Deep text classification

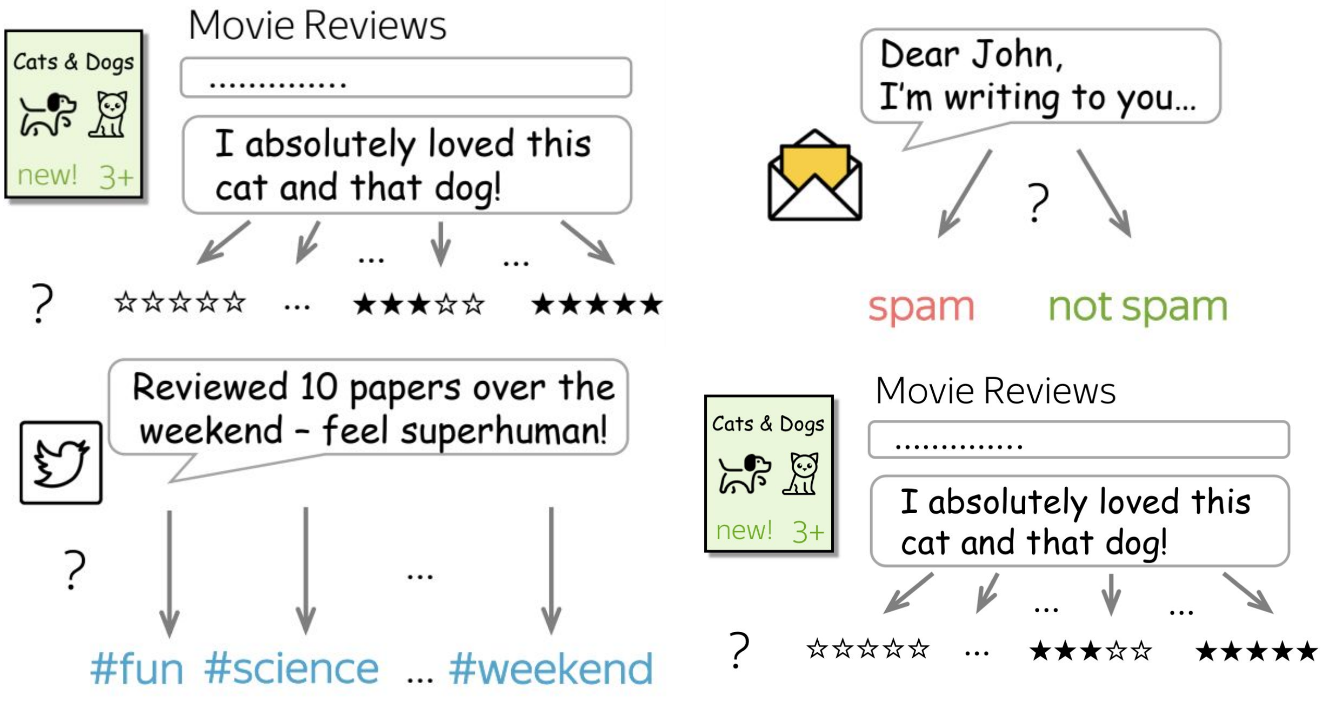
CNN, RNN

Основано на [NLP Course | For You]

