
Foundations for Natural Language Processing

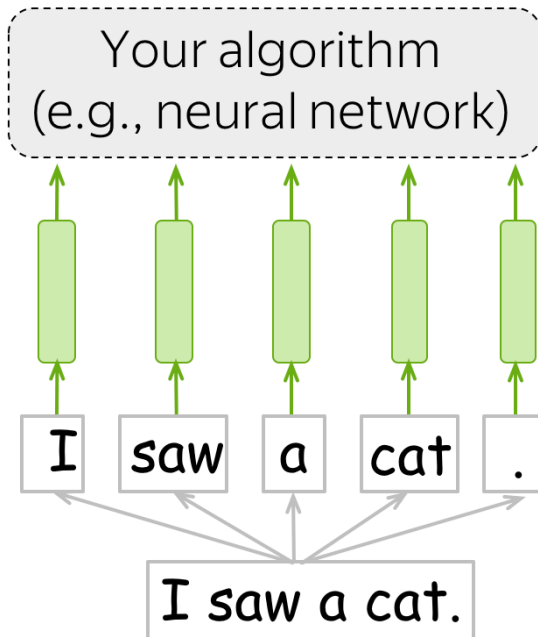
Neural Classifiers

Ivan Titov

(with graphics/materials from Elena Voita)



Neural models and word embeddings



Any algorithm for solving a task

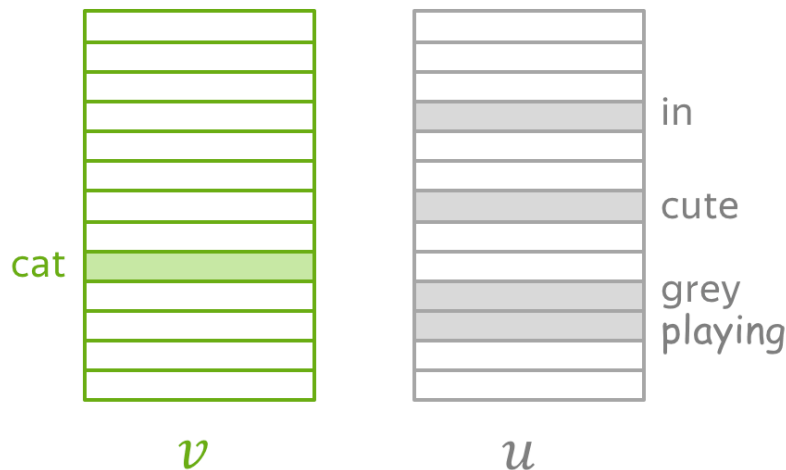
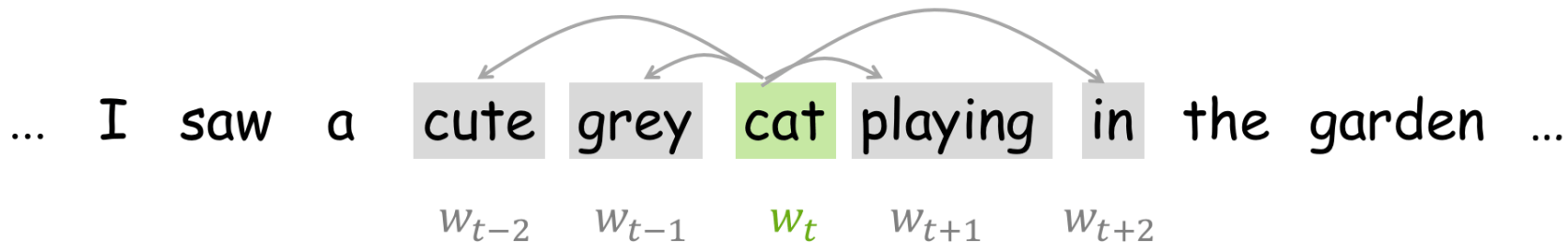
Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)

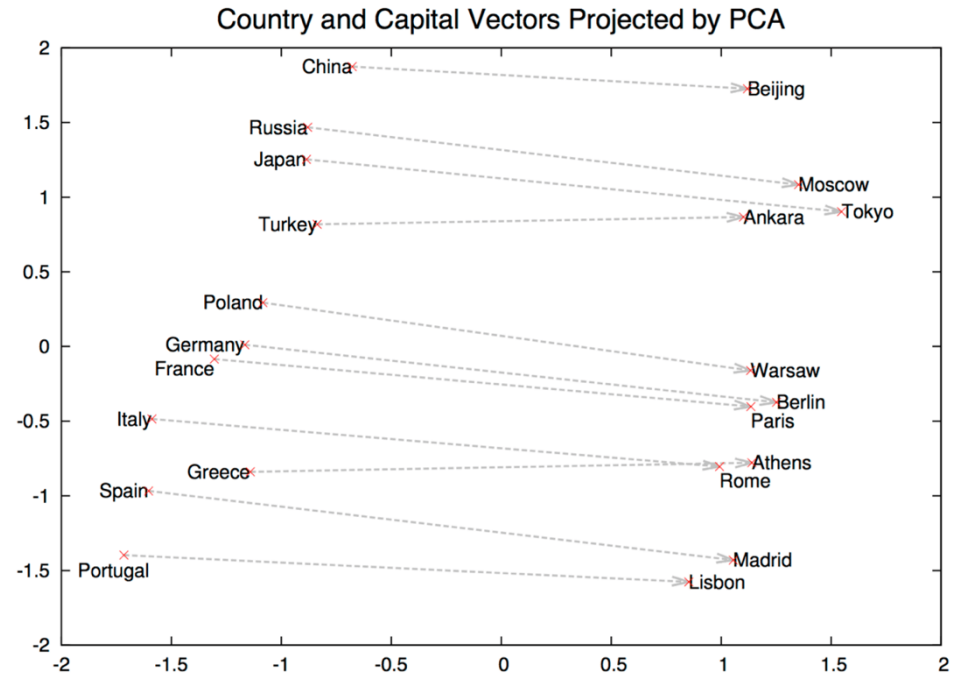
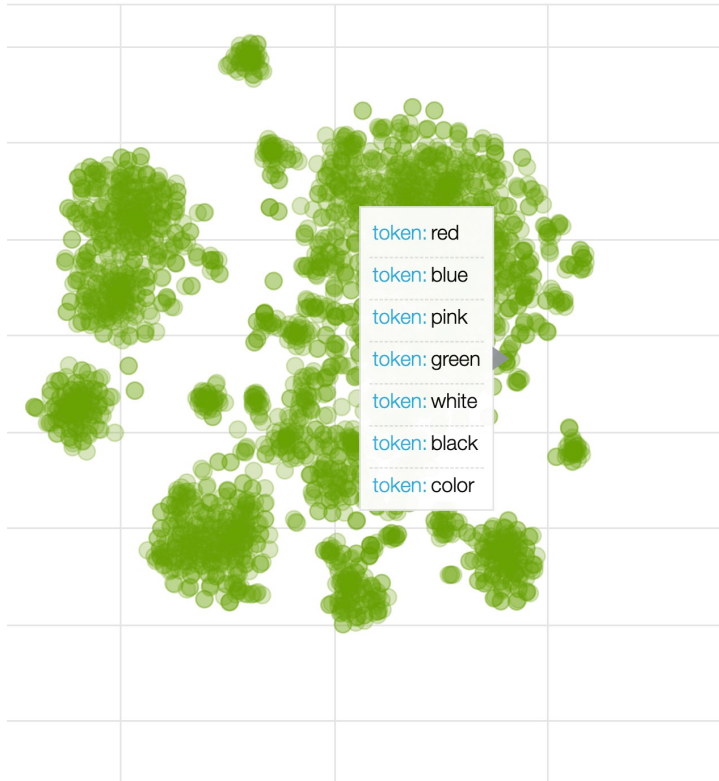
Last time: embeddings with neural predictors

$$P(u_{cute}|v_{cat}) \quad P(u_{grey}|v_{cat}) \quad P(u_{playing}|v_{cat}) \quad P(u_{in}|v_{cat})$$



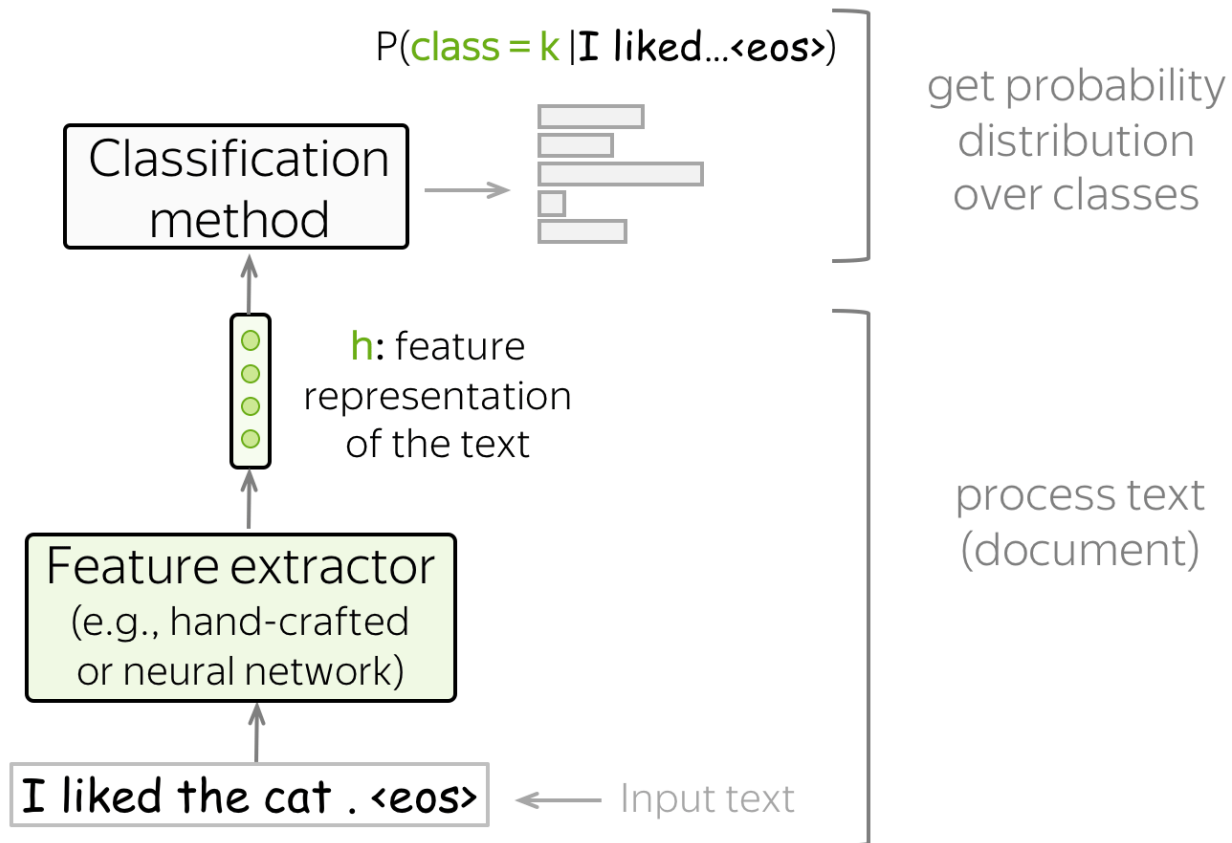
$$P(cute|cat) = -\log \frac{\exp u_{cute}^T v_{cat}}{\sum_{w \in Voc} \exp u_w^T v_{cat}}$$

