Natural Language Processing

Attention

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

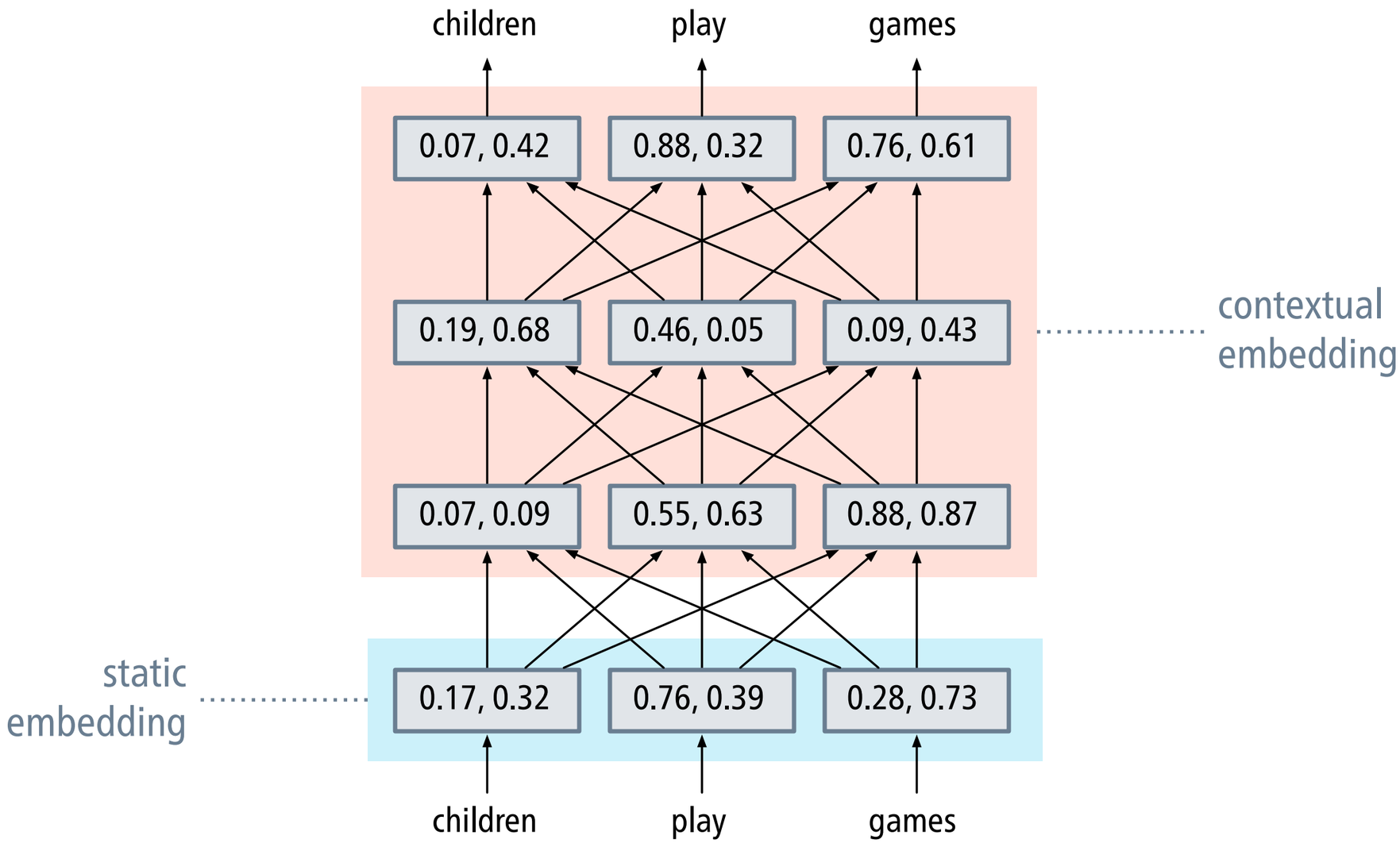
Contextual embeddings

- Standard word embeddings map each word to a fixed vector.
- In natural language, word meaning is not static, but depends on the surrounding words.