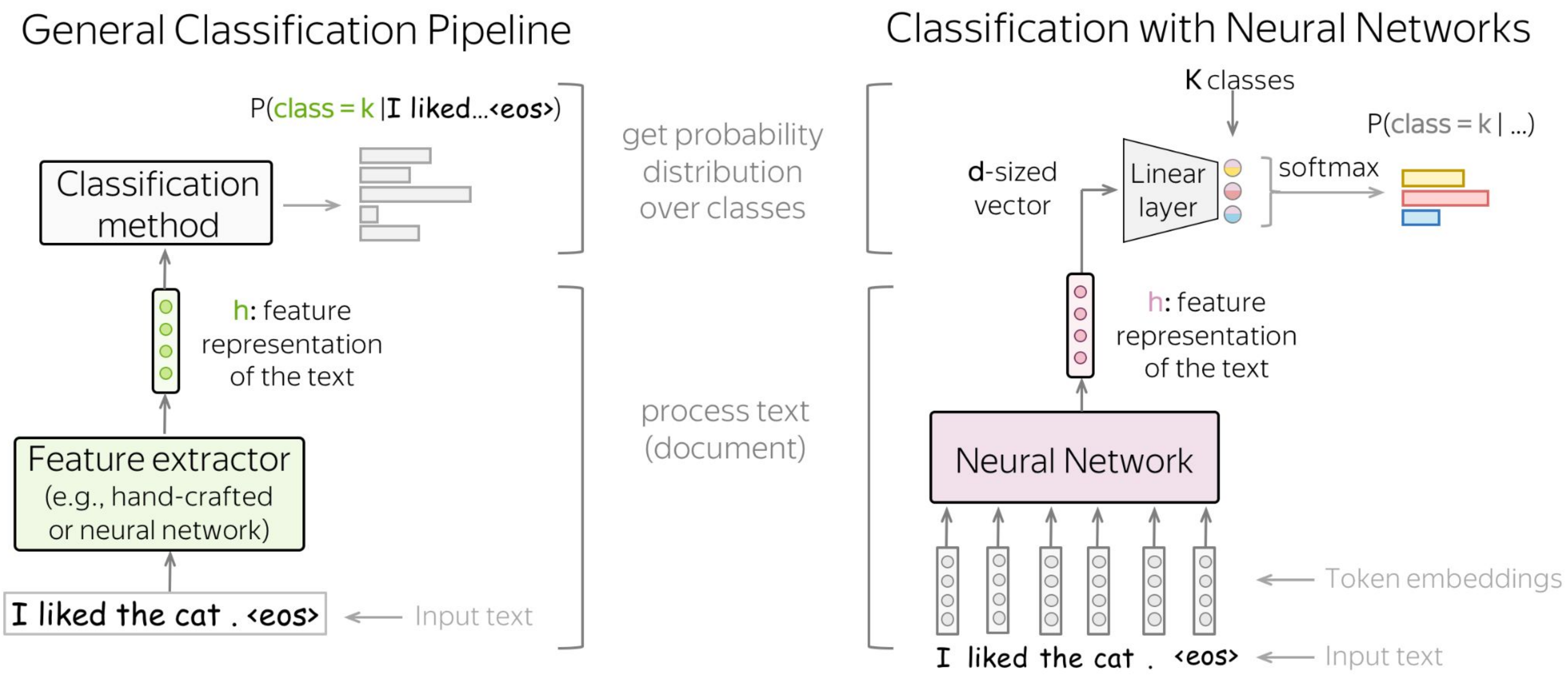
План

- Примеры
- Последний слой
- Свёрточные архитектуры
- Recurent NN
- Embeddings

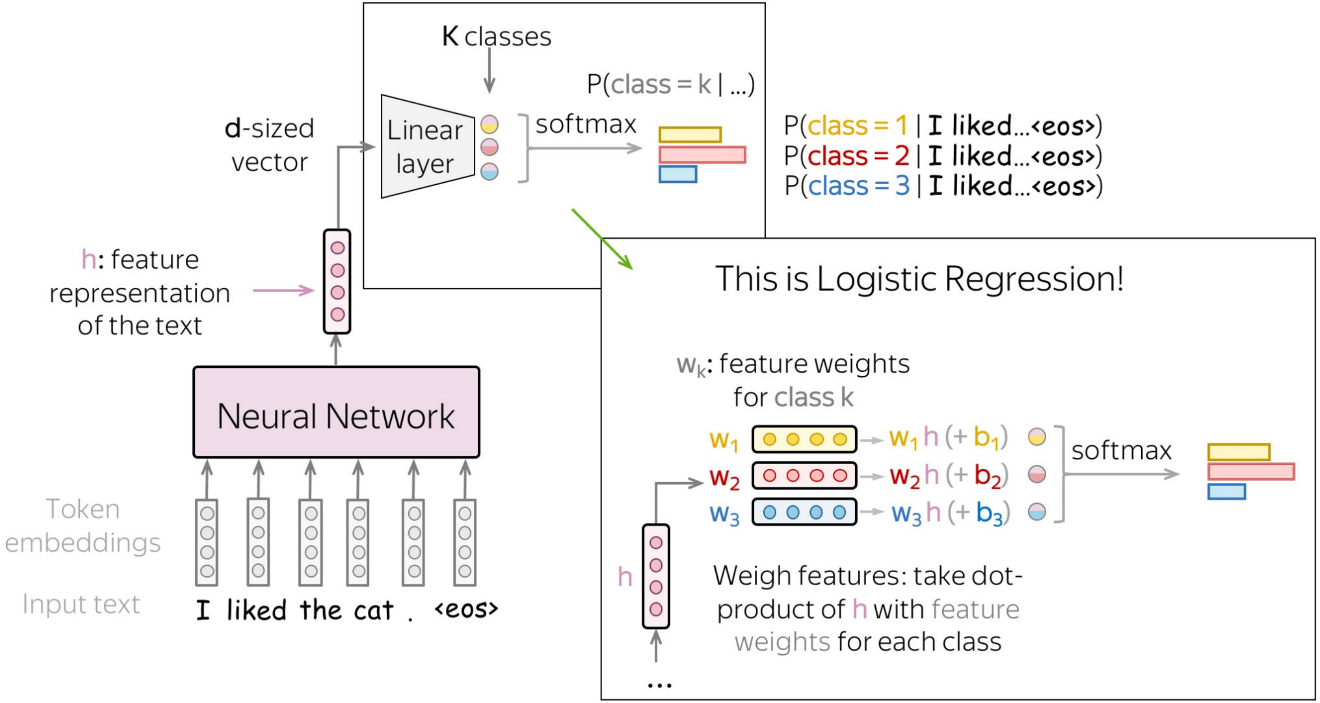
Примеры


Movie Reviews
Cats & Dogs
new! 3+
....
I absolutely loved this cat and that dog!
?
☆☆☆☆☆ ... ★★★☆☆ ★★★★★
Dear John,
I'm writing to you...
?
spam
not spam
Reviewed 10 papers over the weekend - feel superhuman!
?
#fun #science ... #weekend
Movie Reviews
Cats & Dogs
new! 3+
....
I absolutely loved this cat and that dog!
?
☆☆☆☆☆ ... ★★★☆☆ ★★★★★

Общая концепция



Последний слой



Loss-функция

Training example: **I liked the cat on the mat <eos>** Label: k

↑ target

Model prediction:

P(class = i | I liked…<eos>)

Target:

p^*

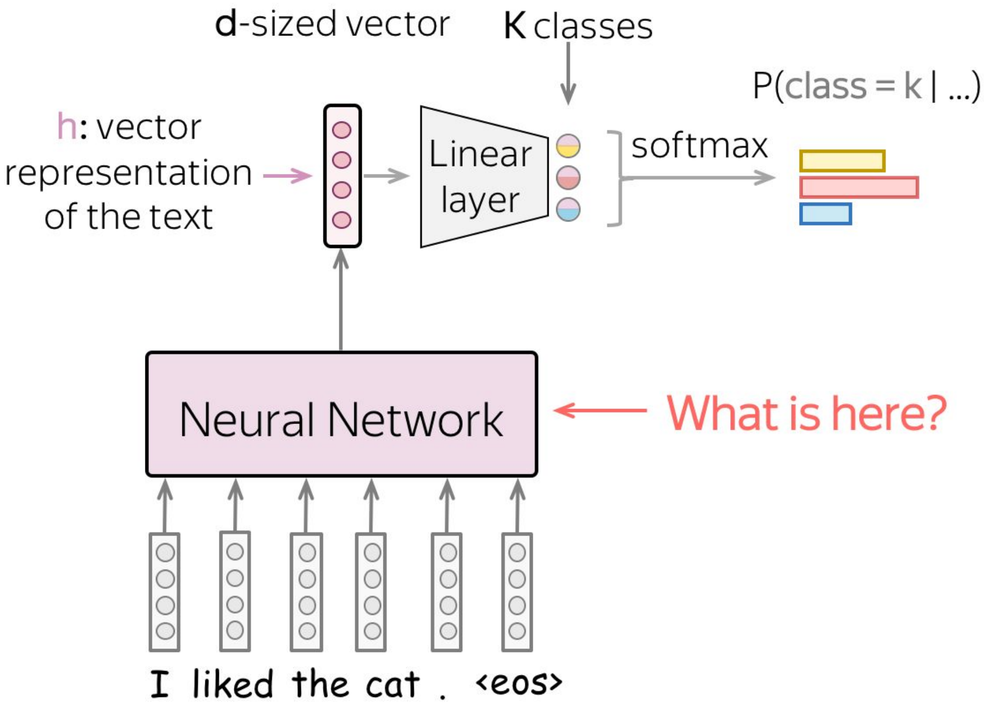
Cross-entropy loss:

$$-\sum_{i=1}^{K} p_i^* \cdot \log P(y=i|x) \to min \qquad (p_k^* = 1, p_i^* = 0, i \neq k\ )$$

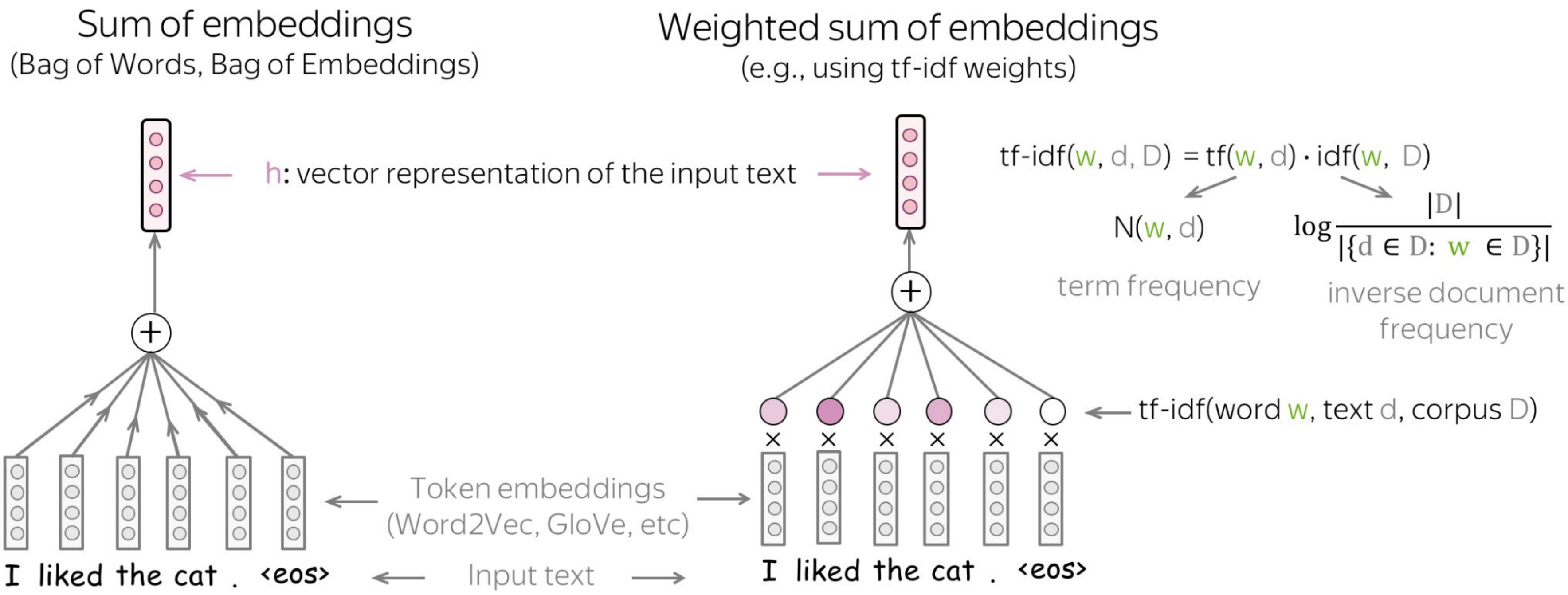
For one-hot targets, this is equivalent to