Last time: intriguing properties

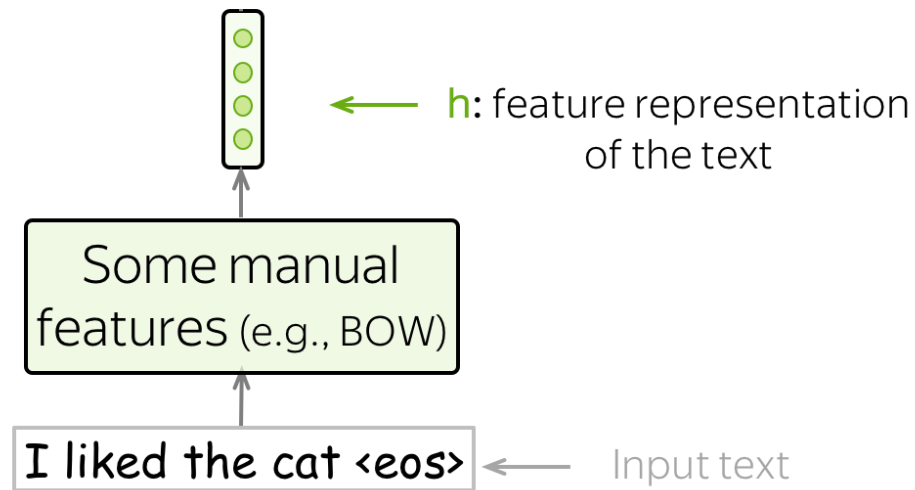


Classification

General Classification Pipeline

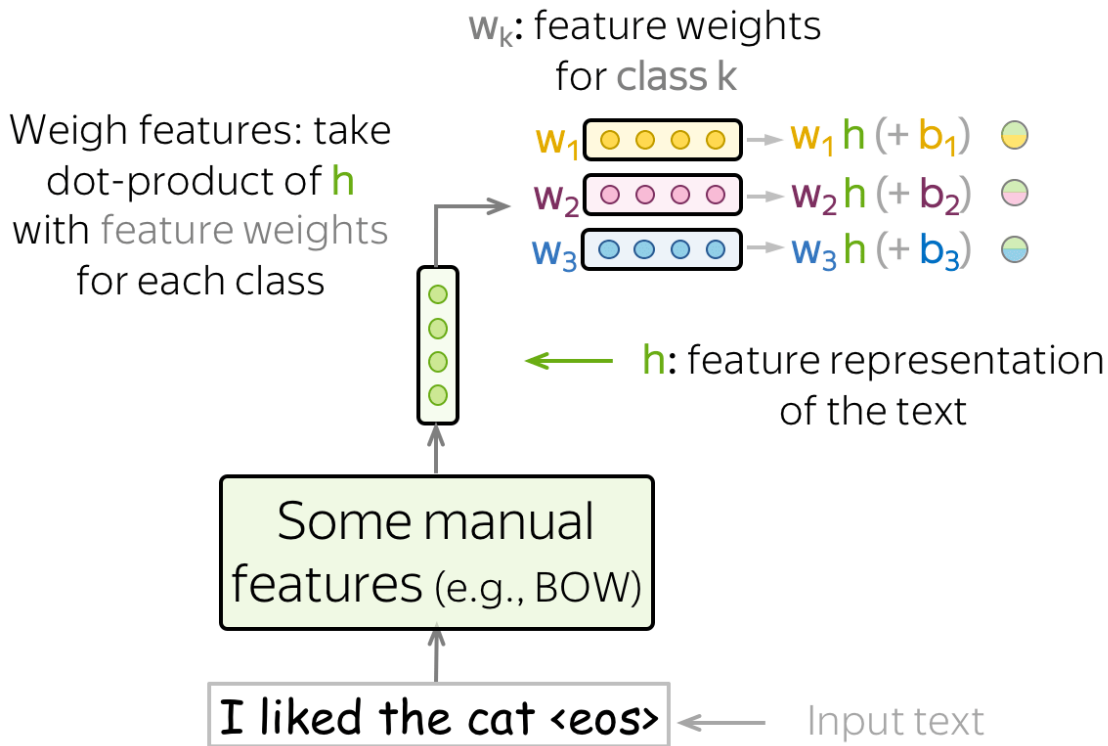


Logistic regression



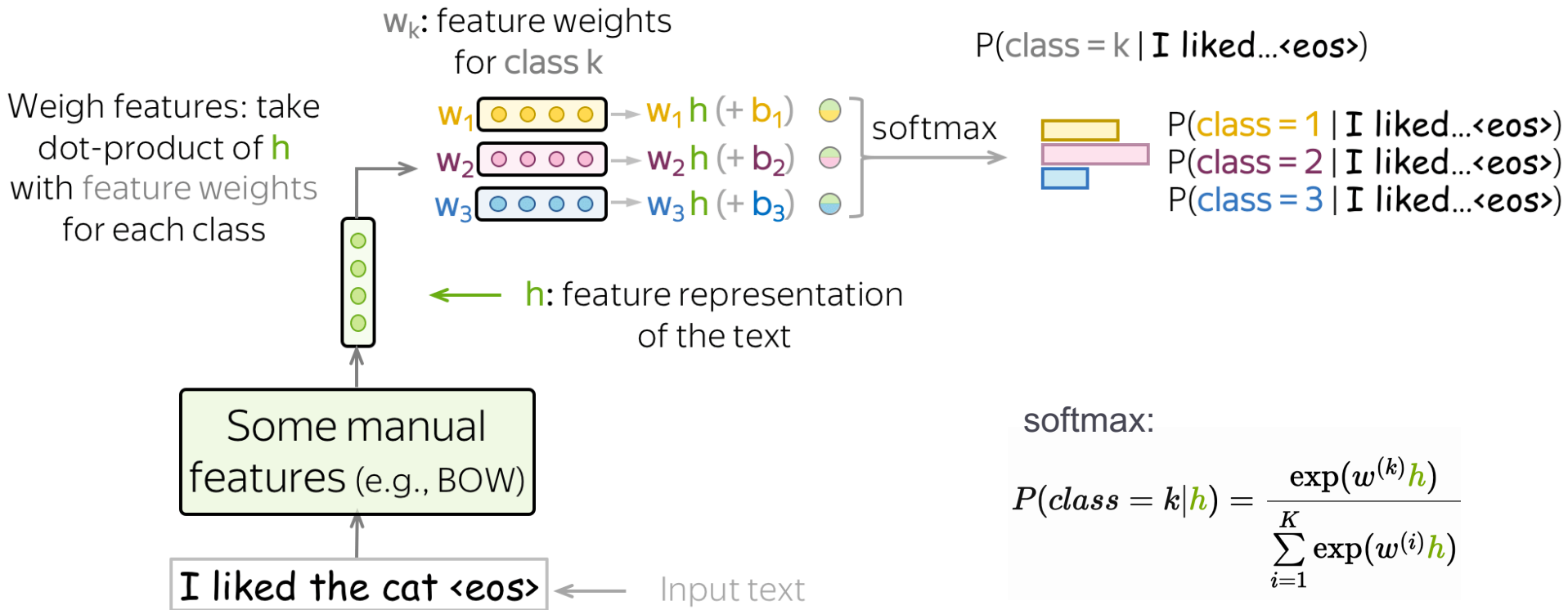
Note: here, we are using an equivalent but notationally different definition of the logistic regression

Logistic regression



Note: here, we are using an equivalent but notationally different definition of the logistic regression

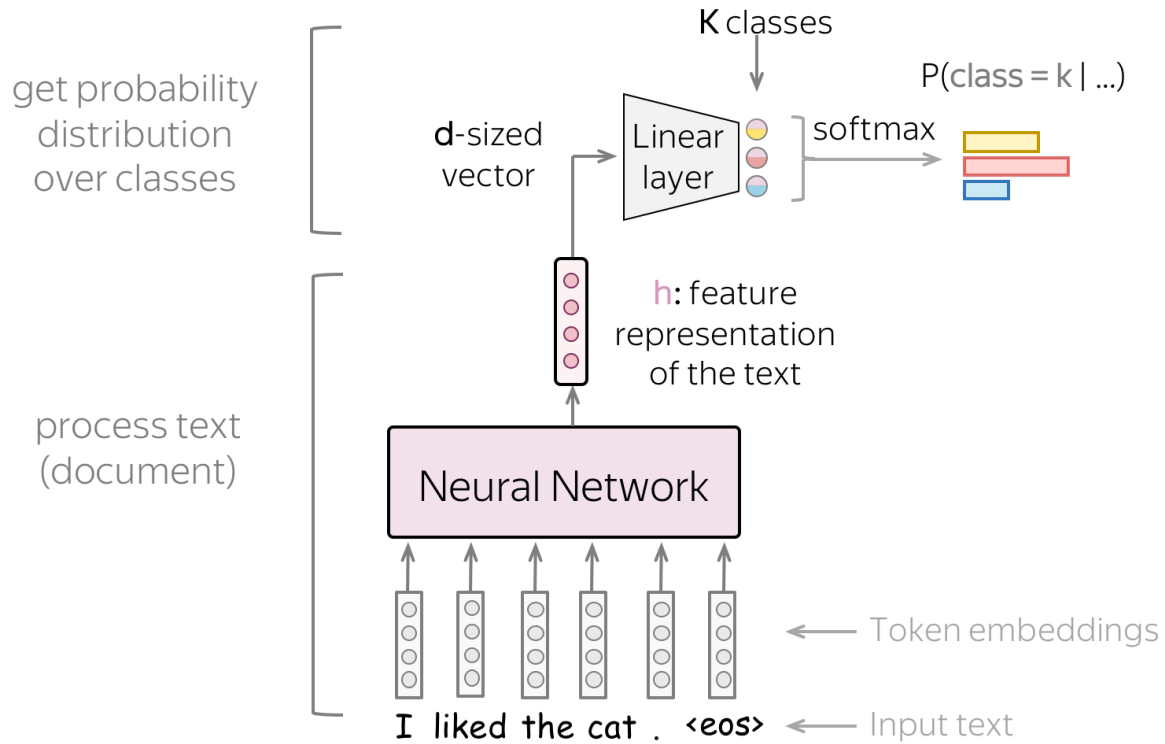
Logistic regression



Note: here, we are using an equivalent but notationally different definition of the logistic regression

