More on LSTMs

In this lecture

Part 1

- Input Representation
- Padding

Input Representation

Input

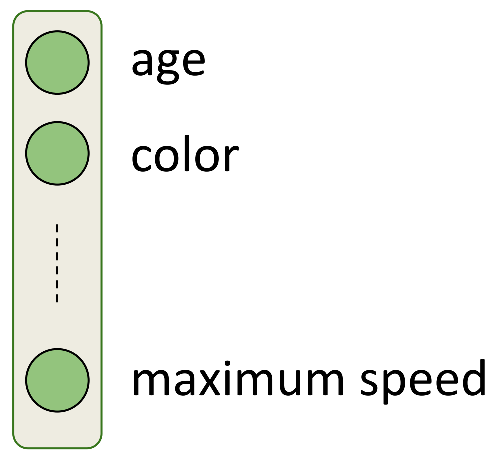
Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

Previous word(s)

Input Representation

Input

Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

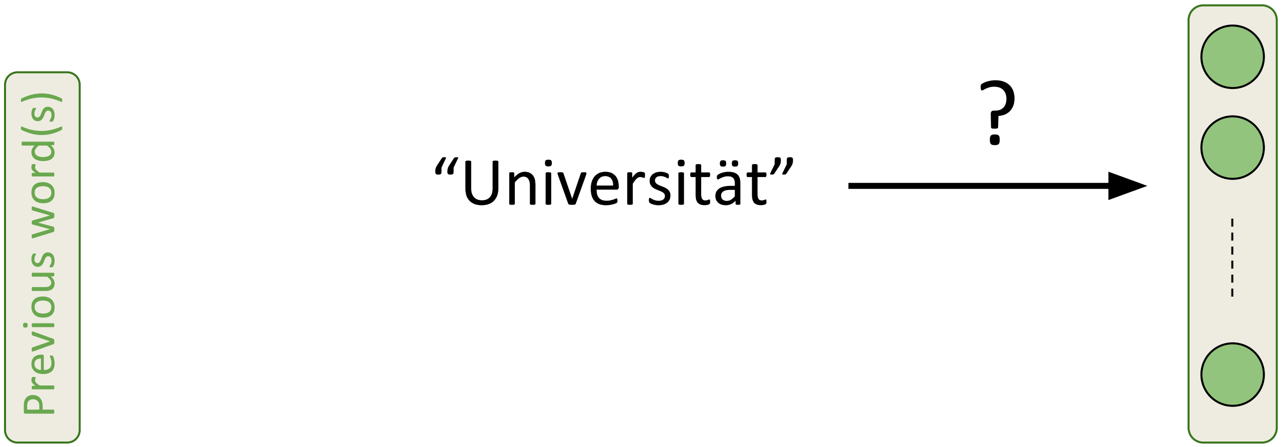


Car

Input Representation

Input

Can we represent a word as a feature vector?



One Hot Vector Representation

- Every word can be represented as a ***one hot vector***

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Universität: 1
cat: 2
house: 3
car: 4
    ⋮
apple: 10,000
```

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Universität: 1
cat: 2
house: 3
car: 4
    ⋮
apple: 10,000
```

Dictionary

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

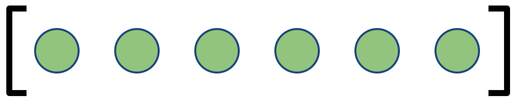
- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Vector size will be the size of the vocabulary, i.e. 10,000 in this case

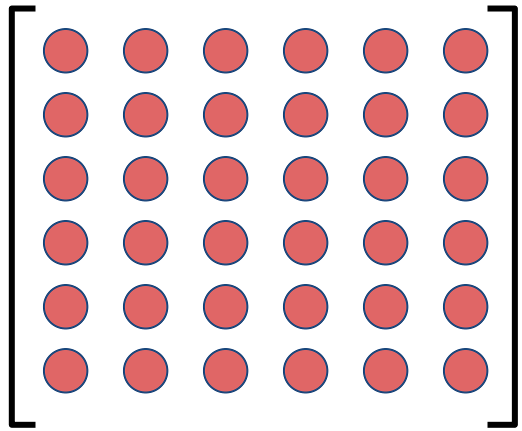
$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