$$-\log P(y=k|x) \to min$$

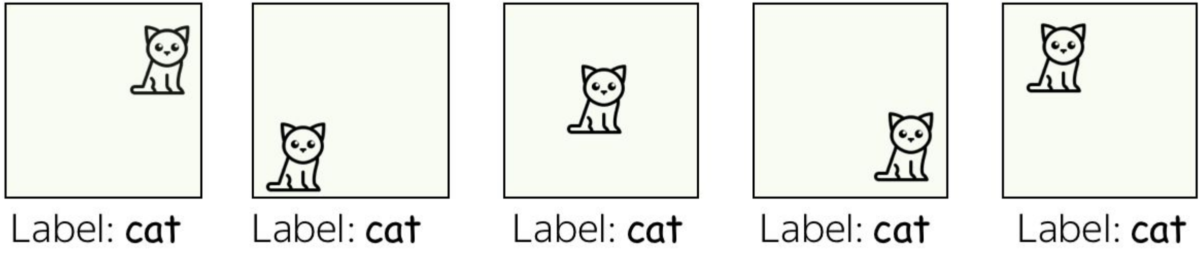
А что внутри?


d-sized vector
K classes
P(class = k | ...)
h: vector representation of the text
Linear layer
softmax
Neural Network
What is here?
I liked the cat . <eos>

№0. Усреднение эмбедингов



CNN



We don't care where the cat is, we care that it is somewhere.

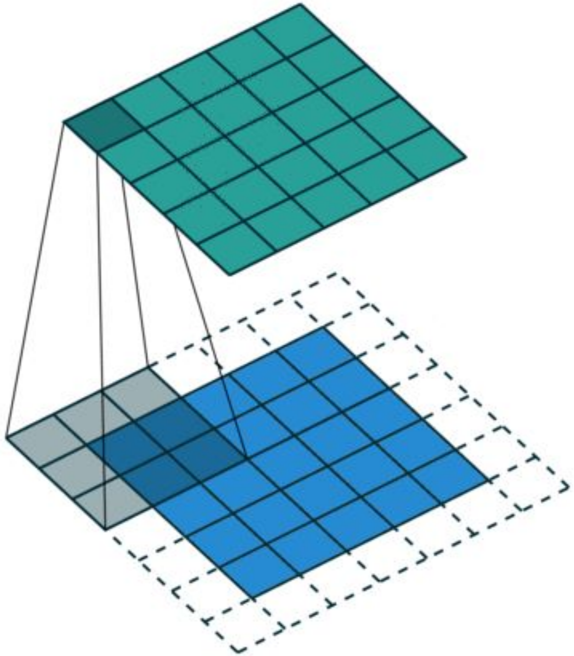
Then why don't we process all these cats similarly?

An absolutely great movie! I watched the premiere with my friends.

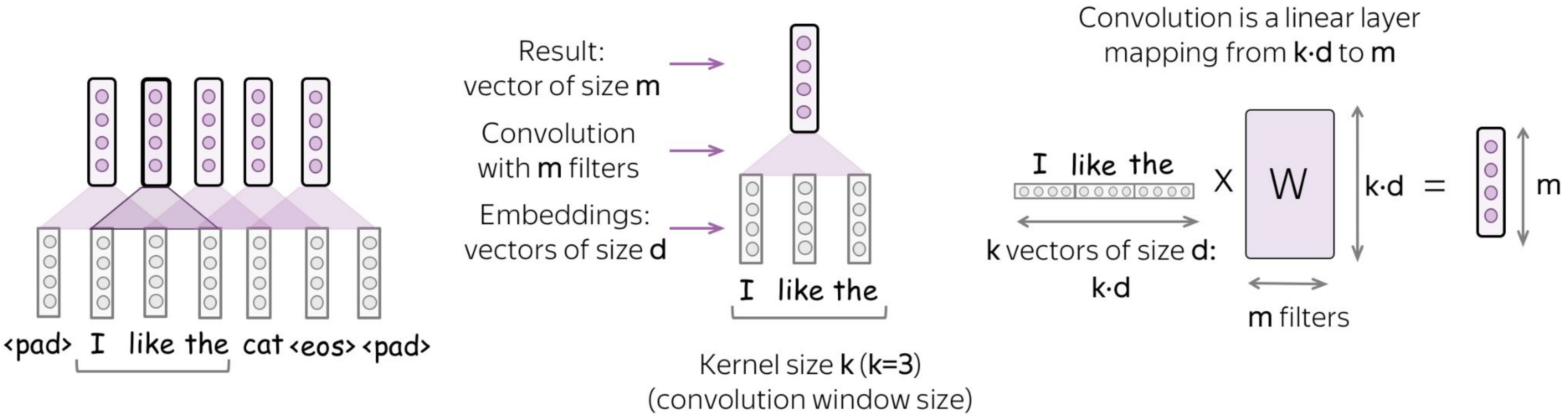
The movie about cats was absolutely great, and the cats were cute.

The movie is about cats running around, and it is absolutely great.

