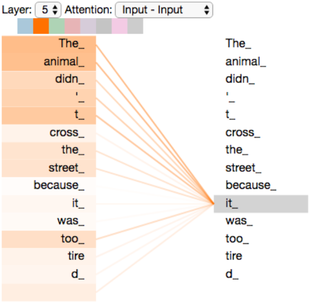
Advanced NN Architectures

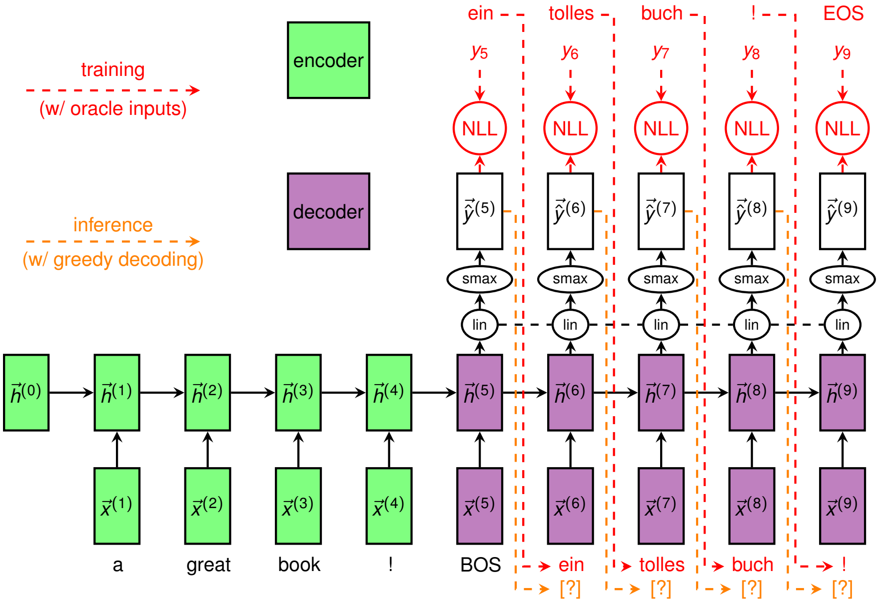
Attention



Learning goals

- Understand attention mechanism
- Learn the different types of attention

ENCODER-DECODER WITH RNNS



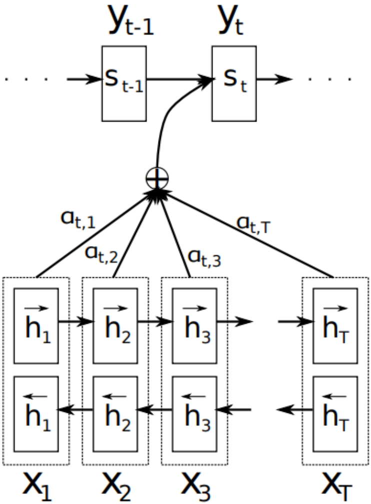
OTHER ENCODER-DECODER APPLICATIONS

- Text summarization
- Text generation
- Keyword generation
- Automatic speech recognition
- Subtitle generation
- Question answering
- Named entity recognition
- Video or image captioning
- Part-of-speech tagging
- ...more

LIMITATIONS OF RNNS

- In an RNN, at a given point in time j, the information about all past inputs $x^{(1)} \dots x^{(j)}$ is "crammed" into the state vector $\vec{h}^{(j)}$ (and $\vec{c}^{(j)}$ for an LSTM)
- So for long sequences, the state becomes a **bottleneck**
- Especially problematic in encoder-decoder models (e.g., for Machine Translation)
- Solution: Attention ▸ Bahdanau et al., 2015 – an architectural modification of the RNN encoder-decoder that allows the model to "attend to" past encoder states

BAHDANAU ATTENTION



Source: Bahdanau et al., 2015

ATTENTION: THE BASIC RECIPE (1)

- **Ingredients:**
- One query vector: $\mathbf{q} \in \mathbb{R}^{d_q}$
- J key vectors: $\mathbf{K} \in \mathbb{R}^{J \times d_k}$; $(\vec{k}_1 \dots \vec{k}_J)$
- J value vectors: $\mathbf{V} \in \mathbb{R}^{J \times d_v}$; $(\vec{v}_1 \dots \vec{v}_J)$
- Scoring function $a : \mathbb{R}^{d_q} \times \mathbb{R}^{d_k} \rightarrow \mathbb{R}$
  - Maps a query-key pair to a scalar ("score")
  - a may be parametrized by parameters θ_a

