Natural Language Processing

Learning word embeddings with neural networks

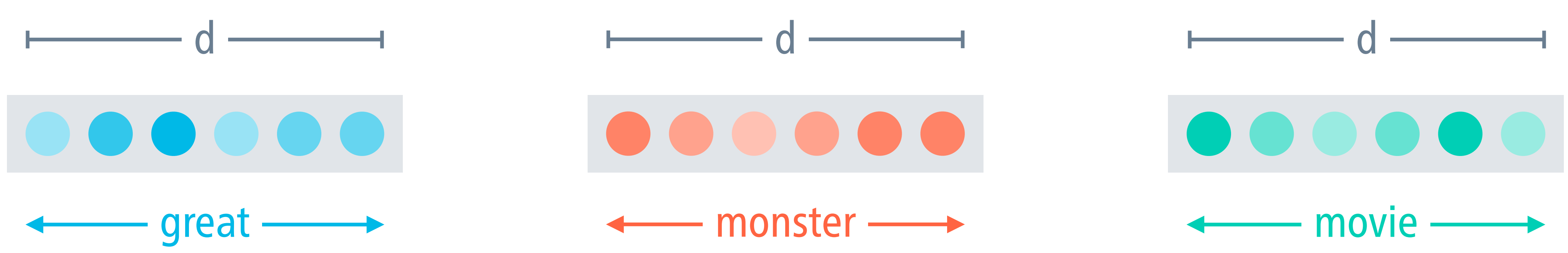
Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

This work is licensed under a Creative Commons Attribution 4.0 International License.

Embedding layers

- In neural networks, word embeddings are realised by **embedding layers**.
- An embedding layer implements a mapping from a vocabulary of words to some *d*-dimensional vector space.



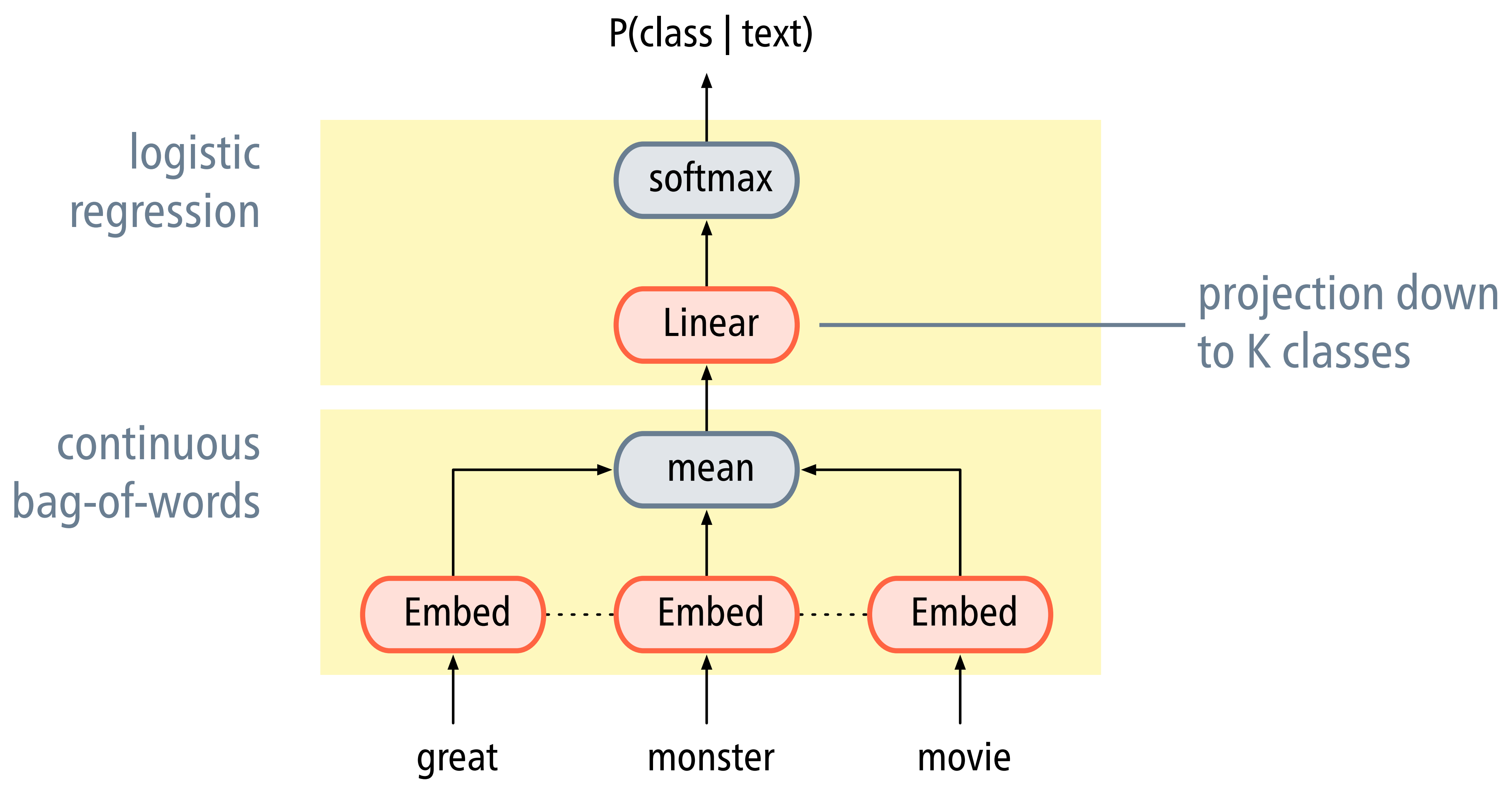
Embedding layers in PyTorch

```
vocab = {'great': 0, 'monster': 1, 'movie': 2}

import torch

e = torch.nn.Embedding(3, 2)

e(torch.tensor(vocab['monster']))
# tensor([0.6399, 0.1779], grad_fn=<EmbeddingBackward>)

e(torch.tensor([0, 1, 2]))
tensor([[ 0.4503, -0.1549],
        [ 0.6399,  0.1779],
        [-0.6537, -0.5875]], grad_fn=<EmbeddingBackward>)
```