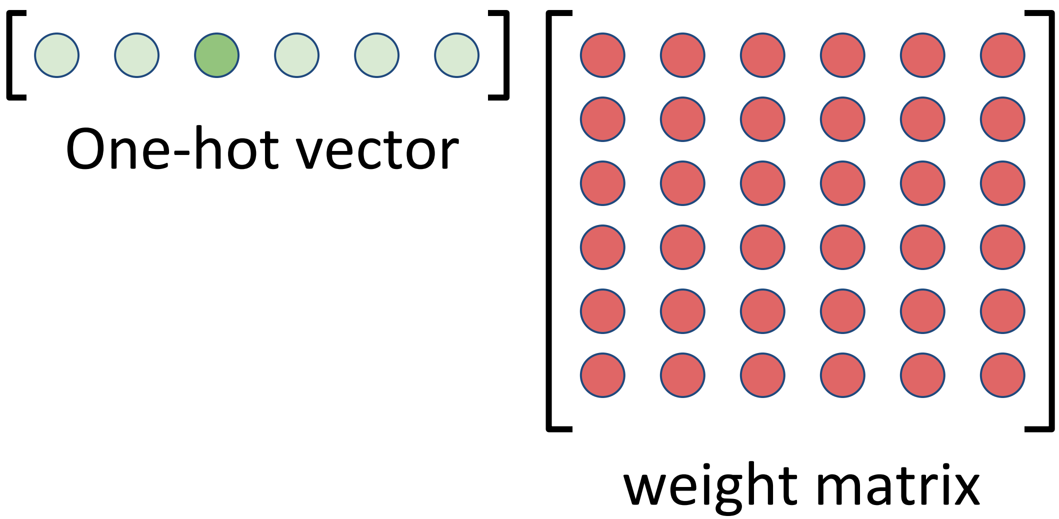


One-hot vector

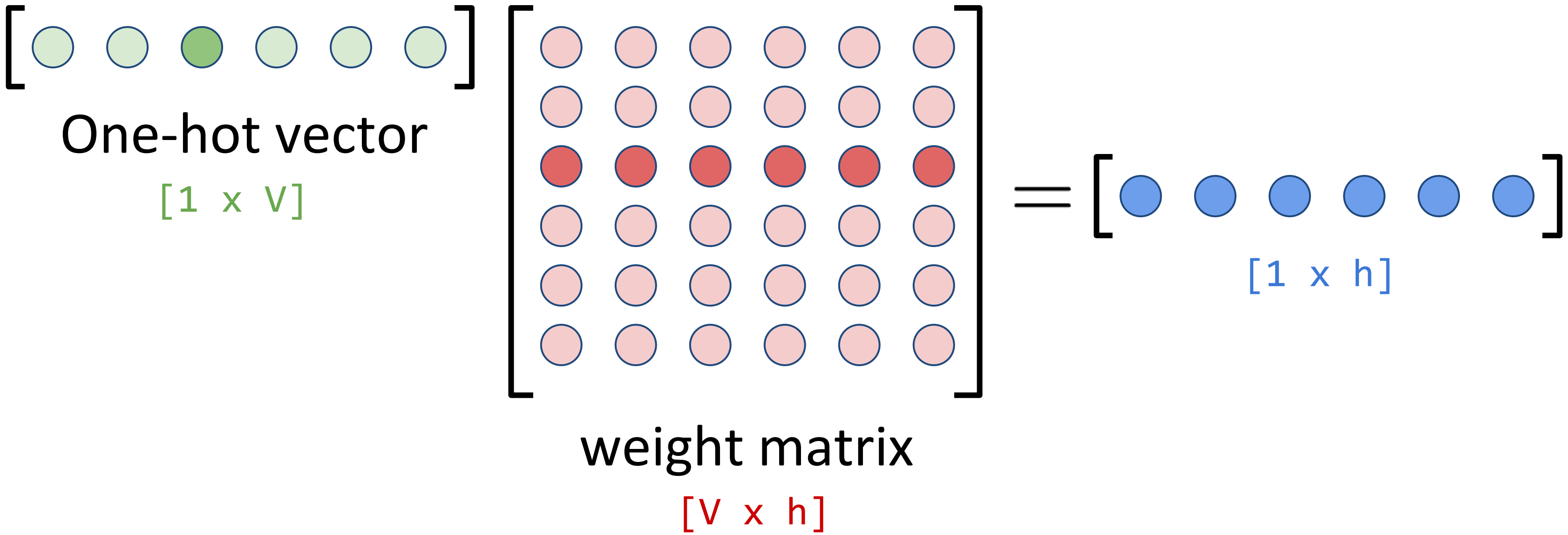


weight matrix

One Hot Vector Representation



One Hot Vector Representation



One-hot vector will “turn on” one row of weights

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boxed{1} \\ 0 \\ 0 \\ \vdots \\ \boxed{1} \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ \boxed{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \boxed{1} \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Trigram: indices of the *three previous words* are 1 in the vector

