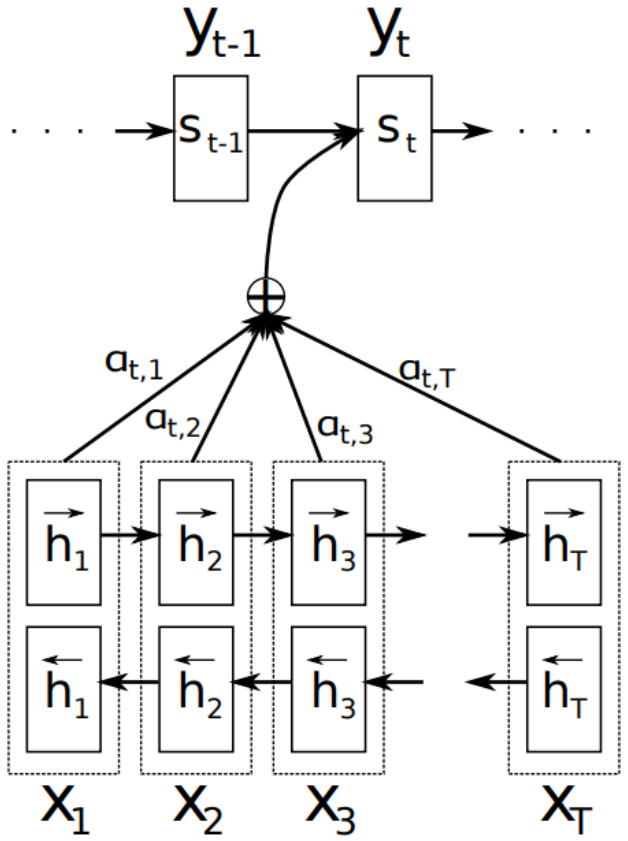