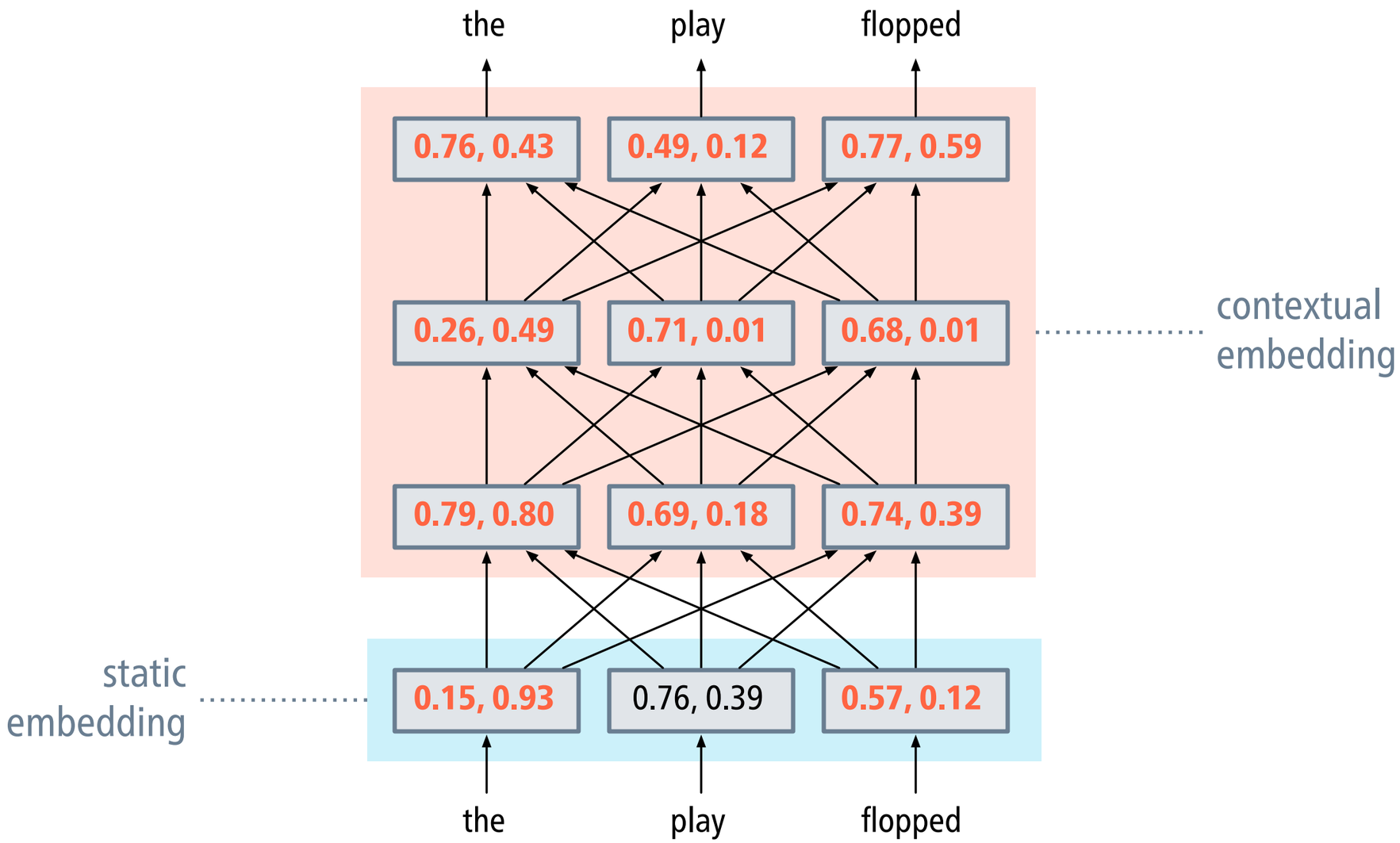
  Dogs may *bark* at strangers. Trees shed *bark* in winter.

- **Contextual embeddings** assign word vectors dynamically, conditioned on context.

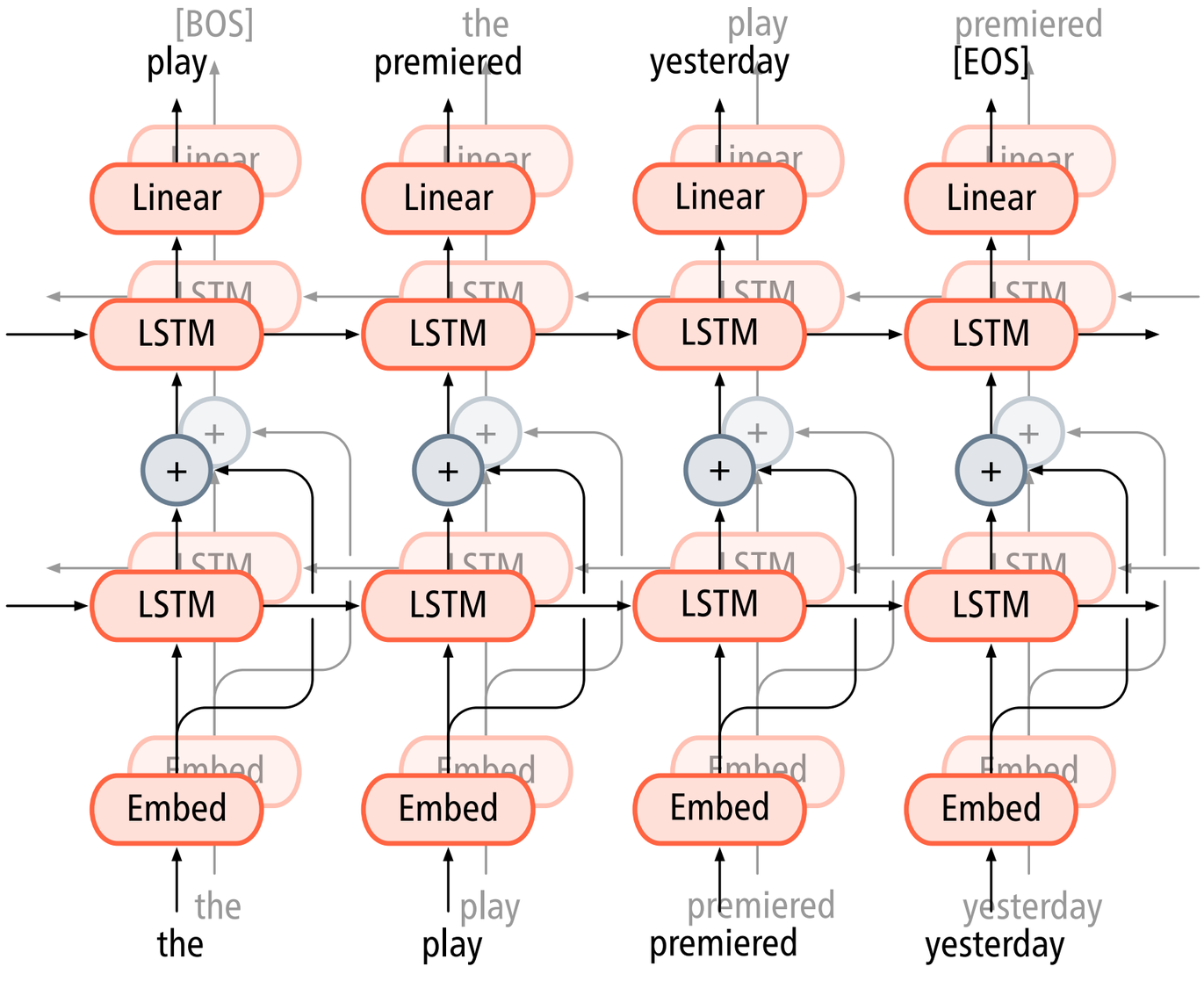
Contextual embeddings



Contextual embeddings



ELMo – Embeddings from Language Models



Peters et al. (2018)

Contextual embeddings via attention

- In attention, each word vector $\boldsymbol{h}_i$ is the weighted sum of all the word vectors $\boldsymbol{h}_j$ computed at the previous layer:

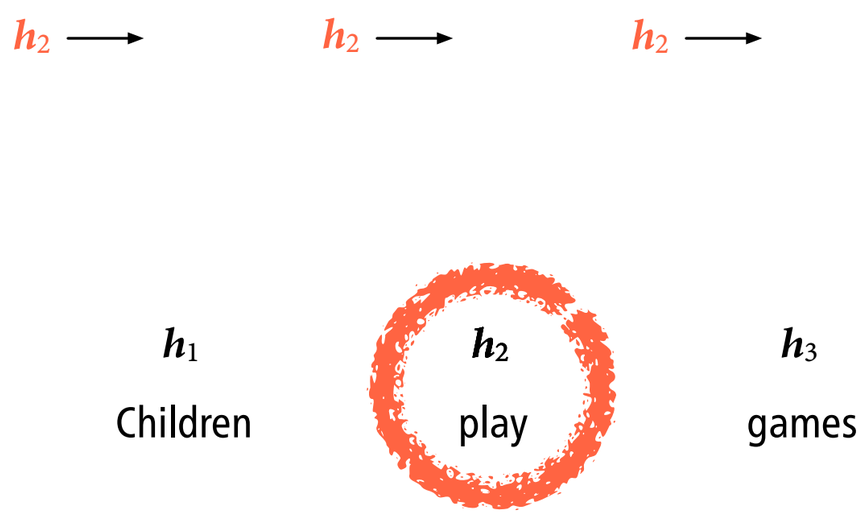
$$\boldsymbol{h}_i^{\ell+1} = \sum_j \alpha_{ij} \boldsymbol{h}_j^{\ell}$$

- The weights α_{ij} express how much the model should "attend to" the context vector $\boldsymbol{h}_j$ at layer ℓ when forming $\boldsymbol{h}_i$ at layer $\ell+1$.

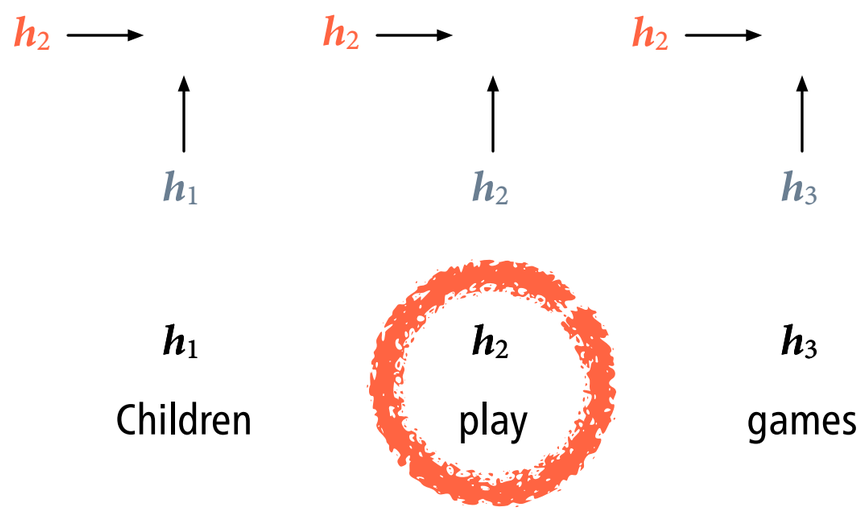
- The weights are proportional to the vector similarity between $\boldsymbol{h}_i$ and the context vectors, and are learned during training.