$$\text{a fast car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boxed{1} \\ 0 \\ \boxed{1} \\ \vdots \\ \boxed{1} \end{bmatrix} \quad \text{a fast cat} = \begin{bmatrix} 0 \\ \boxed{1} \\ 0 \\ 0 \\ 0 \\ \boxed{1} \\ \vdots \\ \boxed{1} \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Solution: for *N* words, concatenate one-hot vectors for each of the words in the correct order

Higher ngram Vector Representation

Context-aware approach

$$\mathrm{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \mathrm{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow \mathrm{a\ car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Higher ngram Vector Representation

Context-aware approach

$$
\mathrm{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}
\qquad
\mathrm{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\qquad \longrightarrow \qquad
\mathrm{a\ car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}
$$

[V x 1]　　[V x 1]　　[2V x 1]

Higher ngram Vector Representation

Context-aware approach

input vector length has increased

$$a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

[V x 1] [V x 1]

$$\longrightarrow \quad a\ \text{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

[2V x 1]

Higher ngram Vector Representation

Context-aware approach

input vector length has increased

- order information is available for the training

Advantages

- long vectors in case of large context size
- number of parameters increases with context size

Disadvantages

Higher ngram Vector Representation

- Bag of words vs. context-aware approach?
  - Given the disadvantages of the context-aware approach, Bag of words is more commonly used
  - Works well in practice

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Some tricks to reduce vocabulary size:

1) Take most frequent top words. For example, consider only 10,000 most frequent words and map the rest to a unique token `<UNK>`
2) Cluster words
   a) based on context
   b) based on linguistic properties

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector

only one element is 1 in the entire vector

vectors of different words do not give us any information about the potential relations between the words!

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

word vectors are ***continuous*** representations of words

vectors of different words give us information about the potential relations between the words - words closer together in meaning have vectors closer to each other

Word Embeddings

$$\text{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow \text{cat} = \begin{bmatrix} 0.523 \\ -0.813 \\ 0.417 \\ 1.738 \\ -0.145 \\ -3.451 \\ \vdots \\ 0.451 \end{bmatrix}$$

