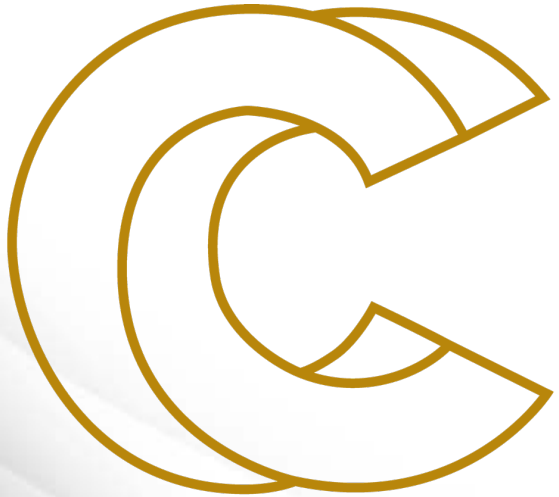


EURO^2

Mastering Transformers: From Building Blocks to Real-World Applications

Attention by Prof.Dr.Tuğba Taşkaya Temizel

12 Sept 2023

What is Attention?

- Attention is, to some extent, motivated by how we pay visual attention to different regions of an image or correlate words in one sentence.

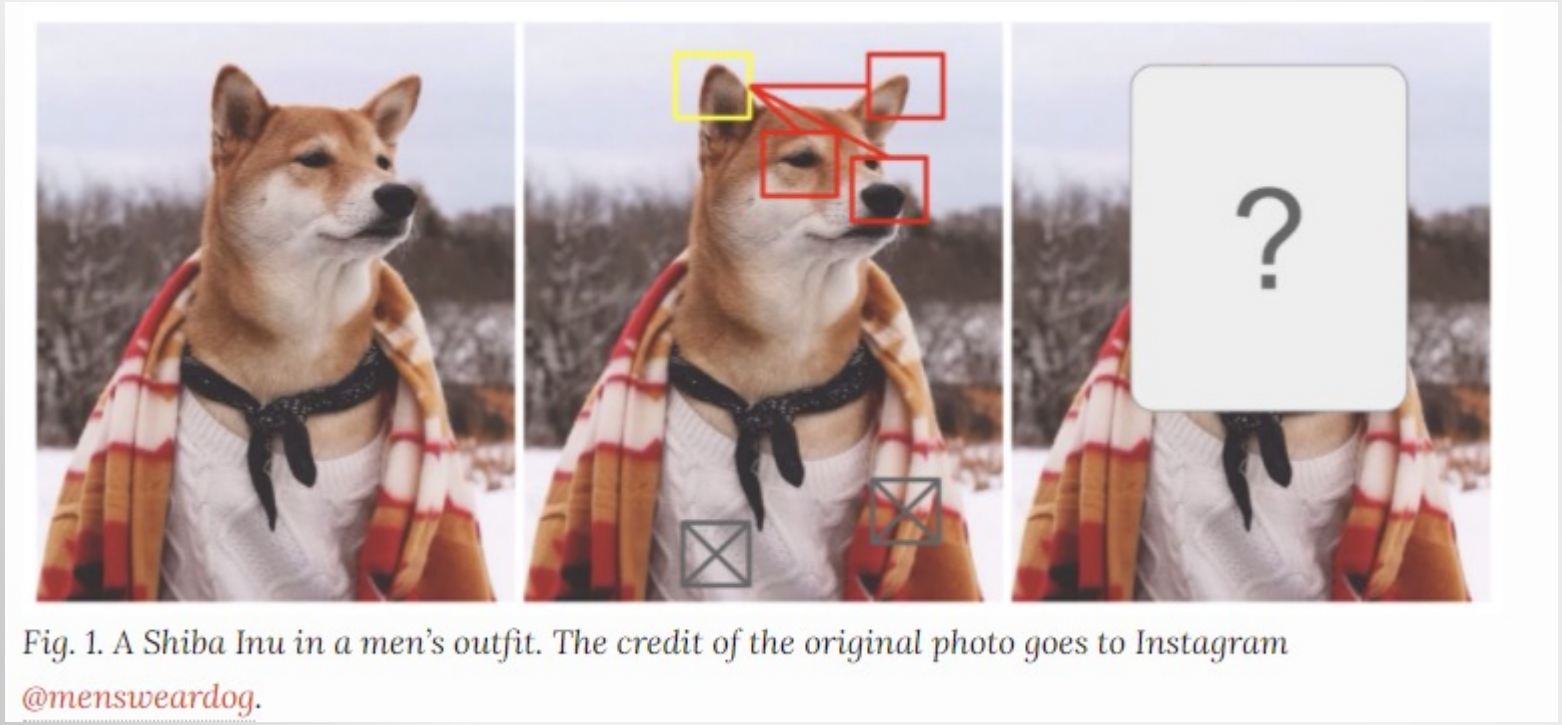


Fig. 1. A Shiba Inu in a men's outfit. The credit of the original photo goes to Instagram @mensweardog.

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

What is Attention?

- Attention is a weight vector, which is dynamically calculated using deep networks.

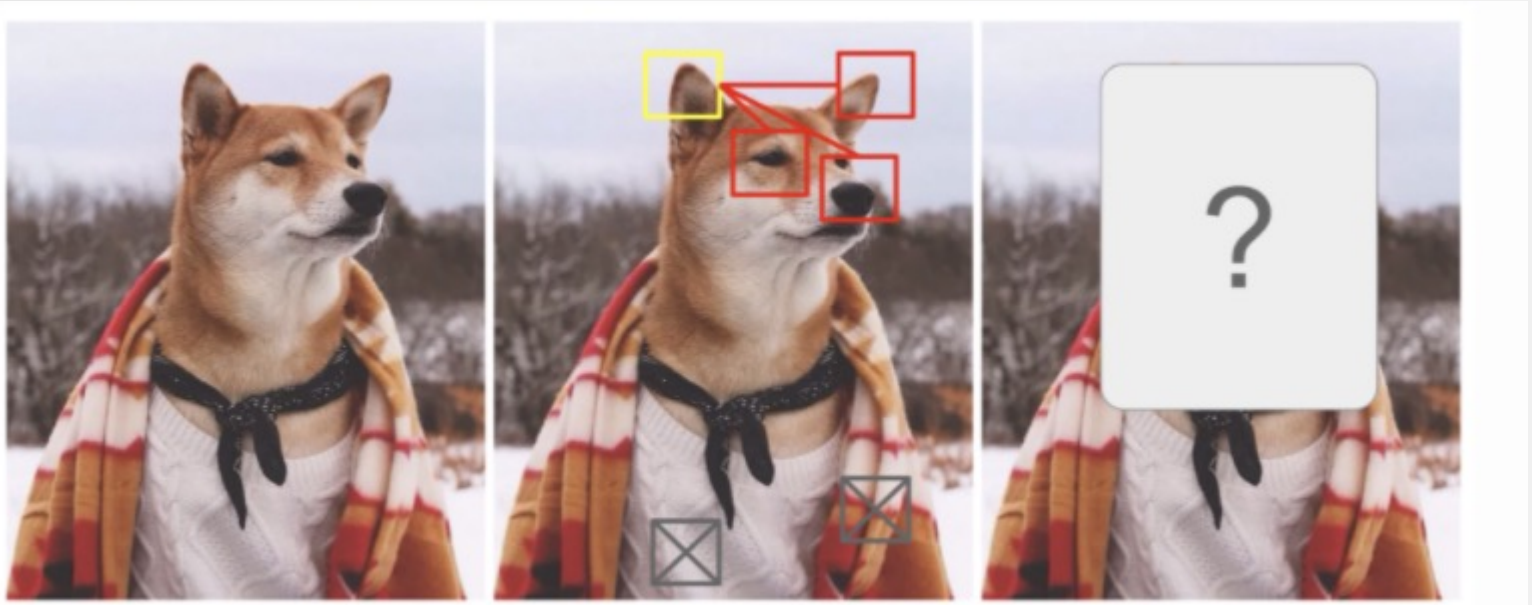
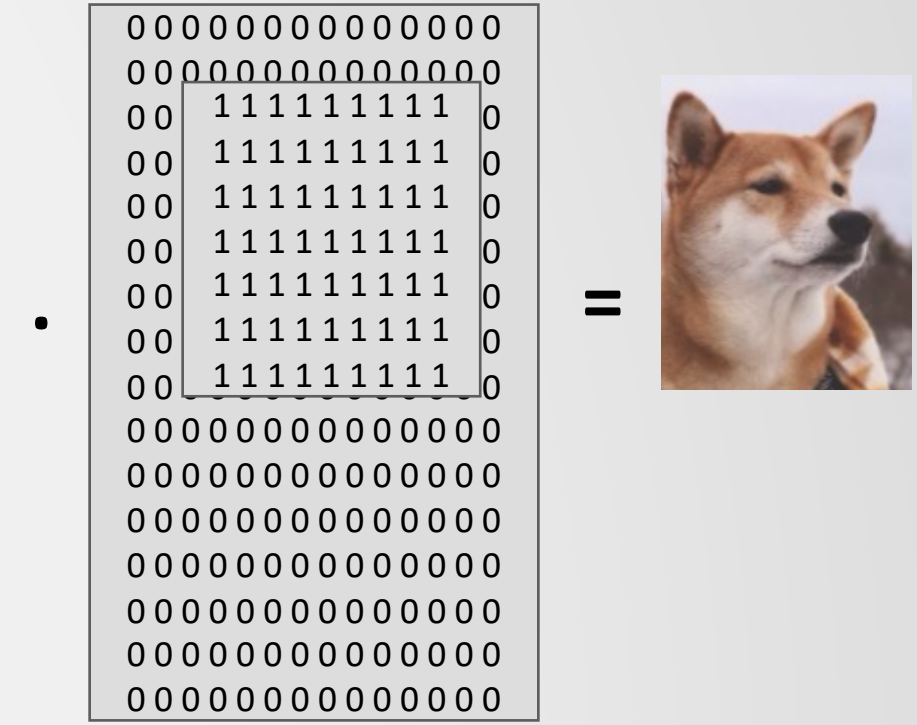


Fig. 1. A Shiba Inu in a men's outfit. The credit of the original photo goes to Instagram @mensweardog.



https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

What is Attention?

- Let’s make an analogy with databases.