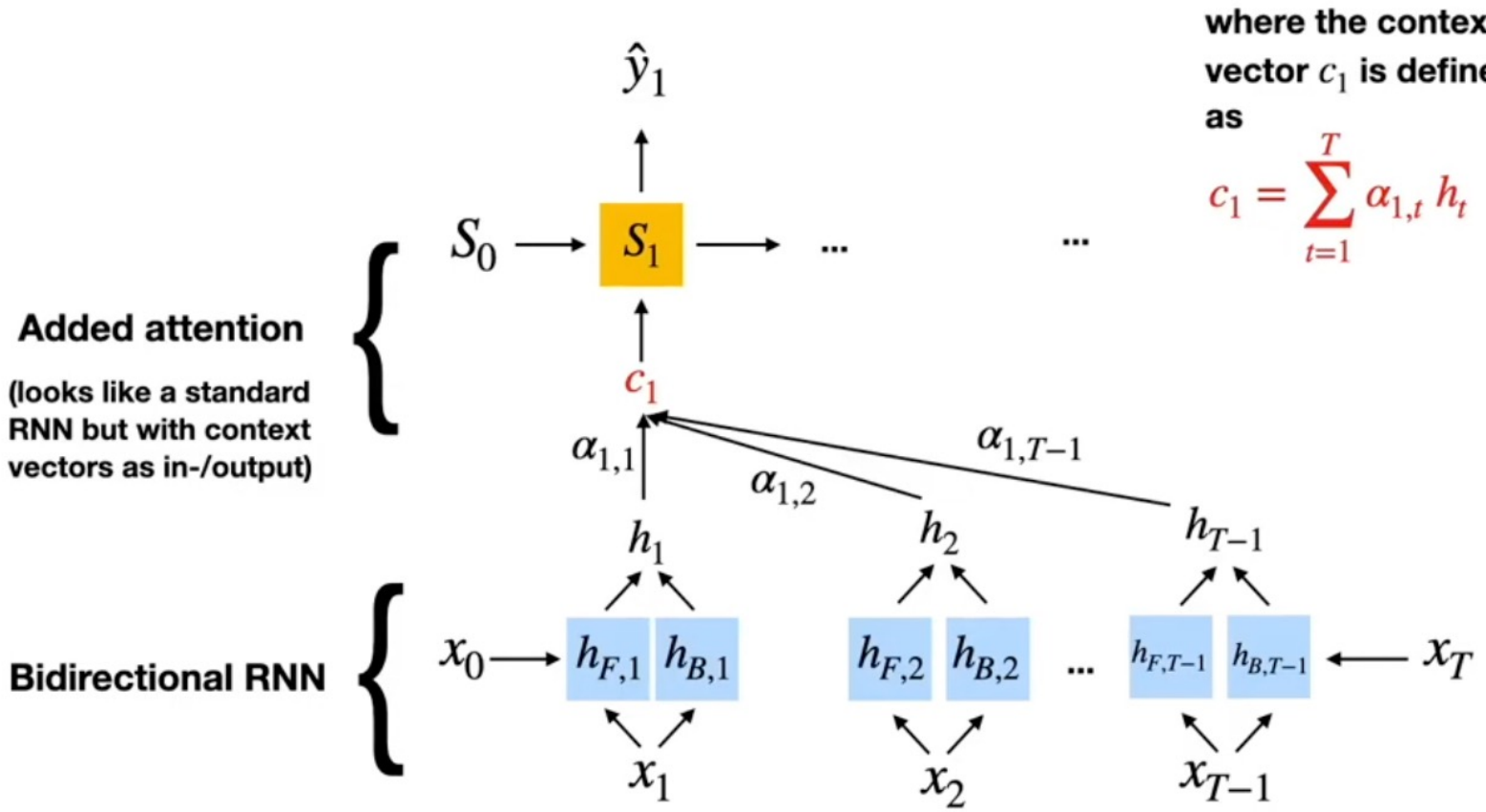