one-hot vector

word embedding

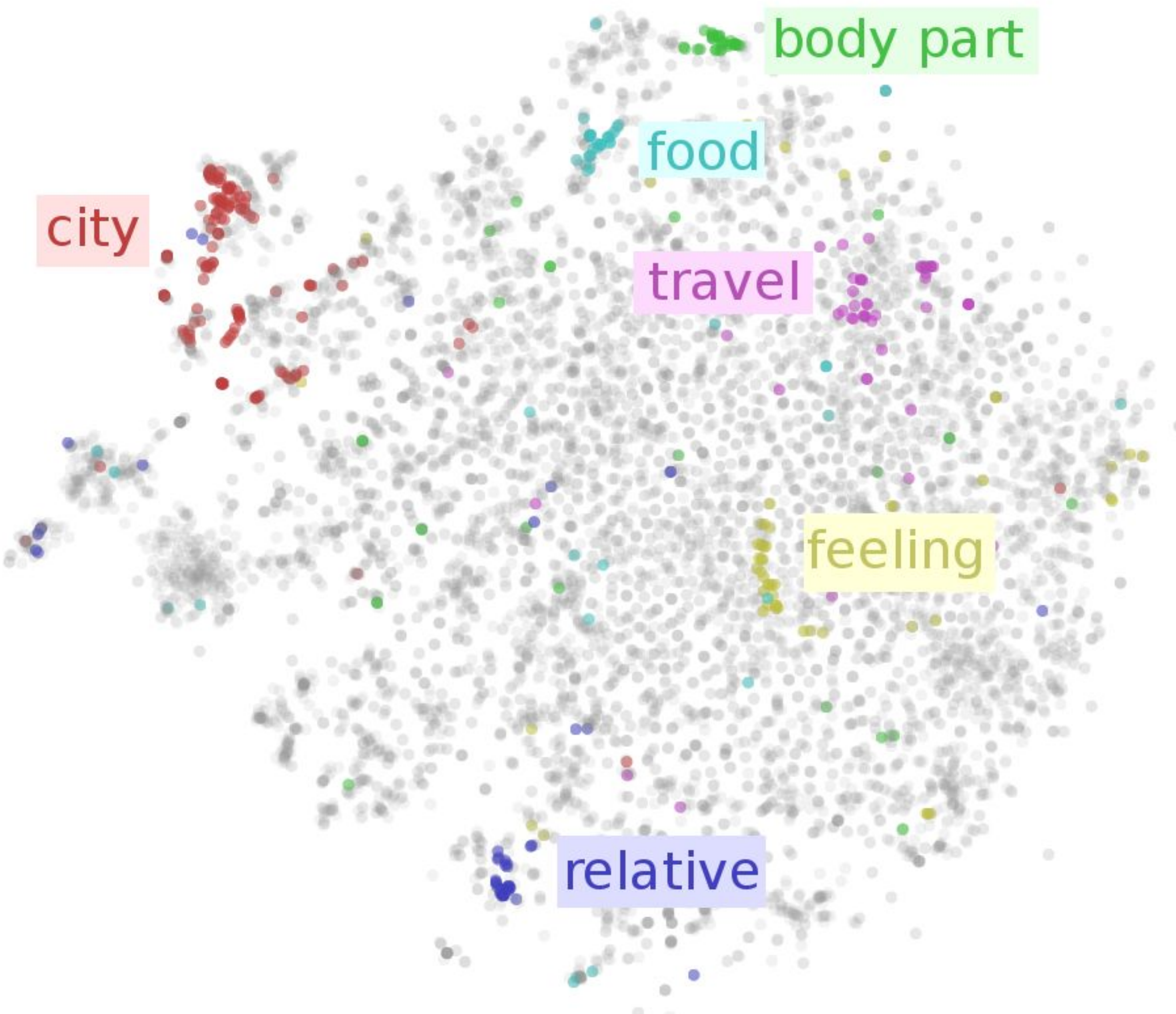
Word Embeddings

"Representation of words in continuous space"

Inherit benefits

- Reduce dimensionality
- Semantic relatedness
- Increase expressiveness
  - one word is represented in the form of several features (numbers)

Word Embeddings



Word Embeddings

Play with some embeddings!

https://rare-technologies.com/word2vec-tutorial/#bonus_app

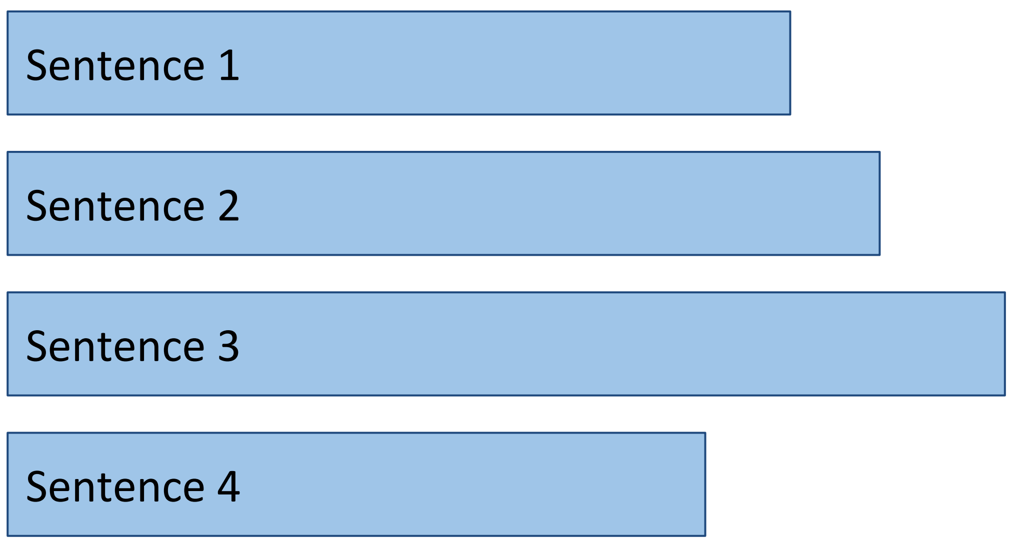
Try various relationships...