ATTENTION: THE BASIC RECIPE (2)

- **Step 1**: Apply a to $\vec{q}$ and all keys $\vec{k}_j$ to get scores (one per key):

$$\vec{e} = \begin{bmatrix} e_1 \\ \vdots \\ e_J \end{bmatrix} = \begin{bmatrix} a(\vec{q}, \vec{k}_1; \theta_a) \\ \vdots \\ a(\vec{q}, \vec{k}_J; \theta_a) \end{bmatrix}$$

- **Step 2**: Turn **e** into a probability distribution with the softmax function

$$\alpha_j = \frac{\exp(e_j)}{\sum_{j'=1}^{J} \exp(e_{j'})}$$

  - Note that $\sum_j \alpha_j = 1$

ATTENTION: THE BASIC RECIPE (3)

- **Step 3**: α-weighted sum over $\vec{V}$ yields one d_v-dimensional output vector:

$$\vec{o} = \sum_{j=1}^{J} \alpha_j \vec{v}_j$$

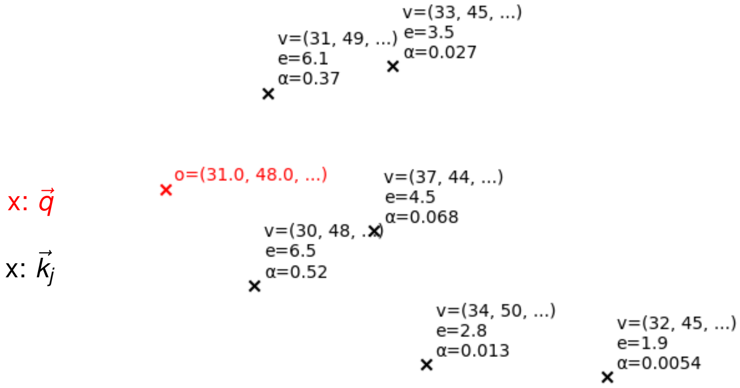
  - Intuition: α_j is how much “attention” the model pays to $\vec{v}_j$ when computing $\vec{o}$.

ATTENTION: AN ANALOGY (1)

- We have J weather stations on a map
- $\vec{K} \in \mathbb{R}^{J \times 2}$ are their geolocations (x,y coordinates)
- $\vec{V} \in \mathbb{R}^{J \times d_v}$ are their current weather conditions (temperature, humidity, etc.)
- $\vec{q} \in \mathbb{R}^2$ is a new geolocation for which we want to estimate weather conditions
- e_j is the relevance of the j'th station (e.g., $e_j = a(\vec{q}, \vec{k}_j) = \frac{1}{||\vec{q} - \vec{k}_j||_2}$), and α_j is e_j as a probability

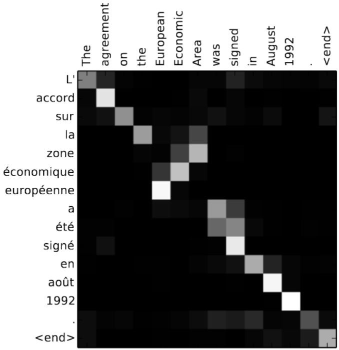
ATTENTION: AN ANALOGY (2)

- $\vec{o}$: a weighted sum of all known weather conditions, where stations that have a small distance (high α) have a higher weight



ATTENTION IN NEURAL NETWORKS

- Contrary to our geolocation example, the $\vec{q}$, $\vec{k}_j$ and $\vec{v}_j$ vectors of a neural network are a function of the input and trainable parameters
- So the *model* learns which keys are relevant for which queries, based on the training data and loss function



Source: Bahdanau et al., 2015

A PRIMER ON THE TRANSFORMER (1)

- The Bahdanau model is still an RNN, just with attention on top.
- Another architecture that consists of attention only:
  - Transformer: “Attention is all you need” ▸ Vaswani et al., 2017
  - No recurrence, *just* Attention (as the name suggests)
  - Better parallelizable (as opposed to RNNs)
  - Will be introduced in chapter 3

A PRIMER ON THE TRANSFORMER (2)

- No (or few) assumptions are baked into the architecture (no notion of which words are neighbors in the sentence, sequentiality, etc.)
- The lack of prior knowledge often means that the Transformer requires more training data than an RNN to achieve a certain performance
- But when presented with sufficient data, it usually outperforms them