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

The First (RNN) Attention Model

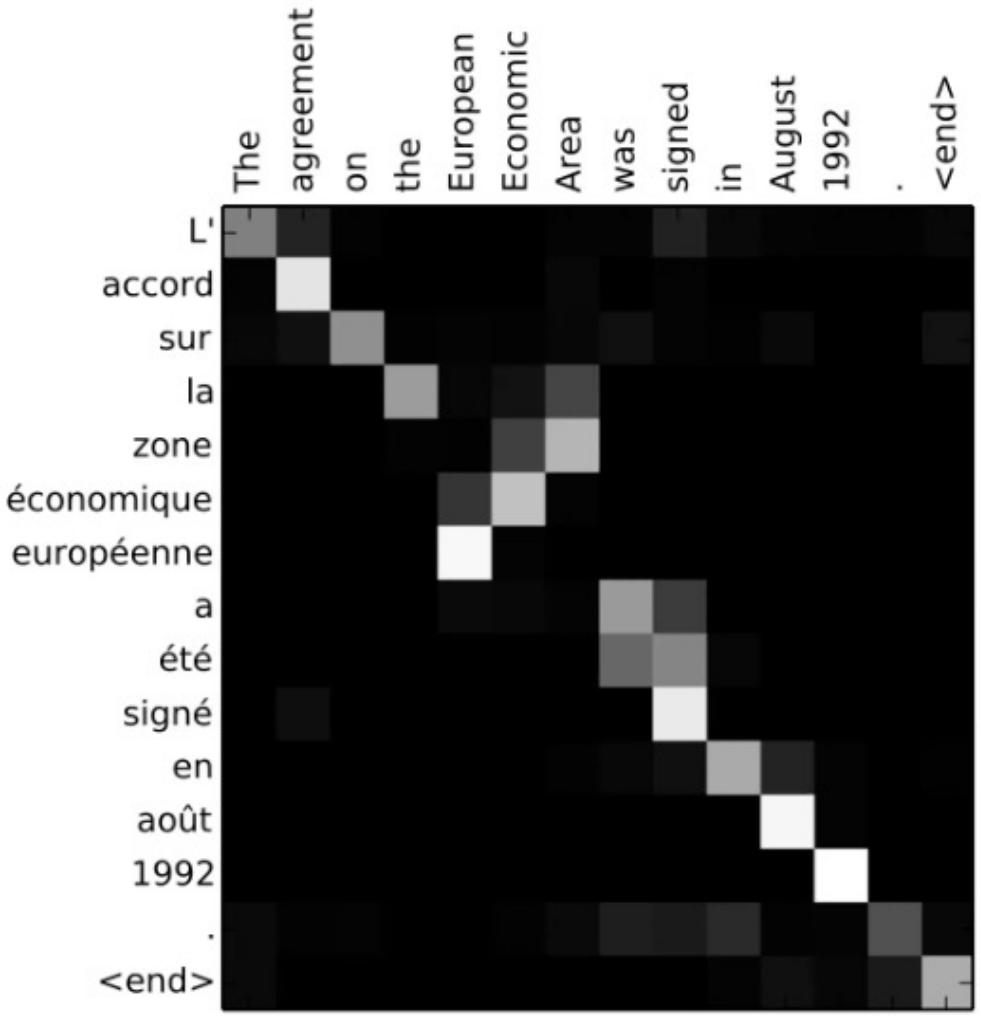


Computing attention weights

$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^{T} \exp(e_{t,t'})}$$

Neural Net $= v_a^\top \tanh(W_a s_{t-1} + U_a h_i)$

The First (RNN) Attention Model



Computing attention weights

S_{t-1}, h_t → Neural Net → $e_{t,t'}$

$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^{T} \exp(e_{t,t'})}$$

Neural Net $= v_a^\top \tanh(W_a s_{t-1} + U_a h_i)$

We Don’t Need an RNN

We Can Drop the RNN and Process the Whole Sequence in Parallel

- Note that transformers work with a similar encoder architecture, but no LSTMs are needed (we use ”stacked attention layers”).
- Transformers are more computationally expensive (more calculations required), but because we can parallelize them, they can be trained faster than RNNs.
- By moving away from RNN specific implementation we can also think about more general forms of attention (e.g.,. For CNNs).

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

A Simpler (Less-Flexible) Attention Mechanism

Main procedure:

1) Derive attention weights: similarity between current input and all other inputs (next slide)
2) Normalize weights via softmax (next slide)
3) Compute attention value from normalized weights and corresponding inputs (below)

Self-attention as weighted sum:

$$\boldsymbol{A}_i = \sum_{j=1}^{T} a_{ij}\boldsymbol{x}_j$$

output corresponding to the i-th input

weight based on similarity between current input $\boldsymbol{x}_i$ and all other inputs

A Simpler (Less-Flexible) Attention Mechanism

Self-attention as weighted sum:

$$A_i = \sum_{j=0}^{T} a_{ij} x_j$$

output corresponding to the i-th input

weight based on similarity between current input x_i and all other inputs

How to compute the attention weights?

here as simple dot product:

$$e_{ij} = x_i^\top x_j$$

repeat this for all inputs $j \in \{1...T\}$, then normalize

$$a_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{j=1}^{T} \exp\left(e_{ij}\right)} = \text{softmax}\left(\left[e_{ij}\right]_{j=1....T}\right)$$