Padding

Practical Considerations

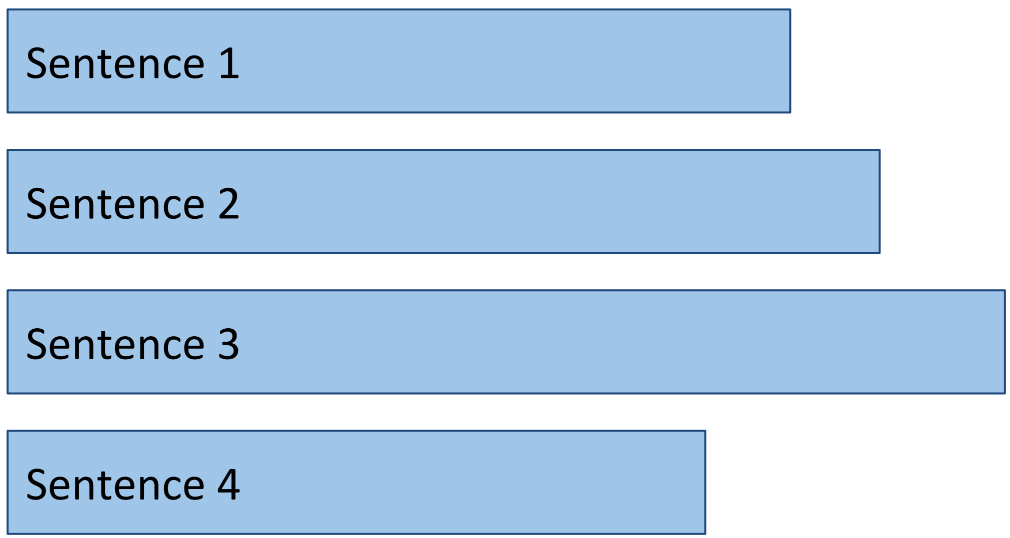
Padding

- Input sentences are of varied length
- Need a fixed length to define a fixed size of weight matrices



