Sequence Encoding Techniques

Sequence Encoding

- Example: the cat sat on floor and is looking at wall

- How do you feed documents to machines to perform NLP tasks?

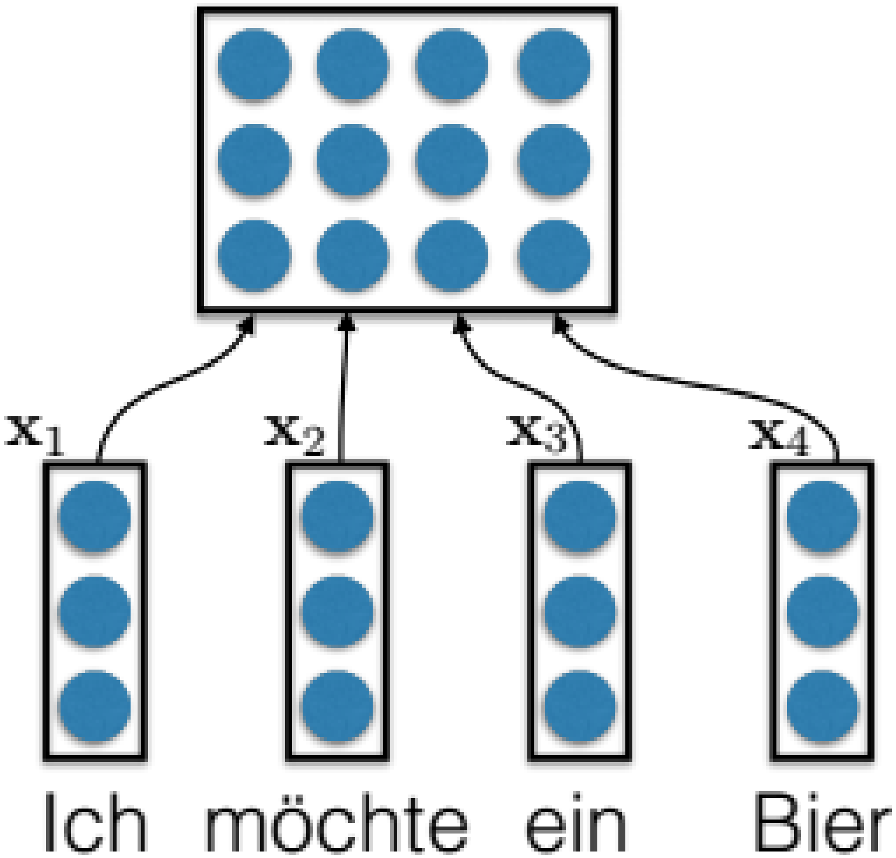
- NN based Solutions
  - Bag of word vectors(sum/average/concat)
  - RNN
  - CNN
  - RNN with big memory

I. Sequence Encoding: concat/averaging word vectors

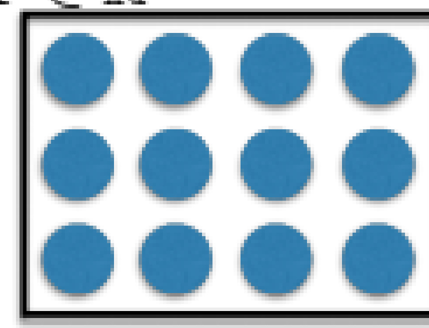
- Each word type is represented by an n-dimensional vector