A Simpler (Less Flexible) Attention Mechanism

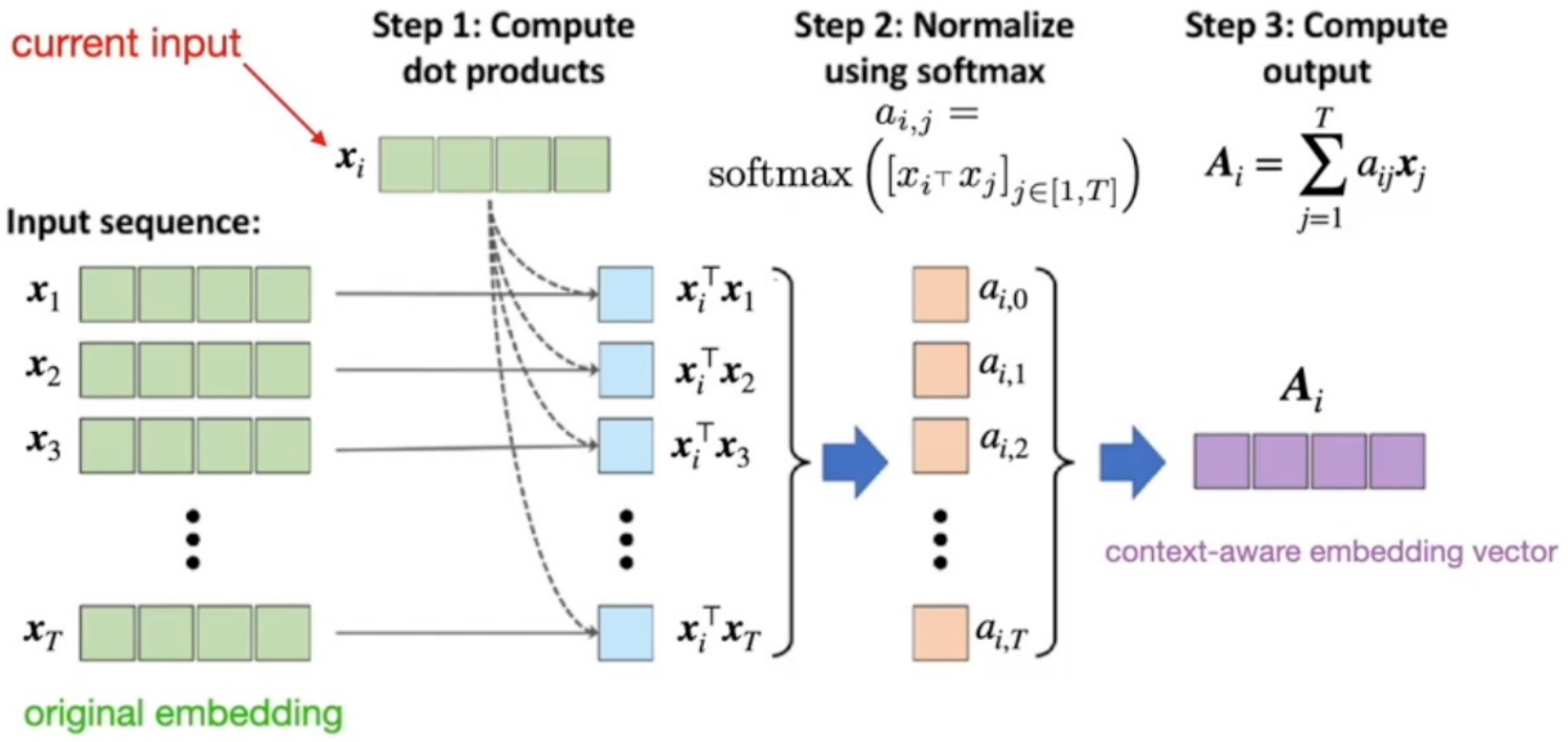
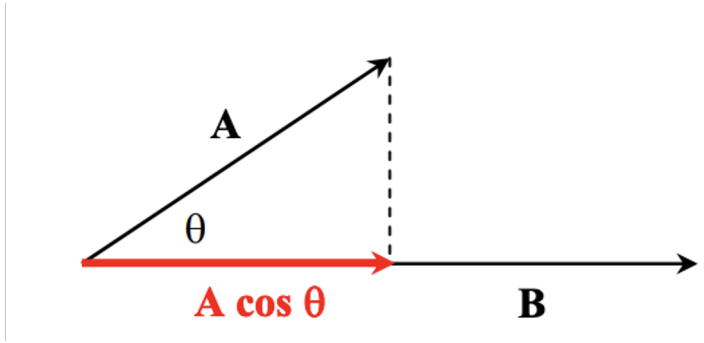


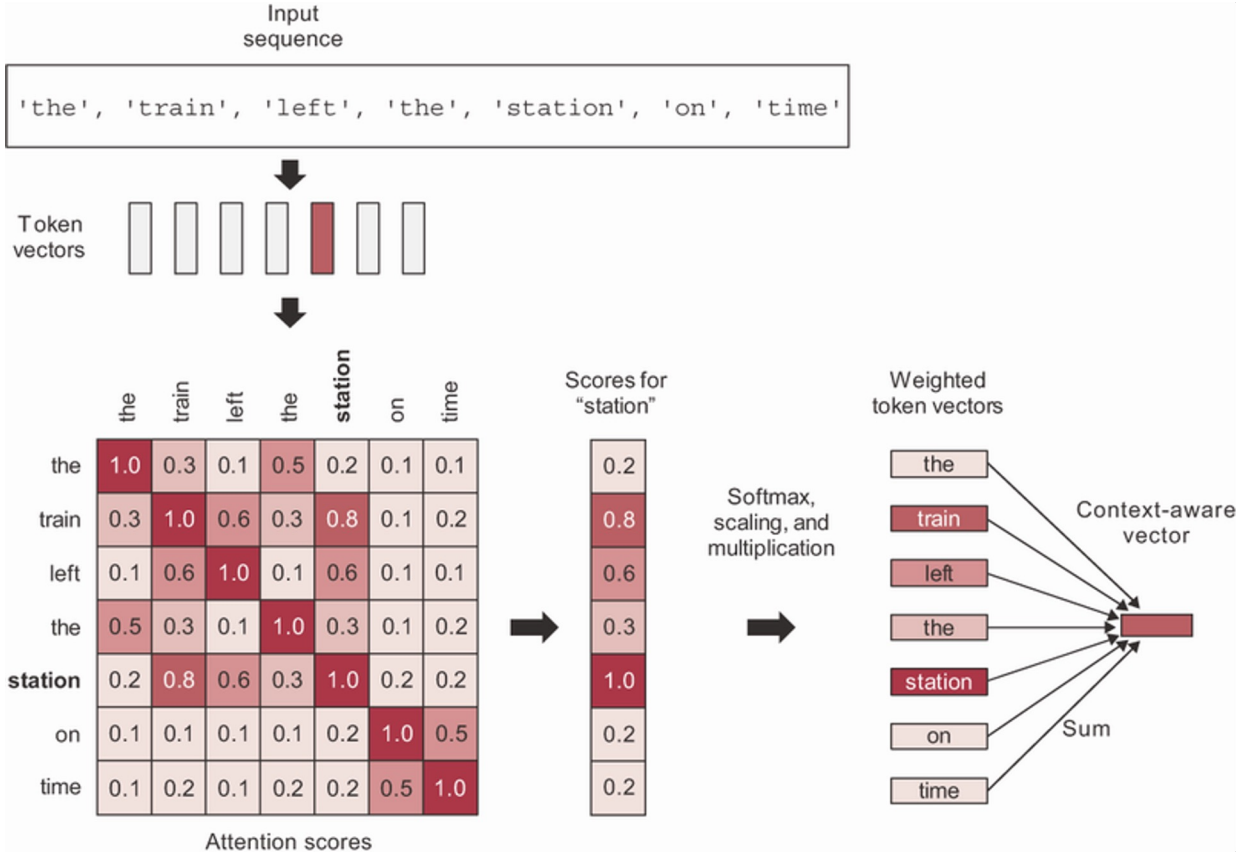
Image source: Raschka & Mirjalili 2019. Python Machine Learning, 3rd edition

A Basic Self-Attention Mechanism

Most Basic Form of Self-Attention

- This is what the textbook introduces.
- We magnify vector representations of tokens based on how parallel (semantically related) they are to the other tokens in the input sequence.