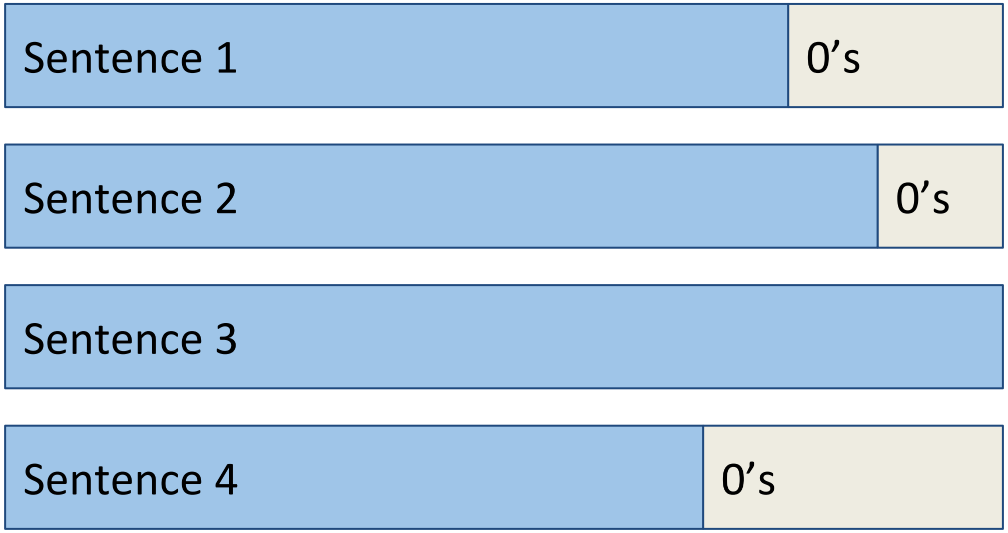
Padding

Solution: Pad smaller sentences with 0's



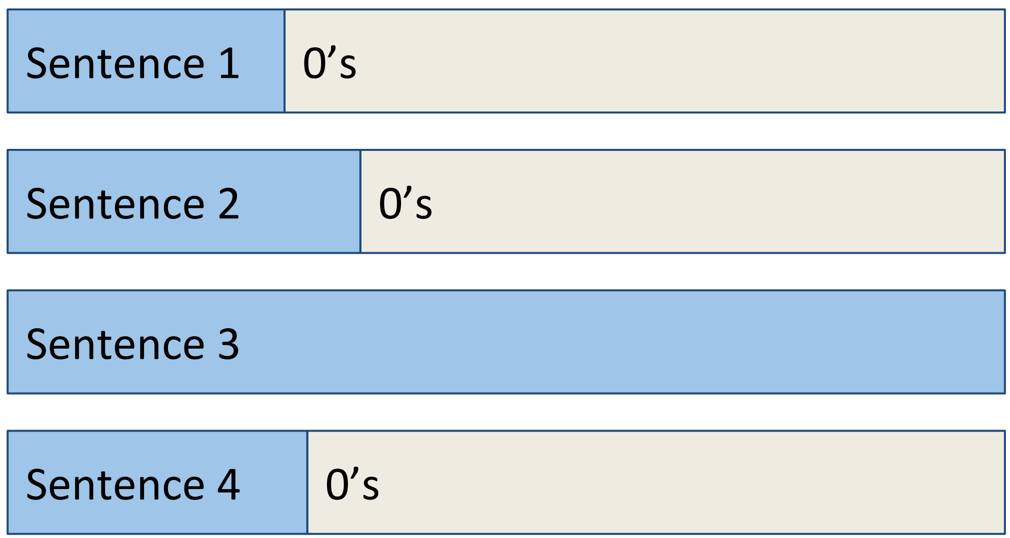
Padding

Solution: Pad smaller sentences with 0's



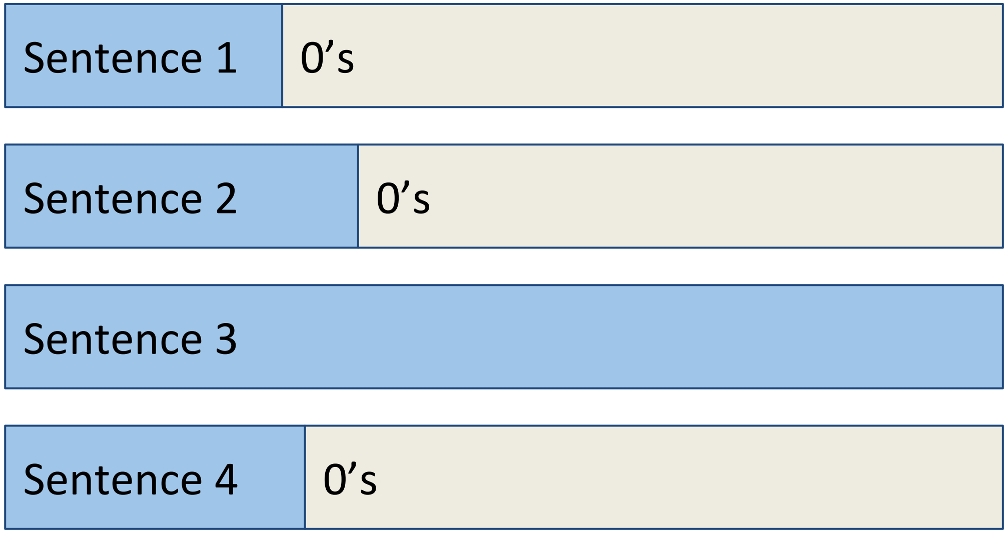
Padding

Problem: What if one sentence is very long in a batch?



