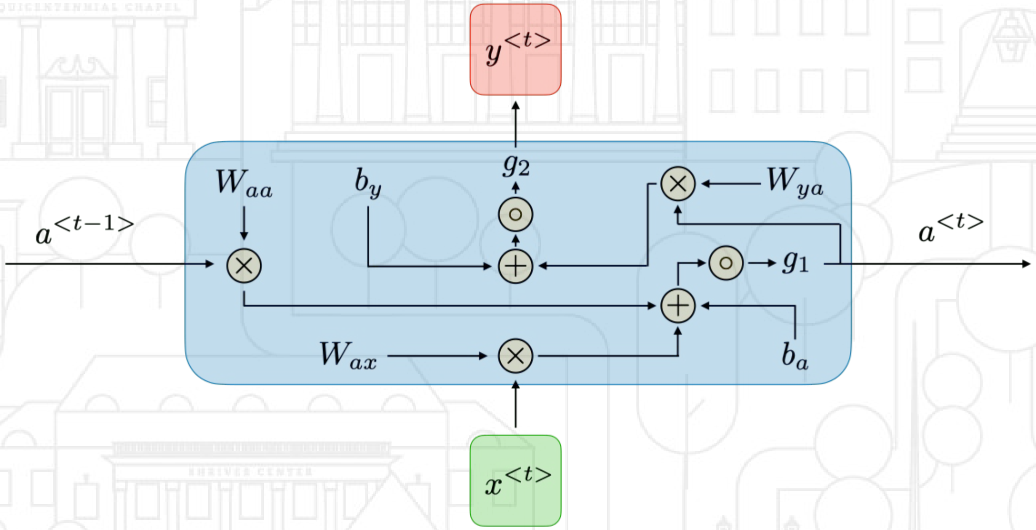
Module 6

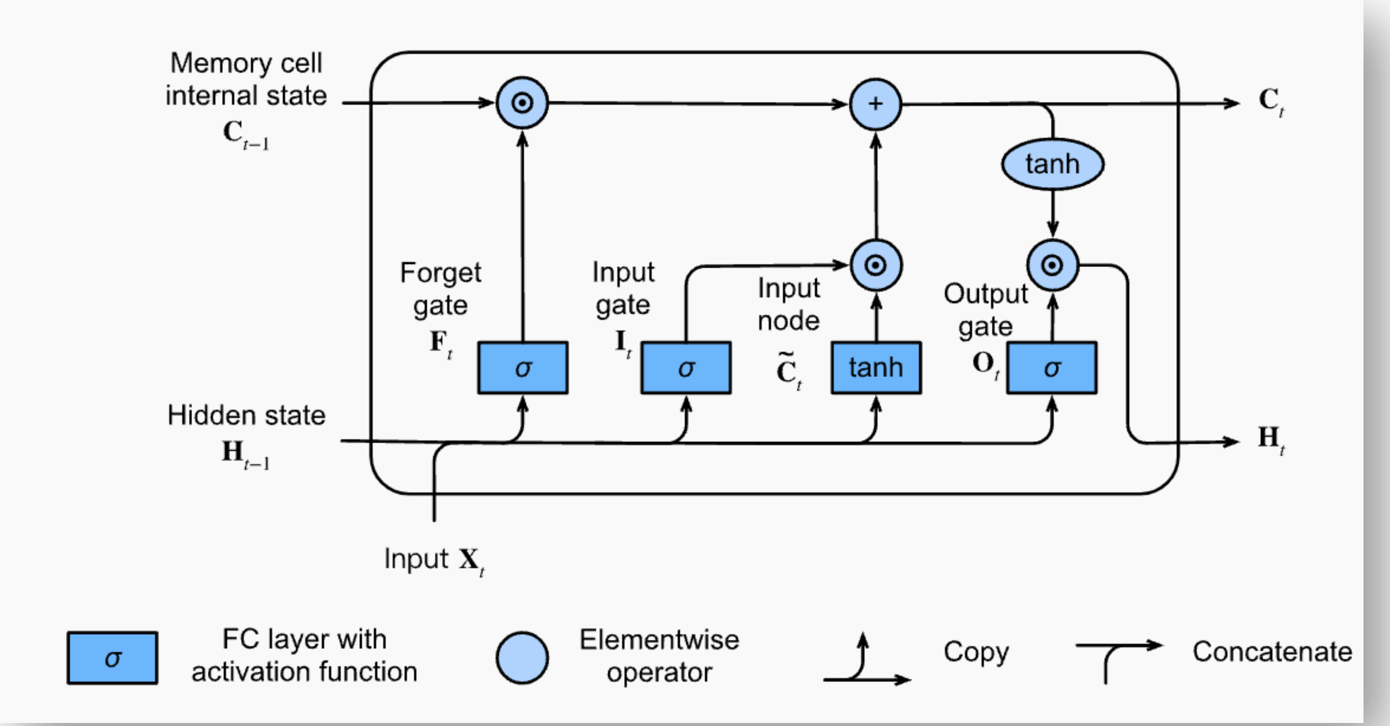
Recurrent Neural Networks

DL: RNNs



LSTM Gates

- Gates allow LSTMs to control short- and long-term memories



$$\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xf}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{hf}} + \mathbf{b}_{\mathrm{f}})$$

$$\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xi}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{hi}} + \mathbf{b}_{\mathrm{i}})$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{\mathrm{xc}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{hc}} + \mathbf{b}_{\mathrm{c}})$$

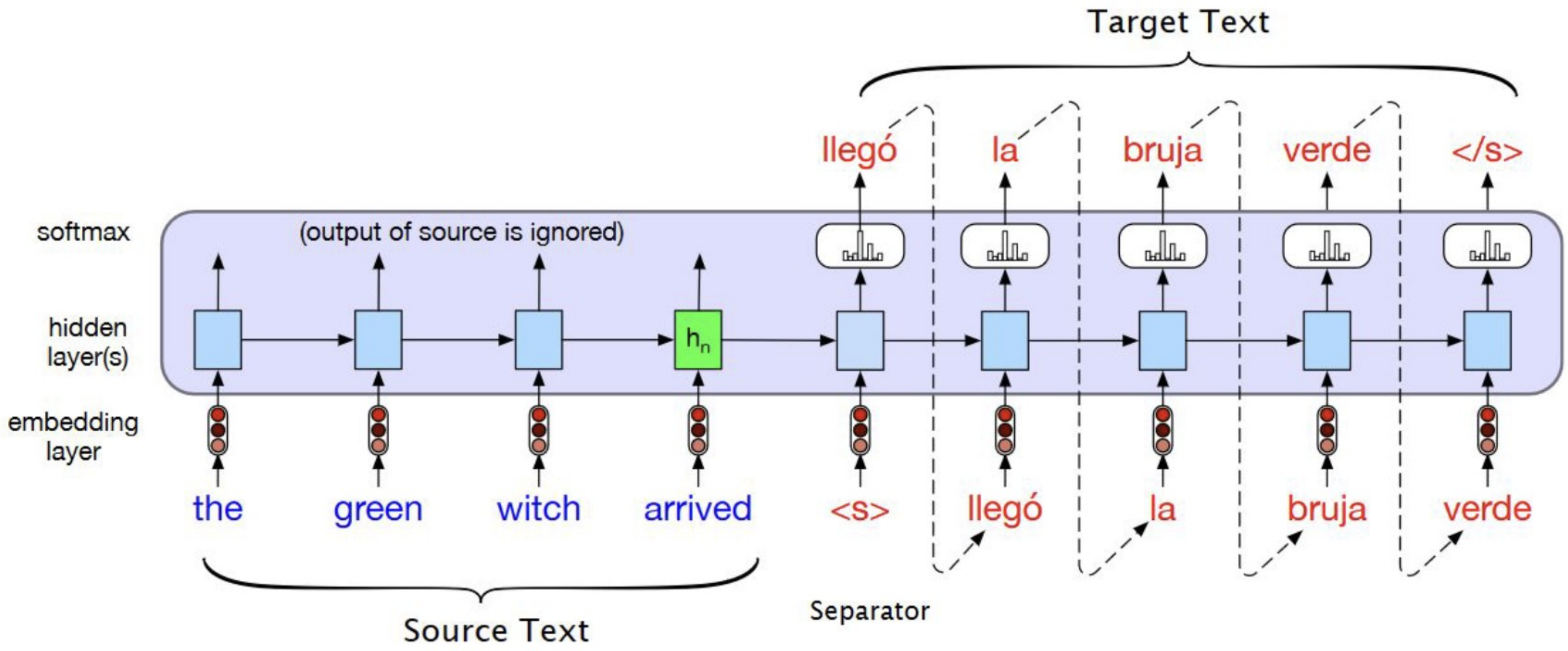
$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t$$

$$\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\mathrm{xo}} + \mathbf{H}_{t-1} \mathbf{W}_{\mathrm{ho}} + \mathbf{b}_{\mathrm{o}})$$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t).$$