number of words to embed

size of each embedding vector

The continuous bag-of-words (CBOW) classifier



Implementation of the CBOW classifier

```
class CBOWClassifier(nn.Module):

    def __init__(self, num_words, embedding_dim, num_classes):
        super().__init__()
        self.embedding = nn.Embedding(num_words, embedding_dim)
        self.linear = nn.Linear(embedding_dim, num_classes)

    def forward(self, x):
        # x is a tensor containing word ids
        return self.linear(torch.mean(self.embedding(x), -2))
```

Task-specific word embeddings

- When we train a neural network, the word embeddings are optimised for the training task.
- **Representation learning:** Words can "mean" different things in different tasks. The network learns the optimal representation.
- There is no guarantee that the embeddings obtained from neural networks model co-occurrence distributions.

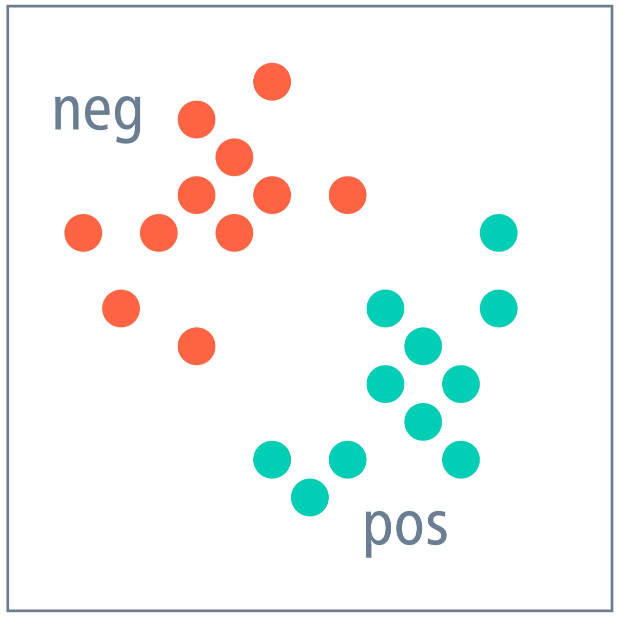
Two different perspectives on word embeddings

- **Count-based approach**

  similar embeddings $\Rightarrow$ the corresponding words have similar distributions

- **Prediction-based approach**

  similar embeddings $\Rightarrow$ the corresponding words behave similarly in learning tasks