Contextual embeddings via attention

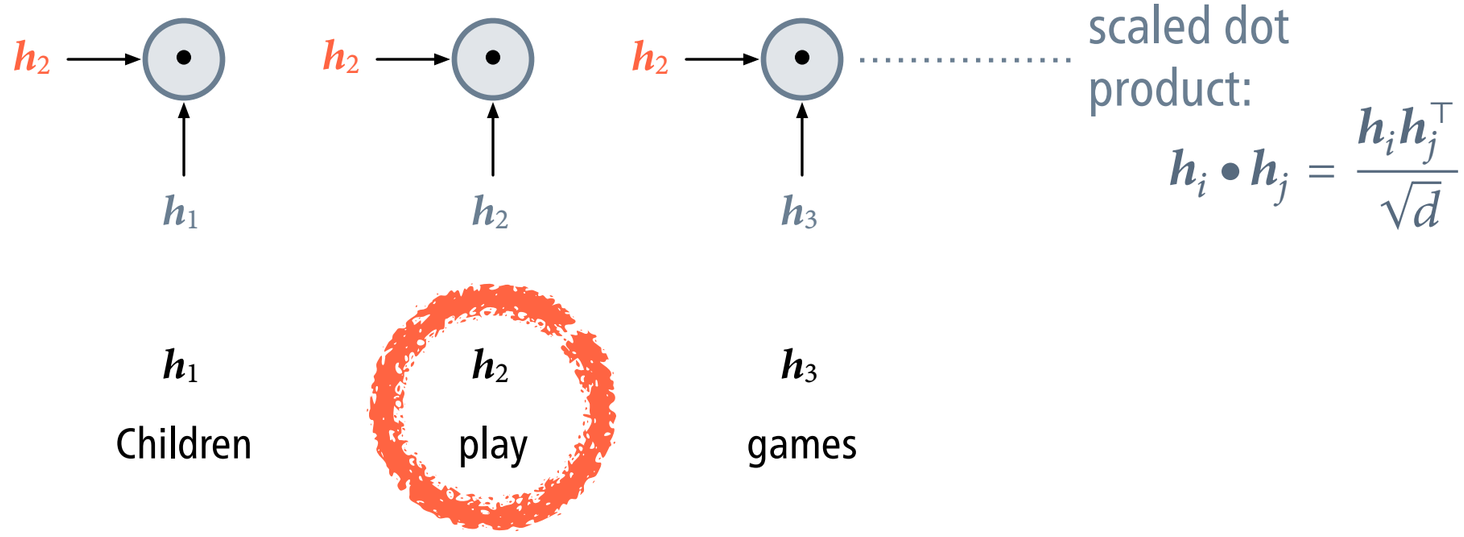
Contextual embeddings via attention



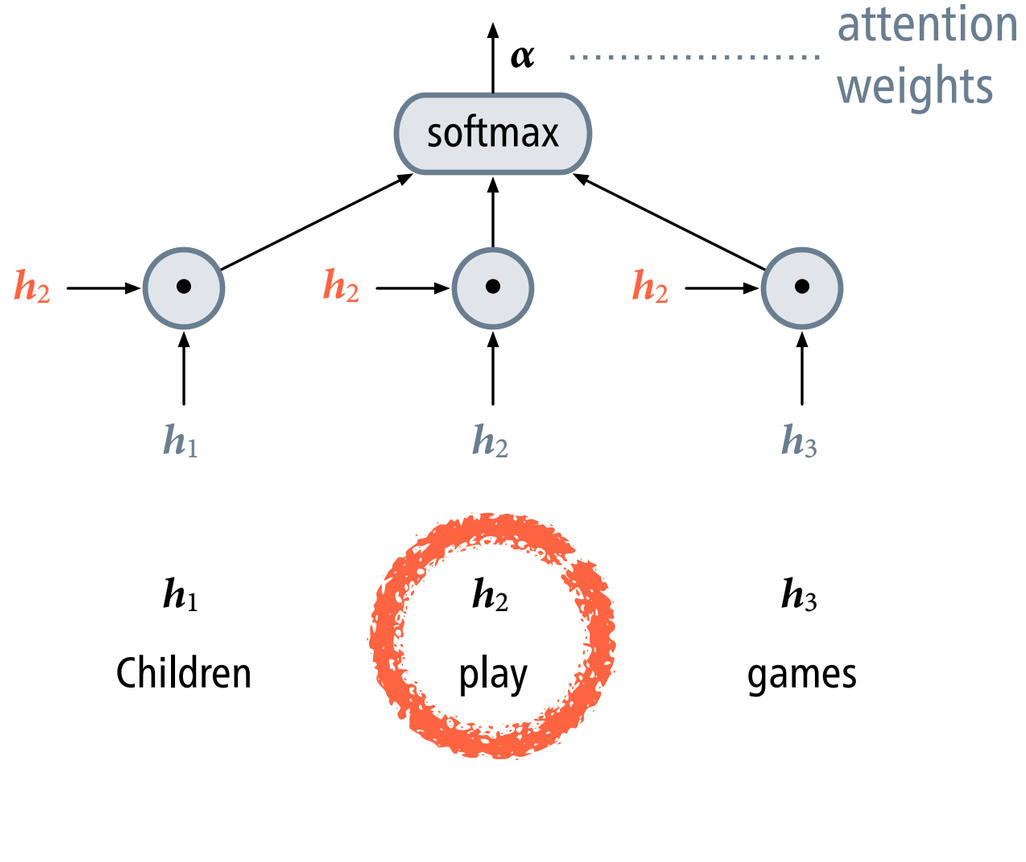
Contextual embeddings via attention



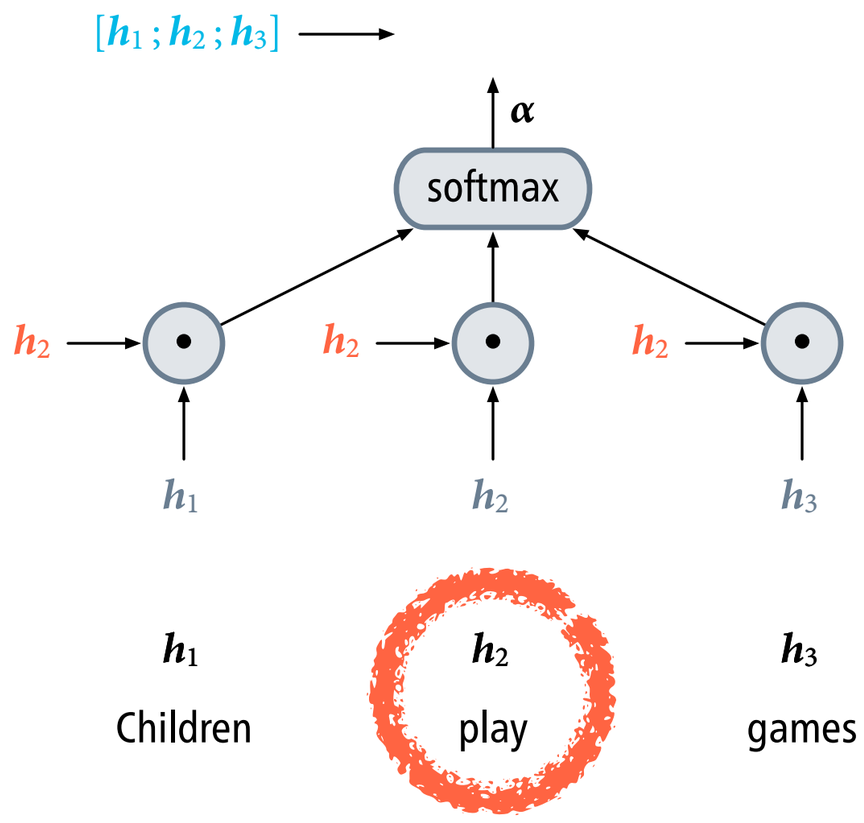
Contextual embeddings via attention



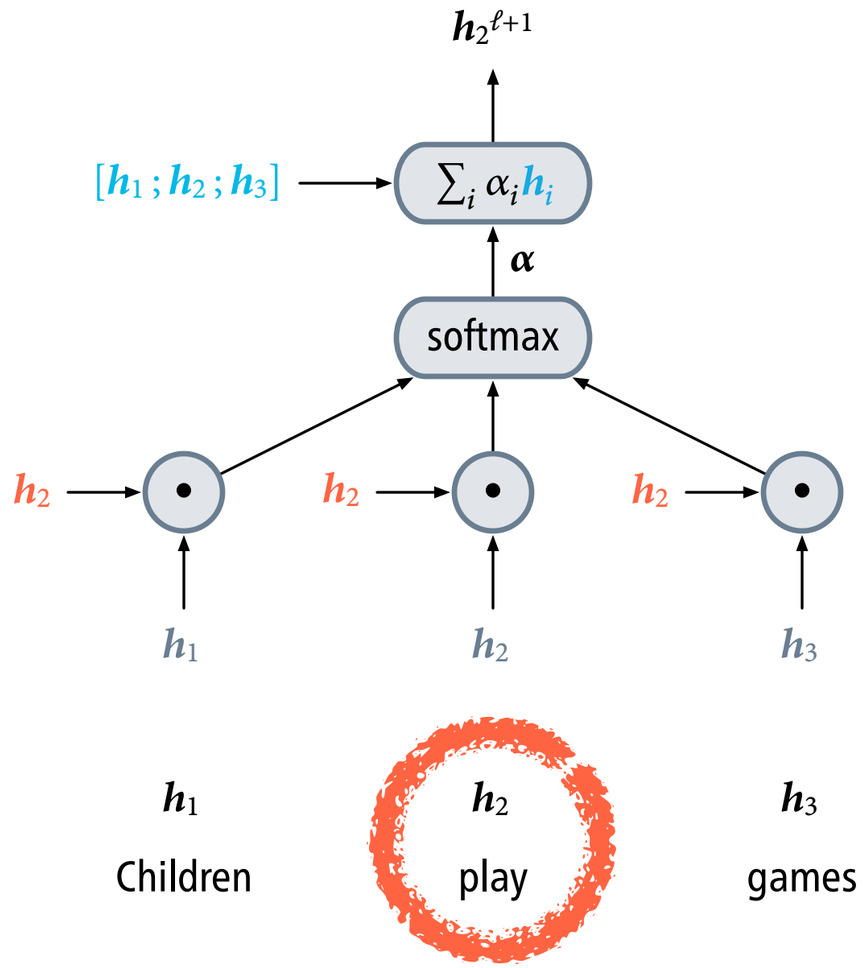
Contextual embeddings via attention



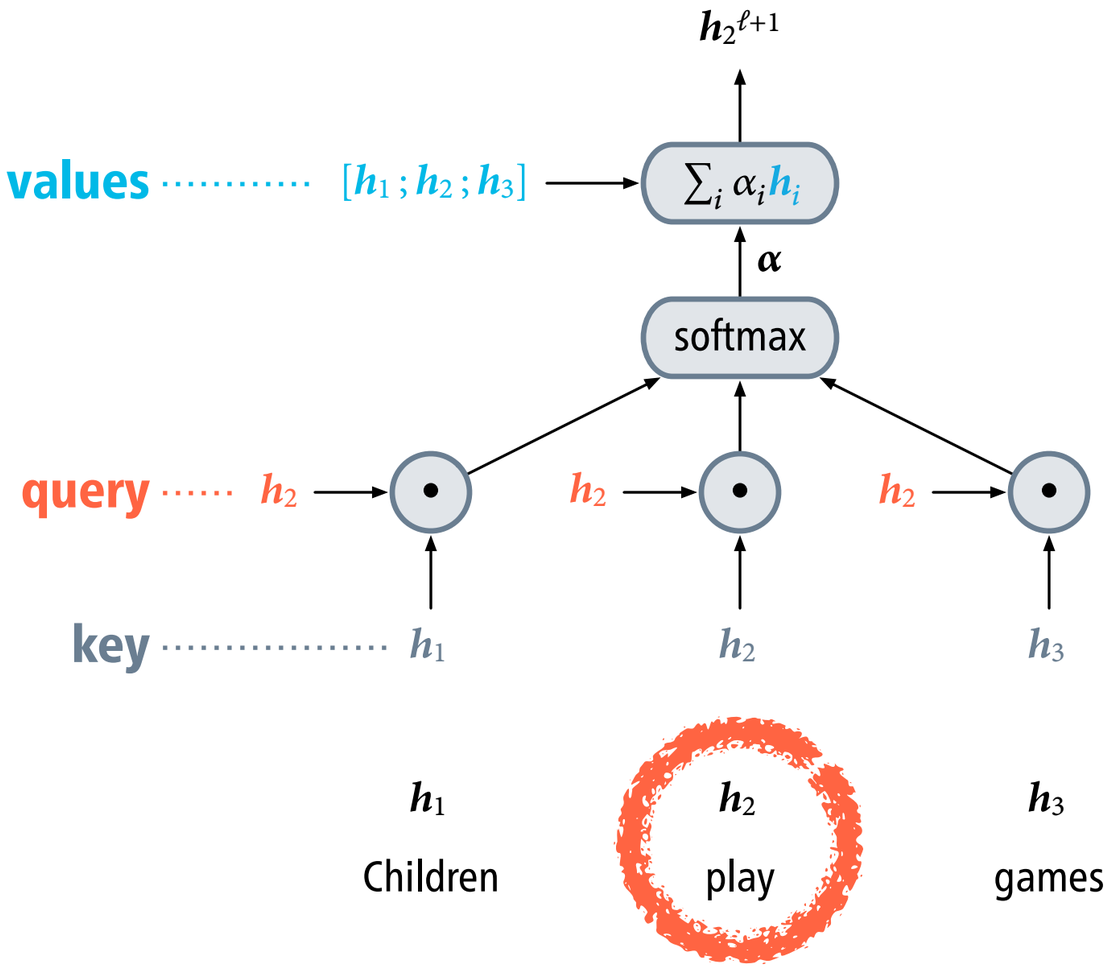
Contextual embeddings via attention