Word ID	Word
1	Dog
2	Cat
3	Rabbit

Key **Value**

Query: Retrieve the Word ID=1

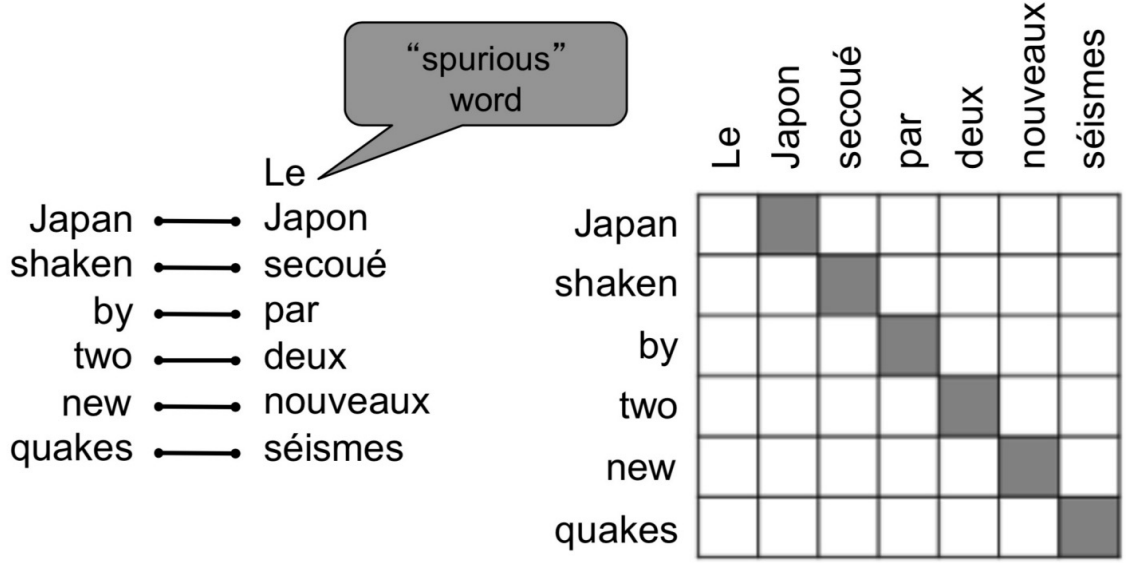
Query is aligned with the key in the database.
The matching key returns the value.

This is one-to-one alignment-> Query is matched with only one word id.

Background: Alignment

Example: Alignment for Machine Translation

- Let's think about machine translation task.
  - Machine translation is the task of translating a sentence *x* in one language (source language) to another sentence *y* in another language (target language).
- Alignment is the correspondence between particular words in the translated sentence pair.
- Typological differences between languages lead to complicated alignments!
- Note: Some words have no counterpart



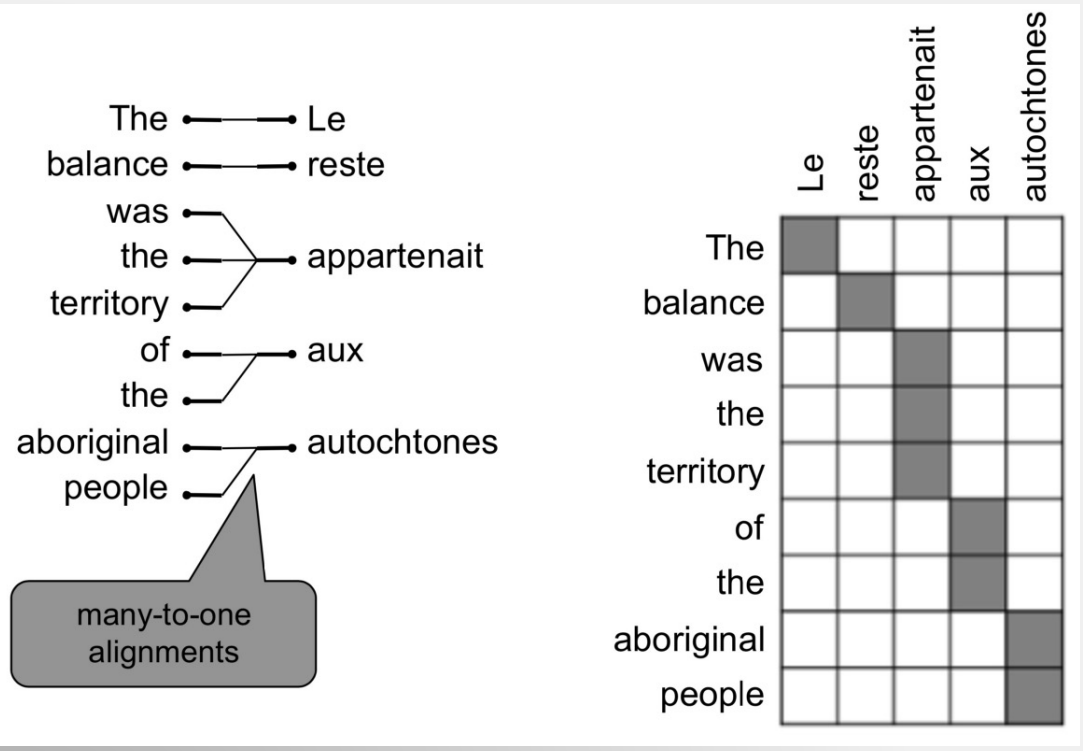
- Alignment can be one-to-one

Examples from: "The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

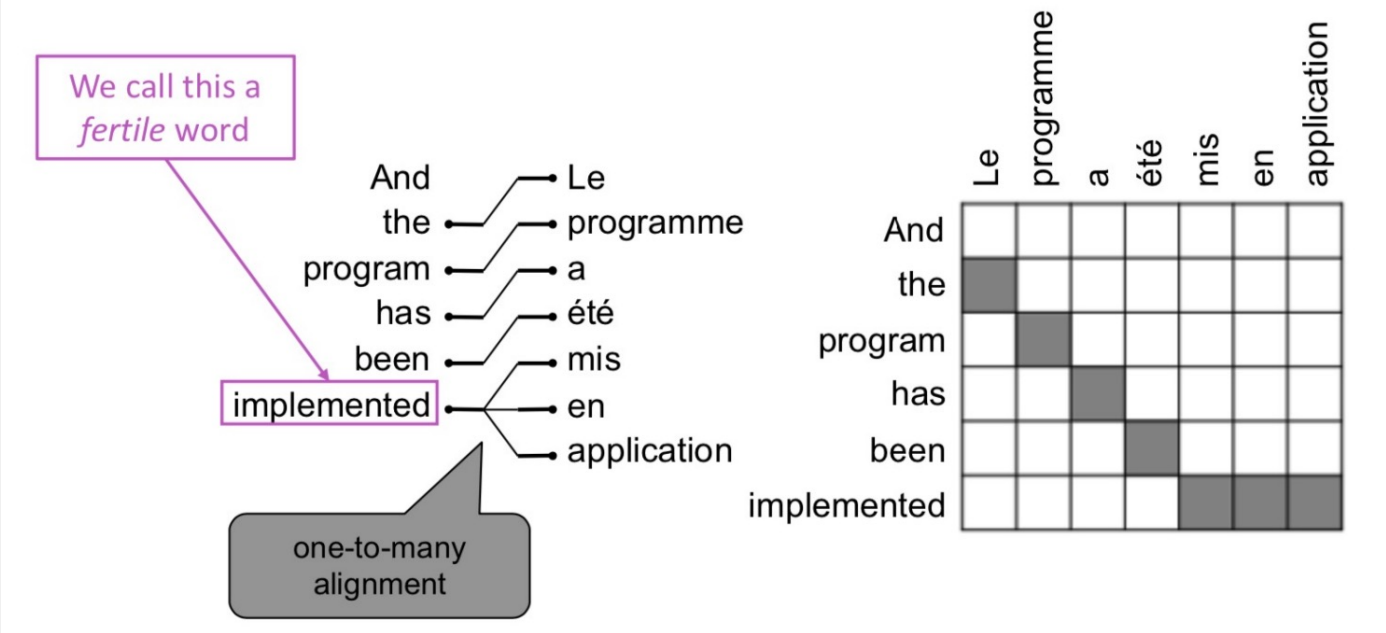
Background: Alignment

Example: Alignment for Machine Translation

- Alignment can be many-to-one



- Alignment can be one-to-many

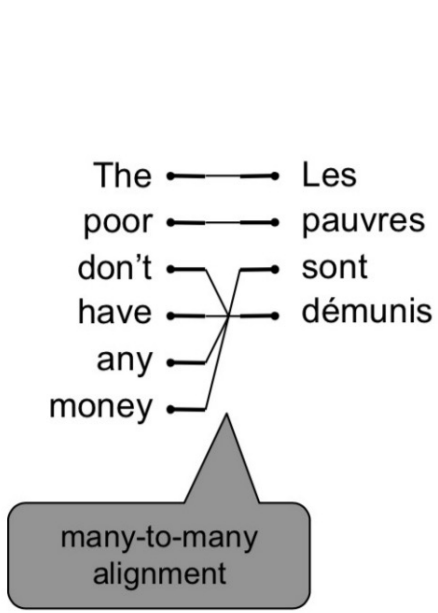


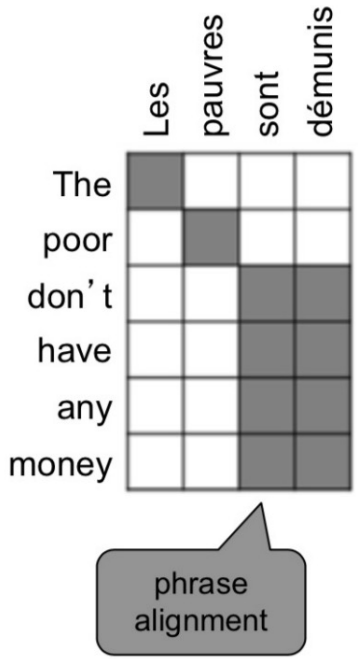
Examples from: "The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

Background: Alignment

Example: Alignment for Machine Translation

- Alignment can be many-to-many (phrase-level)





Examples from: "The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture07-nmt.pdf

Background: Alignment

Example: Alignment for Machine Translation

- We learn as a combination of several factors, including:
  - Probability of particular words aligning (also depends on position in sent)
  - Probability of particular words having a particular fertility (number of corresponding words)
  - etc.
- Alignments a are latent variables: They aren't explicitly specified in the data!
  - Require the use of special learning algorithms (like Expectation-Maximization) for learning the parameters of distributions with latent variables
  - Earlier studies relied on statistical machine translation (SMT)

https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture07-nmt.pdf

What is Attention?