Word embeddings for transfer learning

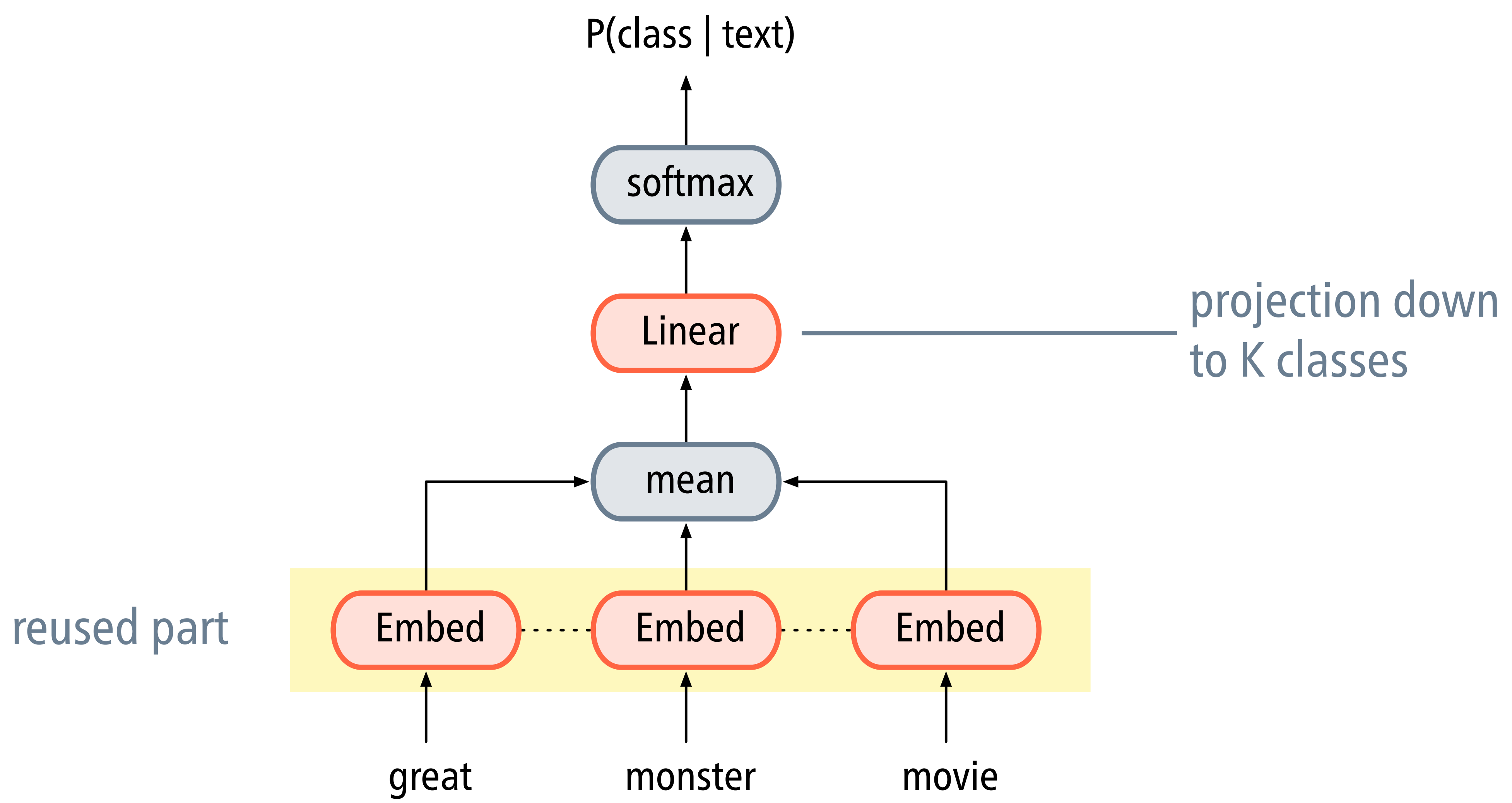
- **Transfer learning** aims to re-use knowledge gained while solving some previous task when solving the next task.

  speed up training, reduce the need for training data

- In the context of deep learning, transfer learning is typically implemented by re-using some part of a trained model.

- In particular, we could try re-using the embedding layers, instead of learning embeddings from scratch for each task.

The continuous bag-of-words (CBOW) classifier



Re-using pre-trained word embeddings

Pre-train embeddings on task *A* and use them to initialise the embedding layers of the network for task *B*. Then:

- **Alternative 1:** Train as usual, effectively fine-tuning the pre-trained embeddings to the task at hand.
- **Alternative 2:** Freeze the weights of the embedding layers, to prevent the pre-trained embeddings from being modified.

What pre-training tasks should we use?

- We want to learn representations that are generally useful, so we prefer pre-training tasks that are general.
- We need to find training data for the pre-training tasks, so we prefer tasks for which data is abundant.

  ideal candidate: raw text
- The standard pre-training task for word embeddings is language modelling, e.g., to predict co-occurrence patterns.

  Remember the Distributional Hypothesis!