Padding

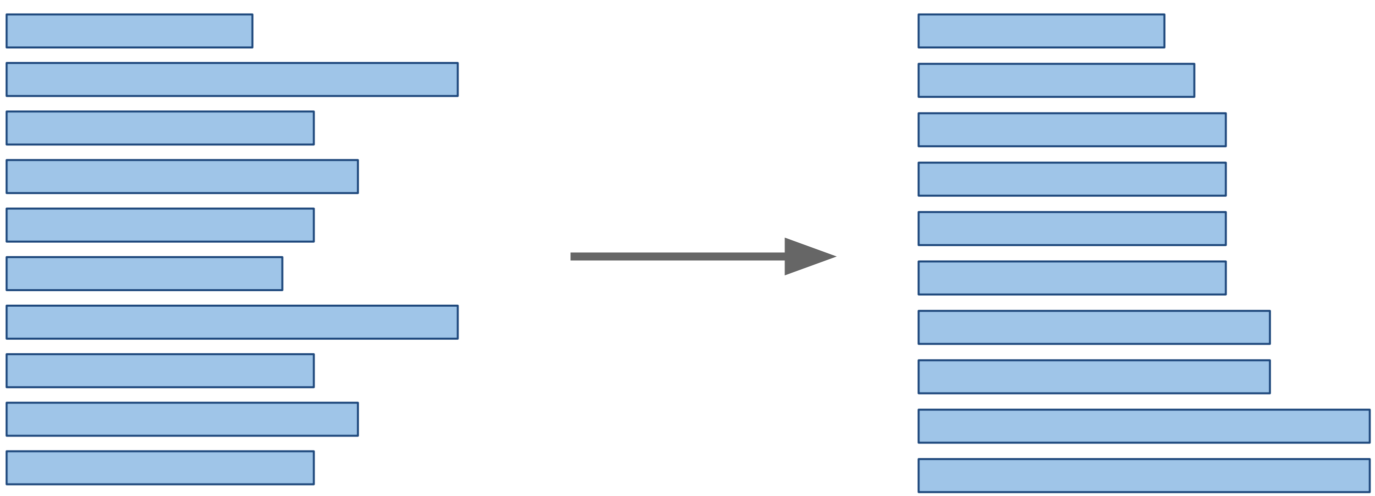
Problem: What if one sentence is very long in a batch?



A lot of wasted computation!

Padding

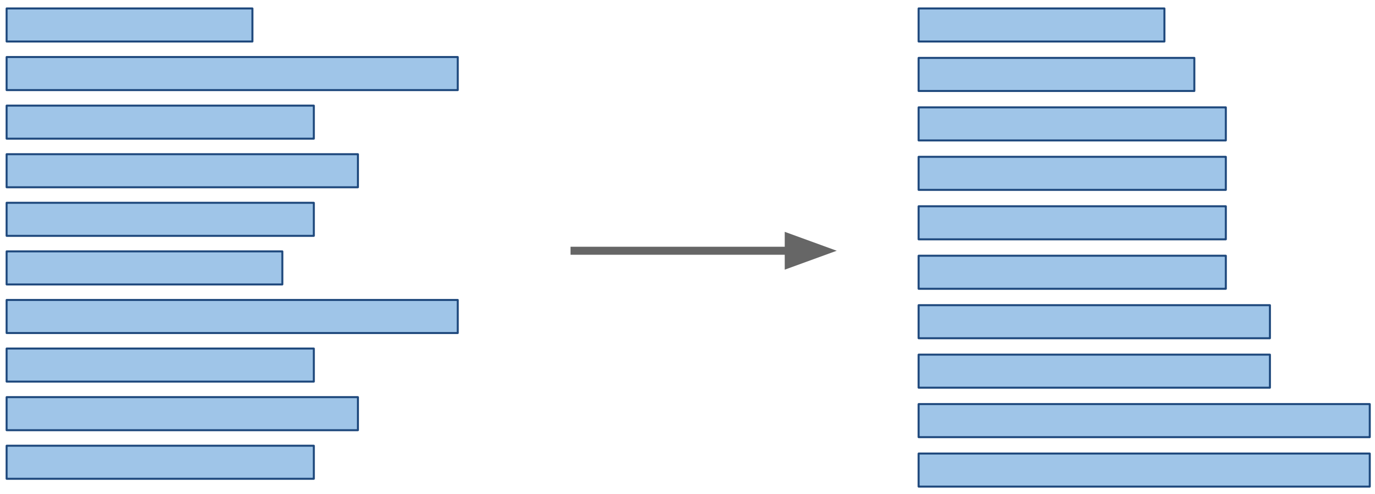
- Alternatively, sort all sentences by length
- Create minibatches by putting sentences of similar length together