- Take all of the vectors for the sentence and concatenate/average them

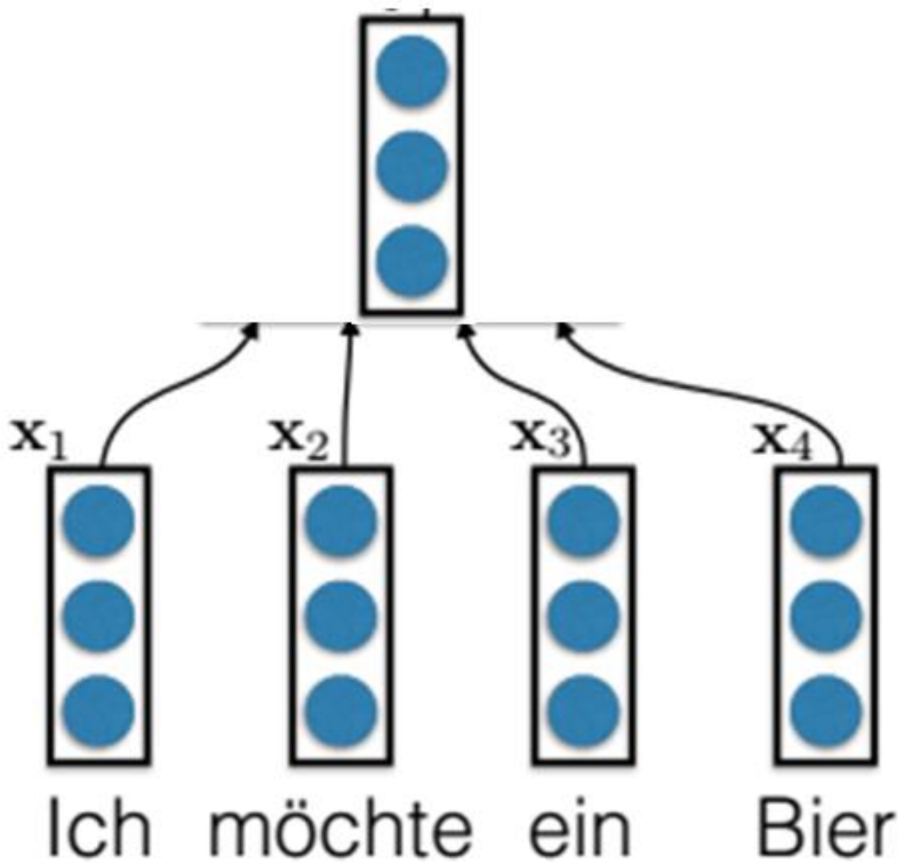
$\mathbf{f}_i = \mathbf{x}_i$



$\mathbf{F} \in \mathbb{R}^{n \times |\boldsymbol{f}|}$



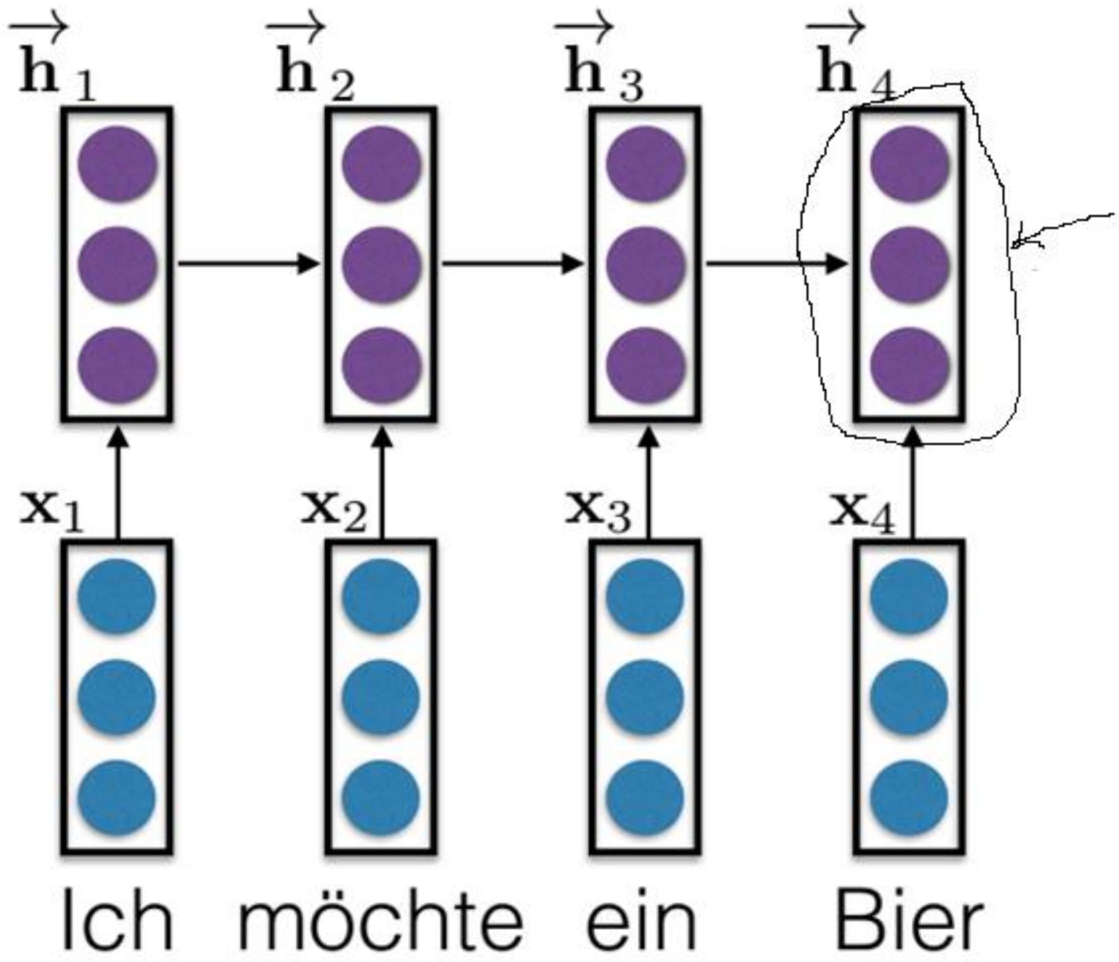
Ich möchte ein Bier

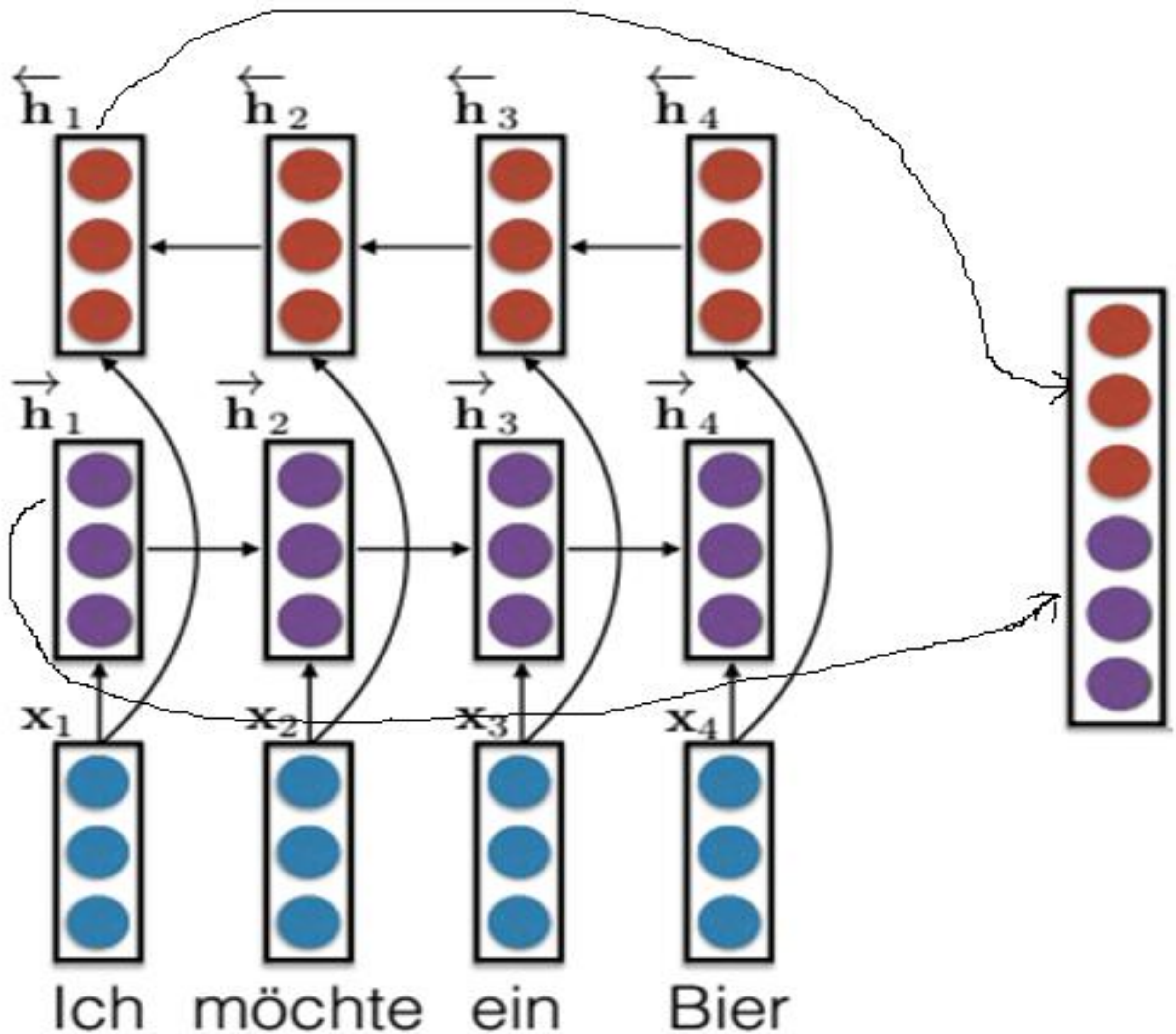


$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

Pros & Cons

III. Sequence Encoding: RNN vector





$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

Pros & Cons

- **Good**
  - RNNs deal naturally with sequences of various lengths
  - LSTMs in principle can propagate gradients a long distance
  - Very simple architecture!
- **Bad**
  - The hidden state has to remember a lot of information!