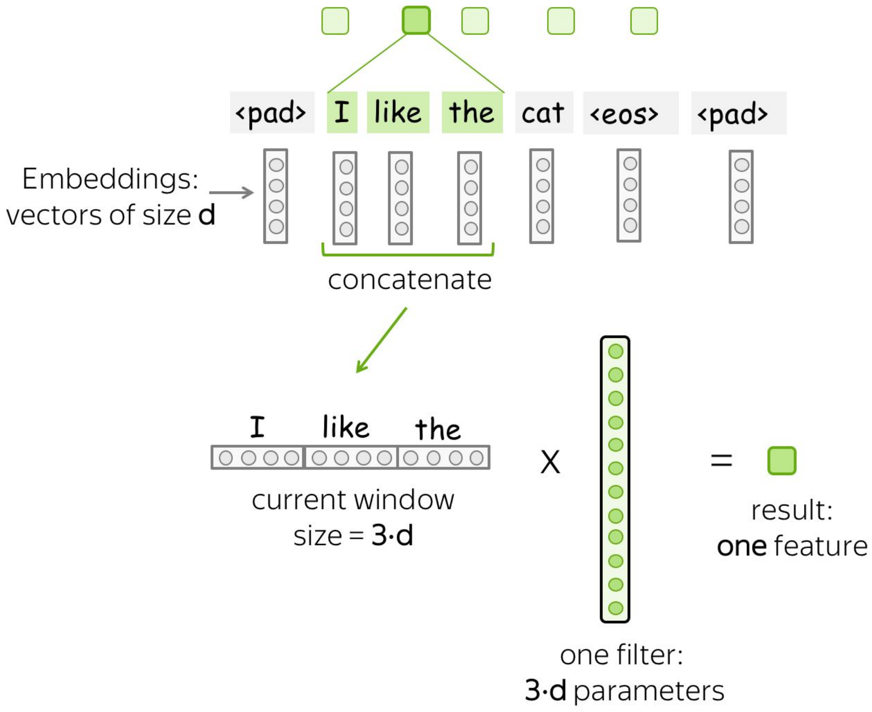
CNN



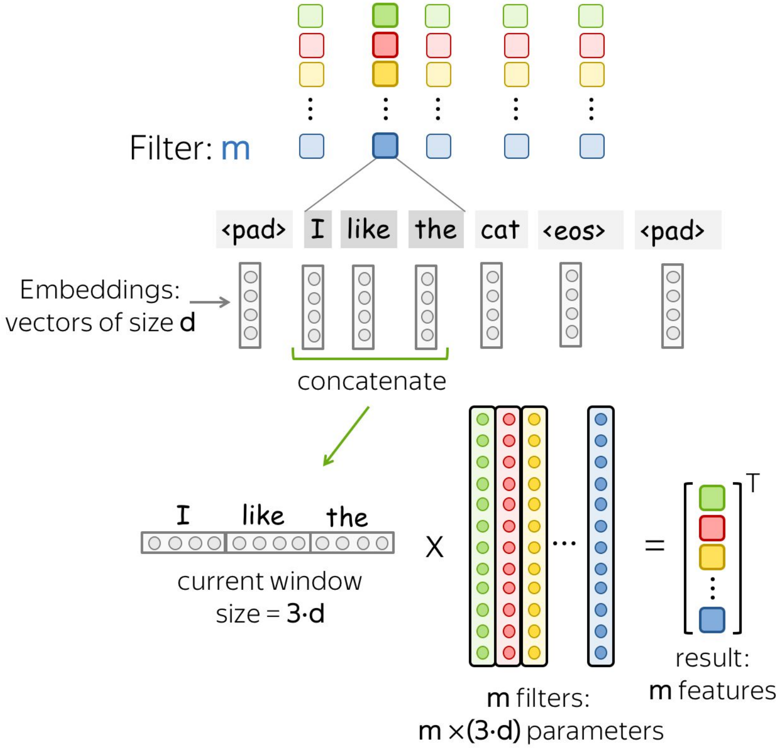
CNN


<pad> I like the cat <eos> <pad>
Result:
vector of size **m**
Convolution
with **m** filters
Embeddings:
vectors of size **d**
I like the
Kernel size **k** (**k=3**)
(convolution window size)
Convolution is a linear layer
mapping from **k·d** to **m**
I like the
X
W
k·d
=
m
k vectors of size **d**:
k·d
m filters

CNN

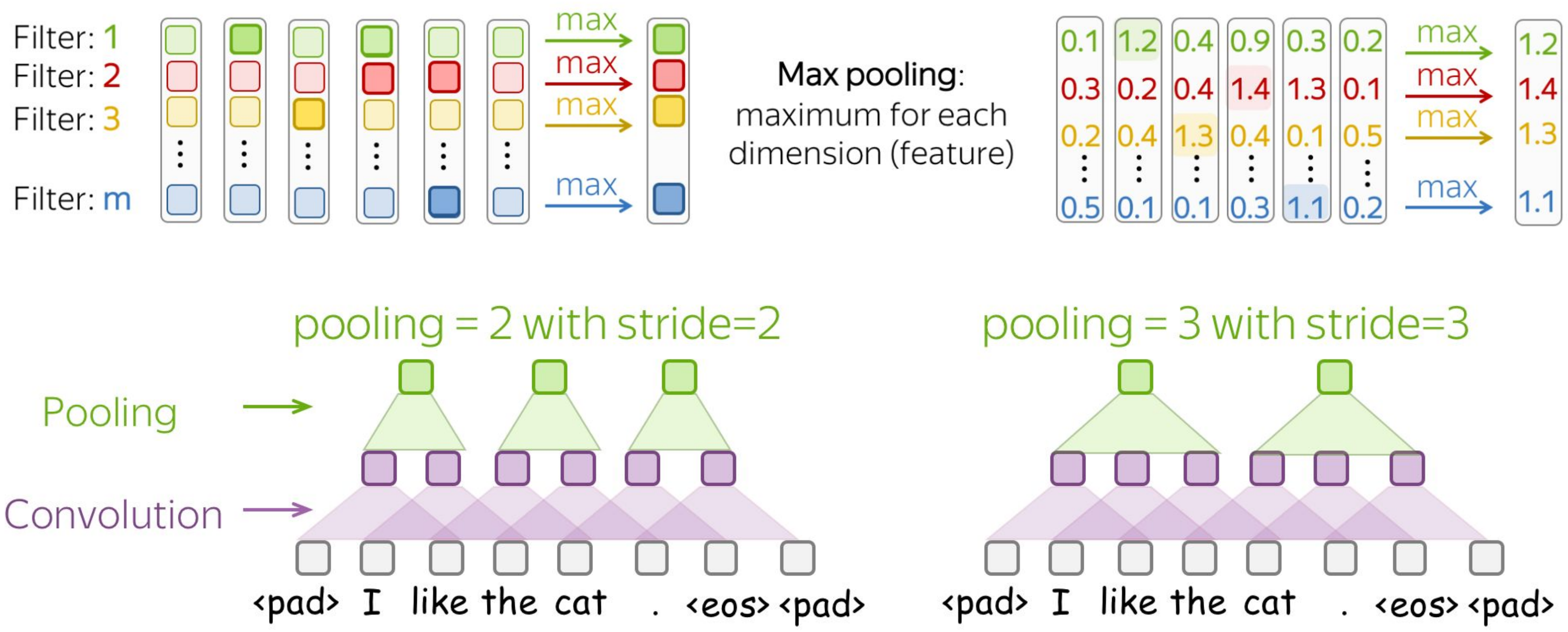


<pad>
I
like
the
cat
<eos>
<pad>
Embeddings:
vectors of size d
concatenate
I like the
current window
size = 3·d
X
=
result:
one feature
one filter:
3·d parameters



Filter: m
<pad>
I
like
the
cat
<eos>
<pad>
Embeddings:
vectors of size d
concatenate
I like the
current window
size = 3·d
X
=
T
result:
m features
m filters:
m ×(3·d) parameters