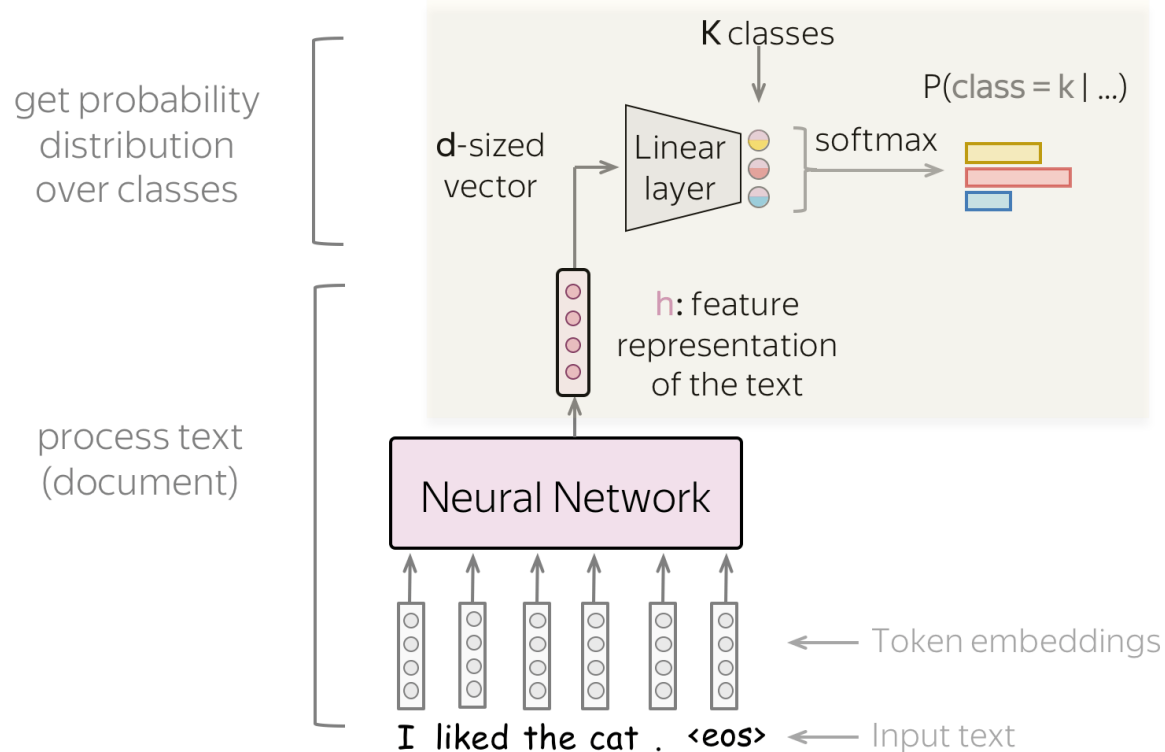
NN Classifier

Classification with Neural Networks



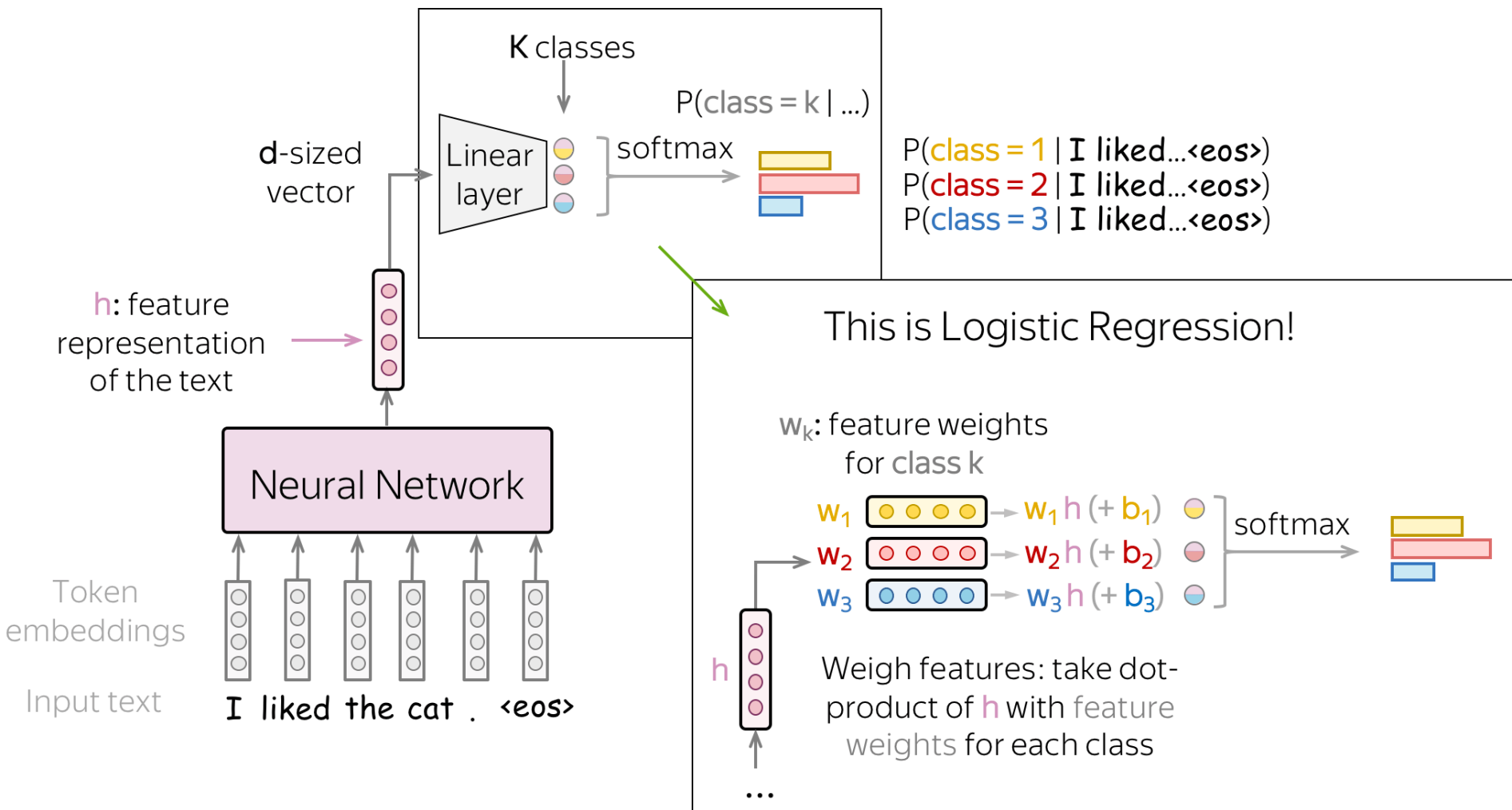
NN Classifier

Classification with Neural Networks

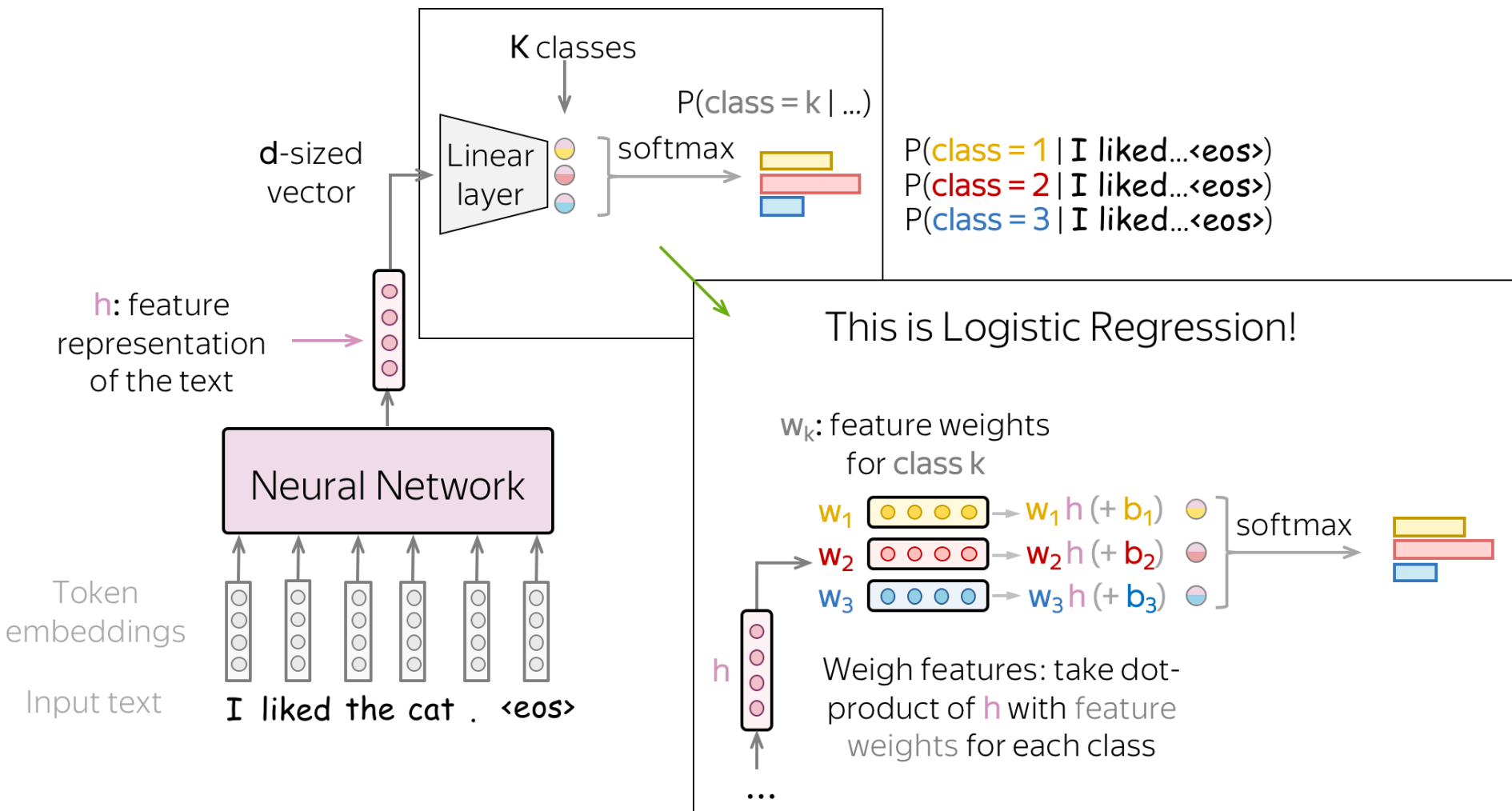


The highlighted part is the logistic regression!