- Attention can be implemented and calculated in numerous ways and architectures.
  - Attention is a general deep learning technique.
- Keys, values and queries vary based on the application task and model.
  - Given a set of vector values, and a vector query, attention is a technique to compute a weighted sum of the values, dependent on the query.
  - In other words, the output is the weighted sum of the values, where the weight of each value is given by the compatibility between query and key. The dot product provides a measure of compatibility.

  - *Query attends to the values!*
  - *Query determines which values to focus on!*
  - *Keys, Values and Query are vectors.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

What is Attention?

- Let's denote $\mathcal{D} \stackrel{\text{def}}{=} \{(\mathbf{k}_1, \mathbf{v}_1), \ldots (\mathbf{k}_m, \mathbf{v}_m)\}$ including *m* tuples of keys and values. Denote *q* as a query. We can define the attention over *D* as:

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$

- where $\alpha(q, k_i) \in \mathbb{R}\ (i = 1, \ldots, m)$ are scalar weights. This operation is also called as attention pooling.
  - If one of the weights $\alpha(q, k_i)$ is 1 and others are zero, it corresponds to a traditional database query.
  - All weights are equal, i.e. $\alpha(q, k_i)=\frac{1}{m}$ for all *i*. This amounts to averaging across the entire database, also called average pooling in deep learning.
  - The weights $\alpha(q, k_i)$ form a convex combination, i.e. $\sum_i \alpha(q, k_i) = 1$ and $\alpha(q, k_i) \geq 0$ for all *i*. This is the common setting in deep learning.

https://d2l.ai/chapter_attention-mechanisms-and-transformers/queries-keys-values.html

Attention in Seq2Seq Models

Background
Encoder-Decoder Model

- Encoder-decoder networks, or sequence-to-sequence networks, are models capable of generating contextually appropriate, arbitrary length, output sequences.

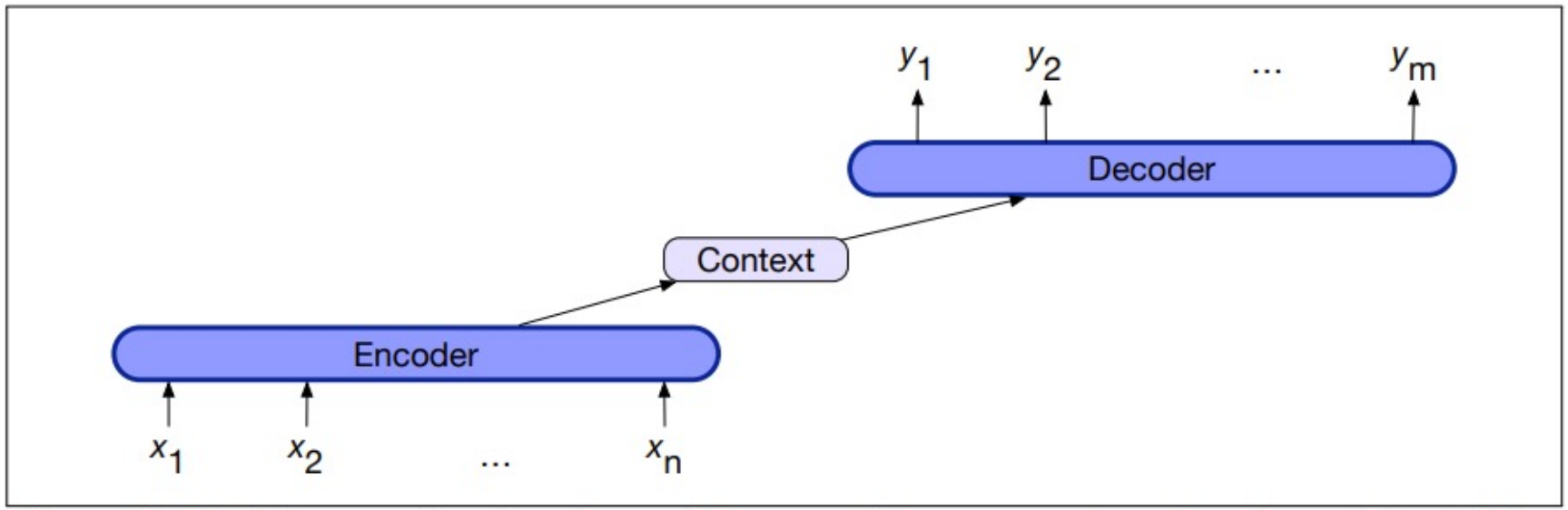
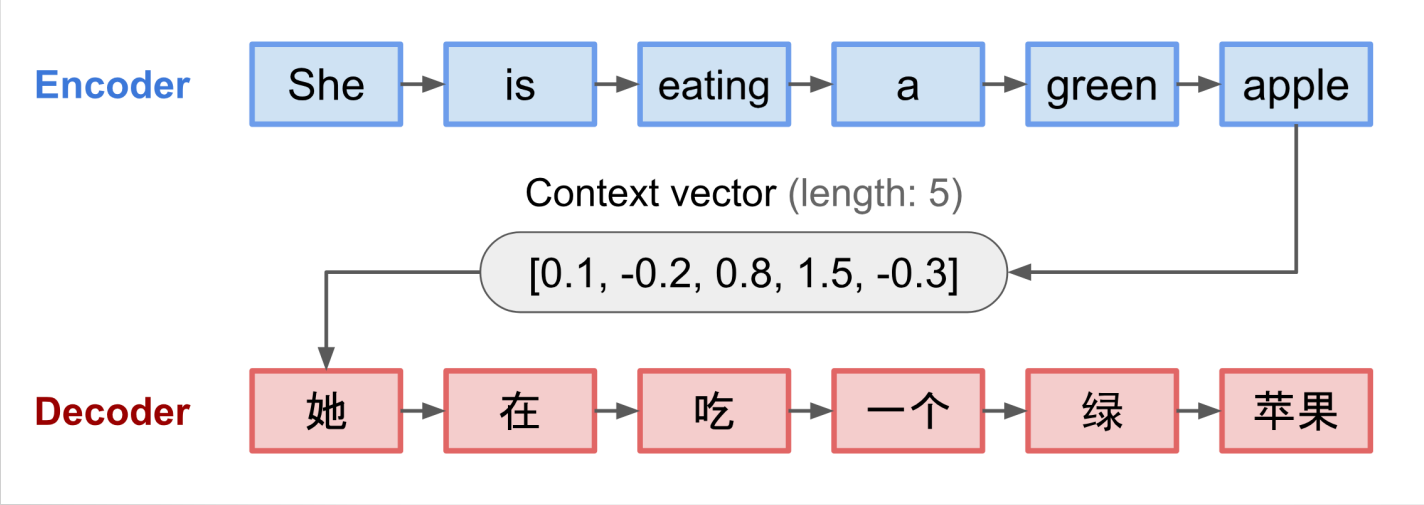


Figure 11.3 The encoder-decoder architecture. The context is a function of the hidden representations of the input, and may be used by the decoder in a variety of ways.

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

Background
Encoder-Decoder Model

- Encoder-decoder networks consist of three components:
  1. An encoder that accepts an input sequence, x_1^n, and generates a corresponding sequence of contextualized representations, h_1^n.
     - LSTMs, GRUs, convolutional networks, and Transformers can all be employed as encoders.
  2. A context vector, c, which is a function of h_1^n, and conveys the essence of the input to the decoder.



https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Background
Encoder-Decoder Model

- Encoder-decoder networks consist of three components:
  3. A decoder, which accepts *c* as input and generates an arbitrary length sequence of hidden states h_1^m, from which a corresponding sequence of output states y_1^m, can be obtained.
     - Just as with encoders, decoders can be realized by any kind of sequence architecture.

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

What's wrong with Seq2Seq Model?

- The final hidden state is acting as a bottleneck: it must represent absolutely everything about the meaning of the source text, since the only thing the decoder knows about the source text is what's in this context vector.

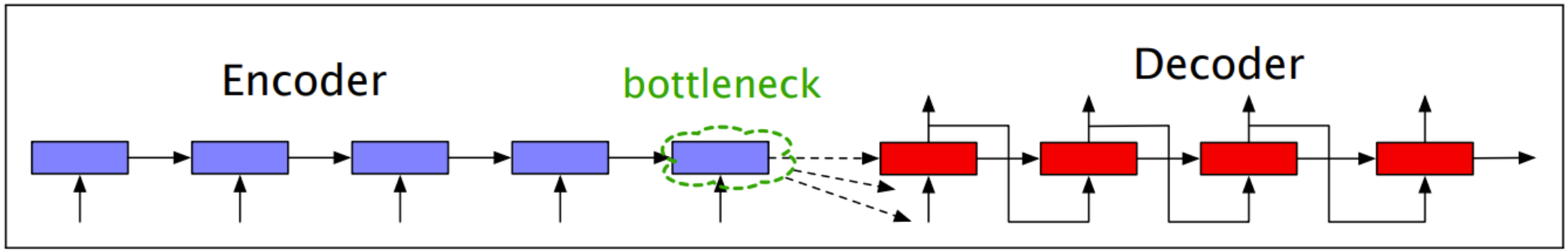


Figure 11.8 Requiring the context c to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