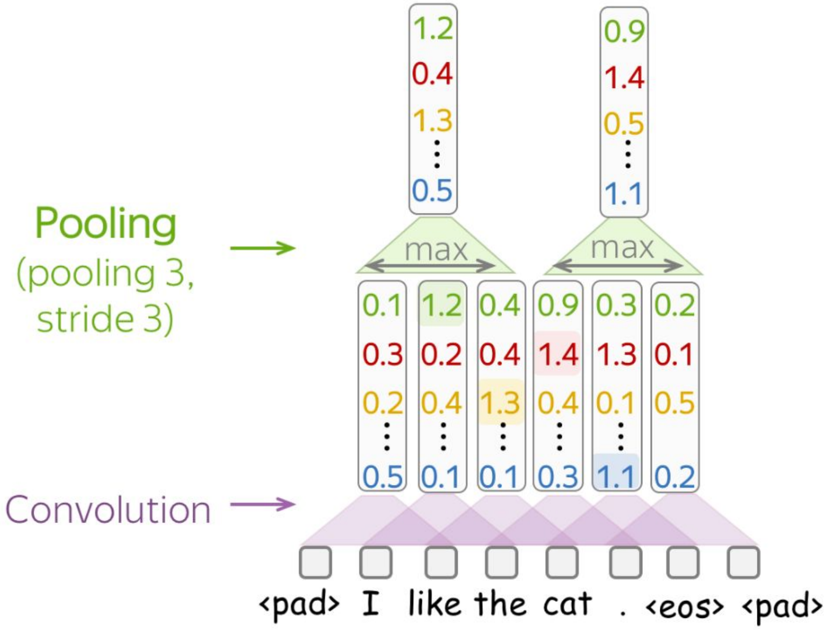
CNN



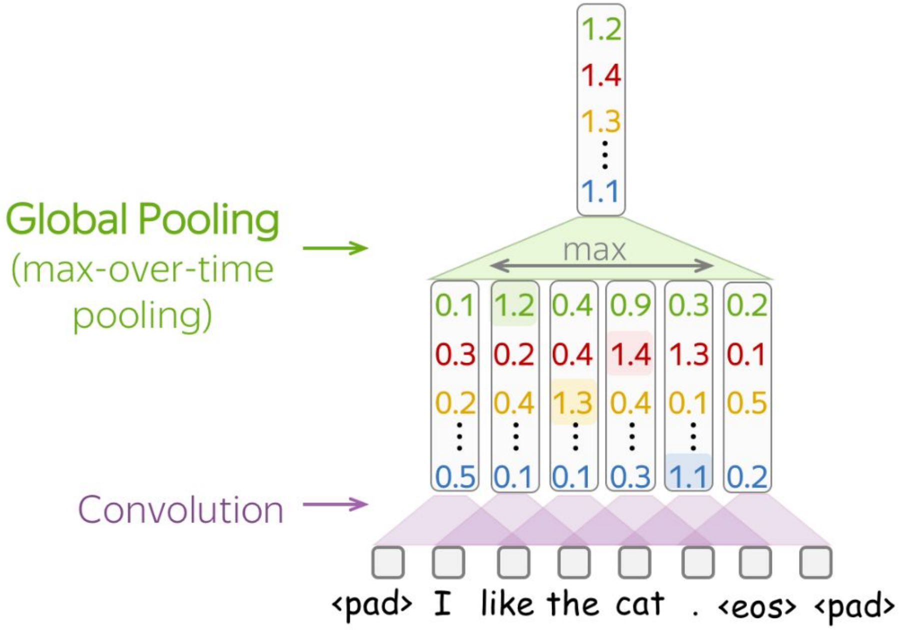
Filter: 1
Filter: 2
Filter: 3
Filter: m
max
max
max
max
Max pooling:
maximum for each dimension (feature)
0.1 1.2 0.4 0.9 0.3 0.2
0.3 0.2 0.4 1.4 1.3 0.1
0.2 0.4 1.3 0.4 0.1 0.5
0.5 0.1 0.1 0.3 1.1 0.2
1.2
1.4
1.3
1.1
pooling = 2 with stride=2
pooling = 3 with stride=3
Pooling
Convolution
<pad> I like the cat . <eos> <pad>
<pad> I like the cat . <eos> <pad>

CNN

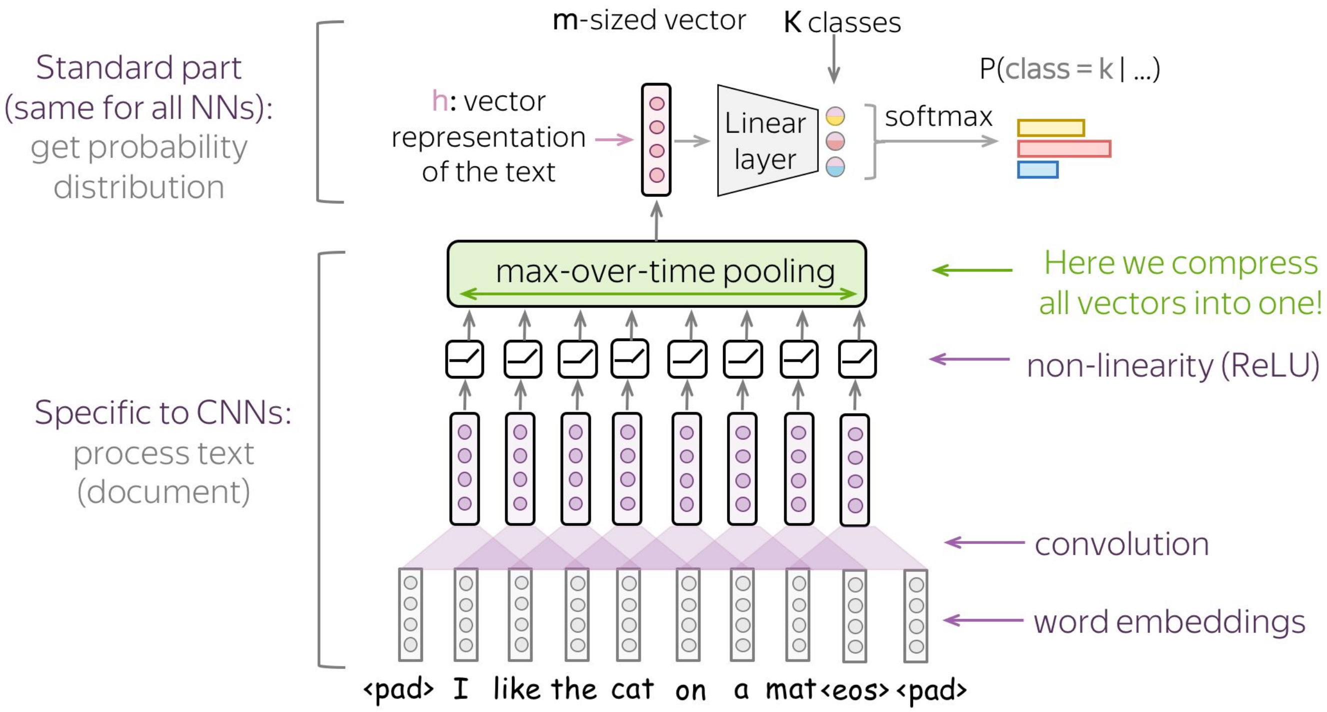
Pooling



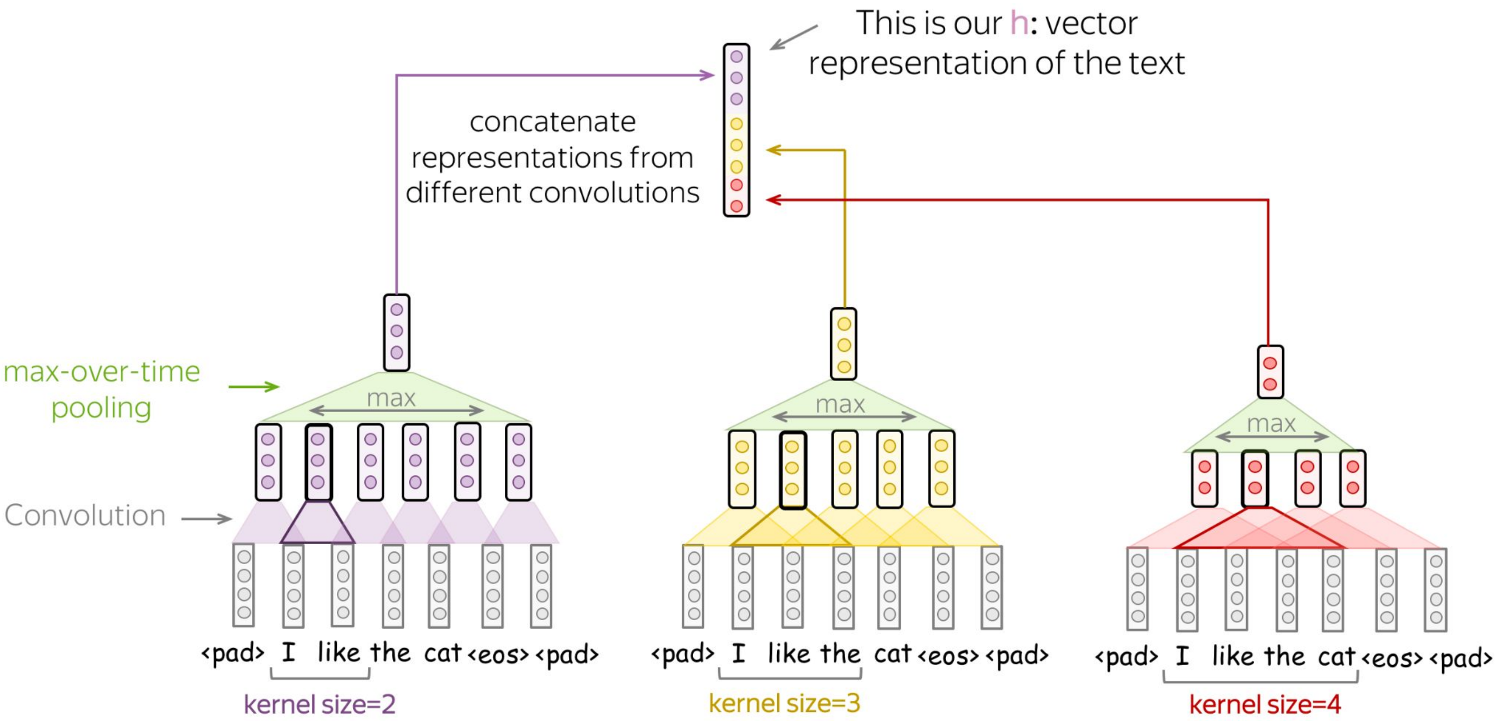
Global Pooling



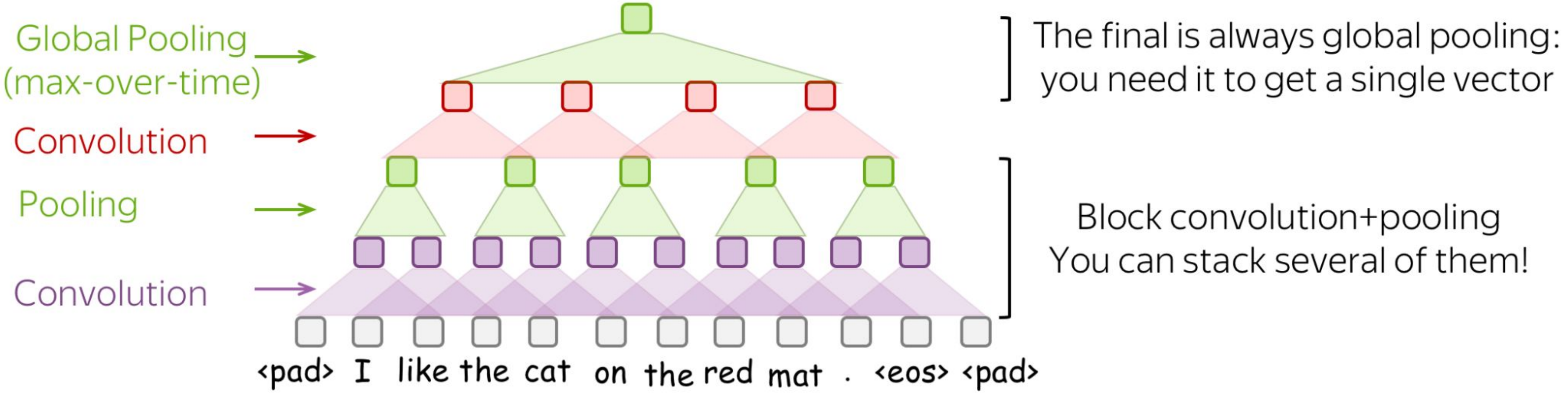
CNN


Standard part
(same for all NNs):
get probability
distribution
m-sized vector
K classes
P(class = k | ...)
h: vector
representation
of the text
Linear
layer
softmax
max-over-time pooling
Here we compress
all vectors into one!
non-linearity (ReLU)
Specific to CNNs:
process text
(document)
convolution
word embeddings
<pad> I like the cat on a mat <eos> <pad>

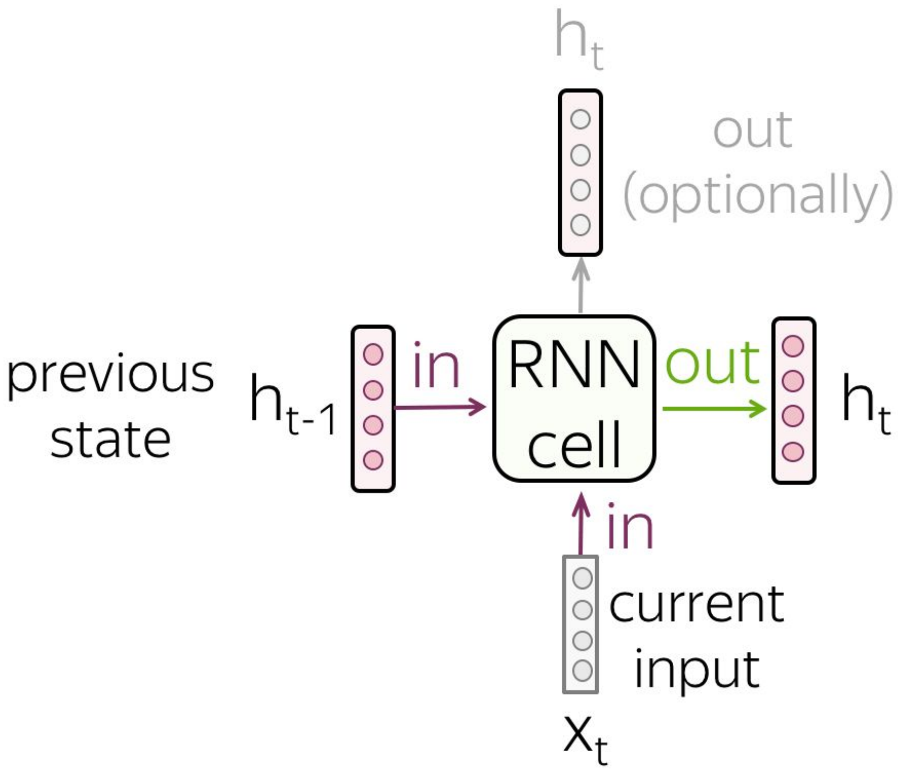
CNN


This is our h: vector representation of the text
concatenate representations from different convolutions
max-over-time pooling
Convolution
max
max
max
<pad> I like the cat <eos> <pad>
kernel size=2
<pad> I like the cat <eos> <pad>
kernel size=3
<pad> I like the cat <eos> <pad>
kernel size=4

CNN

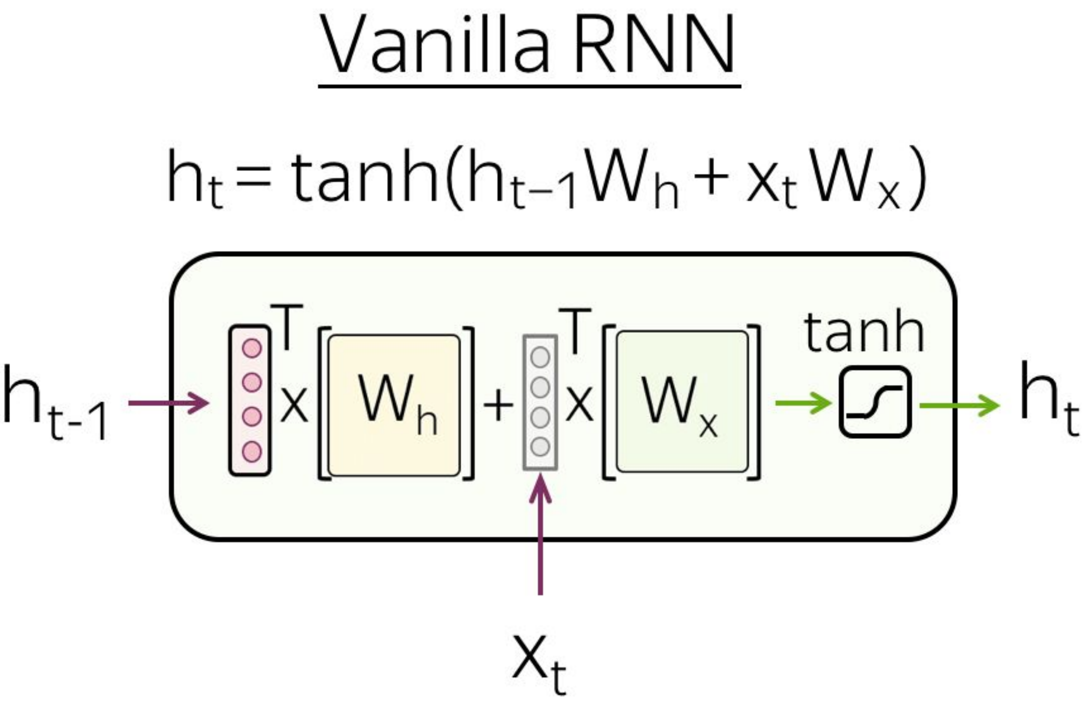

Global Pooling
(max-over-time)
Convolution
Pooling
Convolution
The final is always global pooling:
you need it to get a single vector
Block convolution+pooling
You can stack several of them!
<pad> I like the cat on the red mat . <eos> <pad>

RNN

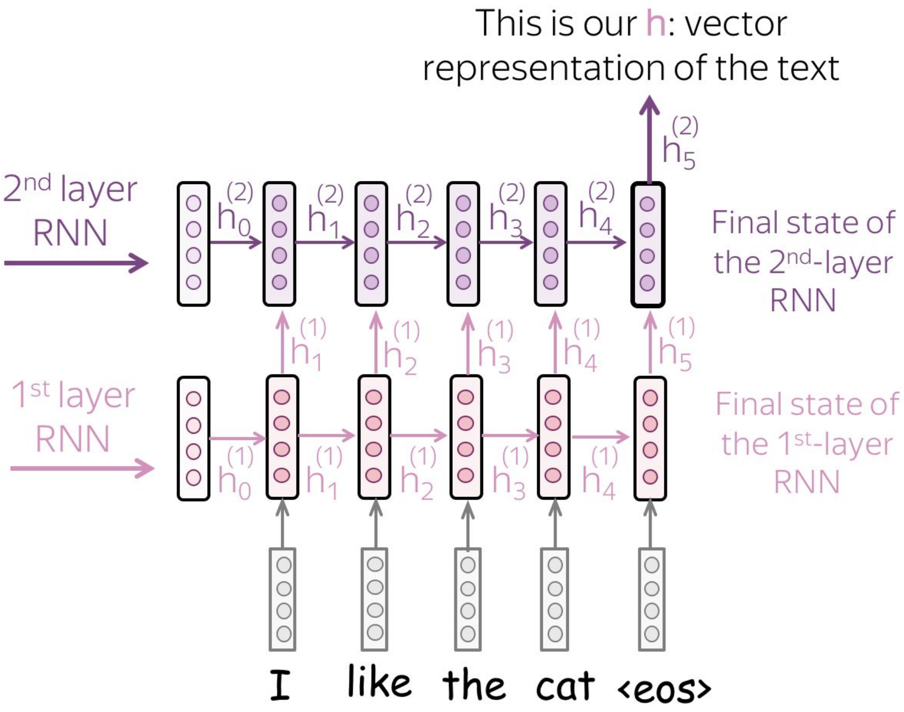


h_t
out
(optionally)
previous
state
h_{t-1}
in
RNN
cell
out
h_t
in
current
input
x_t

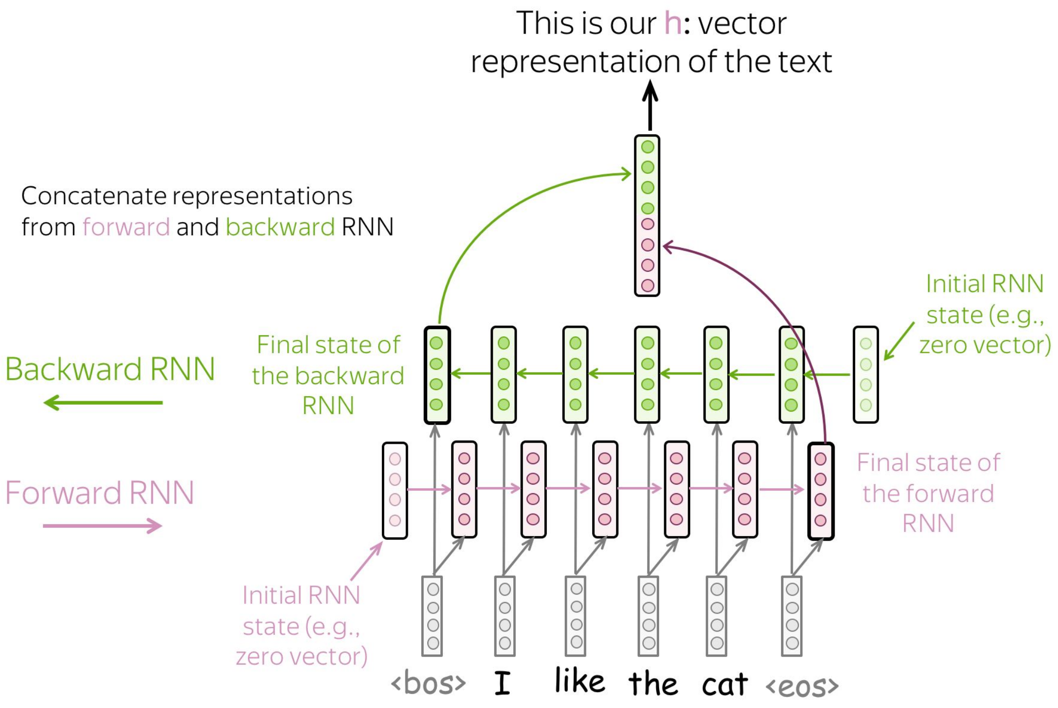
RNN



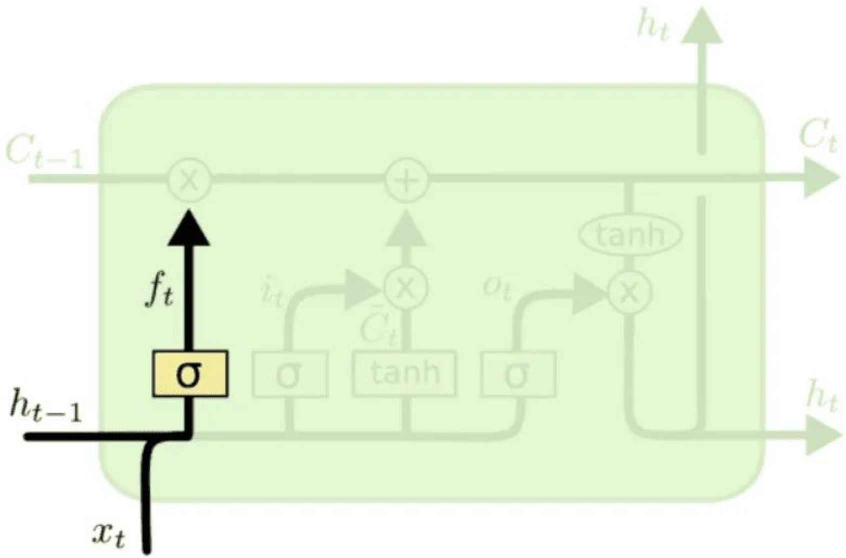
Twp layer RNN



Bidirectional RNN

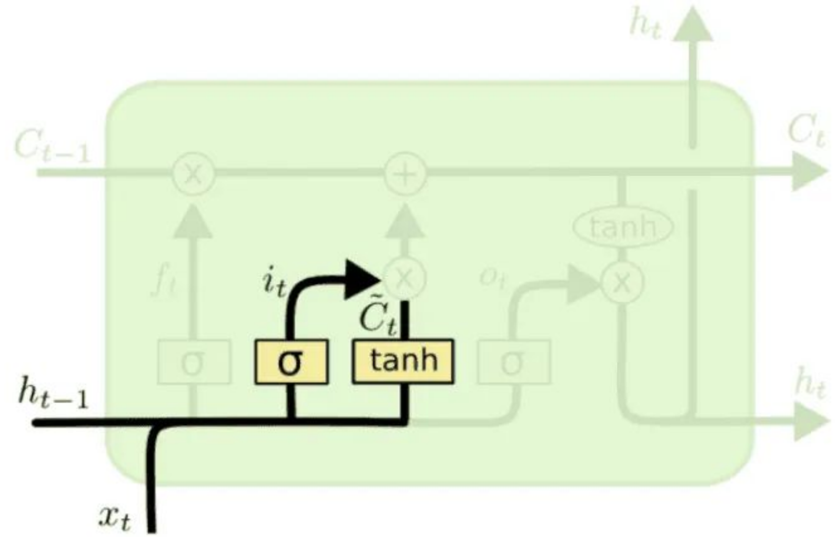


LSTM



$$f_t = \sigma(h_{t-1}W_1^f + x_tW_2^f + b_f)$$

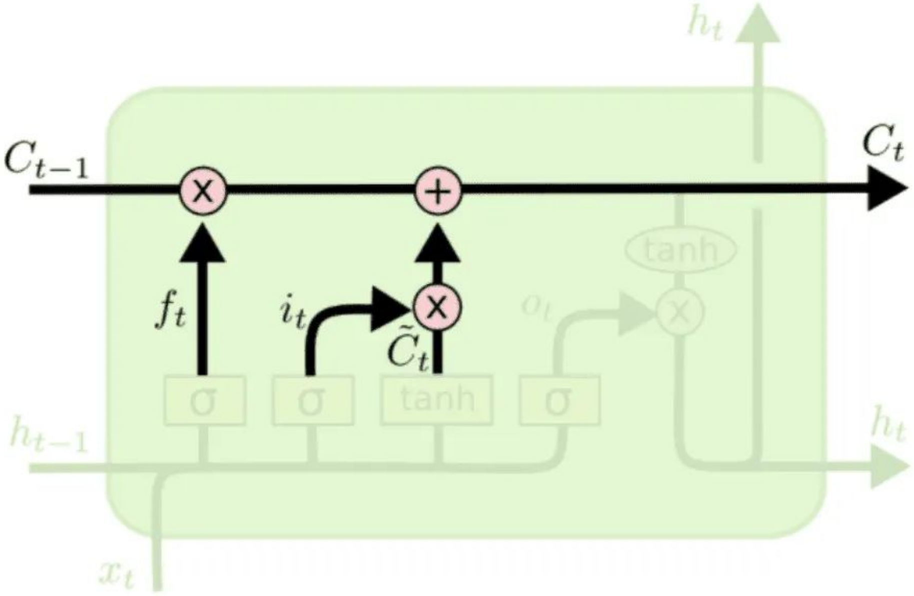
LSTM



$$\tilde{C}_t = \tanh(h_{t-1}W_1^C + x_t W_2^C + b_c)$$

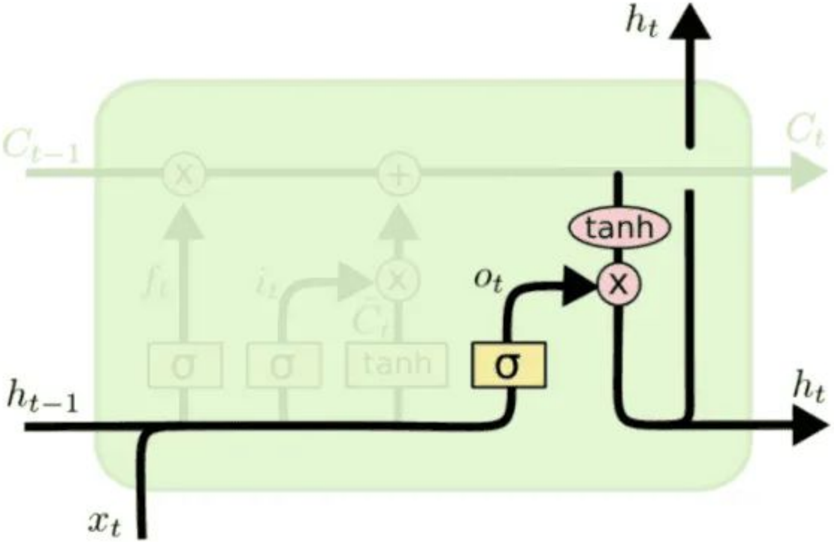
$$i_t = \sigma(h_{t-1}W_1^i + x_t W_2^i + b_i)$$

LSTM



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

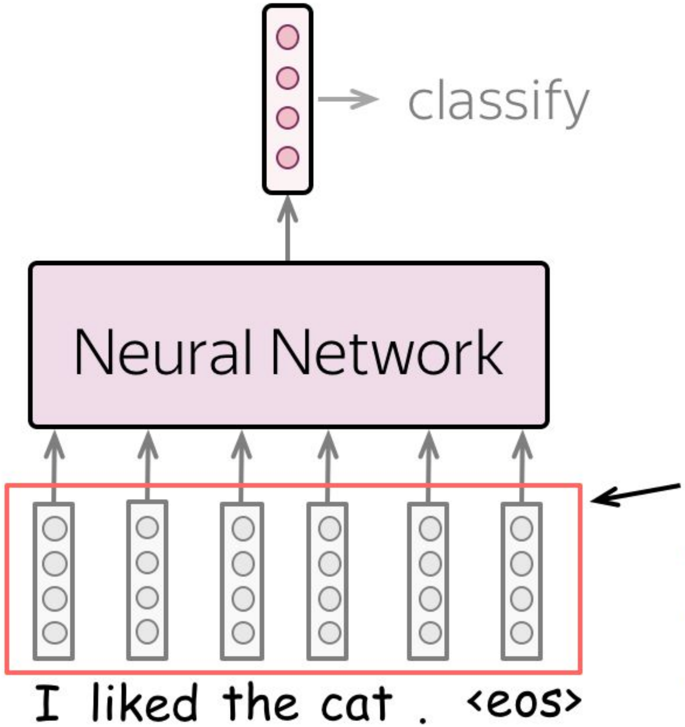
LSTM



$$o_t = \sigma(h_{t-1}W_1^o + x_tW_2^o + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

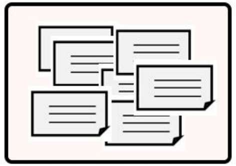
Embeddings



Input word embeddings:

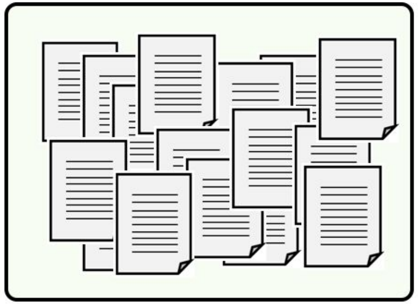
- Train from scratch
- Take pretrained (Word2Vec, GloVe)
- Initialize with pretrained, then fine-tune

Embeddings



Training data for text classification (labeled)

- Not huge, or not diverse, or both
- Domain: task-specific



Training data for word embeddings (unlabeled)

- Huge diverse corpus (e.g., Wikipedia)
- Domain: general

Embeddings

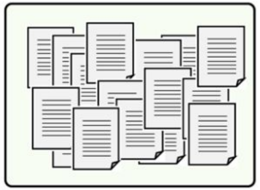
- Train from scratch

What they will know:

May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

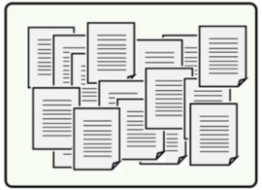
What they will know:



Know relationships between words, but are **not** specific to the task

- **Initialize with pretrained, then fine-tune**

What they will know:



+

Know relationships between words and adapted for the task

"**Transfer**" knowledge from a huge unlabeled corpus to your task-specific model