NN Classifier

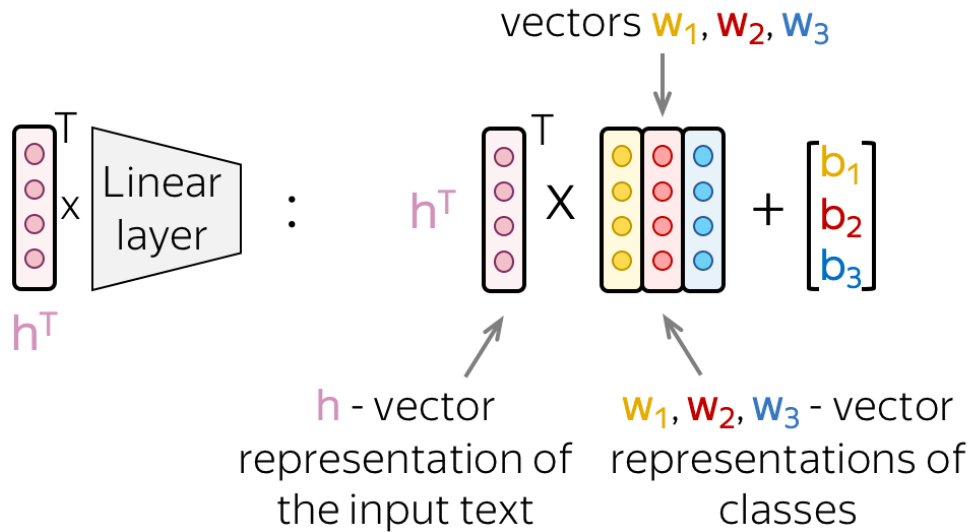


NN Classifier

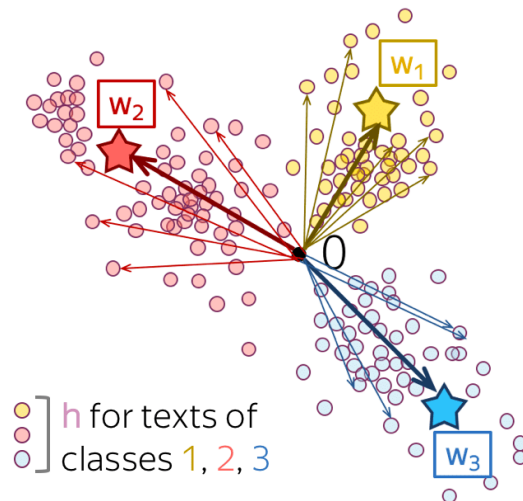


Intuition: the representation of the document points in the direction of the class representation

Representation of the document



Intuition: the representation of the document points in the direction of the class representation



What do we optimize? (recap)

Optimize conditional log-likelihood, as with logistic regression, which is equivalent to using cross-entropy loss

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

Label: **k**
↑
target

Model prediction:

$P(\text{class} = i | \text{I liked...<eos>})$



Target:

p^*



Cross-entropy loss:

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

For one-hot targets, this is equivalent to

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

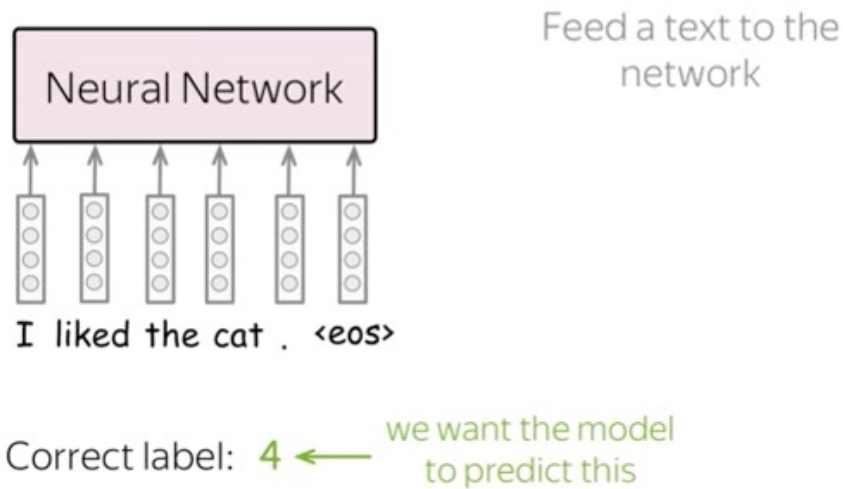
The target distribution is one-hot:

$$p^* = (0, \dots, 0, 1, 0, \dots)$$

Recall: we derived the gradient, and observed some problems

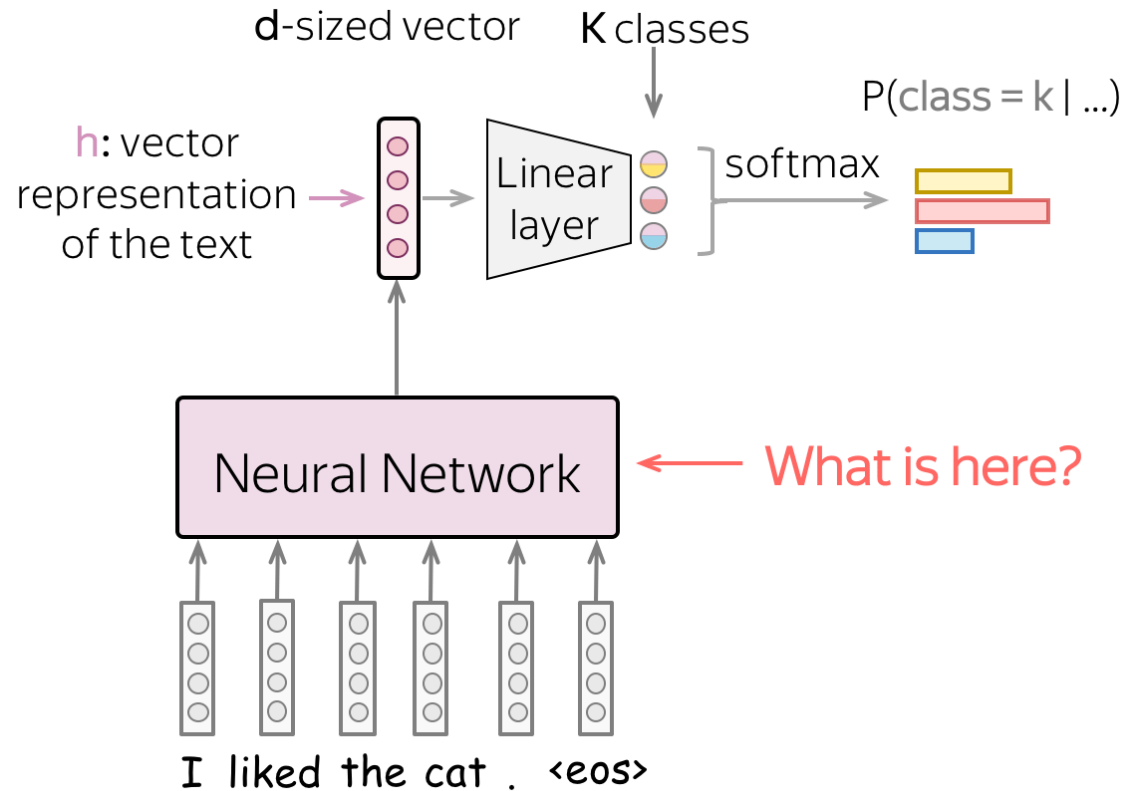
What do we optimize?

Optimize conditional log-likelihood, as with logistic regression



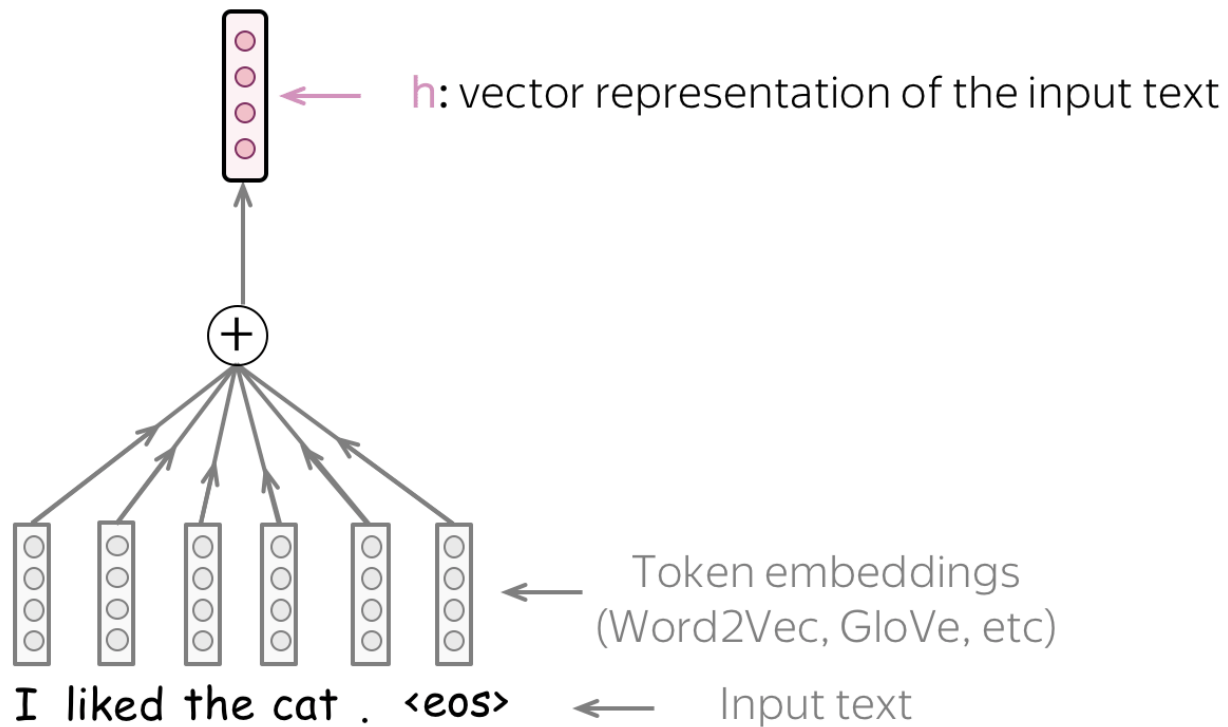
(video, not visible in pdf)

Neural models for text classification



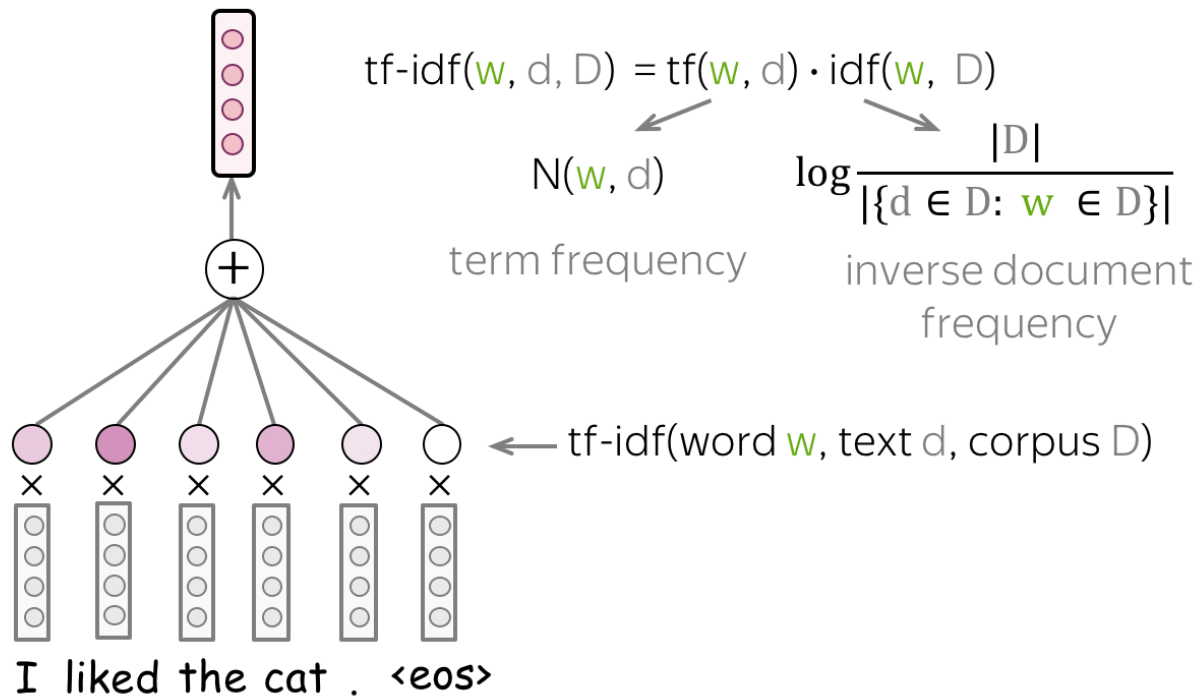
Basic models: bags of words (= Embeddings)

Sum of embeddings
(Bag of Words, Bag of Embeddings)



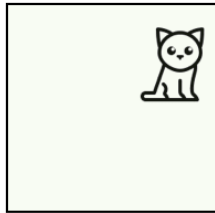
Basic models: bags of words (= Embeddings)

Weighted sum of embeddings
(e.g., using tf-idf weights)

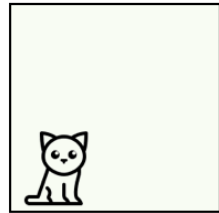


Convolutional Neural Networks

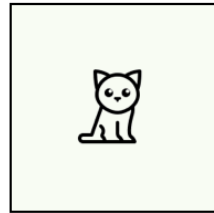
Translational invariance



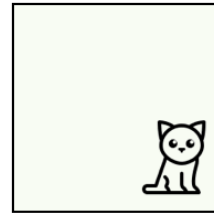
Label: **cat**



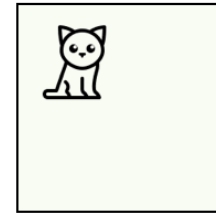
Label: **cat**



Label: **cat**



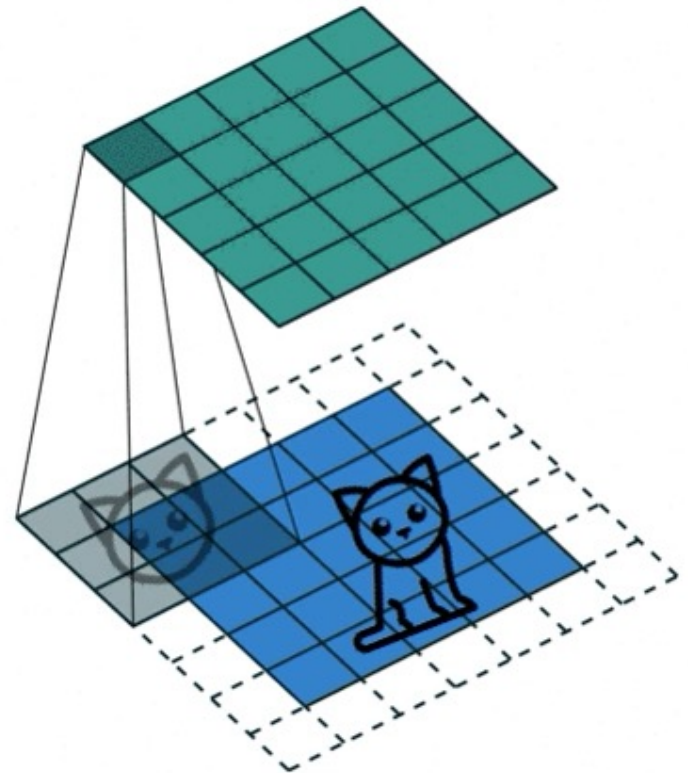
Label: **cat**



Label: **cat**

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

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



(video, not visible in pdf), taken from
https://github.com/vdumoulin/conv_arithmetic

CNNs for Text

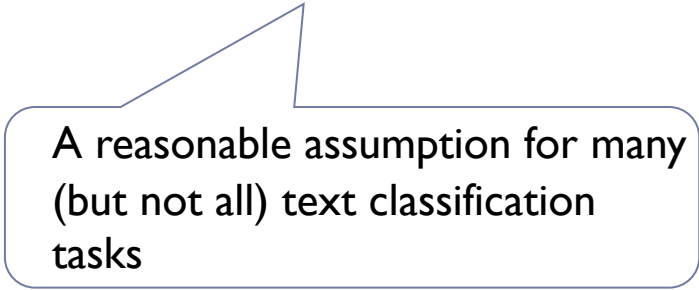
- Is Translational Invariance (and CNNs) an appropriate assumption (or inductive bias) for language tasks?
- It is still trickily better than BoW!

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**.

If a clue is very informative,
maybe we don't care much
where in a text it appears?

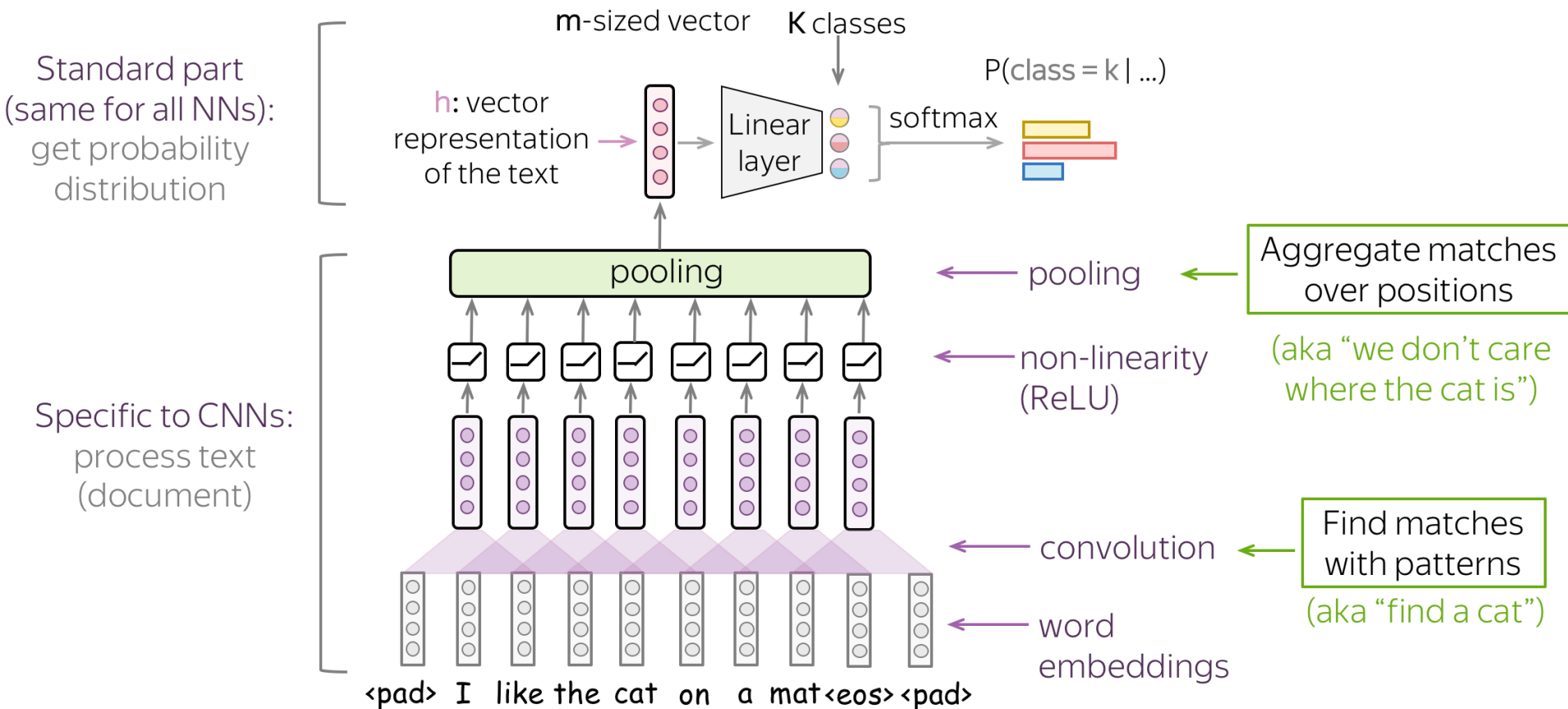


A reasonable assumption for many
(but not all) text classification
tasks

CNNs architectures

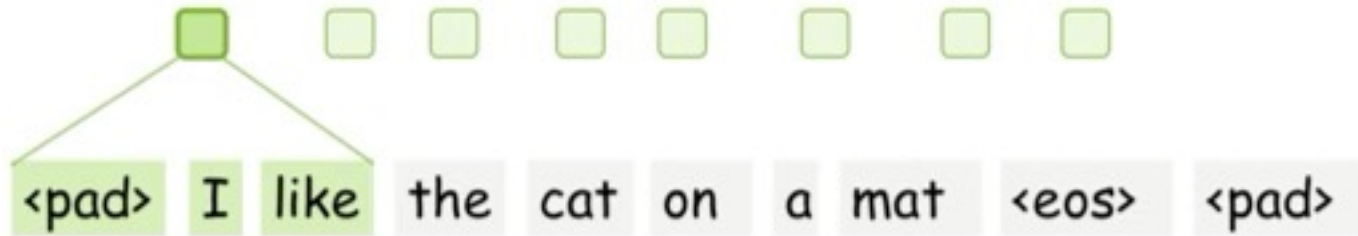
CNN architectures (including for text) consist of blocks

- convolutions: detect matches of patterns
- pooling: aggregate these matches across the positions

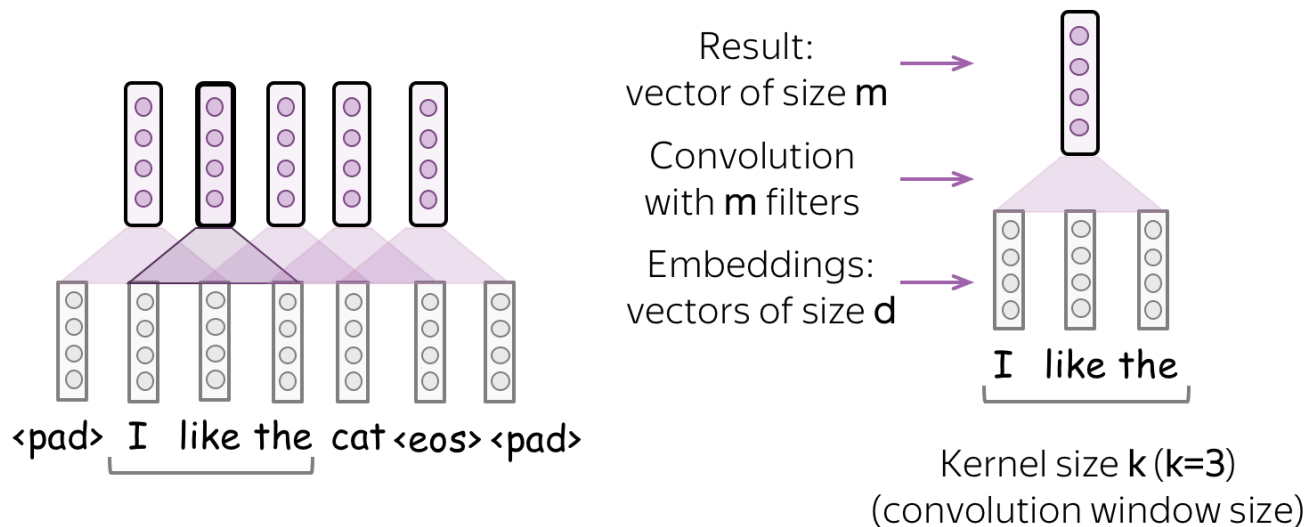


Convolutions for text (1d convolutions)

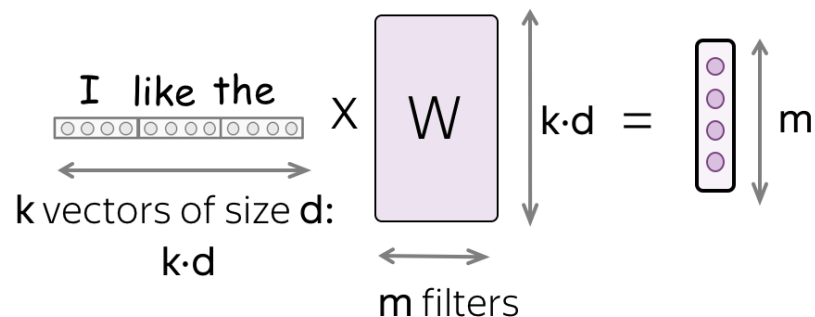
Scan over text and record where you detected matches



Convolutions for text (1d convolutions)

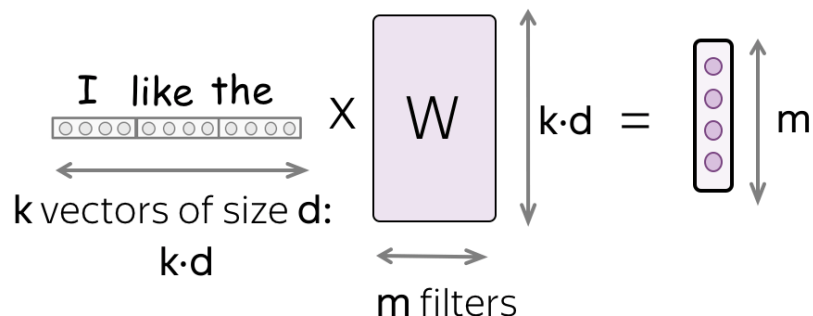


Convolution is a linear layer mapping from $k \cdot d$ to m



Convolutions are linear layers

Convolution is a linear layer mapping from $k \cdot d$ to m



- (x_1, \dots, x_n) - representations of the input words, $x_i \in \mathbb{R}^d$;
- d (input channels) - size of an input embedding;
- k (kernel size) - the length of a convolution window (on the illustration, $k = 3$);
- m (output channels) - number of convolution filters (i.e., number of channels produced by the convolution).

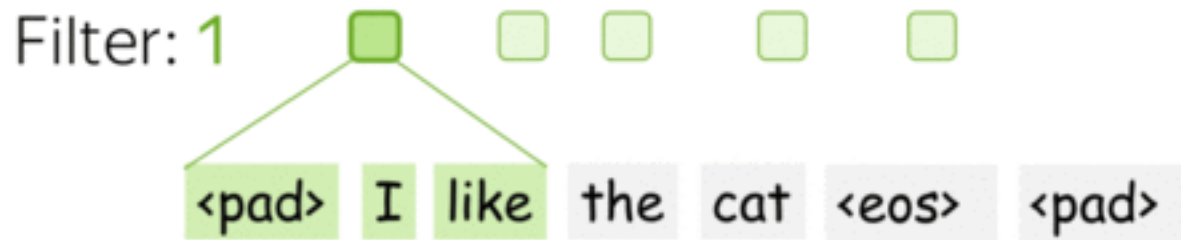
$$u_i = [x_i, \dots, x_{i+k-1}] \in \mathbb{R}^{k \cdot d}$$

A representation of the window (a long vectors)

$$F_i = u_i \times W.$$

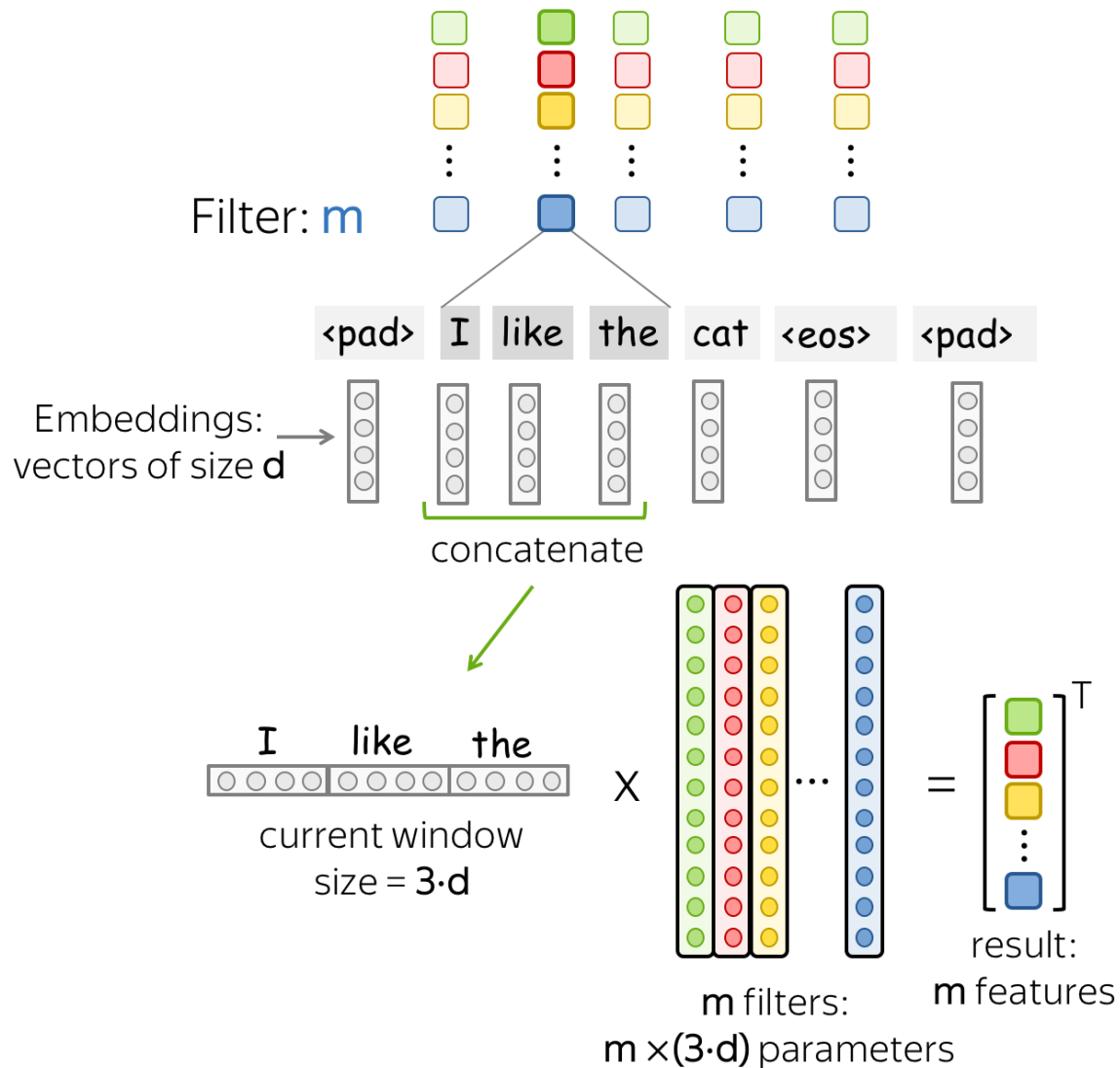
Multiplied by the convolution matrix to produce a feature vector for the window

Individual filters are feature extractors

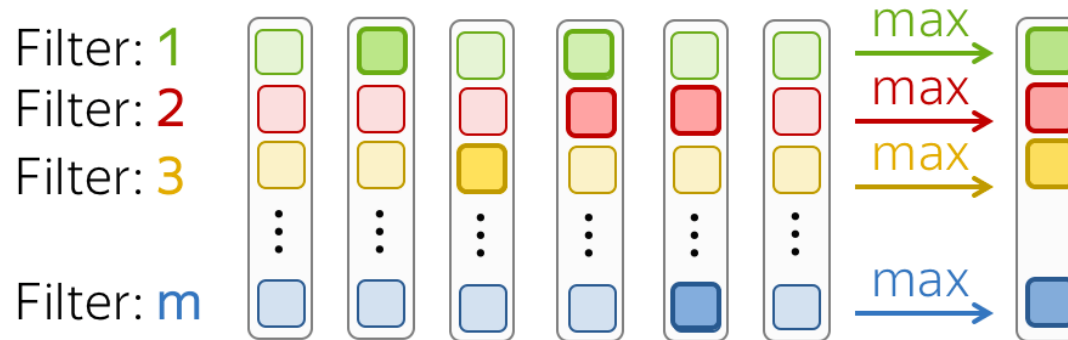


(video, not visible in pdf)

Convolution layer for text



Pooling

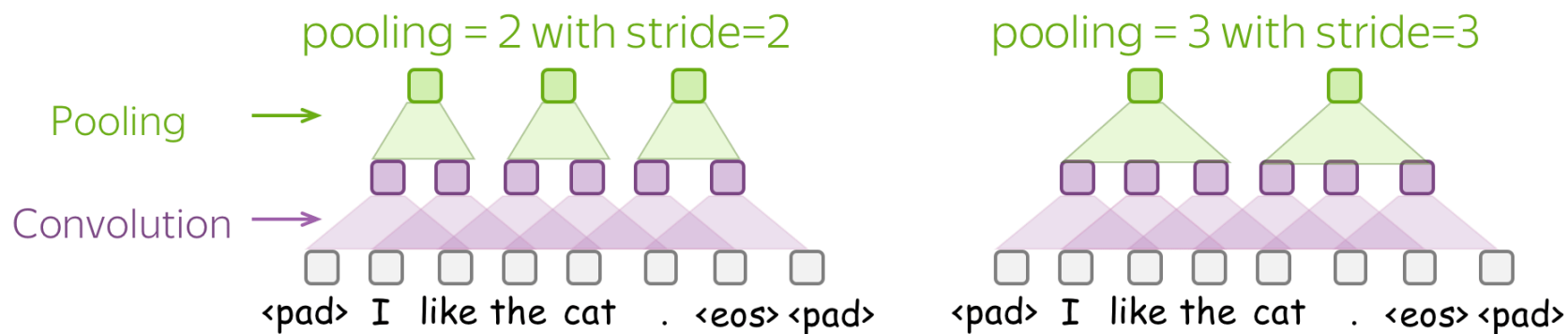


Max pooling:
maximum for each
dimension (feature)



Mean pooling uses averages instead averages

General Pooling

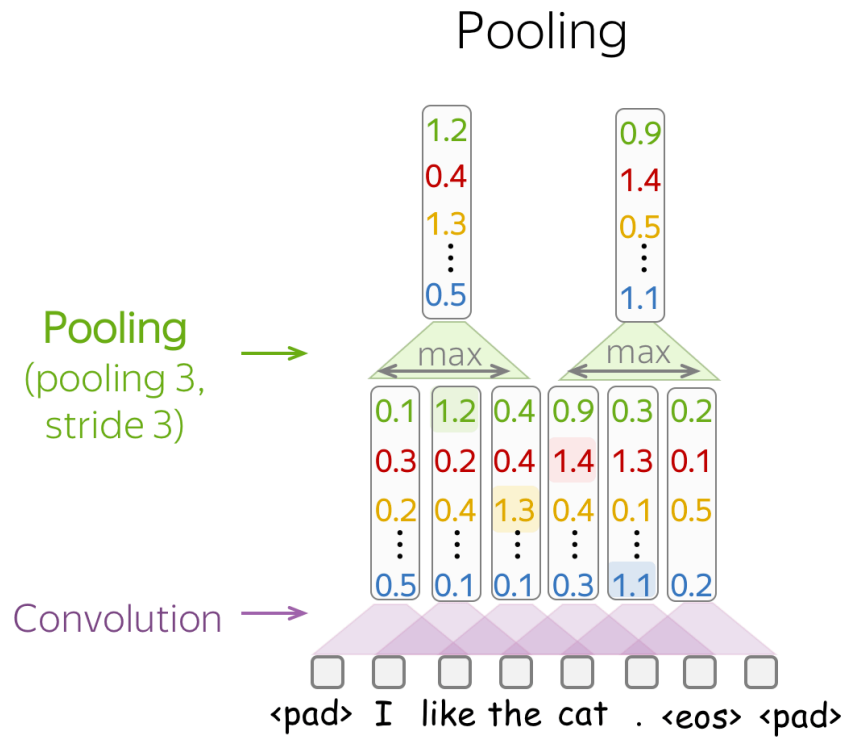


Parameters:

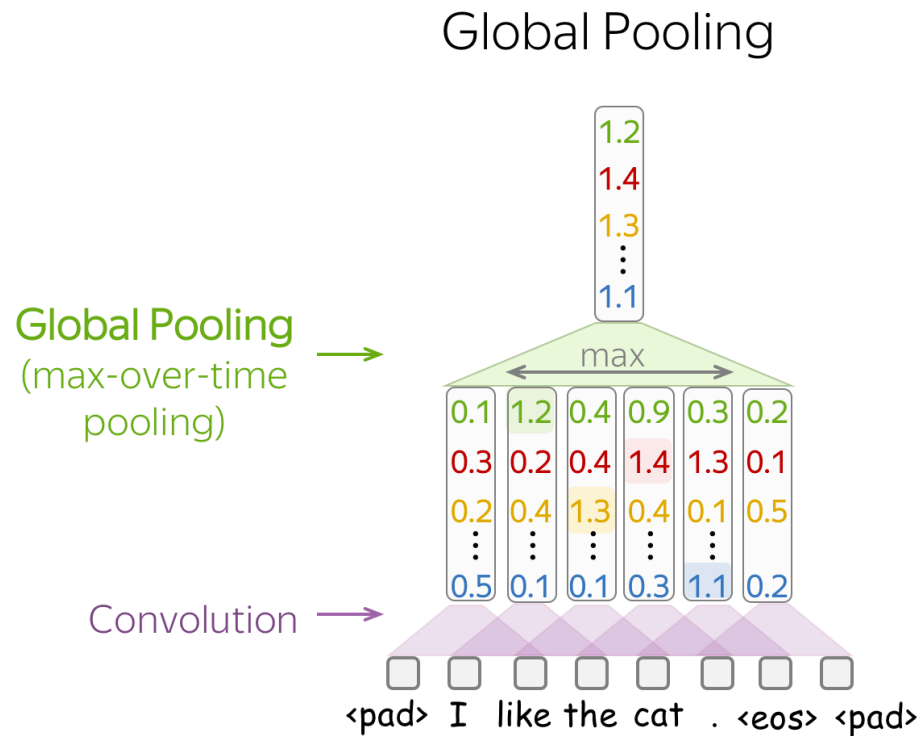
- **pooling:** pool 'size'
- **stride:** shift to the next pool operations

Stride should not be bigger than *pooling* (often equal)

General Pooling

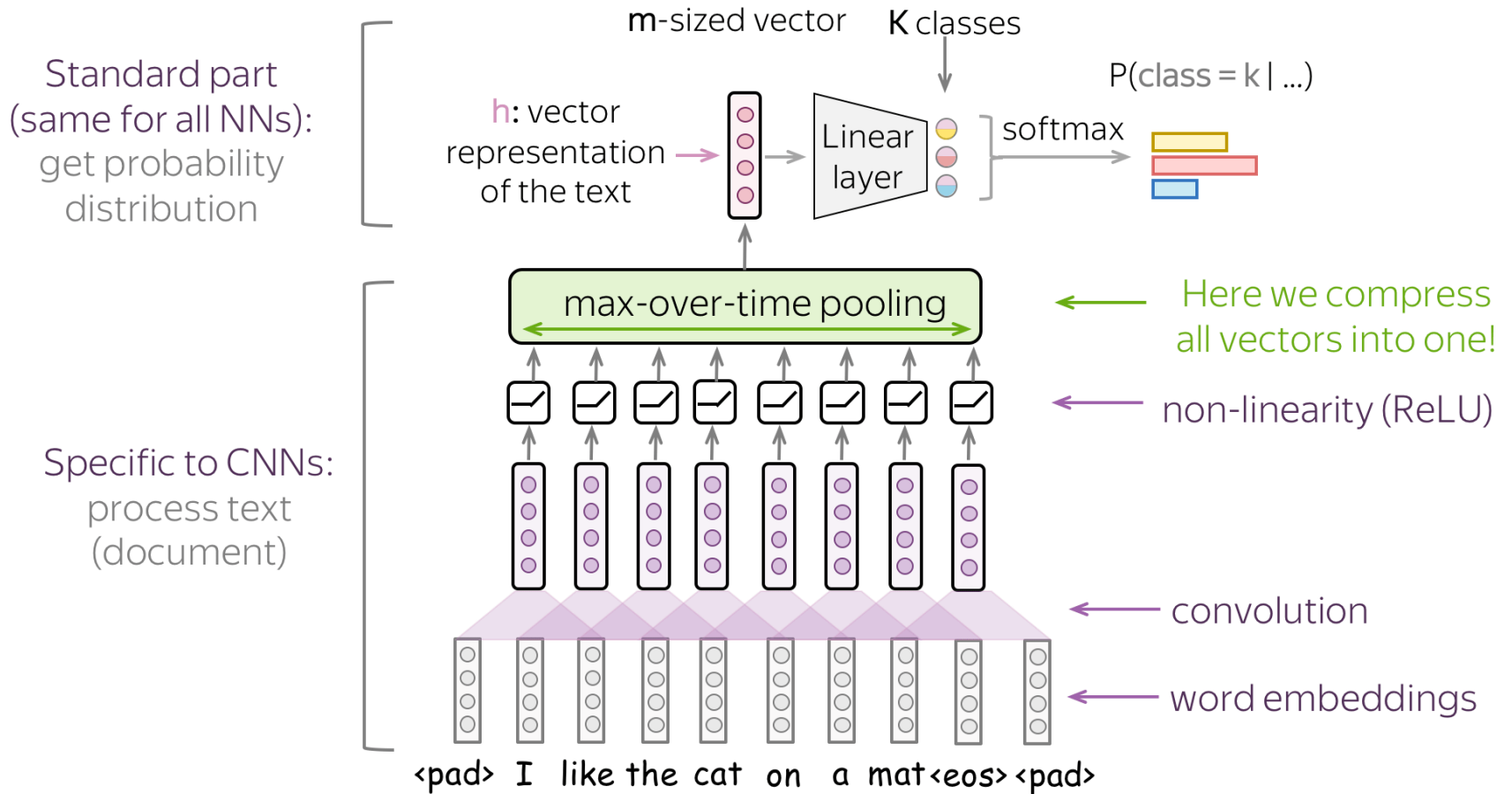


Global Pooling

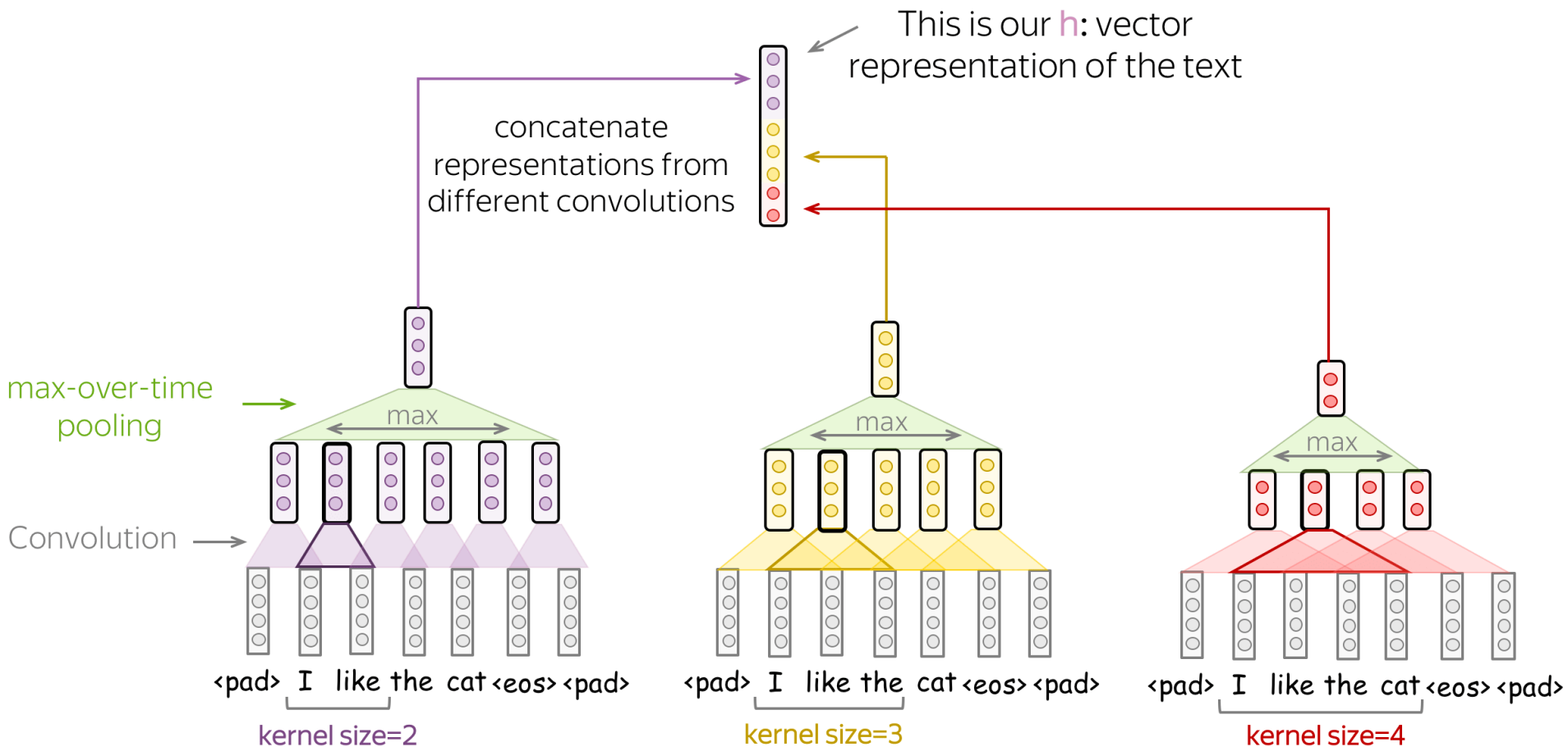


Gives us a single vector per example as needed for classification

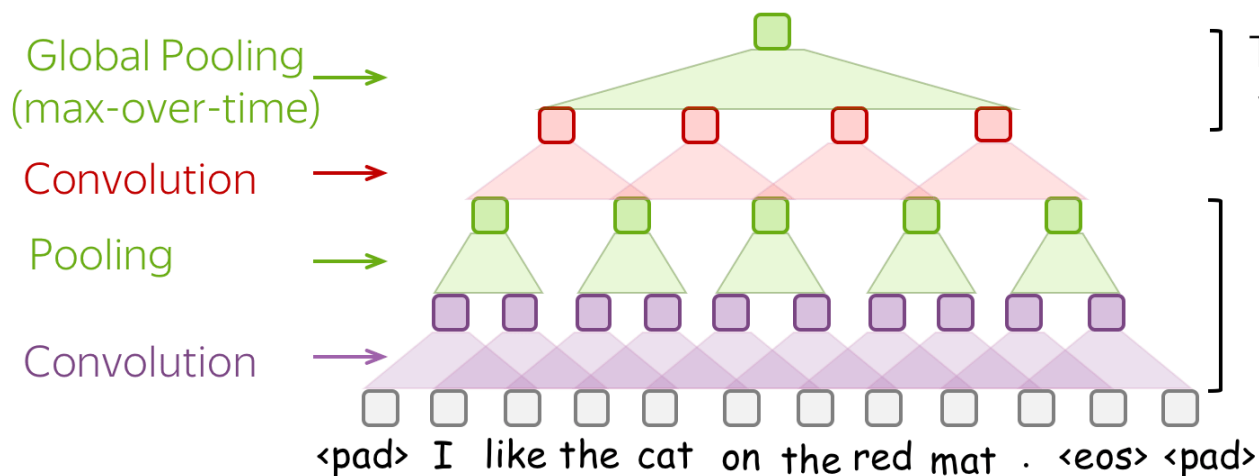
CNN architecture for text



Practical tricks: using multiple kernel sizes



Practical tricks: multiple layers



When crucial to have multiple layers?

- Long documents
- Models which start with character embeddings rather than words
- In principle, complex tasks, requiring modeling interaction between patterns

Interpretability: what are CNNs learning?

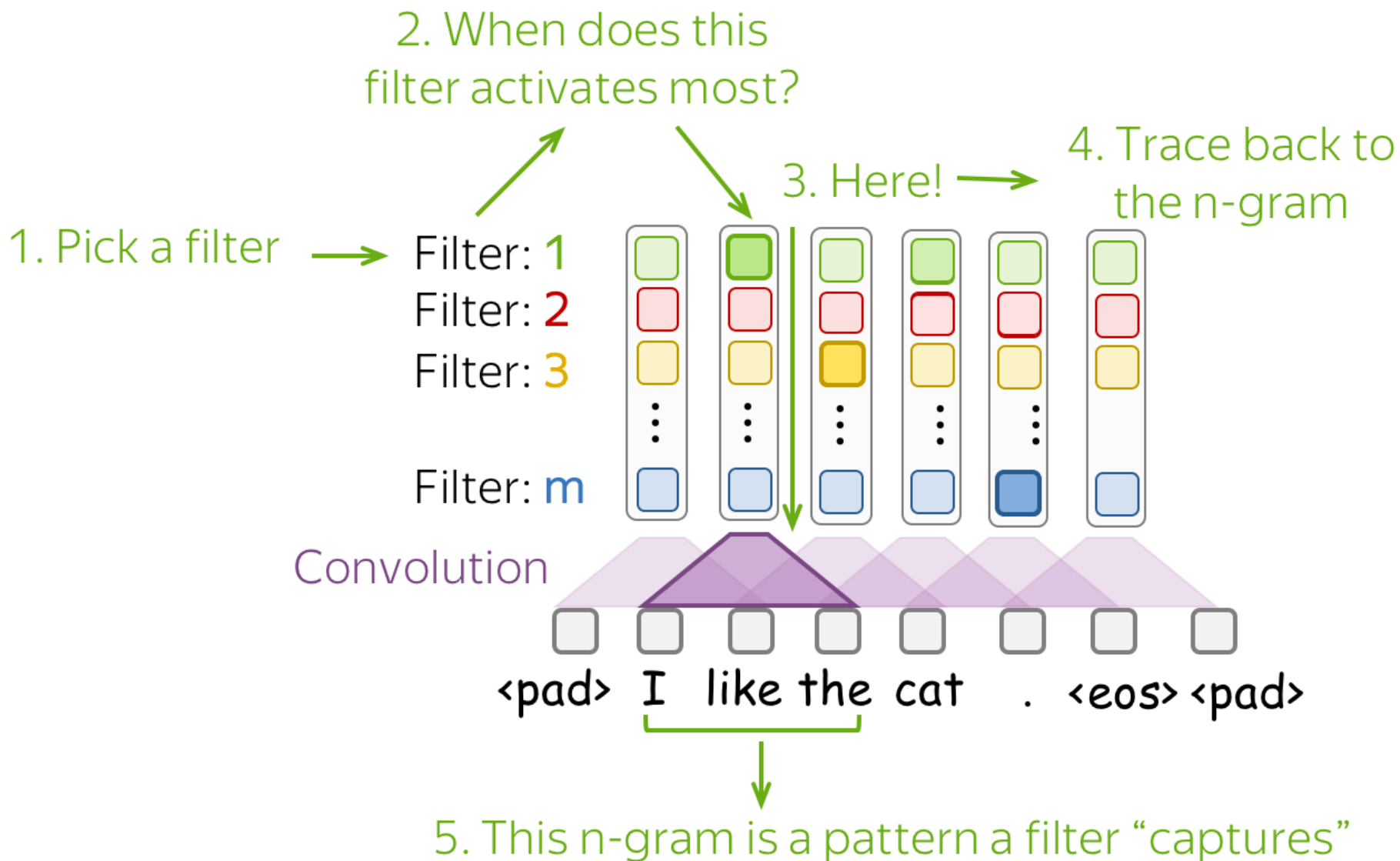
It is informative / interesting to understand what **individual CNN filters capture**, in different layers

CNN filters in image processing models:



Can we get something like this for NLP models?

Interpretability: what are CNNs learning?



Filter activations

filter	Top n-gram	Score
1	poorly designed junk	7.31
2	simply would not	5.75
3	a minor drawback	6.11
4	still working perfect	6.42
5	absolutely gorgeous .	5.36
6	one little hitch	5.72
7	utterly useless .	6.33
8	deserves four stars	5.56
9	a mediocre product	6.91

Top n-grams for filter 4		Score
1	still working perfect	6.42
2	works - perfect	5.78
3	isolation proves invaluable	5.61
4	still near perfect	5.6
5	still working great	5.45
6	works as good	5.44
7	still holding strong	5.37

A filter activates for a family of n-grams with similar meaning

You can draw parallels with logistic regressions, relying on ngrams
What are the key differences?

Summary

Text classification with neural networks

- Generalization of logistic regression
- Easy to integrate embeddings, estimated on unlabeled text
- BoW models, weighted BoW models, CNNs

Not exactly true for multilayered CNNs

All these models model only limited interaction between nonadjacent expressions, how do we handle these?

Next time - Recurrent Neural Networks (RNNs)

Next week - Transformer models