Encoder-Decoder LSTM

- Seq2seq LSTM encoder-decoder networks

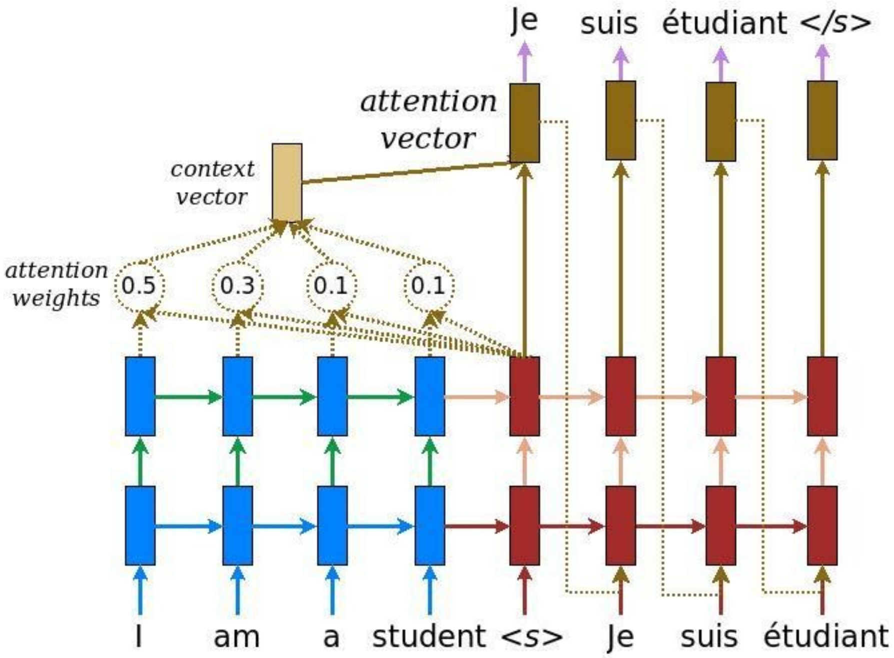


Attention Layers

- Traditional models like RNNs and LSTMs treat all time steps equally.
- In reality, some moments matter more than others.
- Goal: Learn to focus on important time steps dynamically.

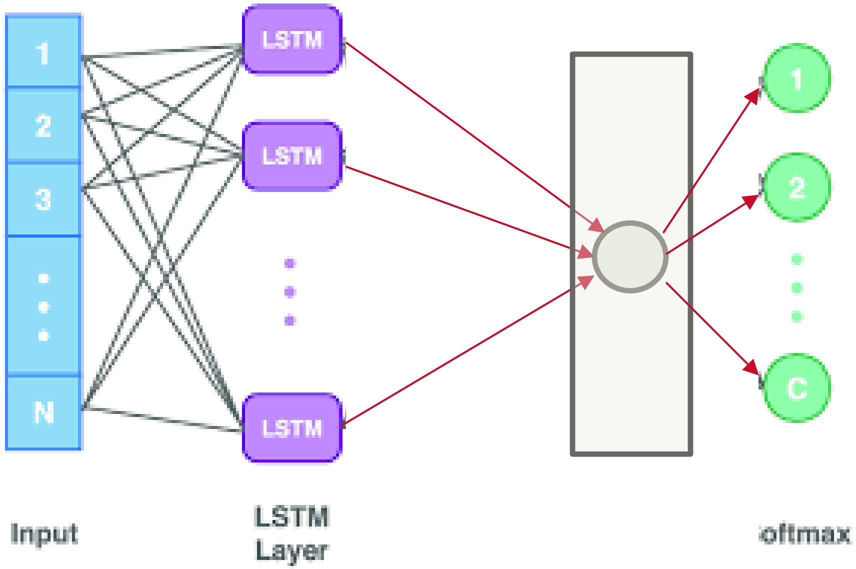
What is Attention?