Contextual embeddings via attention



Queries, keys, and values



A general characterisation of attention

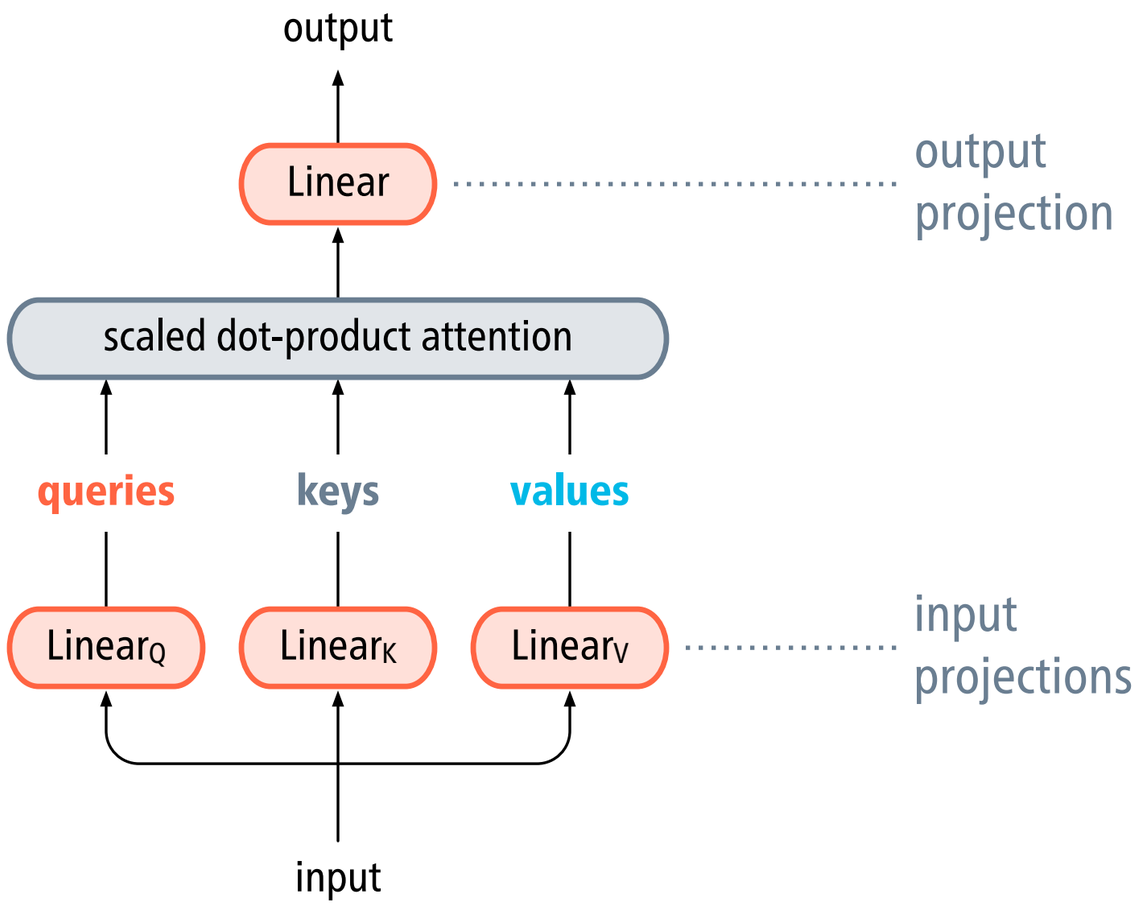
- In general, attention can be described as a mapping from a query $\boldsymbol{q}$ and a set of key–value pairs $\langle \boldsymbol{k}_i, \boldsymbol{v}_i \rangle$ to an output.
- The output is the weighted sum of the $\boldsymbol{v}_i$, where the weight of each $\boldsymbol{v}_i$ is given by the attention score between $\boldsymbol{q}$ and $\boldsymbol{k}_i$:

$$\boldsymbol{\alpha}_i = \text{softmax}(\text{score}(\boldsymbol{q}, \boldsymbol{K}))\boldsymbol{V}$$

$$\boldsymbol{q} \in \mathbb{R}^{d_Q},\ \boldsymbol{K} \in \mathbb{R}^{n \times d_K},\ \boldsymbol{V} \in \mathbb{R}^{n \times d_V},\ d_Q = d_K$$

Vaswani et al. (2017)

Attention in the Transformer



Scaled dot-product attention in PyTorch

```
# Input: queries q, keys k, values v
# shape of q, k, v: [num_words, d]

# Compute the attention scores (scaled dot product)
scores = q @ k.transpose(-1, -2) / hidden_dim**0.5
# shape of scores: [num_words, num_words]

# Normalise the attention scores
alphas = F.softmax(scores, dim=-1)
# shape of alphas: [num_words, num_words]

# The output is the alpha-weighted sum of the values
result = alphas @ v
# shape of result: [num_words, d]
```

Multi-head attention