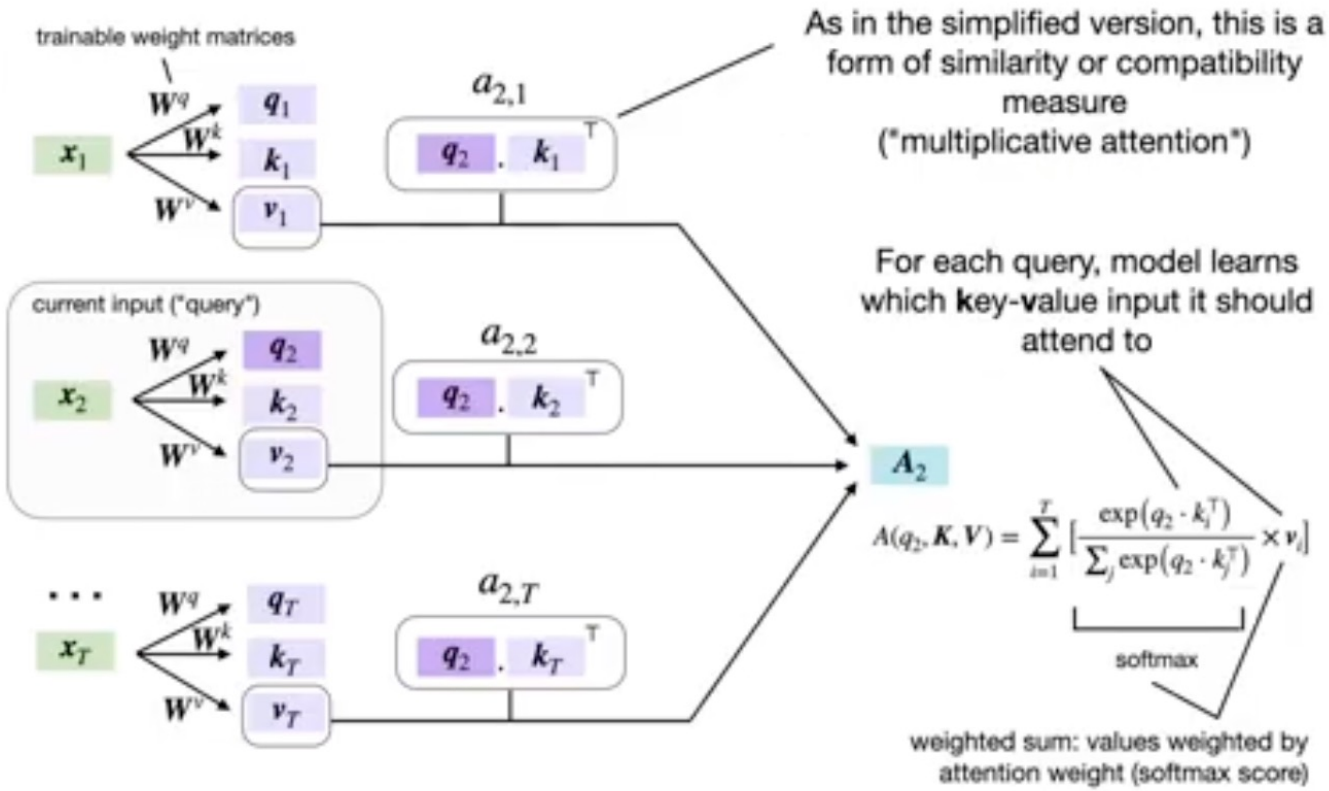
This Basic Attention Mechanism is Rigid

The Approach Considers the Input Context in a Very 'Fixed' Manner

- Although it doesn't assume forward / backward sequence information is most important, it *does* assume a specific attention rule.
- We get context-specific attention weights, but the notion of how context is processed is pre-defined / hard-coded in the dot-product-summation procedure; it's not learned.
- The pre-defined approach to attention does not generalize outside of language.
- The pre-defined approach to attention assumes that shared semantic similarity is the best heuristic for attention, but perhaps it is not in some cases. Most obviously, perhaps in some scenarios it is ideal to focus attention on words that are semantically dissimilar.
- How can we enable a procedure of this sort, but using trainable parameters, so the network can figure out how to allocate attention by itself?