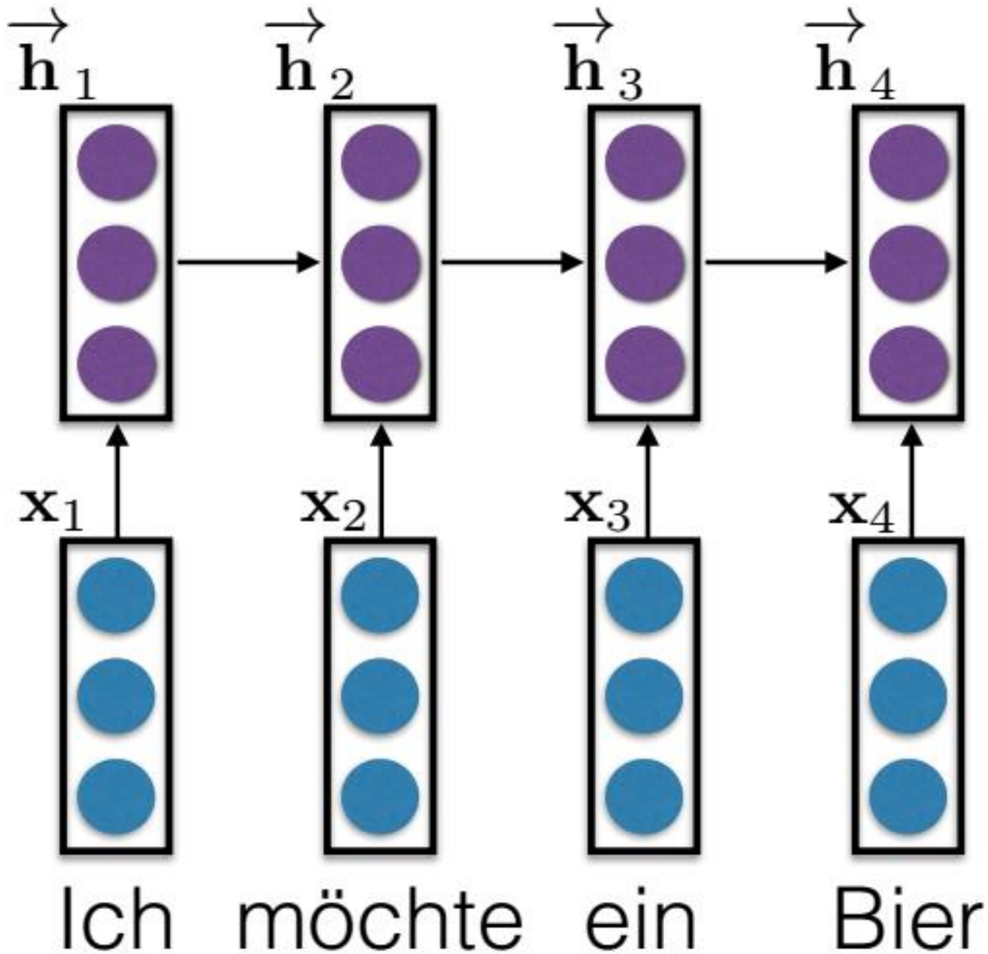
Issue with single global context vector

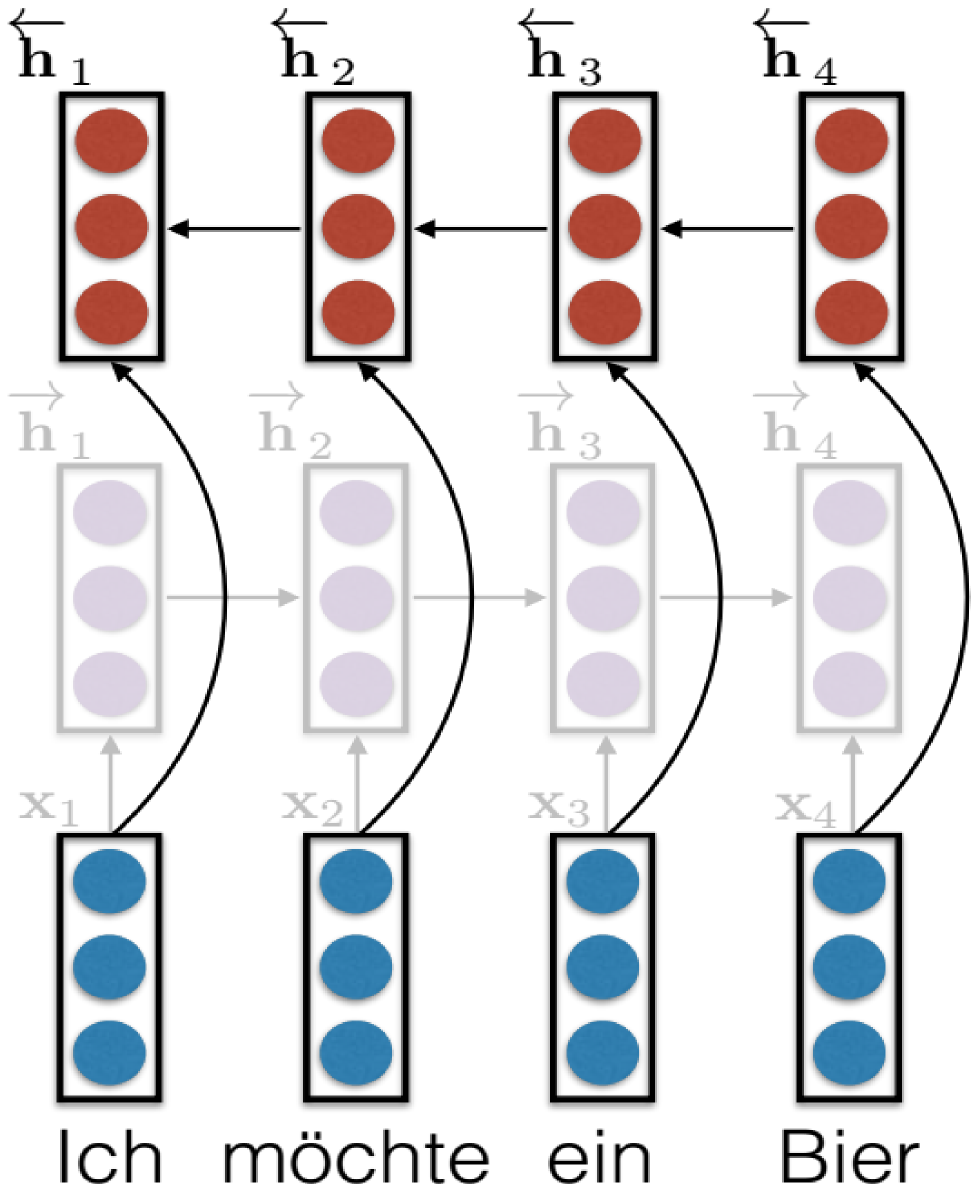
- A single global context vector memory for entire input sequence seems very limited for long sequences.