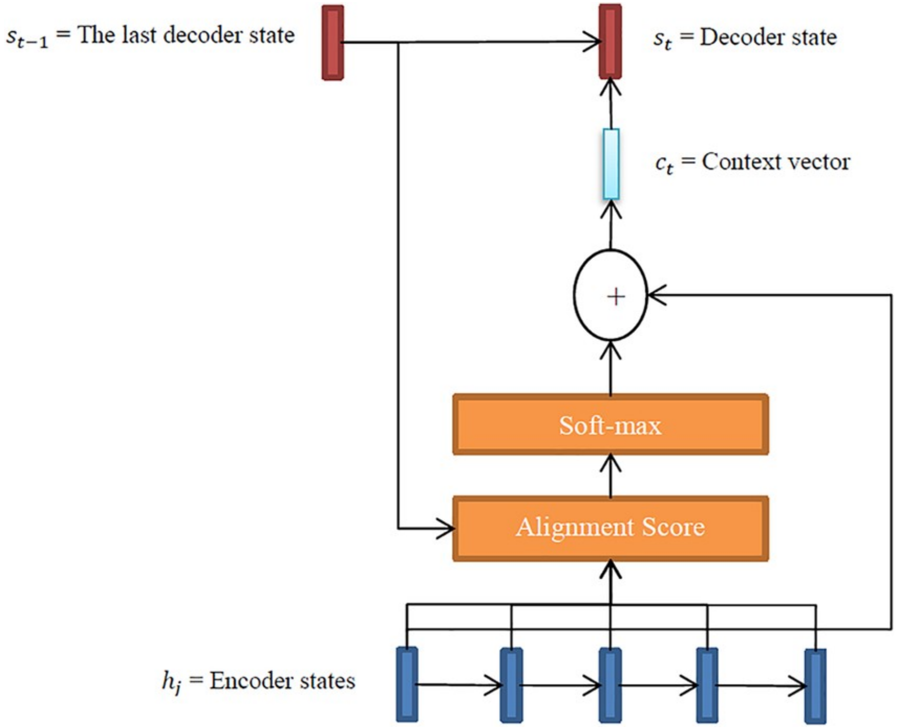
- A mechanism that lets models learn where to focus.
- Assigns a weight to each input element.
- Outputs a context vector summarizing important parts.



How Attention Works

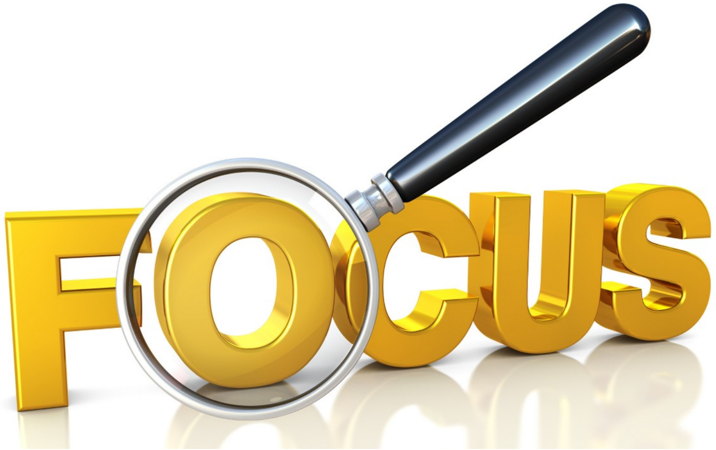
- Score each time step with a small neural network.
- Apply softmax to get attention weights.
- Multiply each hidden state by its weight.
- Sum to get the context vector.





Attention Layers (Business Uses)

- Besides the regular use of LLMs
  - Forecasting
  - Past sales, promotions, holidays, weather
  - Attention focuses on:
    - Holiday seasons
    - Promotional weeks
    - Special weather events



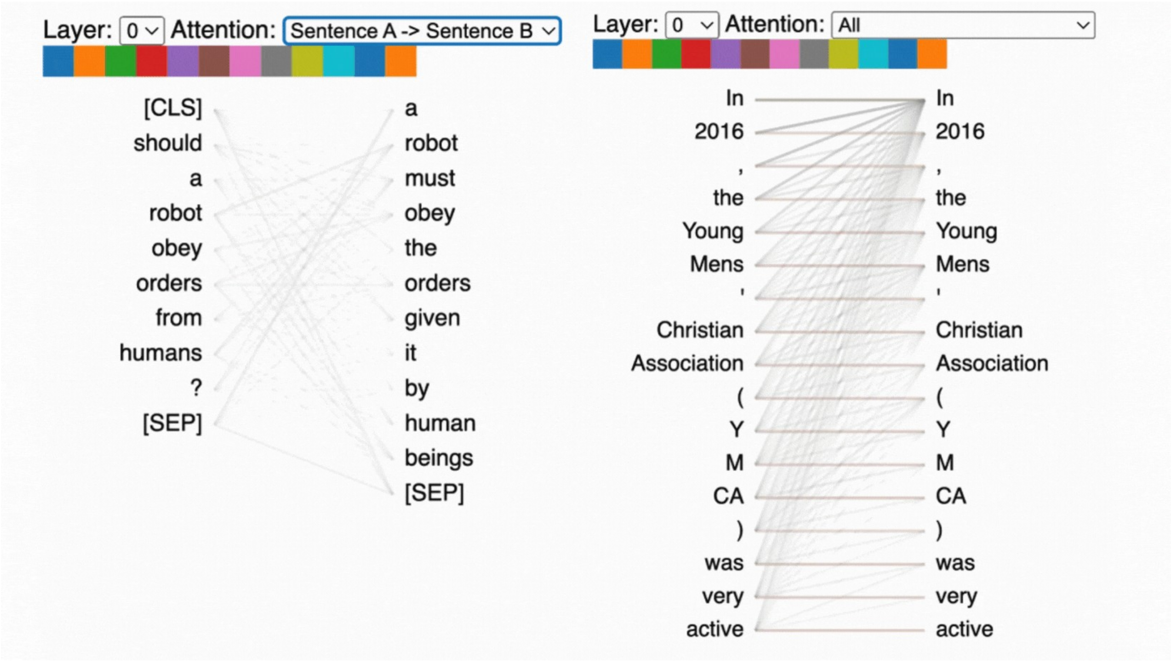
Attention Layers Anatomy

- Example of Attention Layers:
  - Dense(1): Assigns a score to each time step.
  - Softmax: Normalizes into probabilities.
  - Reduce_sum: Dot Product creates the context vector.

```
attention_scores = Dense(1, activation='tanh')(lstm_out)
attention_weights = Activation('softmax')(attention_scores)
context_vector = tf.reduce_sum(attention_weights * lstm_out,
axis=1)
```

Benefits of Attention Layers

- Improved Accuracy: Focus on key time steps.
- Interpretability: Understand what the model "looks at."
- Scalability: Easier learning for long sequences.



Python

Multi-Head Attention Layers

- Introduced in Transformer models.
- Applies multiple attention heads in parallel.
- Each head learns to focus on different aspects of the input.
- Outputs from all heads are concatenated and linearly transformed.

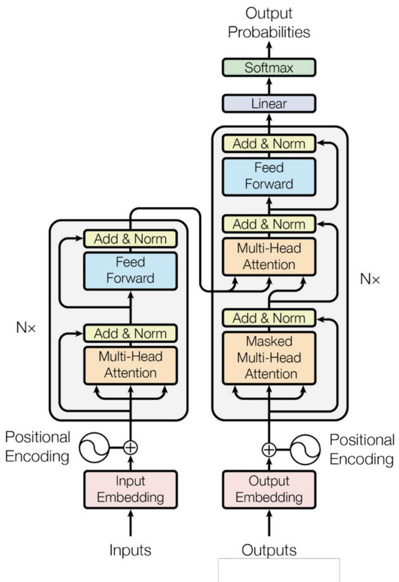
Single Head	Multi-Head
One perspective on sequence	Many parallel perspectives
May miss complex patterns	Captures diverse relationships
Limited capacity	Higher representational power

Multi-Head Attention Layers

- Introduced in Transformer models.
- Applies multiple attention heads in parallel.
- Each head learns to focus on different aspects of the input.
- Outputs from all heads are concatenated and linearly transformed.

BERT

Encoder



GPT

Decoder

Python