- Limit wasted computation in every minibatch

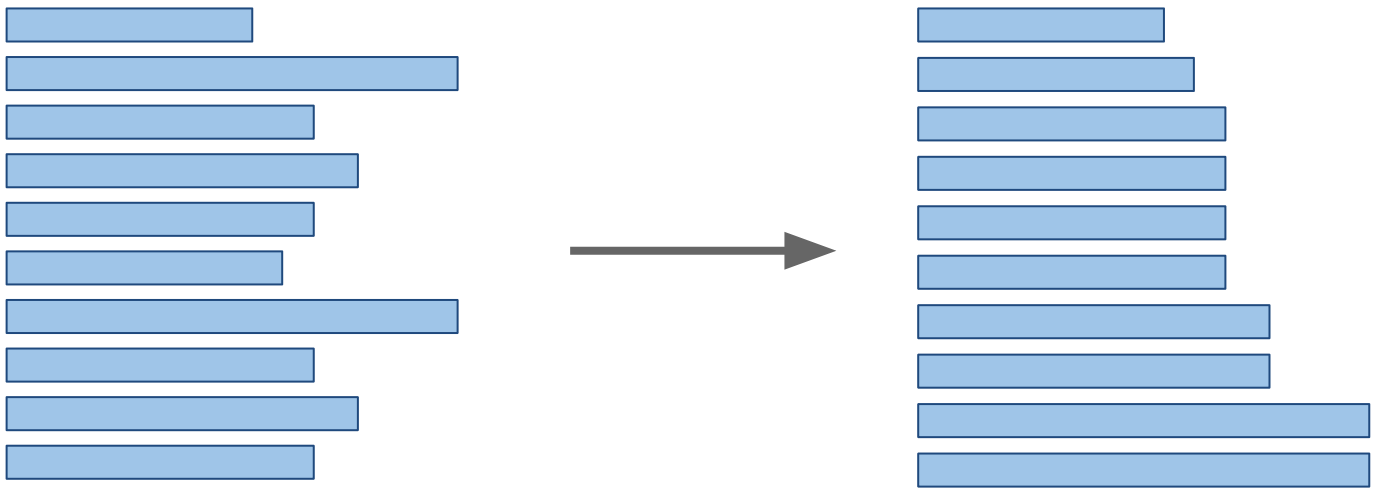
Padding

Q: Any problems with this solution?



Padding

Q: Any problems with this solution?



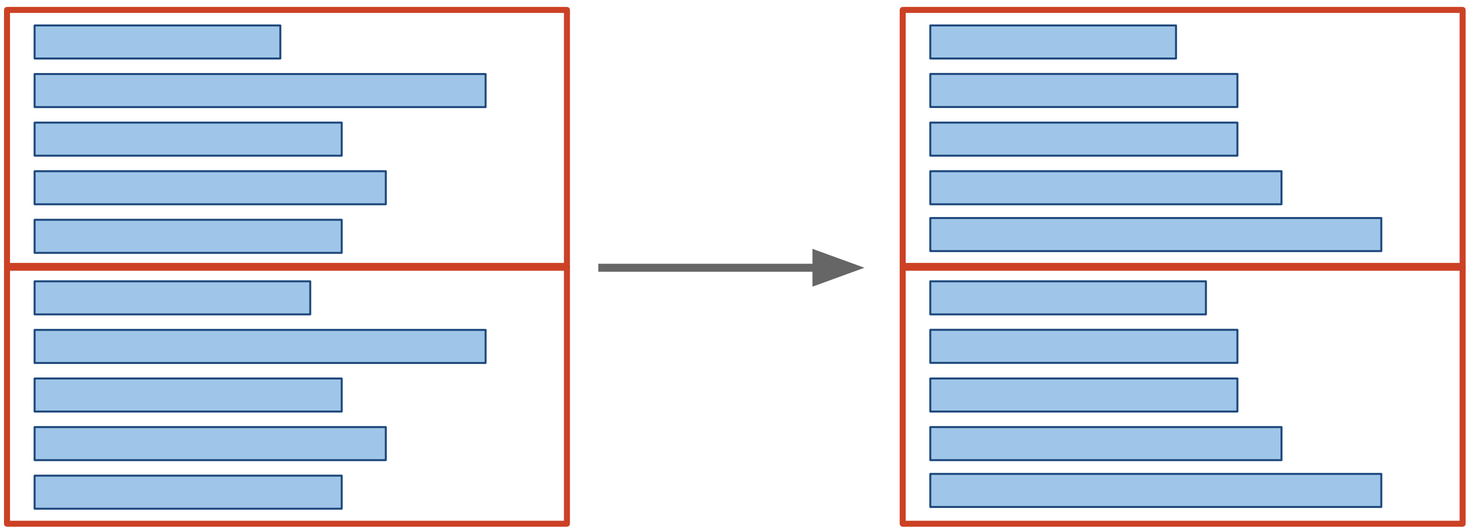
A: Yes! We are inducing a bias so that the model sees all short sentences early and all long sentences later

Padding

Solution: Sort sentences in a **maxi-batch**

Padding

Solution: Sort sentences in a **maxi-batch**



Now choose minibatches from each maxibatch