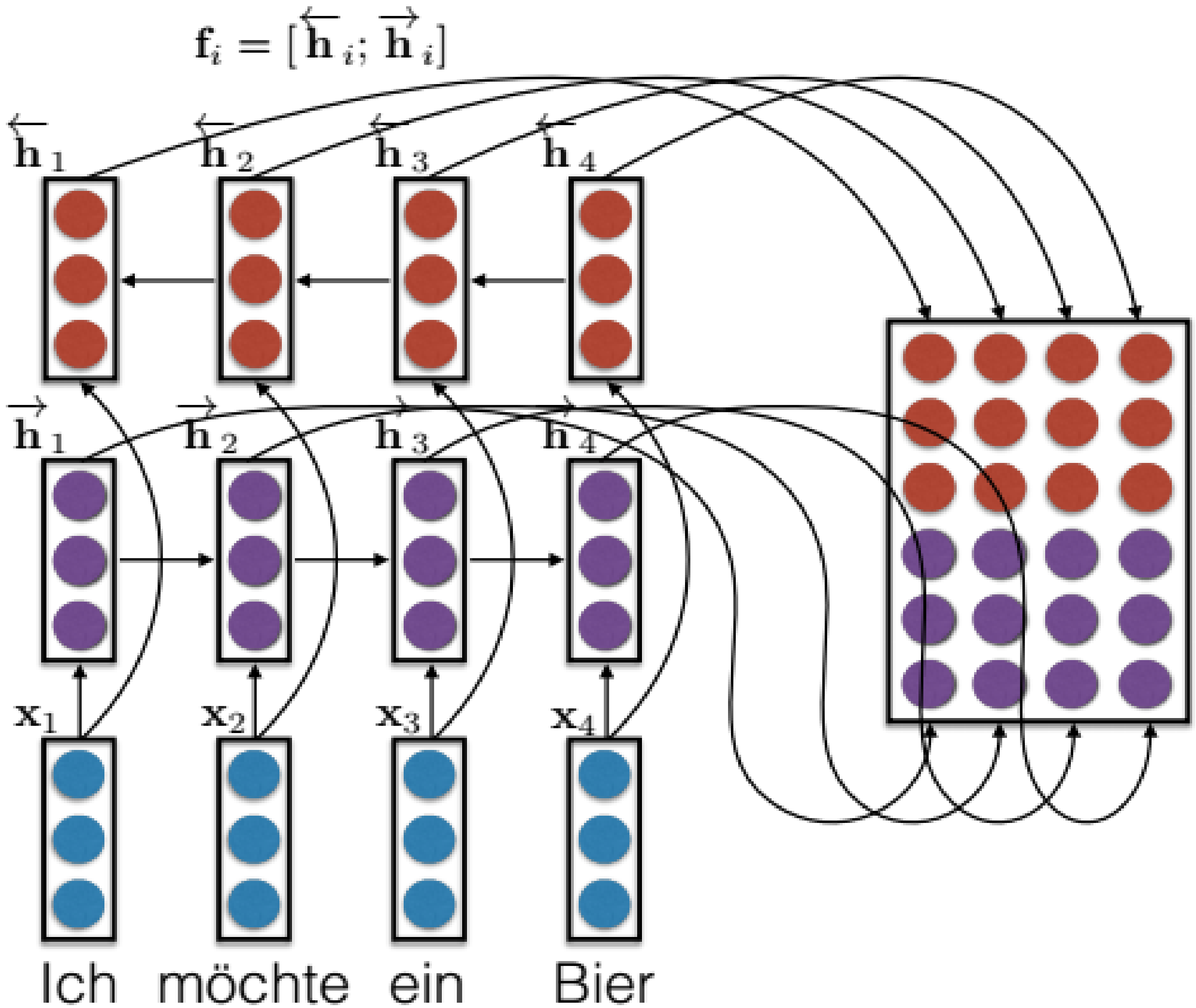
- Use all the intermediate encodings as memory to solve this problem.

