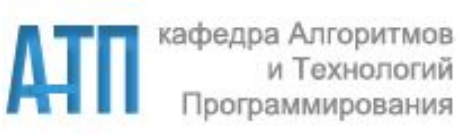


Text classification.
Basic Neural Networks at NLP

MIPT
24.02.2022
Anton Emelianov, Albina Akhmetgareeva

https://github.com/king-menin/mipt-nlp2022

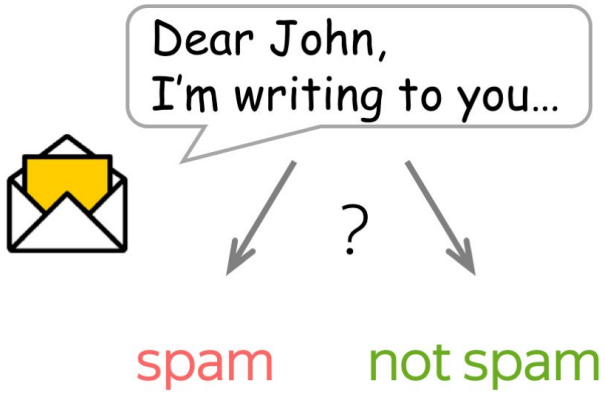
Task description

https://github.com/king-menin/mipt-nlp2022

Task description

- **Input:**

$x \in X - documents$
$y \in Y - classes/labels$



- **Output:**

Assign an unknown document to one of the classes - *binary classification*.

$$Select\ f(x) = y, y \in Y, Y = \{0, 1\}$$

Task description

- **Input:**

$x \in X - documents$
$y \in Y - classes/labels$

- **Output:**

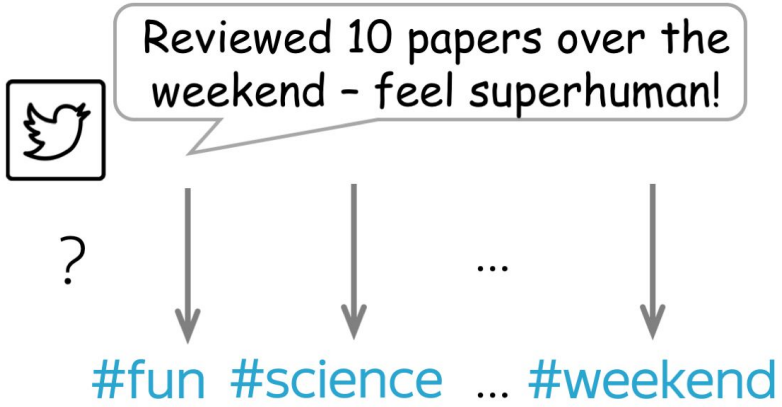
Assign an unknown document to one of the classes - *multiclass classification*.

$$Select\ f(x) = y, y \in Y, Y = \{0, 1, \ldots, K-1\}$$

Task description

- **Input:**

$x \in X - documents$
$y \in Y - classes/labels$



- **Output:**

Assign an unknown document to multiple classes - *multi-label classification*.

$$Select\ f(x) = \{y_0, \dots y_i, \dots\}, i < K, y \in Y, Y = \{0, 1, \dots, K-1\}$$

Task description

- We assume that we have a collection of documents with ground-truth labels. The input of a classifier is a document *x*

$$x = (x_0, x_1, \ldots, x_n)$$

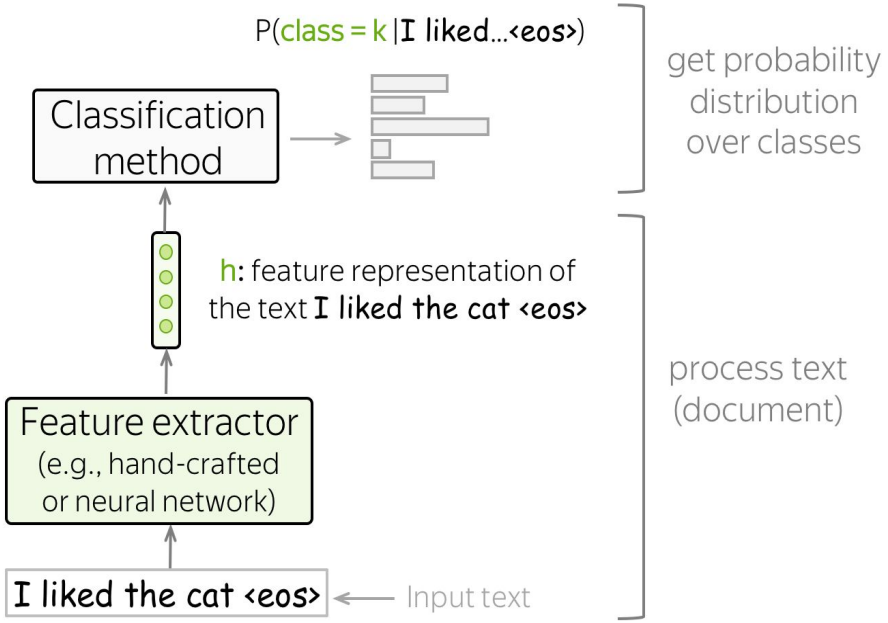
the output is a label:

$$y \in Y, Y = \{0, 1, \ldots, K-1\}$$

- We should select *f(x)=y*. *f* - our ML algorithm.

General classification framework

- Text classifiers have the following structure:
  - A **feature extractor** can be either manually defined (as in classical approaches) or learned (e.g., with neural networks).
  - A **classifier** has to assign class probabilities given feature representation of a text. The most common way to do this is using logistic regression, but other variants are also possible (e.g., Naive Bayes classifier or SVM).



Some methods for text classification

- Naive Bayes Classifier
- SVM
- Logistic Regression
- Neural Networks 🤗

General classification framework

- **Feature representation** of the input text:

$$h = (1, f_1, f_2, \ldots, f_n)$$

- Vectors with **feature weights** for each of the classes

$$w^{(k)} = (w_0^{(k)}, \ldots, w_n^{(k)}), k = 0, \ldots, K-1$$

- For each class, weight features, i.e. take the **dot product** of feature representation *h* with feature weights:

$$w^{(k)} h = w_0^{(k)} + w_1^{(k)} \cdot f_1 + \cdots + w_n^{(k)} \cdot f_n, k = 0, \ldots, K-1.$$

- Get **class probabilities** using softmax:

$$P(y = k|h) = \frac{\exp(w^{(k)} h)}{\sum_{i=1}^{K} \exp(w^{(i)} h)}.$$

General classification framework

- Define *h* as **function** of *x*, where *x* is the document from collection *X*:

$$y = \arg\max_k P(y = k|h) = \arg\max_k P(y = k|logits_k),$$

$$logits_k = w^{(k)} h(x)$$

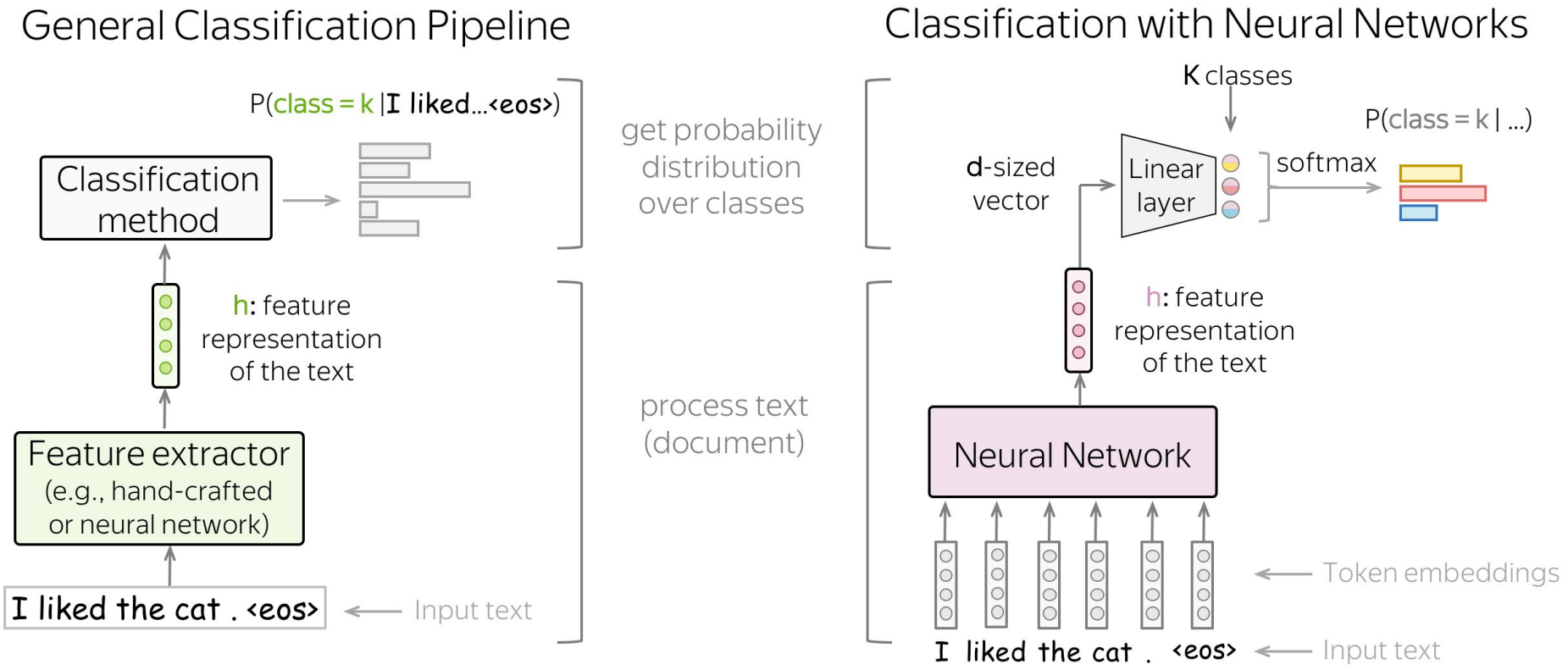
- Or at matrix form:

$$logits = Wh(x)$$

$$y = argmax(logits, dim = -1)$$

- Here *h(x)* is your **favorite** algorithm (include neural nets)!
- Function *h(x)* is **feature extractor**.
- Also *h(x) generates* **representation** of text *x.*

General classification framework



Training

- Neural classifiers are trained to predict probability distributions over classes. Intuitively, at each step we maximize the probability a model assigns to the correct class.
- The standard loss function is the cross-entropy loss.

Cross-entropy loss for the target probability distribution:

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

(1 for the target label, 0 for the rest) and the predicted by the model distribution:

$$p = (p_1, \ldots, p_K), p_i = p(i|x)$$

- And loss:

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{K} p_i^* \log(p_i).$$

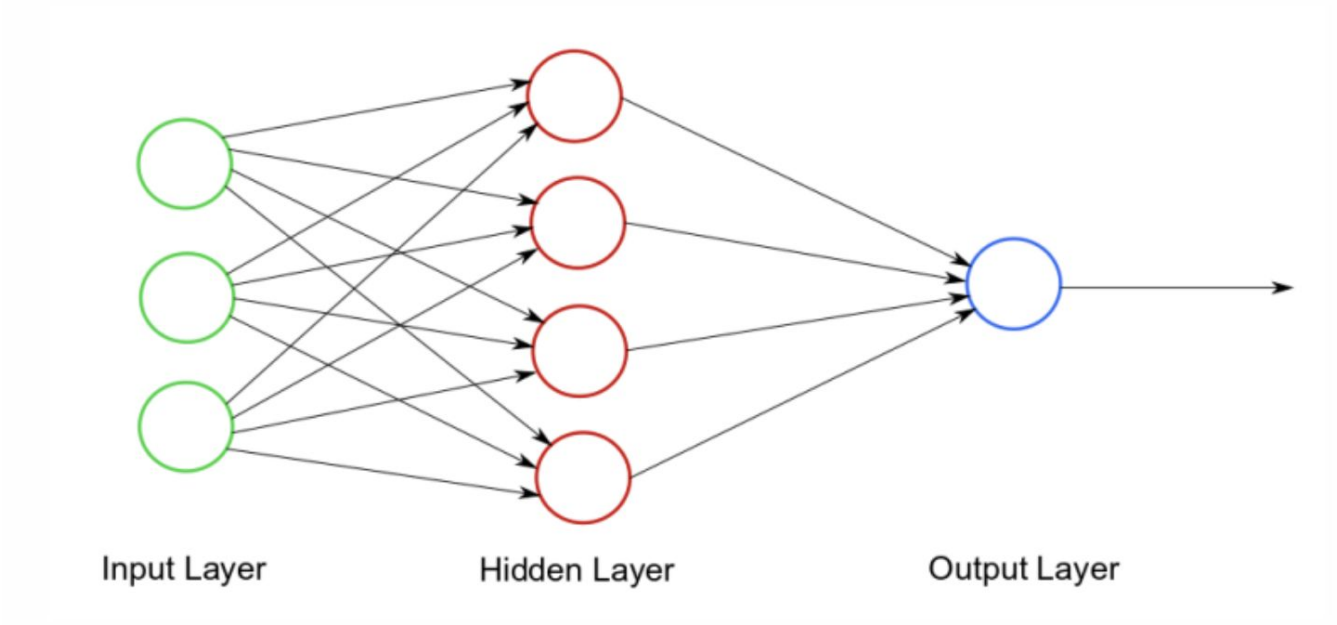
Basic Neural networks for classification

2022

https://github.com/king-menin/mipt-nlp2022

Networks with one hidden layer

- **Theorem (universal approximator)** Any continuous function on a compact can be uniformly approximated by a neural network with one hidden layer.

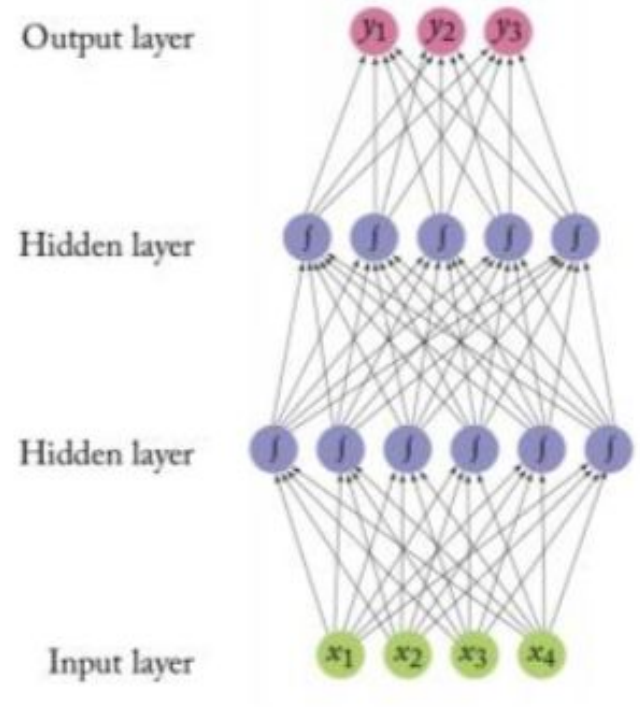


Multilayer feedforward networks

$$\mathrm{NN}_{\mathrm{MLP2}}(x) = y$$

$$h_1 = g^1(xW^1 + b^1)$$
$$h_2 = g^2(h^1W^2 + b^2)$$
$$y = h^2W^3$$

$$x \in \mathbb{R}^{d_{in}}, y \in \mathbb{R}^{d_{out}}$$
$$W^1 \in \mathbb{R}^{d_{in} \times d_1}, b^1 \in \mathbb{R}^{d_1}$$
$$W^2 \in \mathbb{R}^{d_1 \times d_2}, b^2 \in \mathbb{R}^{d_2}$$
$$W^3 \in \mathbb{R}^{d_2 \times d_{out}}$$

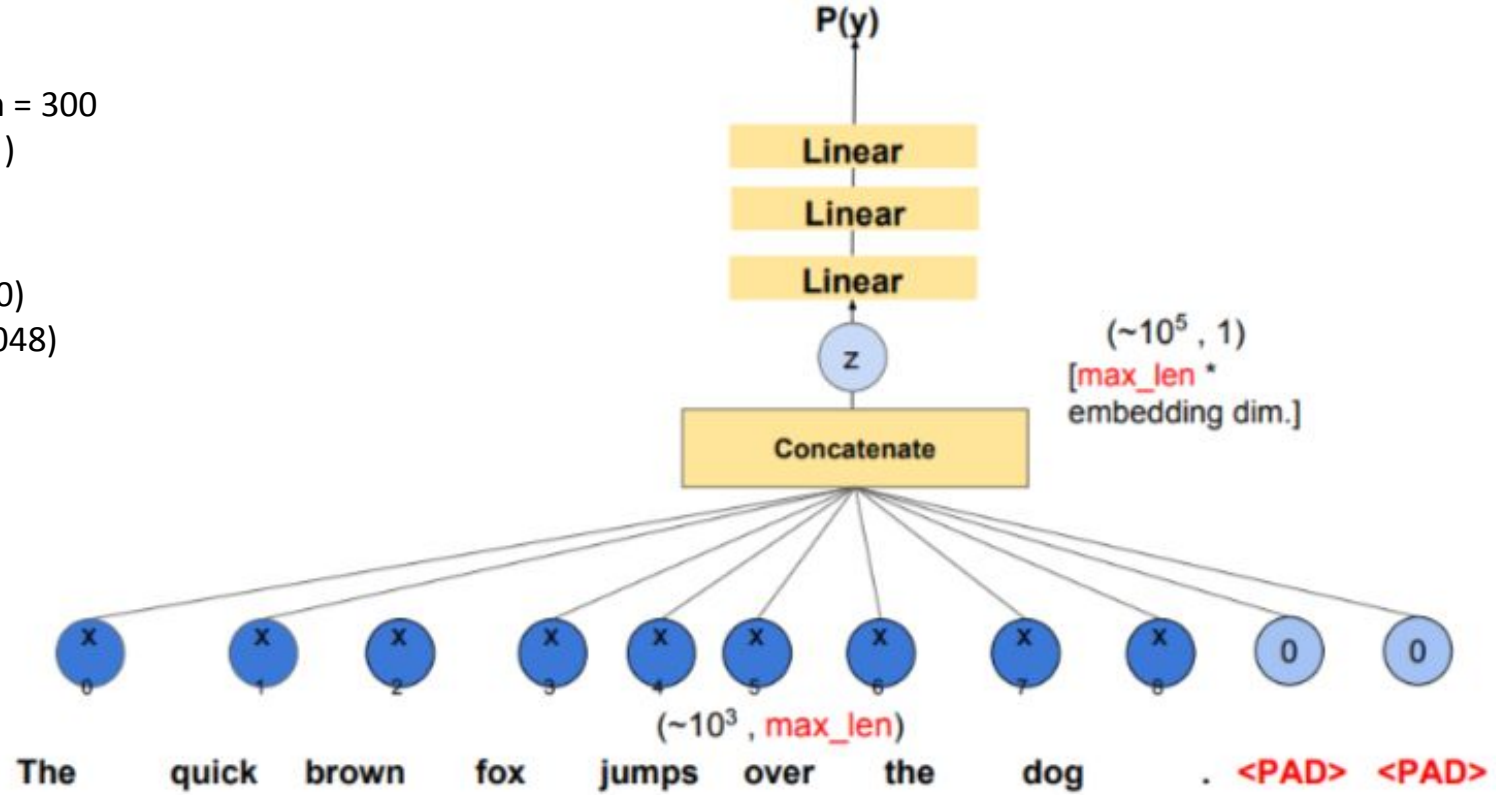


Multilayer perceptron

max_len = 2048 (set text len)

embed_dim = 300
(size of h(x))

X (dim=300)
W (dim=2048)



Recurrent Neural Networks (RNNs)

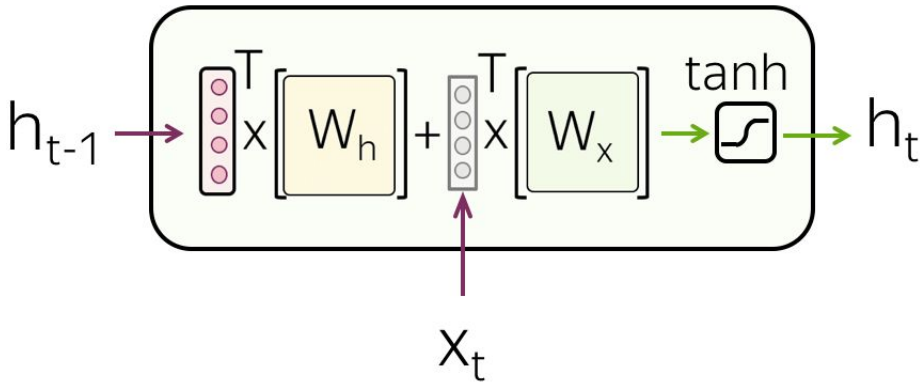
2022

https://github.com/king-menin/mipt-nlp2022

Vanilla RNN

- **RNN** reads a text token by token, at each step using a new token embedding and the previous state.
- **Note** that the RNN cell is the same at each step!
- **Vanilla RNN**, transforms *h(t-1) and x(t)* linearly, then applies a non-linearity (most often, the *tanh* function)

$$h_t = \tanh(h_{t-1}W_h + x_t W_t).$$



How to learn RNN?

- **Backpropogation Through Time**:

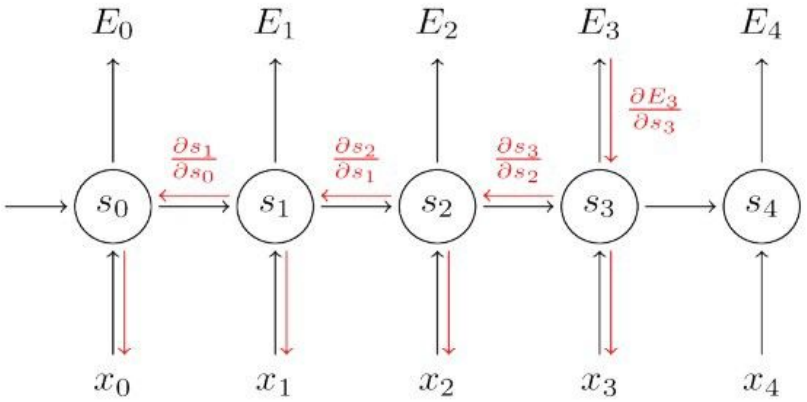
$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$

But $s_3 = \tanh(Ux_t + Ws_2)$

S_3 depends on s_2, which depends on W and s_1, and so on.

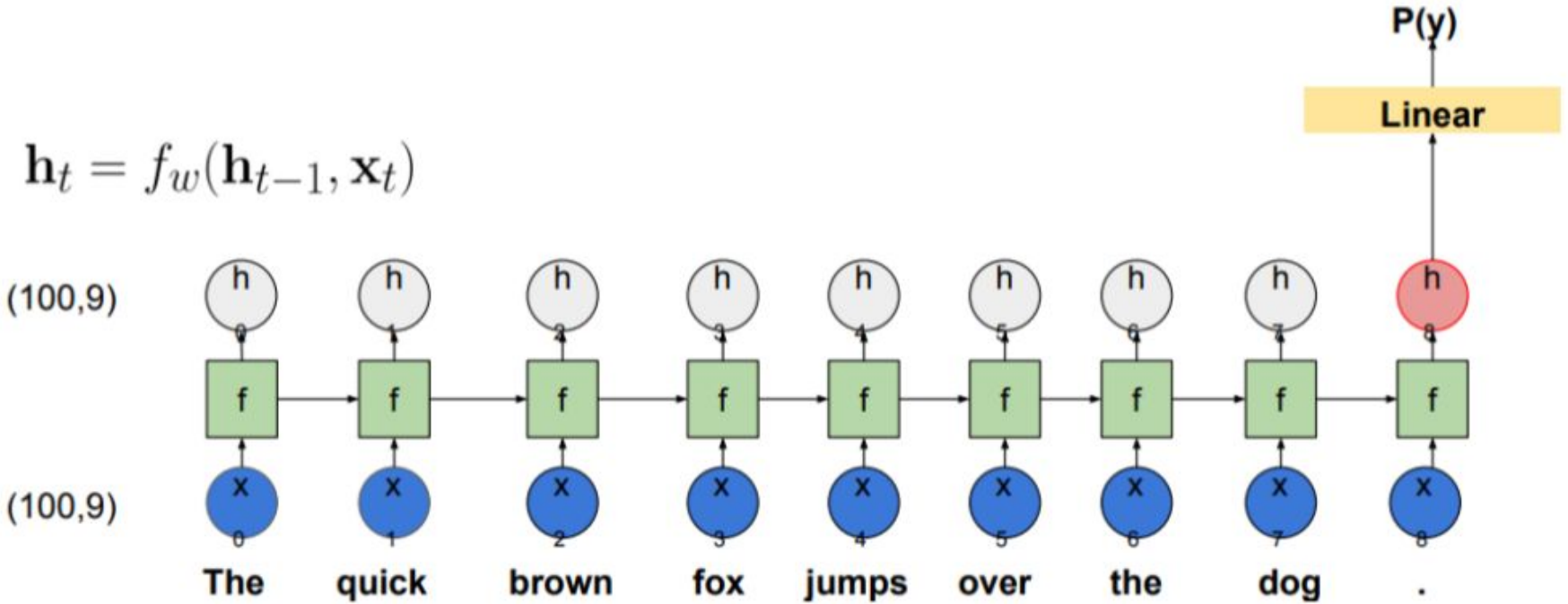
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$



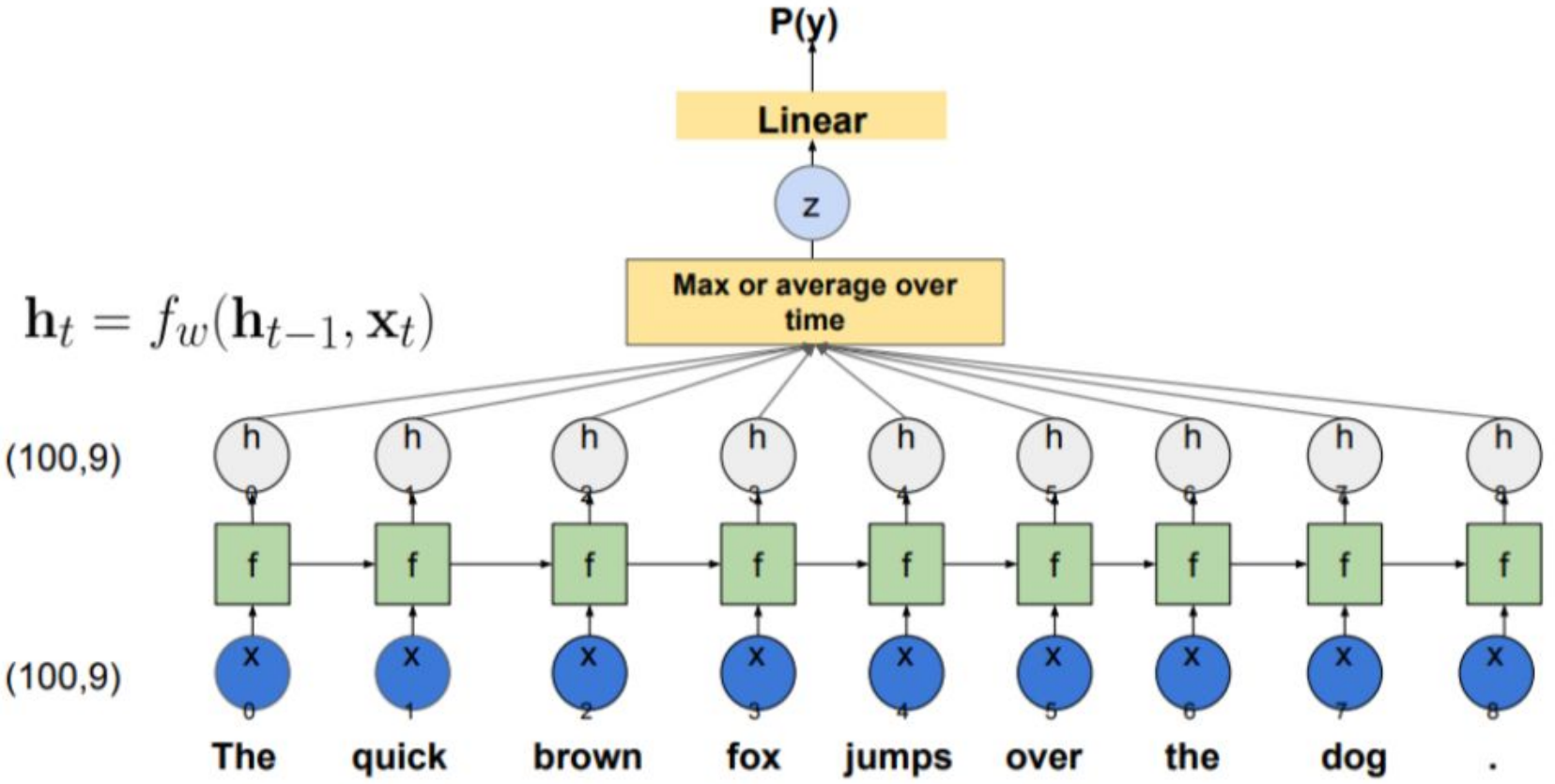
More details here

RNN for classification

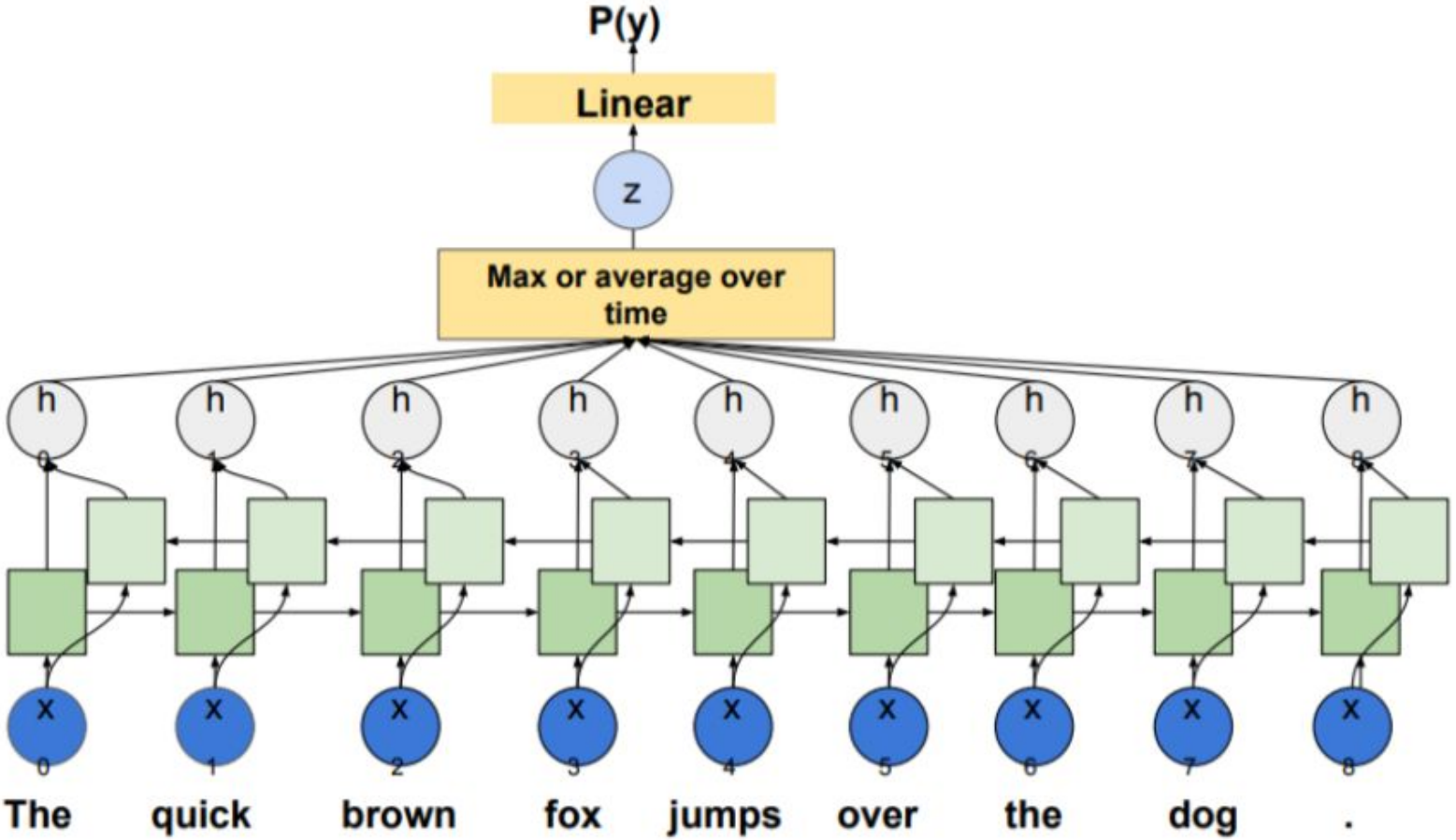
$$\mathbf{h}_t = f_w(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



RNN for classification



RNN for classification



RNN overview

- RNN is difficult to train:
  - vanishing gradient problem
  - the problem of fast forgetting
- Solution: guided neurons of a special type: LSTM and GRU.
- Other modifications: peephole lstm (2014), QRNN (2016), AWD LSTM (2017), Mogrifier LSTM (2019-2020).

Problems of Fully Connected Neural Networks

- **Problems:**
  - Vanishing/Exploding gradients

Problems of Fully Connected Neural Networks

- **Problems:**
  - Vanishing/Exploding gradients
  - Requires a huge number of neurons

Problems of Fully Connected Neural Networks

- **Problems:**
  - Vanishing/Exploding gradients
  - Requires a huge number of neurons
  - Overfitting

Problems of Fully Connected Neural Networks

- **Problems:**
  - Vanishing/Exploding gradients
  - Requires a huge number of neurons
  - Overfitting
  - Lack of translational invariance (weights are specific to the absolute coordinate of a word).

Problems of Fully Connected Neural Networks

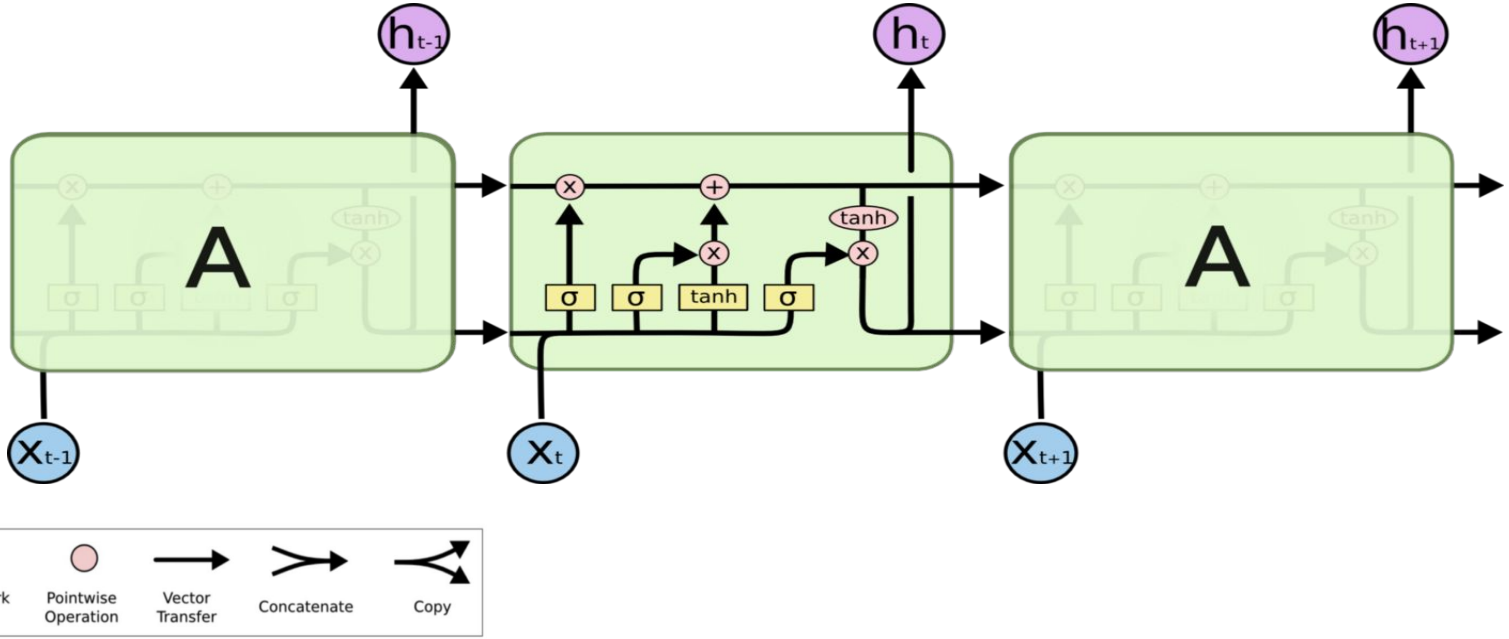
- **Problems:**
  - Vanishing/Exploding gradients
  - Requires a huge number of neurons
  - Overfitting
  - Lack of translational invariance (weights are specific to the absolute coordinate of a word).

Problems of Fully Connected Neural Networks

- **Problems:**
  - Vanishing/Exploding gradients
  - Requires a huge number of neurons
  - Overfitting
  - Lack of translational invariance (weights are specific to the absolute coordinate of a word).
- Possible **solution -** introduction of new types of layers:
  - Do not use Fully Connected NN
  - Dropout
  - Normalization

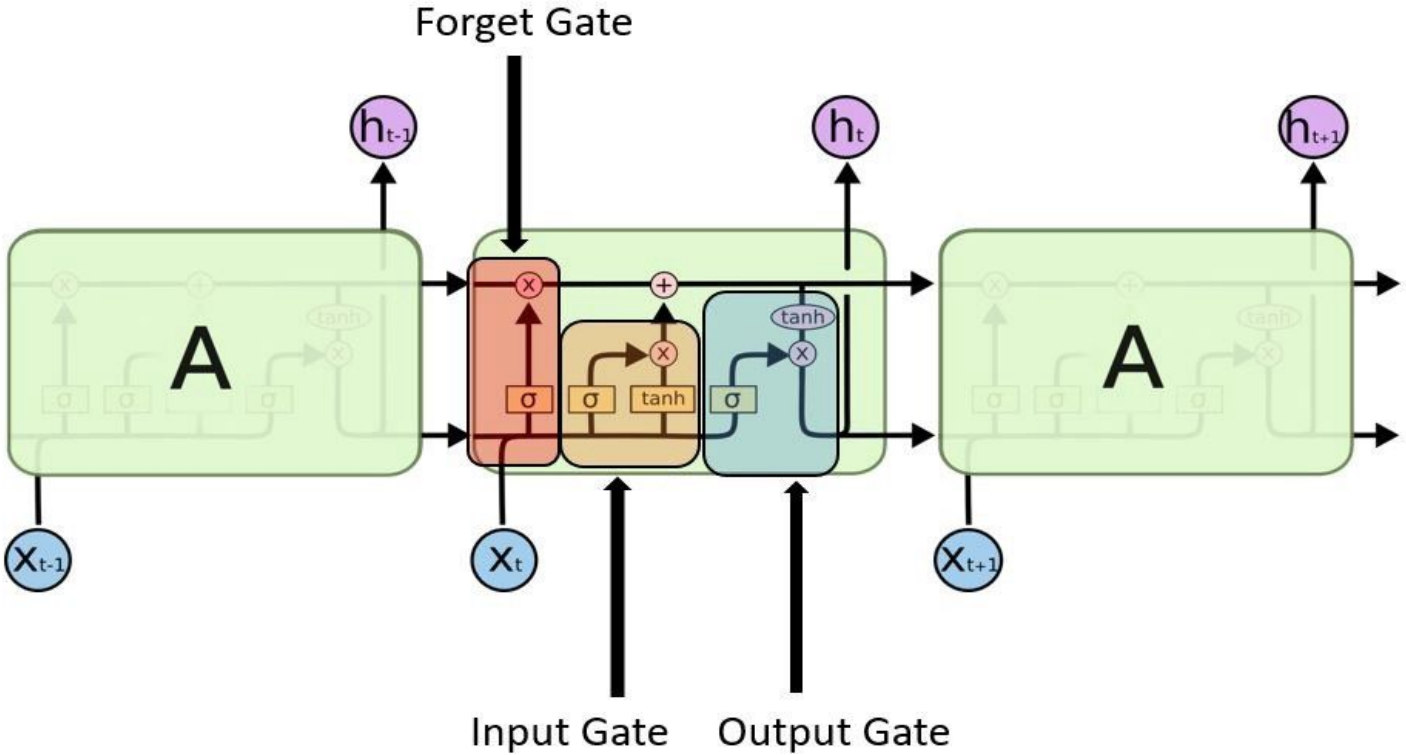
Long Short Term Memory (LSTM)

- A special kind of RNN's, capable of **Learning Long-term dependencies**.
- **LSTM's** have a Nature of Remembering information for a long periods of time is their Default behaviour.



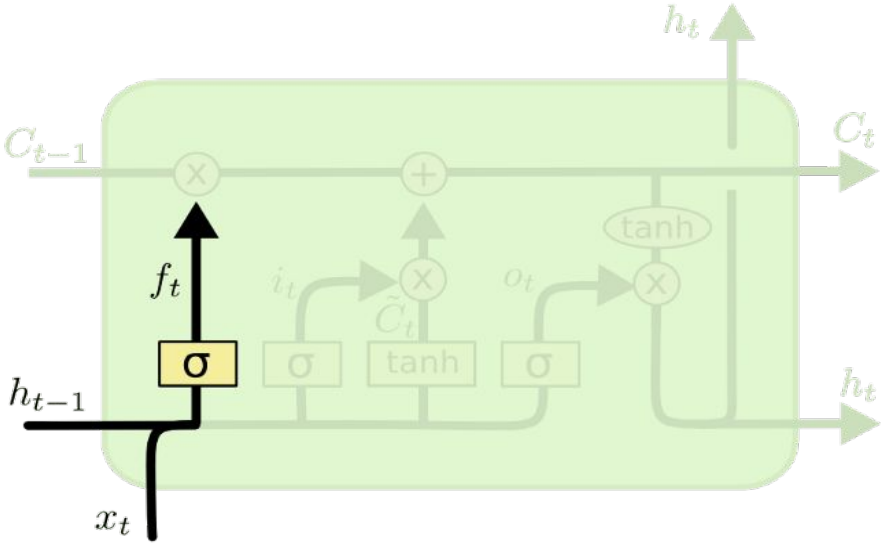
LSTM

- LSTM had a **three step** Process: **Every LSTM** module will have 3 gates named as **Forget gate**, I**nput gate**, **Output gate**.



LSTM

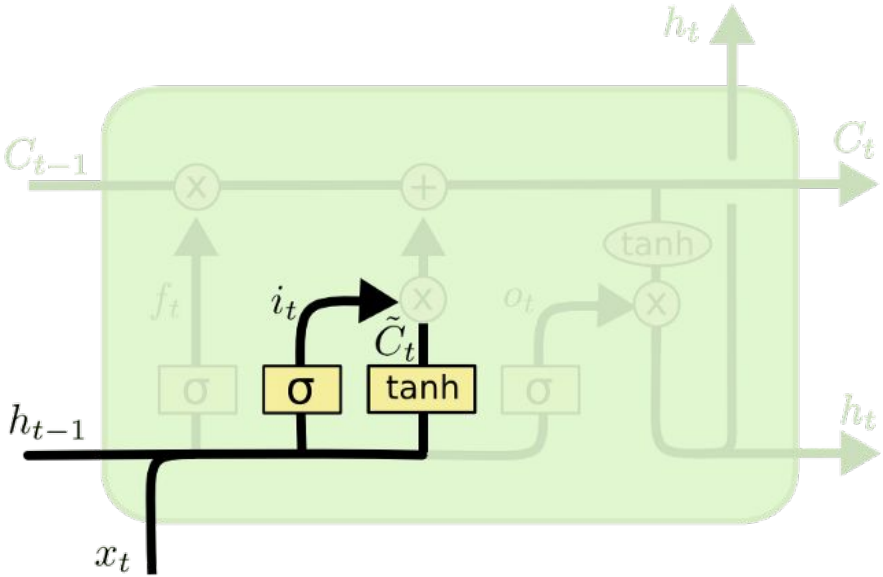
- **Forget Gate**
  - *Decides how much of the past you should remember.*
  - This gate Decides which information to be omitted in from the cell in that particular time stamp. It is decided by the **sigmoid function.** it looks at the previous state(**ht-1**) and the content input(**Xt**) and outputs a number between **0(***omit this*)and **1(***keep this***)**for each number in the cell state **Ct−1**.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM

- Update Gate/input gate:
  - Decides How much of this unit is added to the current state
  - Sigmoid function decides which values to let through **0,1**. and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from-**1** to **1**.

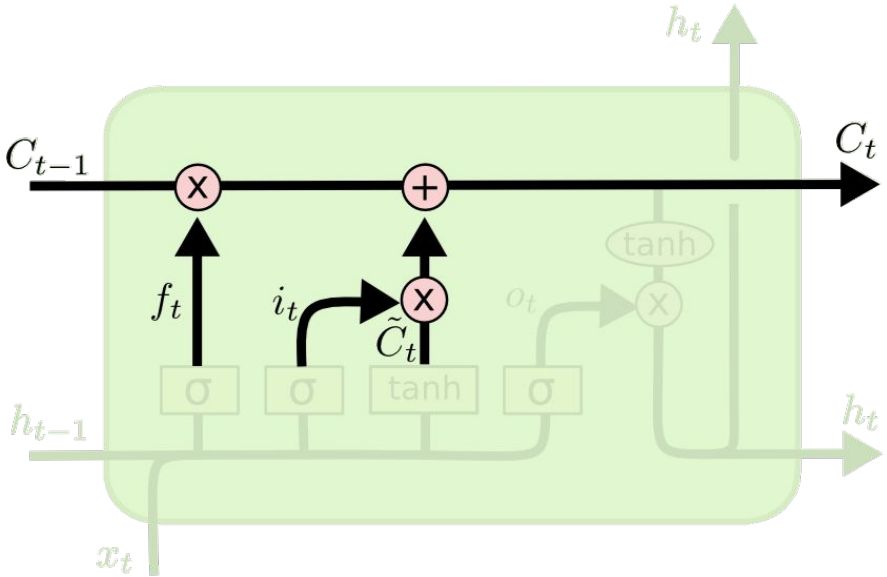


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM

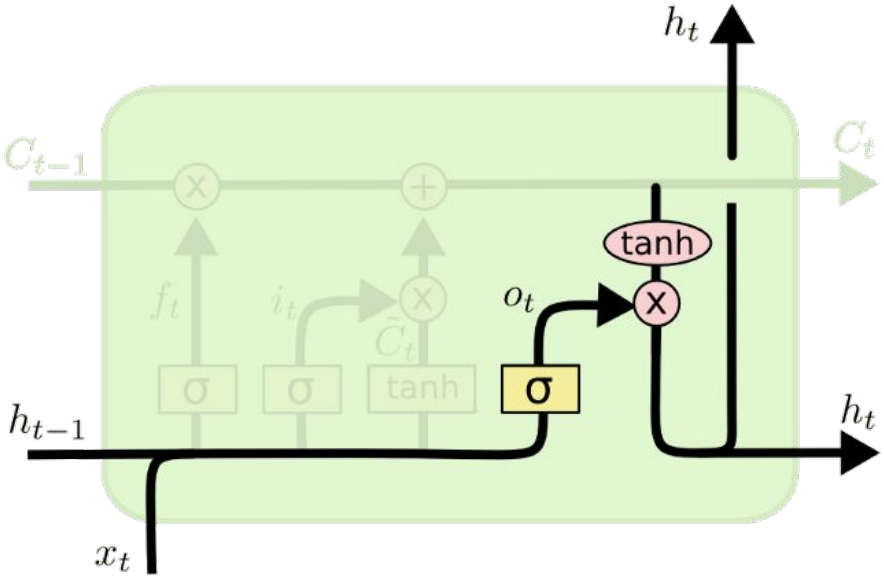
- We multiply the old state by *f(t)*, forgetting the things we decided to **forget** earlier. Then we add *i(t)∗C̃(t)*. This is the new candidate values, scaled by how much we decided to **update** each state value.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM

- **Output Gate:**
  - Decides which part of the current cell makes it to the output.
  - Sigmoid function decides which values to let through **0,1**. and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from**-1** to 1 and multiplied with output of **Sigmoid**.

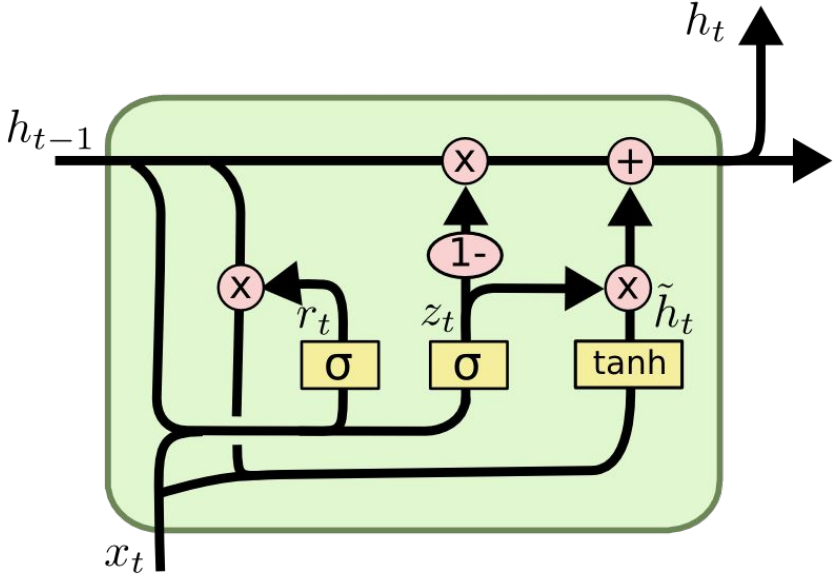


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Unit (GRU)

- GRU **combines** the **forget** and **input** gates into a single "**update gate**." It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Convolution Neural Networks (CNN)

2022

https://github.com/king-menin/mipt-nlp2022

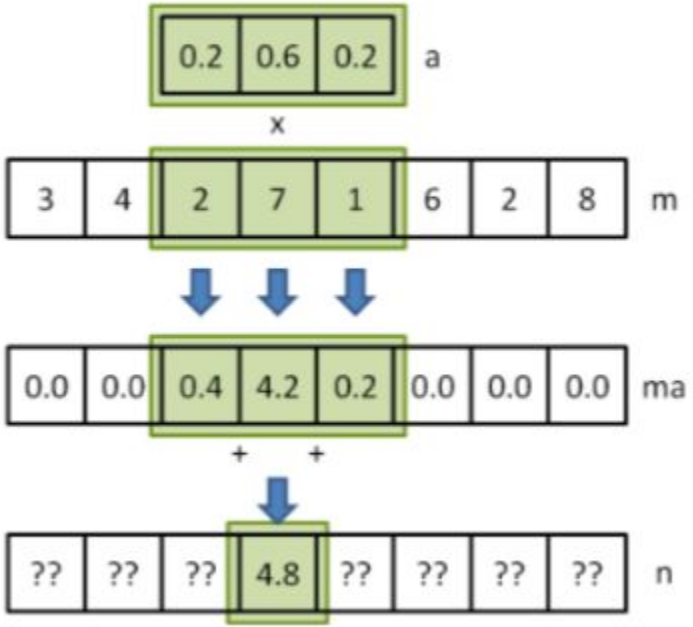
Convolutional neural networks

- **Convolutional neural networks**:
  - Borrowed from the field of computer vision.
  - The peak of popularity was in 2014 (up to + 10% accuracy in classification problems), over time they were supplanted by recurrent neural networks.
- **Help solve problems**:
  - Often, inputs are of variable length (texts, paragraphs, offers)
  - Translational invariance

Convolution

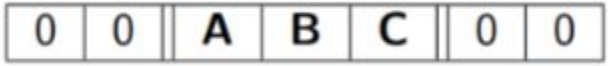
- **Definition**: The result of the operation of **convolution** of an array m with a kernel a is a signal n. Notation: *n=m*a*

$$n[k] = \sum_{i=-w/2}^{i=w/2} m[k+i] \cdot a[-i][k+i]$$

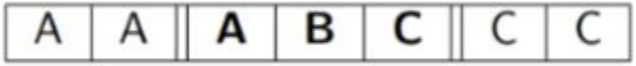


Padding

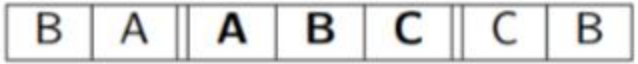
- Zero padding

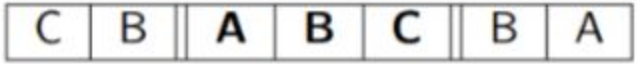


- Copy of boundary

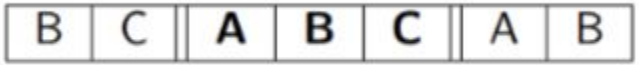


- Mirror padding

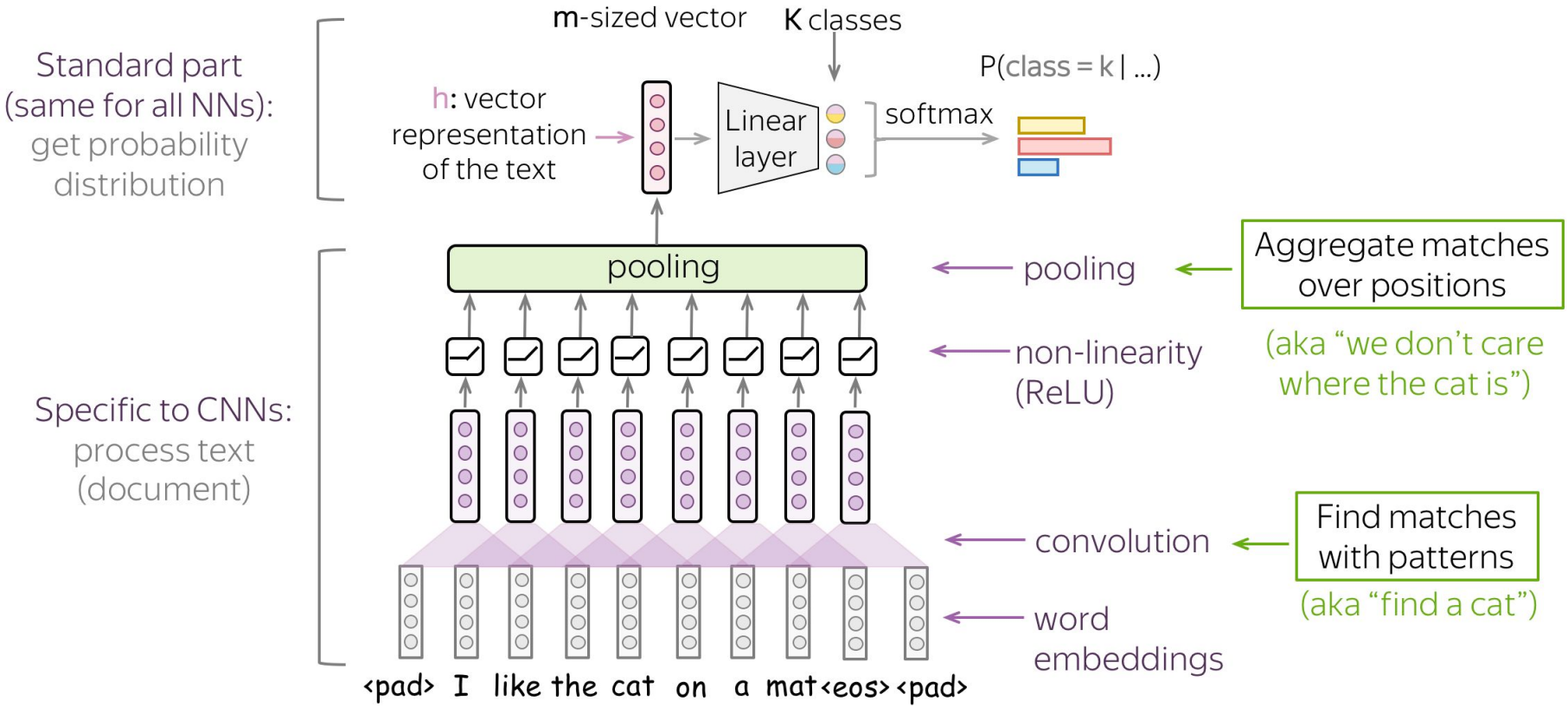




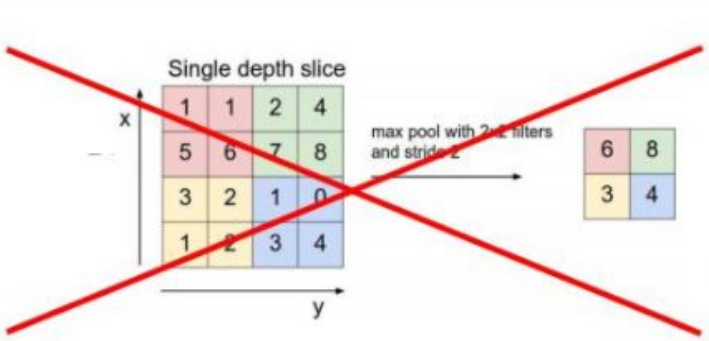
- Cycle padding

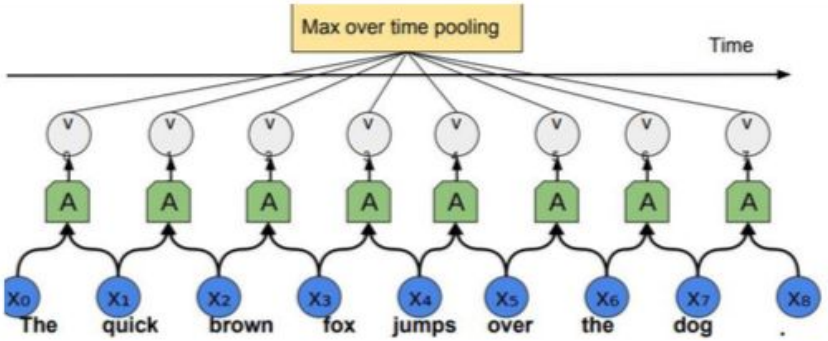


A Typical Model: Convolution + Pooling Blocks



Task description





- **Voting**: the most active neurons wins.
- Developed **invariance** to small shifts (within window).
- **Reduced computational** costs.
- There is avg pooling, but **max pooling** over time works better in text classification tasks.

Yoon Kim. Convolutional Neural Networks for Sentence Classification. 2014

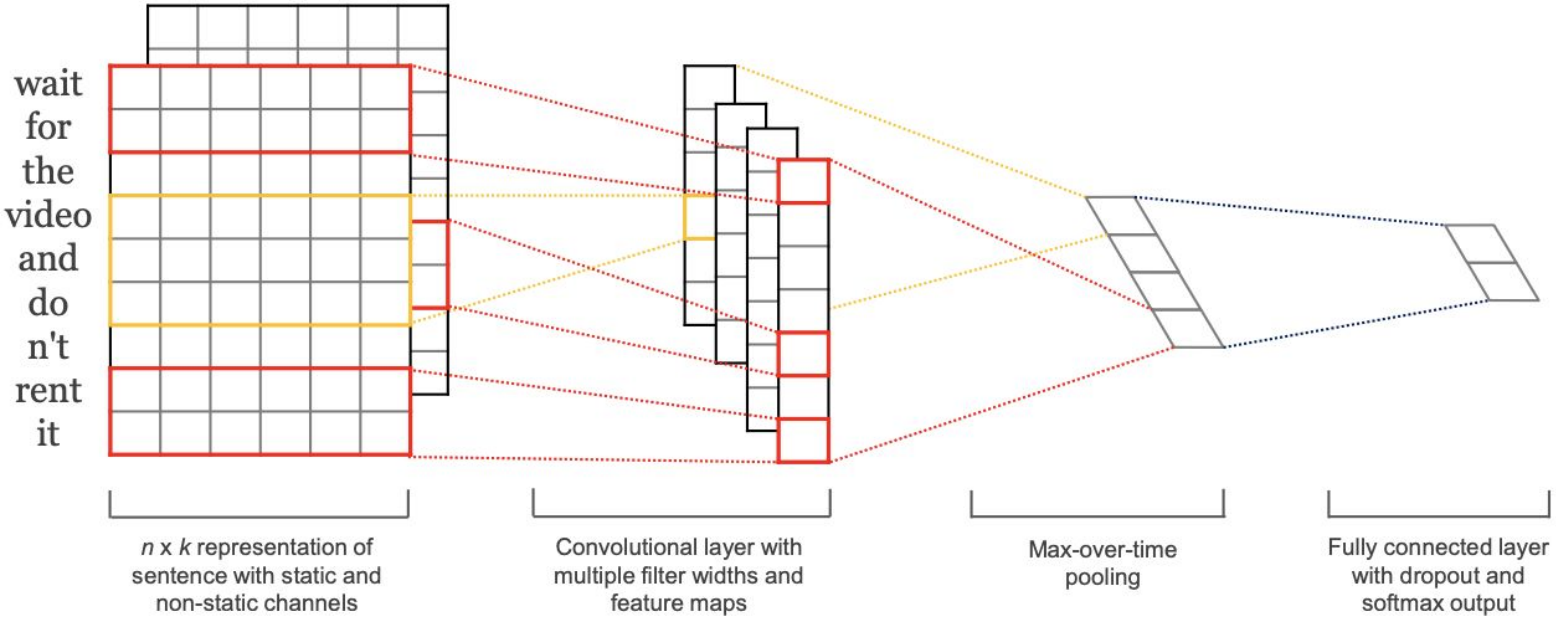


Figure 1: Model architecture with two channels for an example sentence.

[Convolutional Neural Networks for Sentence Classification]

CNNs vs RNNs

- With a lot of reservations RNNs demonstrates slightly better results on the benchmark classification tasks.
- CNNs work well on the tasks that can be reduced to keyword search. Keyword mean NEs and so on.
- Also, RNNs have slower inference than CNNs. CNNs are easier to train.
- For RNN you need more data.

Comparative Study of CNN and RNN for Natural Language Processing [Yin et al 2017] - https://arxiv.org/abs/1702.01923.

CNNs vs RNNs

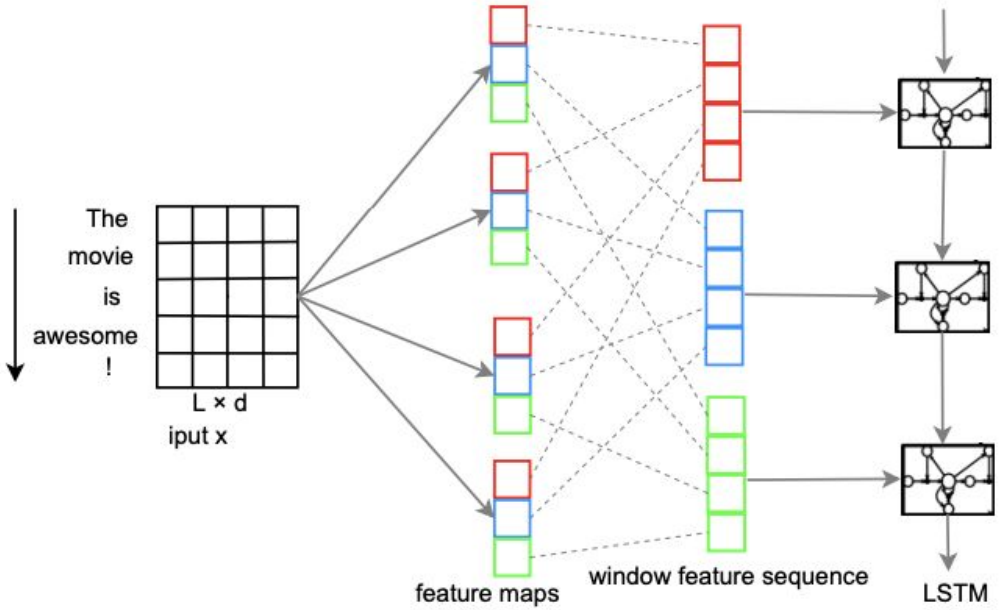
- With a lot of reservations RNNs demonstrates slightly better results on the benchmark classification tasks.
- CNNs work well on the tasks that can be reduced to keyword search. Keyword mean NEs and so on.
- Also, RNNs have slower inference than CNNs. CNNs are easier to train.
- For RNN you need more data.

It's seems to be very task-dependent thing.
So you should try both options.

Comparative Study of CNN and RNN for Natural Language Processing [Yin et al 2017] - https://arxiv.org/abs/1702.01923.

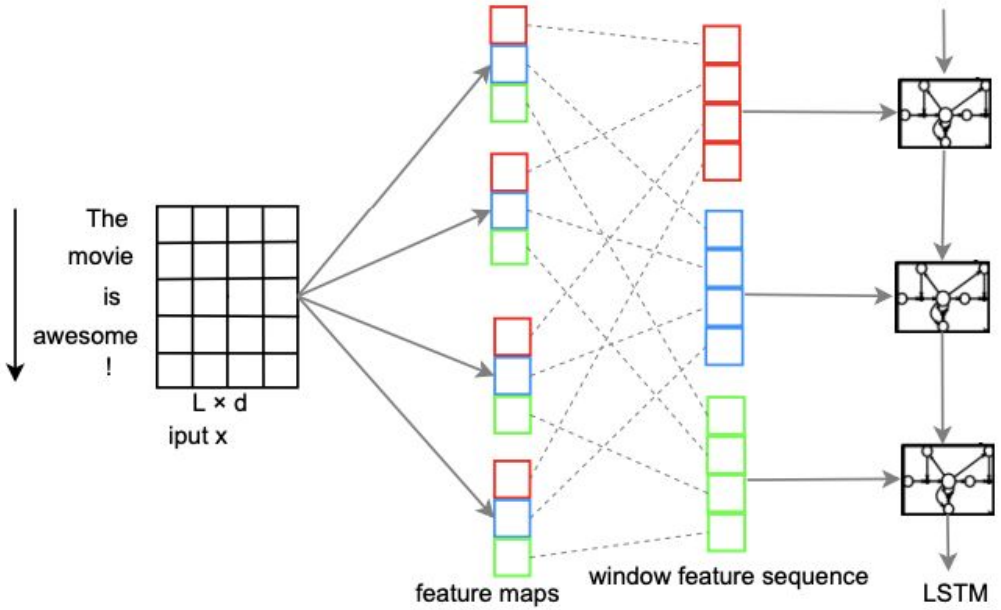
A C-LSTM Neural Network for Text Classification. 2015

- The **C-LSTM** uses **CNN** to extract a sequence of higher-level phrase representations and feeds into the LSTM to obtain a sentence representation.



A C-LSTM Neural Network for Text Classification. 2015

- The **C-LSTM** uses **CNN** to extract a sequence of higher-level phrase representations and feeds into the LSTM to obtain a sentence representation.



Recurrent Convolutional Neural Networks for Text Classification

- RCNN. [Bi-RNN]->[conv.].

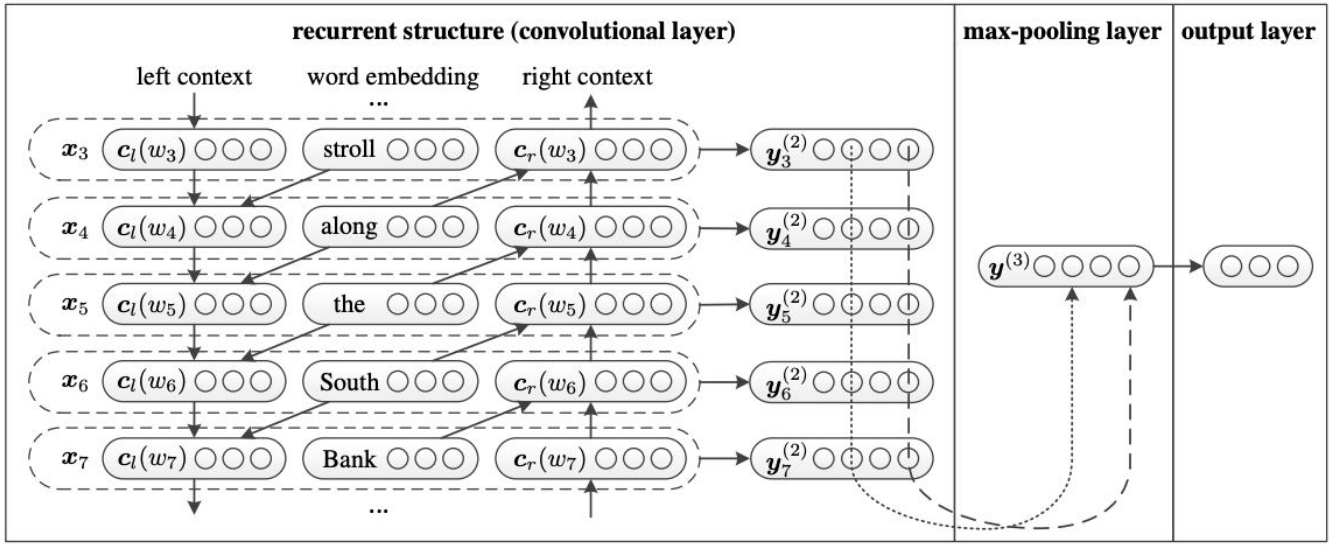


Figure 1: The structure of the recurrent convolutional neural network. This figure is a partial example of the sentence "A sunset stroll along the South Bank affords an array of stunning vantage points", and the subscript denotes the position of the corresponding word in the original sentence.

Learning pytorch

- Very good examples (**for google colab!**):
  - https://github.com/param087/Pytorch-tutorial-on-Google-colab
- And official https://pytorch.org/tutorials/

Questions

https://github.com/king-menin/mipt-nlp2022

Reference

- https://lena-voita.github.io/nlp_course
- https://medium.com/better-programming/generative-vs-discriminative-models-d26def8fd64a
- Vanishing & Exploding Gradient
- Dropout
- CNNs papers
  - A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification
  - Convolutional Neural Network Architectures for Matching Natural Language Sentences
  - A Convolutional Neural Network for Modelling Sentences
  - Convolutional Neural Networks for Sentence Classification
  - Convolutional Neural Network for Paraphrase Identification
  - Relation Classification via Convolutional Deep Neural Network
  - Character-level Convolutional Networks for Text Classification

Reference

- Different RNNs and explanations
  - peephole lstm
  - QRNN
  - AWD LSTM
  - Mogrifier LSTM
  - https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
  - https://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - RNNs explained
- Comparative Study of CNN and RNN for Natural Language Processing
- C-LSTM
- RCNN