Encoder: Builds up sentence meaning

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

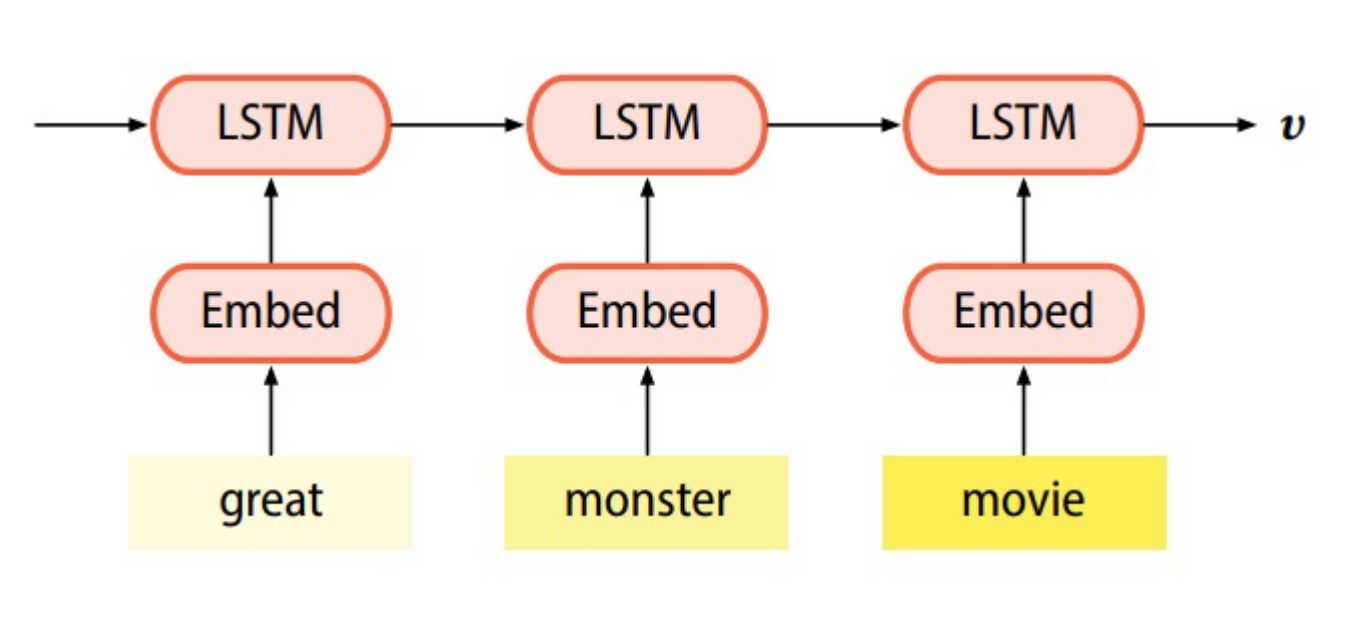
What's wrong with Seq2Seq Model?

- As the input sequence gets compressed and blended into a single dense vector, sufficiently large memory capacity is required to store past information.
  - As a result, the network generalizes poorly to long sequences while wasting memory on shorter ones.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

What's wrong with Seq2Seq Model?

- Recency bias: Information at the beginning of the sentence, especially for long sentences, may not be equally well represented in the context vector.
- The last hidden state is prone to bias towards the recent past.



https://liu-nlp.github.io/dl4nlp/slides/slides-163.pdf

Attention in Seq2Seq Model

- The attention mechanism is a solution to the bottleneck problem, a way of allowing the decoder to get information from all the hidden states of the encoder, not just the last hidden state.
- Here, the context vector *c* is a single vector that is a function of the hidden states of the encoder, that is, $c = f(h_1^n)$.
  - Because the number of hidden states varies with the size of the input, we can't use the entire tensor of encoder hidden state vectors directly as the context for the decoder.
  - The idea of attention is instead to create the single fixed-length vector *c* by taking a weighted sum of all the encoder hidden states h_1^n .

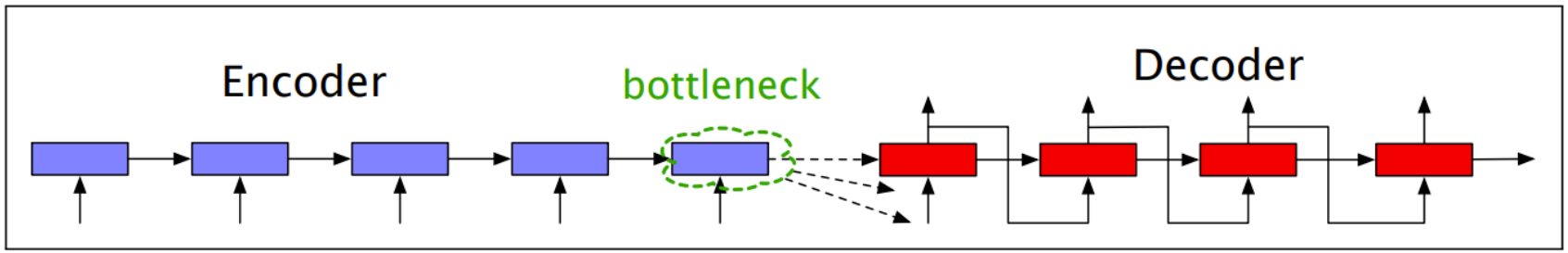


Figure 11.8 Requiring the context *c* to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

Attention in Seq2Seq Model

- Attention replaces the static context vector with one that is dynamically derived from the encoder hidden states at each point during decoding.
- This context vector, c_i, is generated anew with each decoding step *i* and takes all the encoder hidden states into account in its derivation.

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

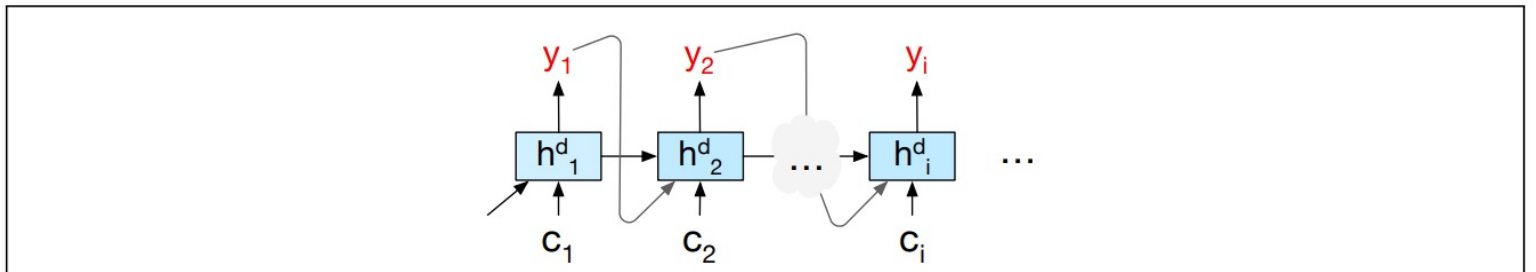


Figure 11.9 The attention mechanism allows each hidden state of the decoder to see a different, dynamic, context, which is a function of all the encoder hidden states.

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

Attention in Seq2Seq Model

- Let's calculate how relevant each encoder state is to the decoder state captured in h_{i-1}^{d}
- We capture relevance by computing— at each state *i* during decoding—a score(h_{i-1}^{d}, h_{j}^{e}) for each encoder state *j*.

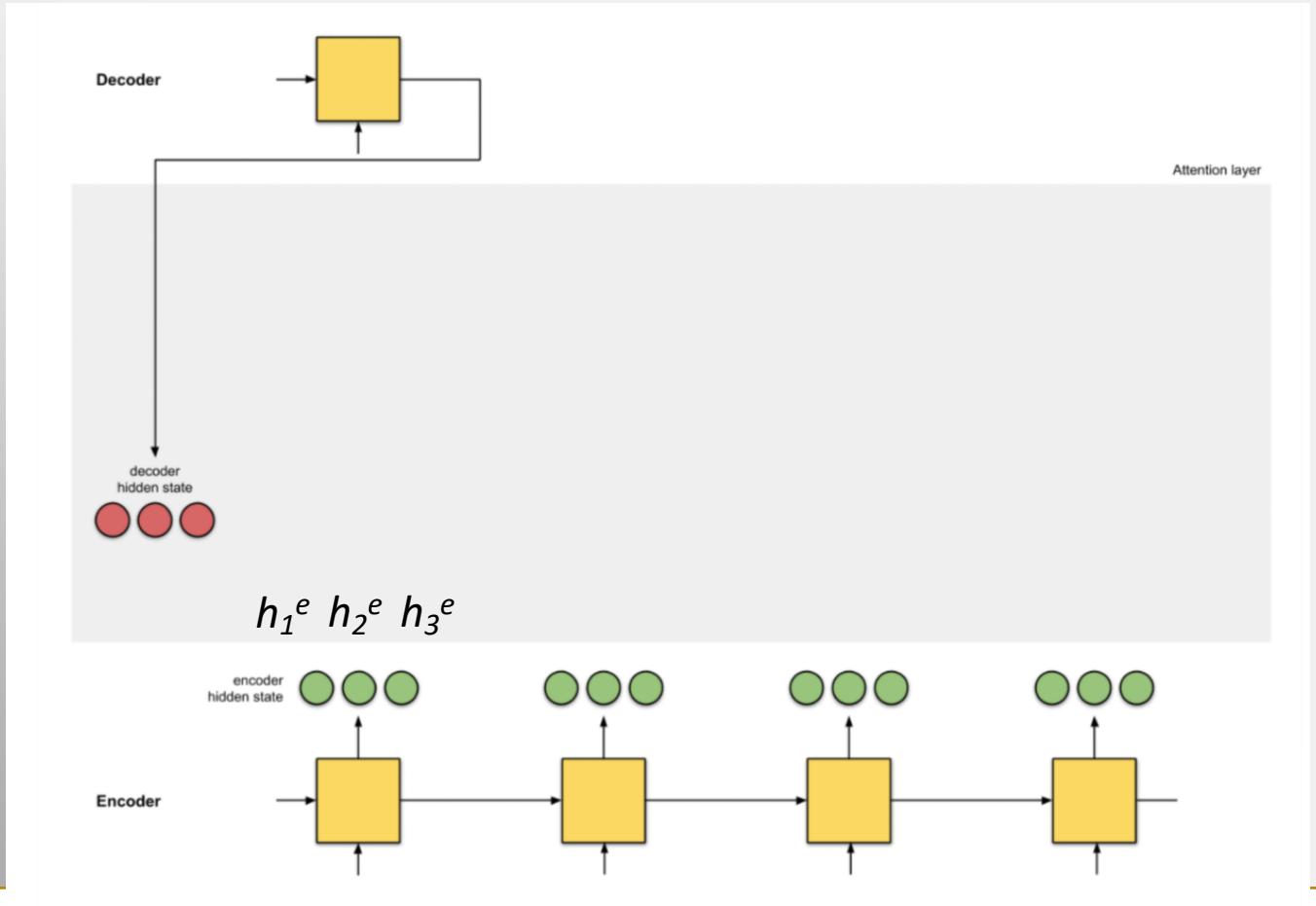
$$score(h_{i-1}^{d}, h_{j}^{e}) = h_{i-1}^{d} \cdot h_{j}^{e}$$

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

Attention in Seq2Seq Model

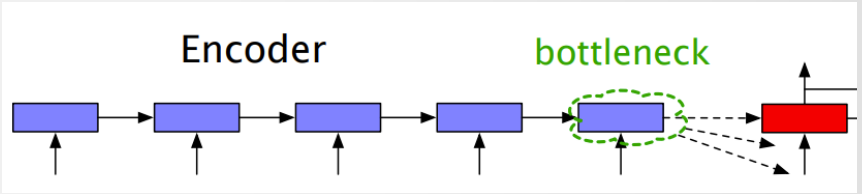
A Toy Example

- The vector of these scores across all the encoder hidden states gives us the relevance of each encoder state to the current step of the decoder.