III. Sequence Encoding: RNN with big memory

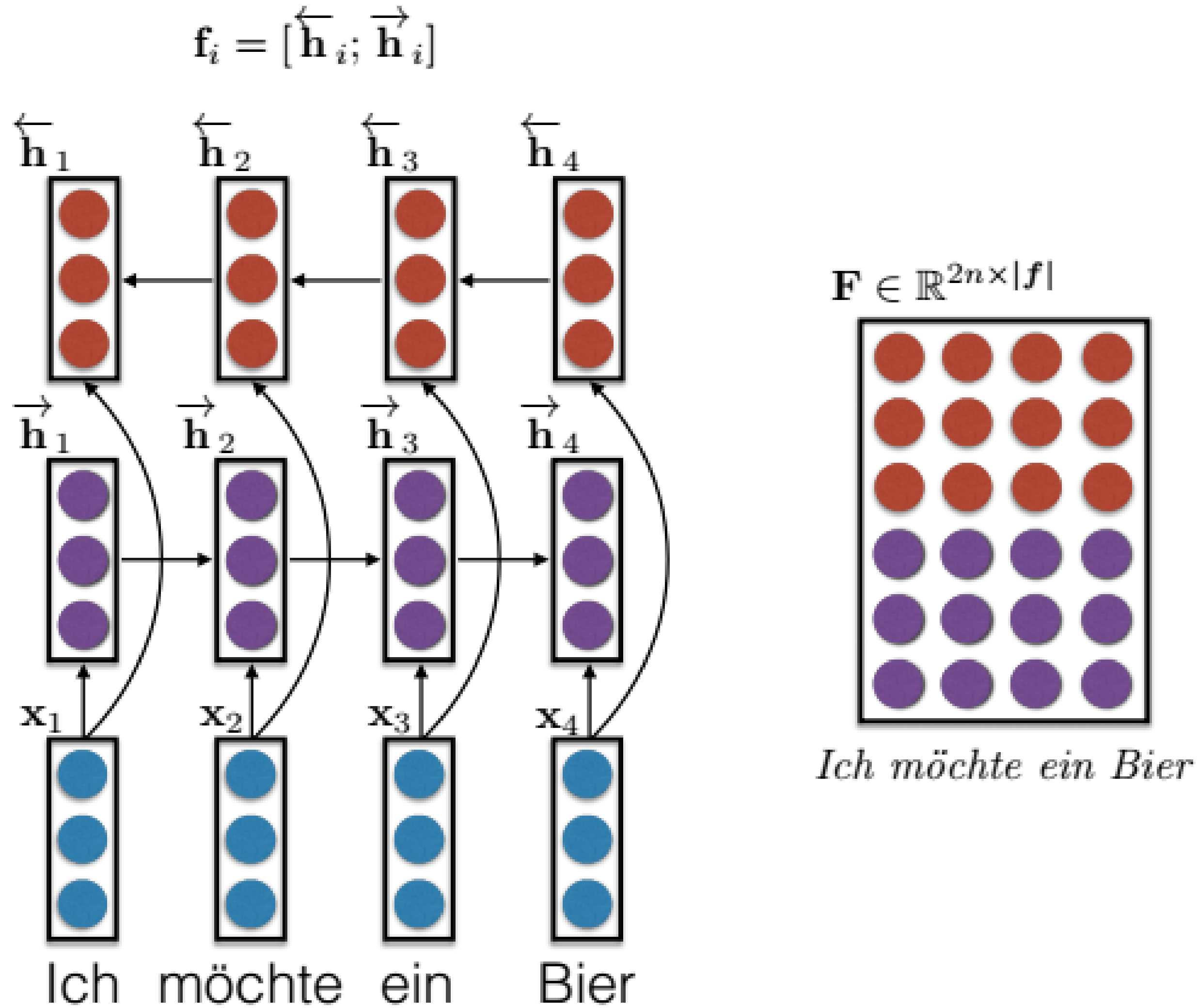
- Encode sentence as a matrix with one column per word
- Each column (word) has two halves concatenated together:
  - a “forward representation”, i.e., a word and its left context
  - a “reverse representation”, i.e., a word and its right context
- Use bidirectional RNNs (GRUs or LSTMs) to read from left to right and right to left, concatenate representations