Query Key Value Framework

$$\text{query} = \boldsymbol{W}^q \boldsymbol{x}_i$$
$$\text{key} = \boldsymbol{W}^k \boldsymbol{x}_i$$
$$\text{value} = \boldsymbol{W}^v \boldsymbol{x}_i$$



‘Scaled’ Dot Product

Scaled Dot-Product Attention…

- One other nuance is that we scale the Softmax input.
- This is done to prevent the dot products from getting too big, which can cause the output of softmax to tend toward 1 or 0 and lead to vanishing gradient issues.
- sqrt(d_k) is the square root of the dimensionality of the input key embeddings (in the original paper, this was the same as the dimension of the word embeddings).
- We write this in big-letter notation because we can parallelize everything as tensor calculations.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

From Self-Attention to Cross-Attention

- We now have the freedom to learn attention scores for input in one modality of input data to inform what we do with another modality of data.

Example: Image Q&A

- You input an **image** (e.g., via a CNN or Vision Transformer), producing visual features.
- Then input a **text prompt** (e.g., for captioning or VQA).
- Use **cross-attention**: the text tokens query the image features to decide what visual content is relevant.

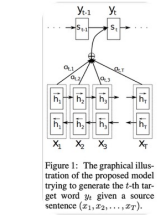


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

What is this figure all about?

This figure is a classic illustration of the **attention mechanism in neural machine translation (NMT)**, particularly the **Bahdanau et al. (2015)** model, also known as **additive attention**.

Let me break it down:

- The **bottom row** represents the **encoder**. The input sentence $(x_1, x_2, ..., x_T)$ is processed by a bidirectional RNN (often LSTM or GRU). For each input word x_i, the forward RNN computes $\vec{h}_i$ and the backward RNN computes \(\reflectbox{\(\vec{\reflectbox{\(h\)}}\)}_i). These are concatenated to form the annotation vector \(h_i = [\vec{h}_i; \reflectbox{\(\vec{\reflectbox{\(h\)}}\)}_i]), capturing context from both directions.
- The **top row** is the **decoder**, another RNN that generates the target sentence one token at a time. At time step t, it generates y_t using the decoder state s_t, which depends on the previous decoder state s_{t-1}, the previously generated word y_{t-1}, and a **context vector**.
- The **context vector** (middle of the diagram, shown as a circle with ⊕) is a weighted sum of all encoder hidden states h_i, using attention weights $\alpha_{t,i}$. These weights determine how much attention the decoder at time t pays to each source word.

Mathematically:

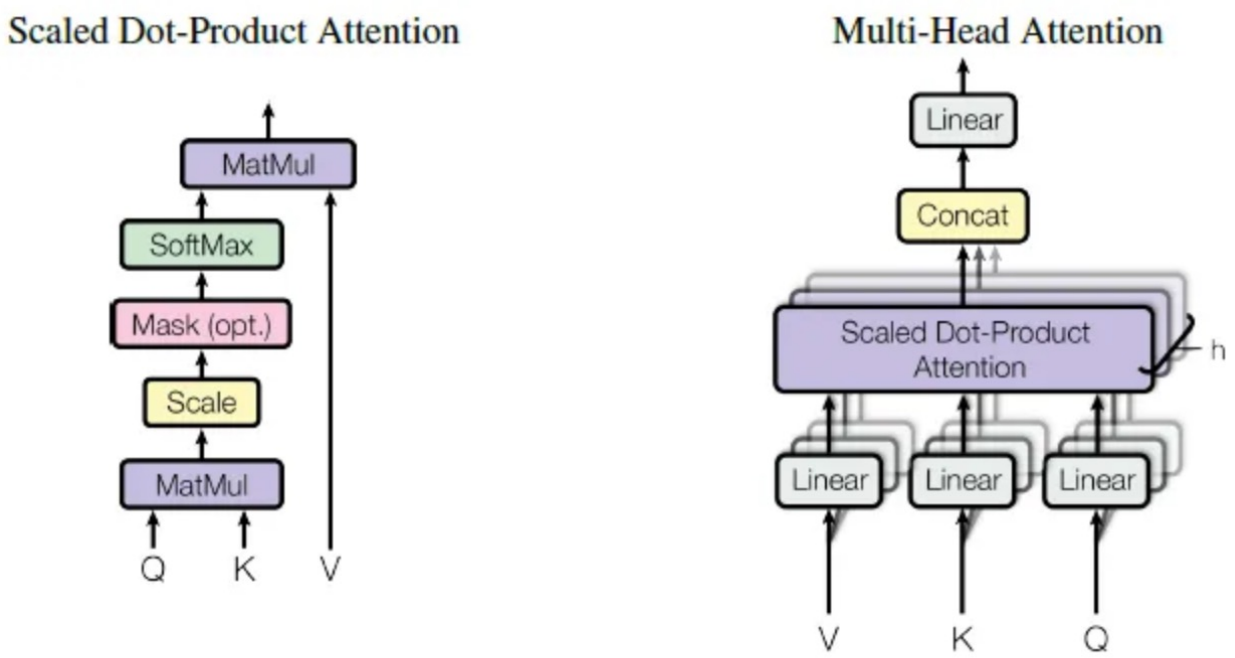
$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i$$

The weights $\alpha_{t,i}$ are computed via a **softmax** over an alignment score between the current decoder state and each encoder hidden state.

Multi-Head Attention

We have Multiple 'Heads' Paying Attention

- Each head can be thinking about something very different.
- By using multiple sets of query, key and value weight matrices, the network can learn to consider different combinations of input features.
- This lets it capture more complex patterns in the data.



From "Attention is all you need" paper by Vaswani, et al., 2017 [1]

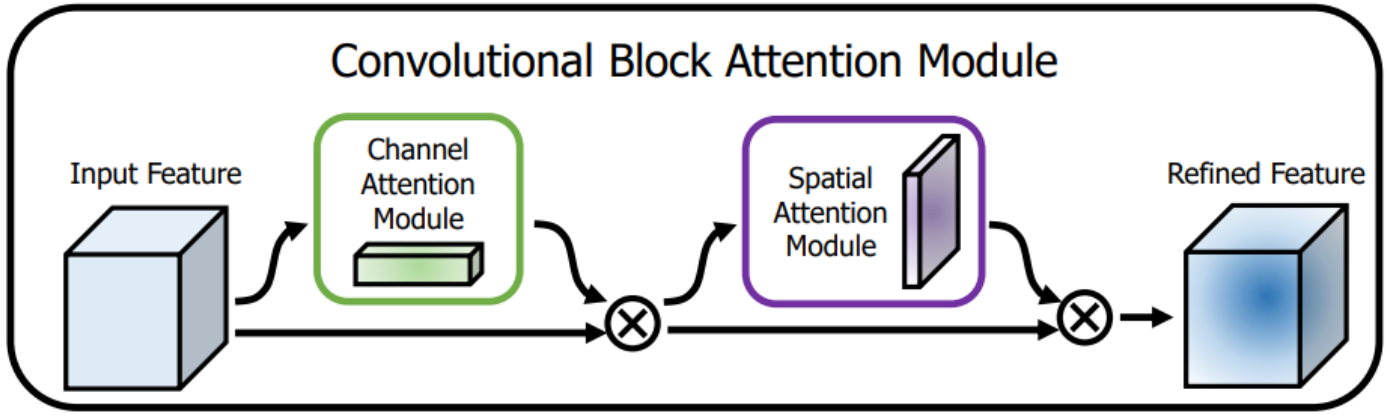
Attention in Vision Models

Spatial Attention

We chunk images spatially and implement an attention mechanism to allow a model to decide which portions of the image it should focus on.

Channel Attention

We implement an attention mechanism that allows the model to decide which feature maps are important for a prediction task.



Questions?