In this example, we have 4 encoder hidden states and the current decoder hidden state ($d=4$).

Note that the last consolidated encoder hidden state is fed as input to the first time step of the decoder.



Encoder last state.

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

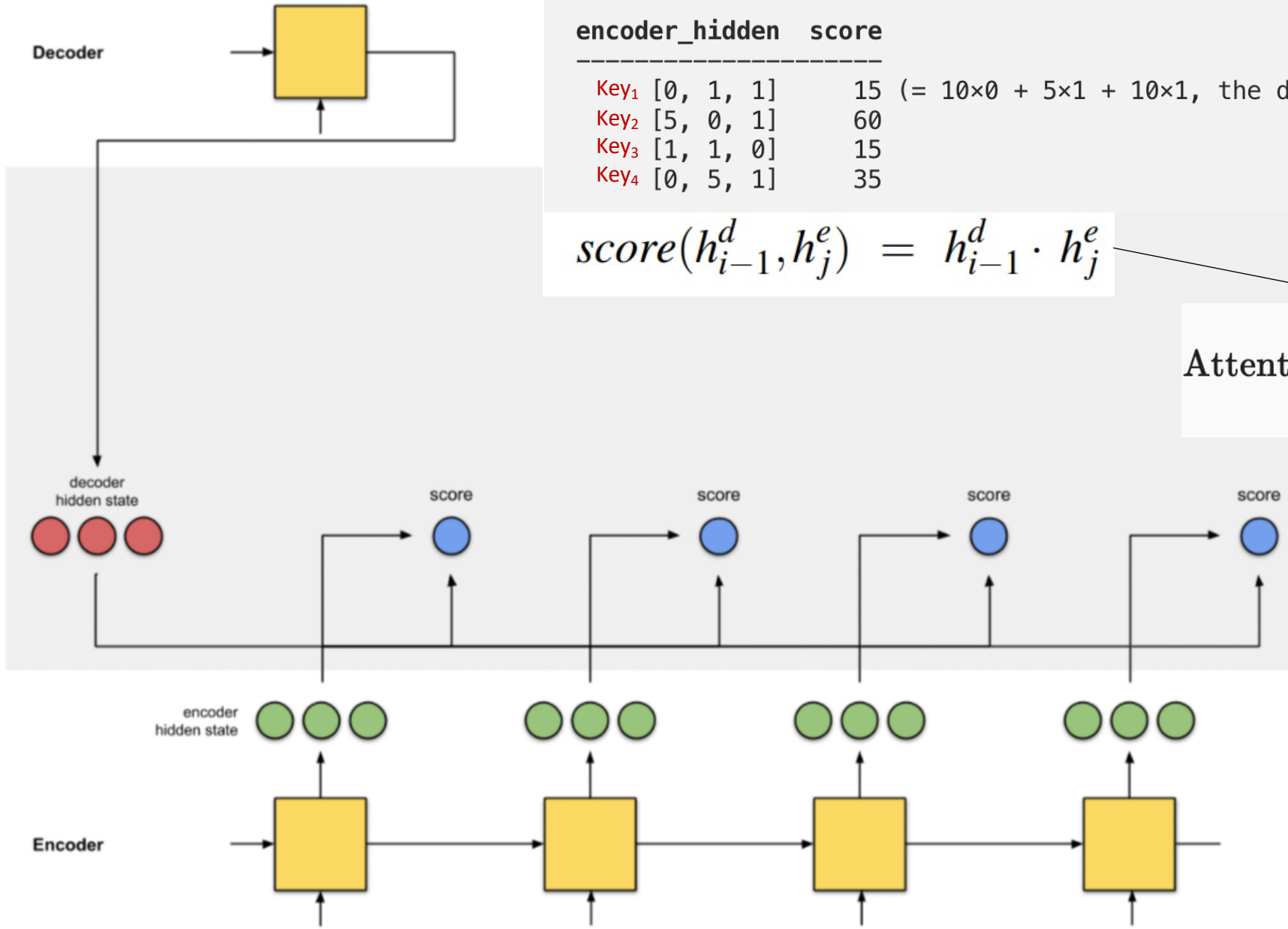
```
decoder_hidden = [10, 5, 10]  ⟶ Query

encoder_hidden  score
--------------------
Key1 [0, 1, 1]     15 (= 10×0 + 5×1 + 10×1, the dot product)
Key2 [5, 0, 1]     60
Key3 [1, 1, 0]     15
Key4 [0, 5, 1]     35
```

$$score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$

We work on $\alpha(q, k_i)$

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \overset{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

```
decoder_hidden = [10, 5, 10]  ⟶ Query

encoder_hidden  score
--------------------
Key1 [0, 1, 1]     15 (= 10×0 + 5×1 + 10×1, the dot product)
Key2 [5, 0, 1]     60
Key3 [1, 1, 0]     15
Key4 [0, 5, 1]     35
```

$$score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$

We work on $\alpha(q, k_i)$

The higher the score means the more relevant the hidden states of encoder and decoder hidden states are.

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$

Calculate the dot product between the query and the key: The higher the similarity, the higher the dot products between vectors is.

Motivation: To identify the corresponding words in the queries that are similar to the keys.

Attention in Seq2Seq Model

- We'll normalize them with a softmax to create a vector of weights, α_{ij}, that tells us the proportional relevance of each encoder hidden state *j* to the prior hidden decoder state, h_{i-1}^{d}

$$\alpha_{ij} = \text{softmax}(score(h_{i-1}^{d}, h_{j}^{e}) \ \forall j \in e)$$

$$= \frac{\exp(score(h_{i-1}^{d}, h_{j}^{e})}{\sum_{k} \exp(score(h_{i-1}^{d}, h_{k}^{e}))}$$

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*

Attention in Seq2Seq Model

- Finally, given the distribution in α, we can compute a fixed-length context vector for the current decoder state by taking a weighted average over all the encoder hidden states.

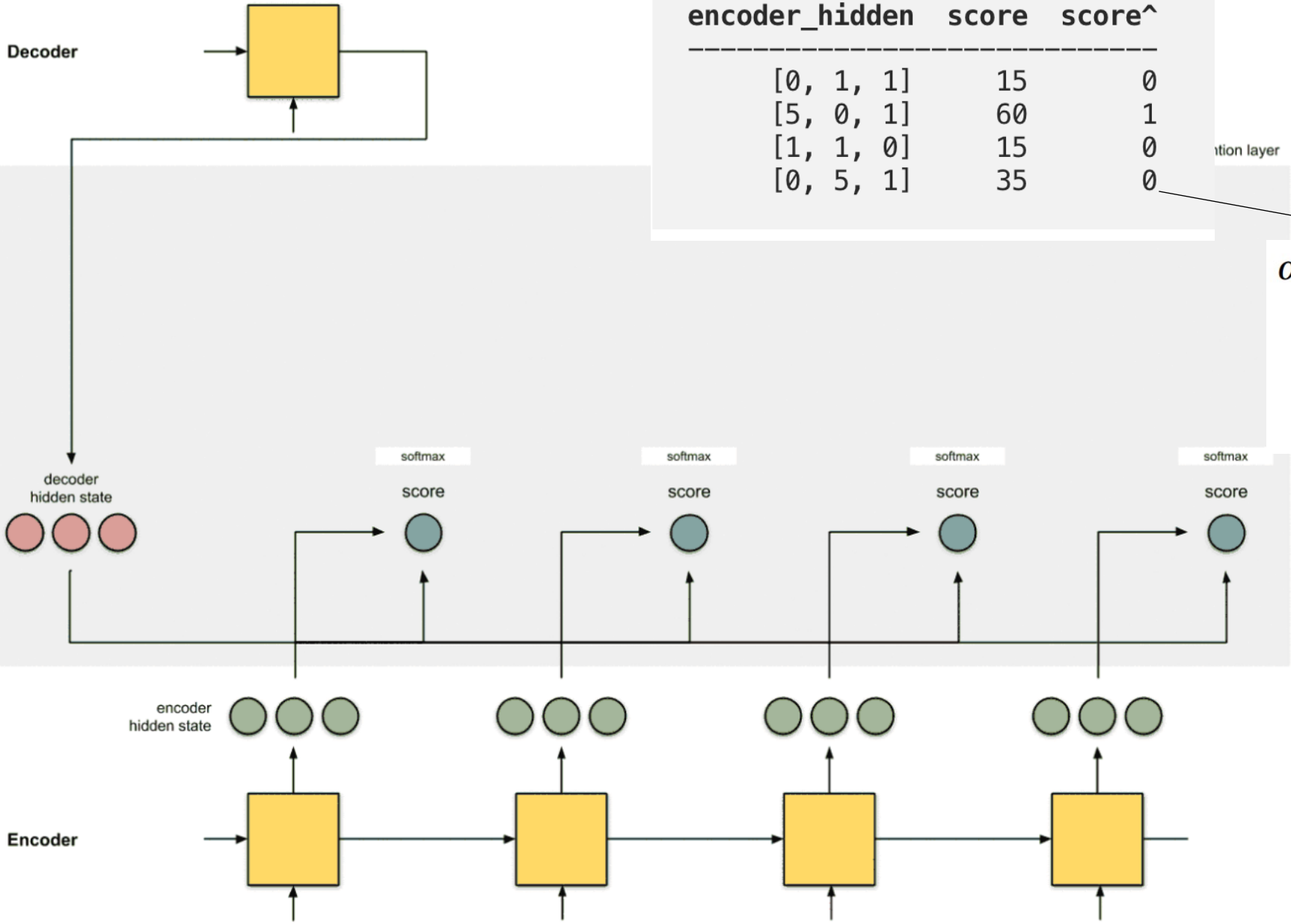
$$c_i = \sum_j \alpha_{ij} h_j^e$$

Corresponds to the Value

Recall:

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$

Jurafsky, D., & Martin, J. H. (2018). Speech and language processing. Machine Translation and Encoder-Decoder Models, *https://web.stanford.edu/~jurafsky/slp3/. 2020,* https://web.stanford.edu/~jurafsky/slp3/11.pdf *and Deep Learning Architectures for Sequence Processing, https://web.stanford.edu/~jurafsky/slp3/9.pdf*



encoder_hidden	score	score^
[0, 1, 1]	15	0
[5, 0, 1]	60	1
[1, 1, 0]	15	0
[0, 5, 1]	35	0

$$\alpha_{ij} = \text{softmax}(score(h^d_{i-1}, h^e_j) \ \forall j \in e)$$

$$= \frac{\exp(score(h^d_{i-1}, h^e_j)}{\sum_k \exp(score(h^d_{i-1}, h^e_k))}$$

The final scores after softmax here were rounded, simplied to help you follow the calculations better.

Total= 15+60+15+35=125

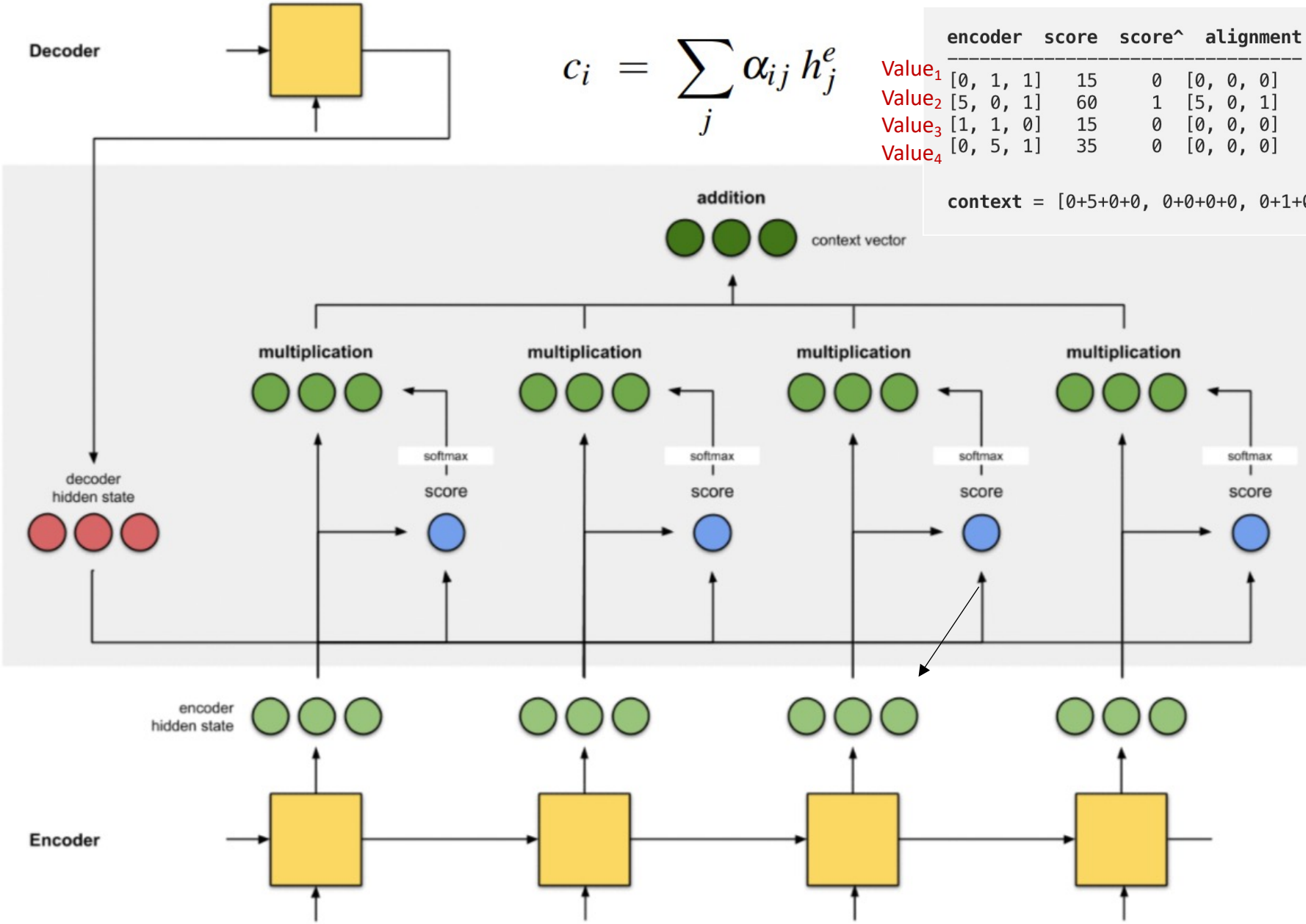
$$\alpha_{11} = \frac{15}{125} = 0.12 \sim 0$$

$$\alpha_{12} = \frac{60}{125} = 0.48 \sim 1$$

$$\alpha_{13} = \frac{15}{125} = 0.12 \sim 0$$

$$\alpha_{14} = \frac{35}{125} = 0.28 \sim 0$$

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3



	encoder	score	score^	alignment
$Value_1$	[0, 1, 1]	15	0	[0, 0, 0]
$Value_2$	[5, 0, 1]	60	1	[5, 0, 1]
$Value_3$	[1, 1, 0]	15	0	[0, 0, 0]
$Value_4$	[0, 5, 1]	35	0	[0, 0, 0]

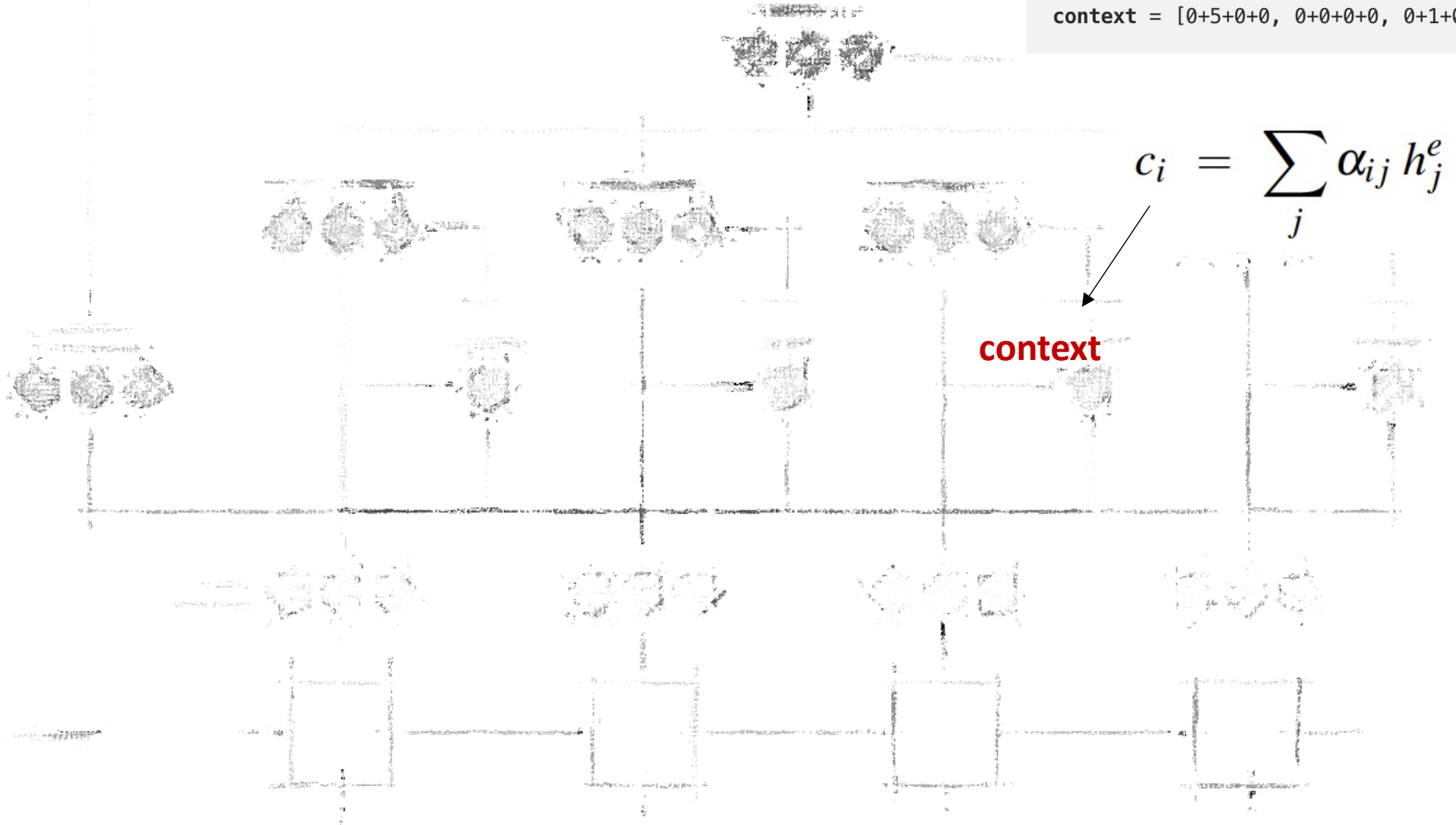
`context = [0+5+0+0, 0+0+0+0, 0+1+0+0] = [5, 0, 1]`

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

$$c_i = \sum_j \alpha_{ij} h_j^e$$

	encoder	score	score^	alignment
$Value_1$	[0, 1, 1]	15	0	[0, 0, 0]
$Value_2$	[5, 0, 1]	60	1	[5, 0, 1]
$Value_3$	[1, 1, 0]	15	0	[0, 0, 0]
$Value_4$	[0, 5, 1]	35	0	[0, 0, 0]

`context = [0+5+0+0, 0+0+0+0, 0+1+0+0] = [5, 0, 1]`



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Attention in Seq2Seq Model

- The set of $\{\alpha_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each output.

$$\alpha_{ij} = \text{softmax}(score(h^d_{i-1}, h^e_j) \ \forall j \in e)$$

$$= \frac{\exp(score(h^d_{i-1}, h^e_j)}{\sum_k \exp(score(h^d_{i-1}, h^e_k))}$$

In Bahdanau's paper, the alignment score α is parametrized by a feed-forward network with a single hidden layer and this network is jointly trained with other parts of the model.

$$score(h^d_{i-1}, h^e_j) = V_a^T \tanh(W_a[h^d_{i-1}; h^e_j])$$

The score function is therefore in the following form, given that tanh is used as the non-linear activation function. Here both V_a^T and W_a are weight matrices to be learned in the alignment model.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR

Attention in Seq2Seq Model

- Instead of simple dot product attention, we get a more powerful function that computes the relevance of each encoder hidden state to the decoder hidden state by parameterizing the score with its own set of weights, W_a.
- The matrix of alignment scores is a nice byproduct to explicitly show the correlation between source and target words.

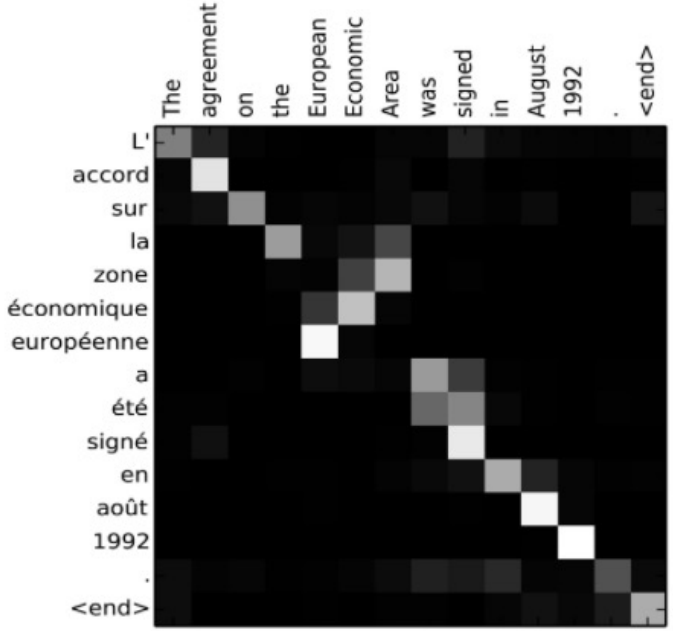


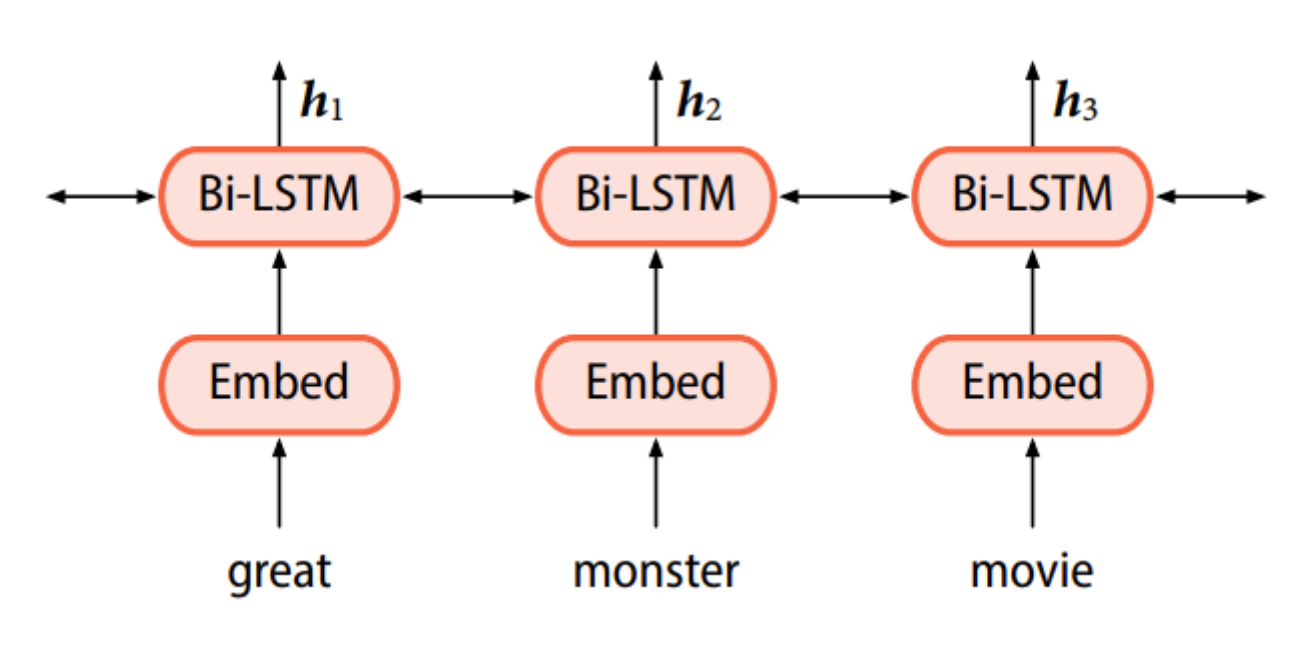
Fig. 5. Alignment matrix of "L'accord sur l'Espace économique européen a été signé en août 1992" (French) and its English translation "The agreement on the European Economic Area was signed in August 1992". (Image source: Fig 3 in Bahdanau et al., 2015)

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#summary

Attention in Classification

Attention in Classification

- In the classification architecture, keys and values are the same; they correspond to the hidden states h_i.



Assume that in this example we want to predict the sentiment of a given sentence.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

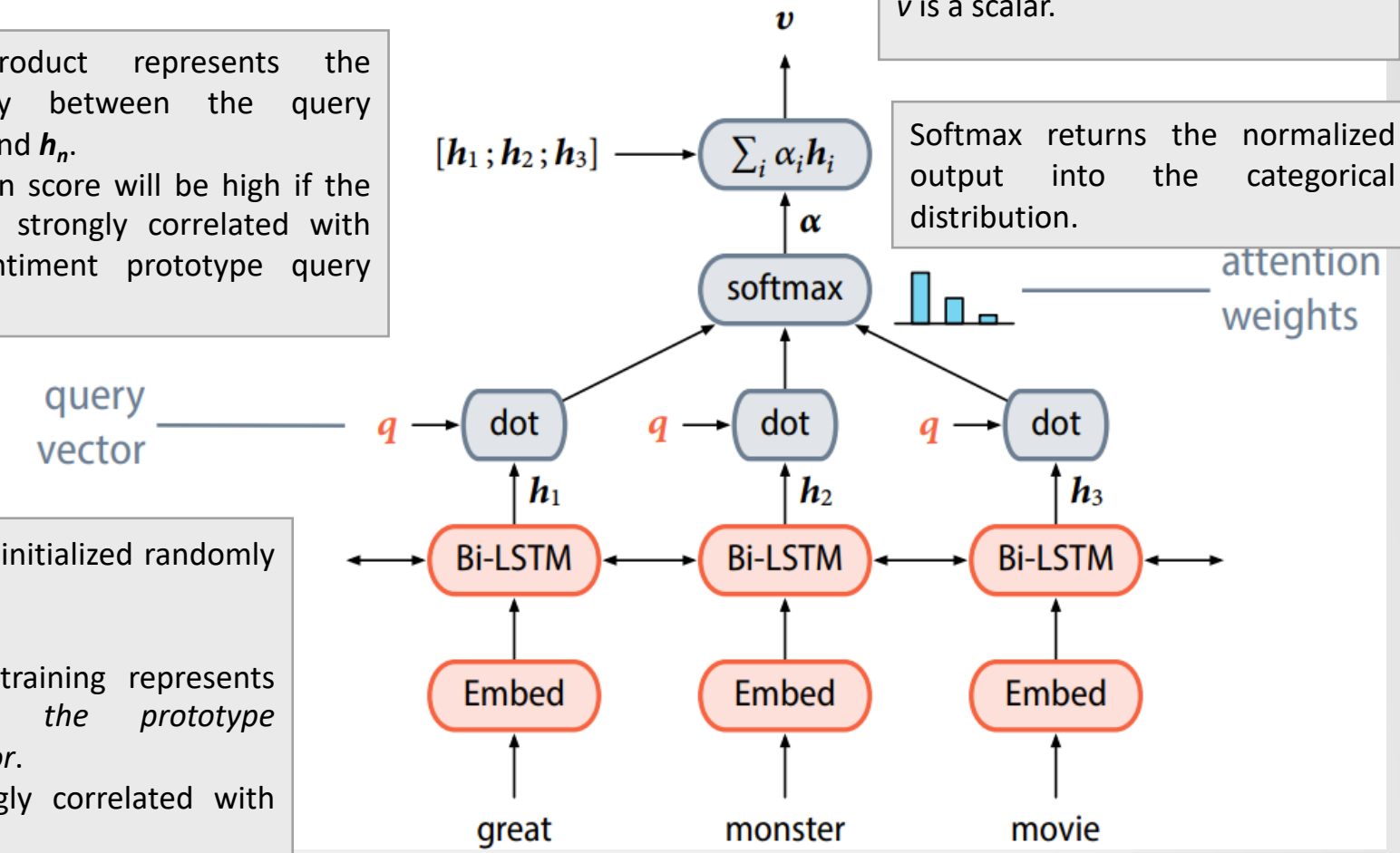
https://liu-nlp.github.io/dl4nlp/slides/slides-163.pdf

Attention in Classification

Dot product represents the similarity between the query vector and $\boldsymbol{h_n}$.
Attention score will be high if the word is strongly correlated with the sentiment prototype query vector.

v is a scalar.

Softmax returns the normalized output into the categorical distribution.



Query vector ***q*** is initialized randomly at the beginning.

Eventually after training represents something like *the prototype sentiment indicator*.
e.g. words strongly correlated with sentiment.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

https://liu-nlp.github.io/dl4nlp/slides/slides-163.pdf

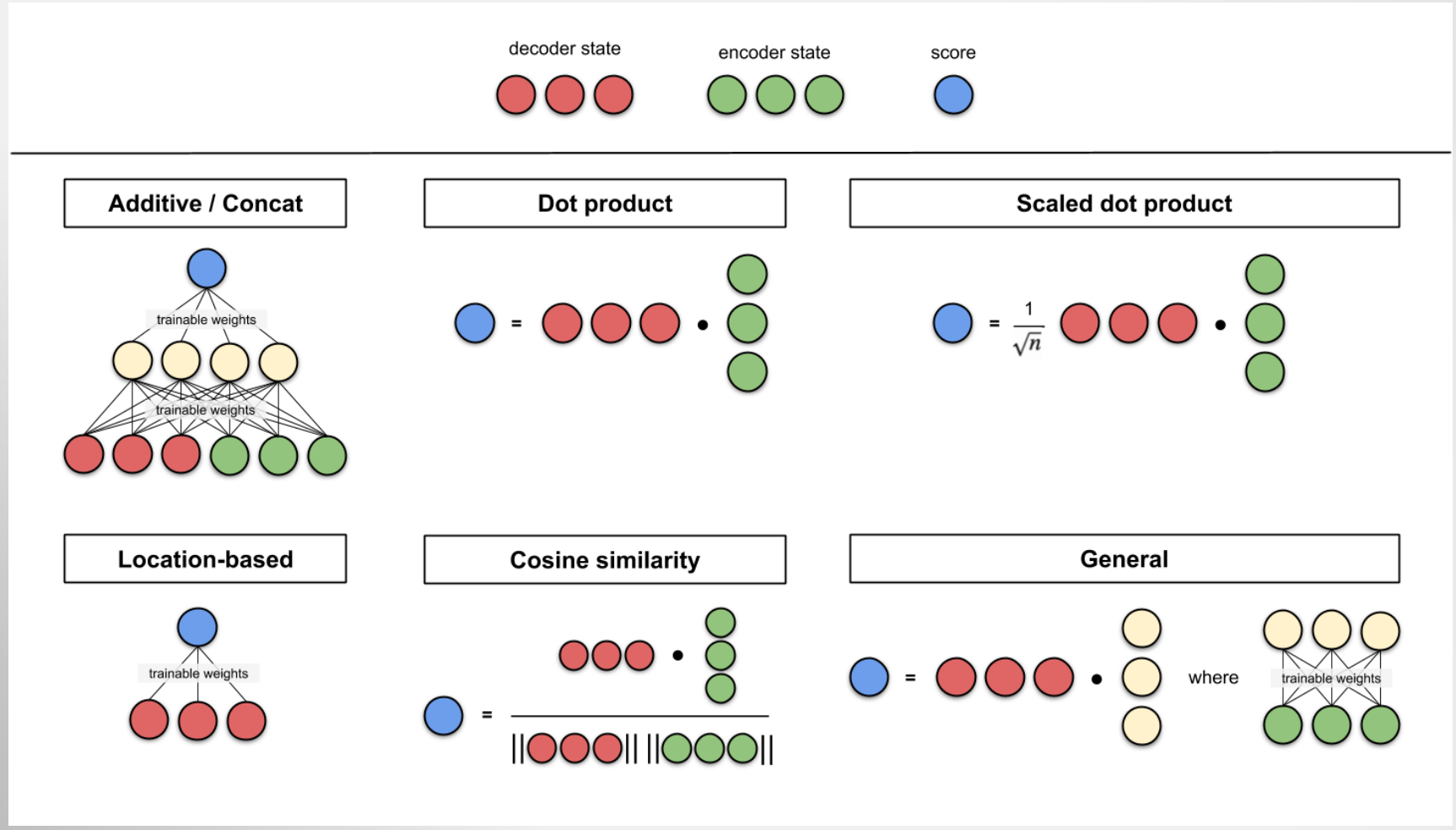
Attention Variants

Attention Variants

- There are several ways you can compute attention scores $e \in \mathbb{R}^N$ from the values $v_1, \ldots, v_N \in \mathbb{R}^{d_1}$ and the query $q \in \mathbb{R}^{d_2}$
- Basic dot-product attention: $e_i = q^T v_i \in \mathbb{R}$
  - Note that this assumes $d_1 = d_2$.
  - This is the version we saw earlier.
- Multiplicative attention: $e_i = q^T W v_i \in \mathbb{R}$
  - Where $W \in \mathbb{R}^{d_2 x d_1}$ is a weight matrix
- Additive attention: $e_i = y^T \tanh(W_1 v_i + W_2 q) \in \mathbb{R}$
  - Where $W_1 \in \mathbb{R}^{d_3 x d_1}$, $W_2 \in \mathbb{R}^{d_3 x d_2}$ are weight matrices and $y \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter.

Course slides: https://www.youtube.com/watch?v=QEw0qEa0E50&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z&index=7

Attention Variants



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Attention Types

- Attention-based models are classified into two broad categories, *global* and *local*.
  - These classes differ in terms of whether the “attention” is placed on all source positions or on only a few source positions.
  - While these models differ in how the context vector c_t is derived, they share the same subsequent steps.
- Attention can be classified into two based on how hidden states are incorporated in the final state.
  - Soft attention is when we calculate the context vector as a weighted sum of the encoder hidden states.
  - Hard attention is when, instead of weighted average of all hidden states, we use attention scores to select a single hidden state.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Attention Types

- Global Attention: The one we have studied where all the hidden layers attend the attention layer.

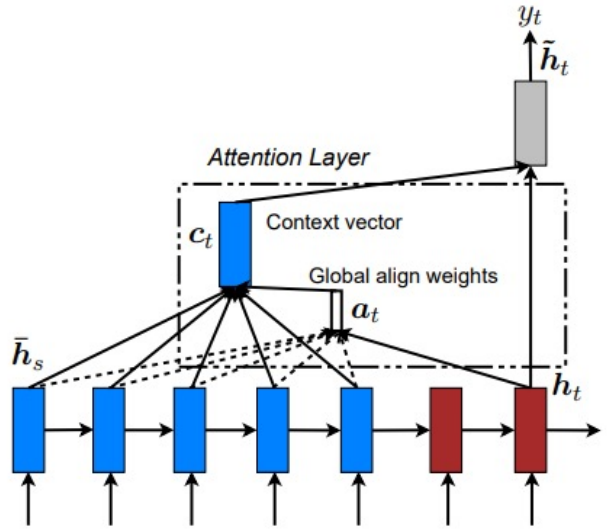


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector a_t based on the current target state h_t and all source states $\bar{h}_s$. A global context vector c_t is then computed as the weighted average, according to a_t, over all the source states.

The global attention has a drawback that it has to attend to all words on the source side for each target word, which is expensive and can potentially render it impractical to translate longer sequences, e.g., paragraphs or documents.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Attention Types

- Local Attention: a mechanism that chooses to focus only on a small subset of the source positions per target word.
- It selectively focuses on a small window of context and is differentiable.
- This approach has an advantage of avoiding the expensive computation incurred in the soft attention and at the same time, is easier to train than the hard attention approach.

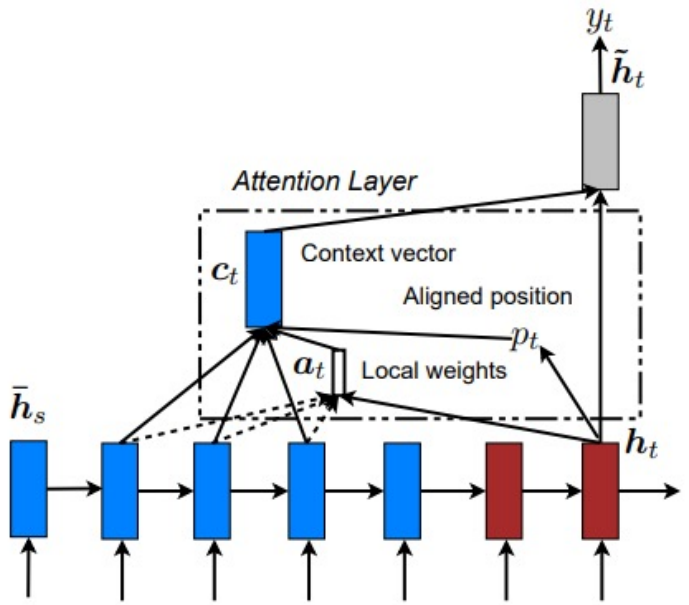


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Attention Types

- Local Attention: the model first generates an aligned position p_t for each target word at time t.
- The context vector c_t is then derived as a weighted average over the set of source hidden states within the window $[p_t - D, p_t + D]$; D is empirically selected.
- Unlike the global approach, the local alignment vector at is now fixed-dimensional, i.e., $\in R^{2D+1}$.

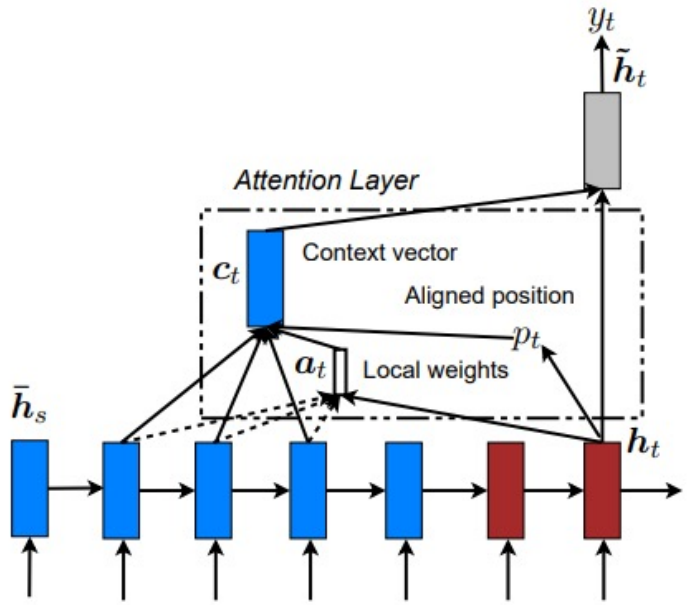


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Attention
Summary

- Attention significantly improves neural machine translation performance
  - It is very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem.
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem.
  - Provides shortcut to faraway states.
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on.
  - The network just learned alignment by itself

Course slides: https://www.youtube.com/watch?v=QEw0qEa0E50&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z&index=7

https://medium.com/analytics-vidhya/translation-seq2seq-attention-4f8b23f33965