$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier

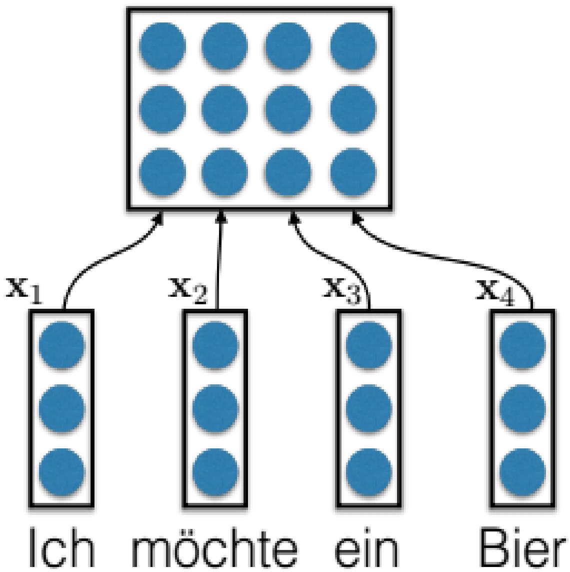


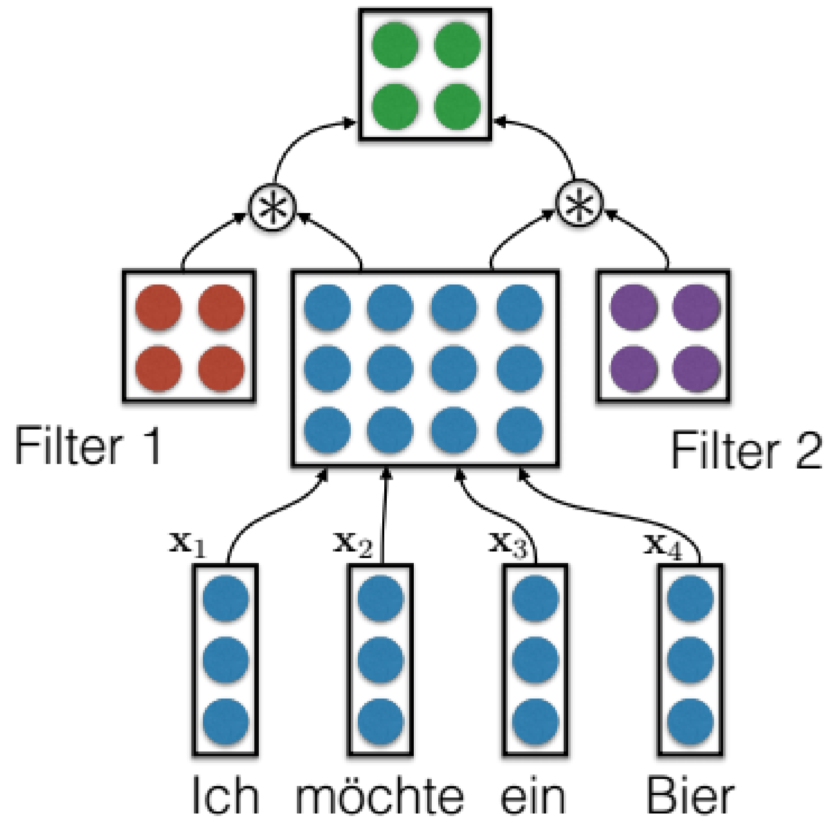
$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$
$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$
$\overleftarrow{\mathbf{h}}_1$ $\overleftarrow{\mathbf{h}}_2$ $\overleftarrow{\mathbf{h}}_3$ $\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$ $\overrightarrow{\mathbf{h}}_2$ $\overrightarrow{\mathbf{h}}_3$ $\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$
Ich möchte ein Bier
$\mathbf{F} \in \mathbb{R}^{2n \times |f|}$
Ich möchte ein Bier

IV. Sequence Encoding: CNN

- Apply convolutional networks to transform the naïve concatenated matrix to obtain a context-dependent matrix


$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier


Filter 1
Filter 2
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier