Głębokie sieci neuronowe w przetwarzaniu języka naturalnego

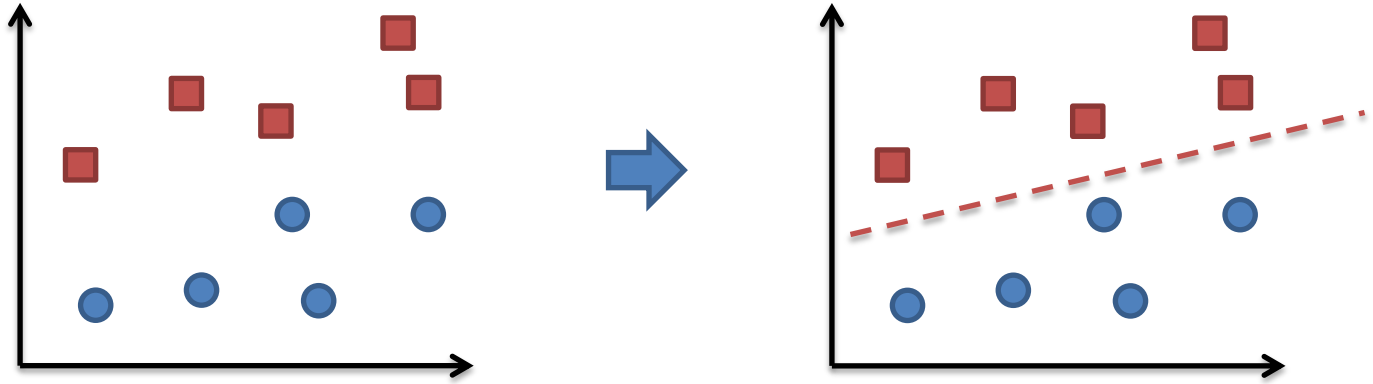
CaseWeek 2018

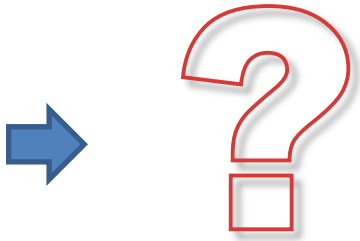
SAMSUNG

Agenda

- Sieci neuronowe
- Zastosowania sieci w NLP
- Modele języka
- Część praktyczna

Sieci neuronowe

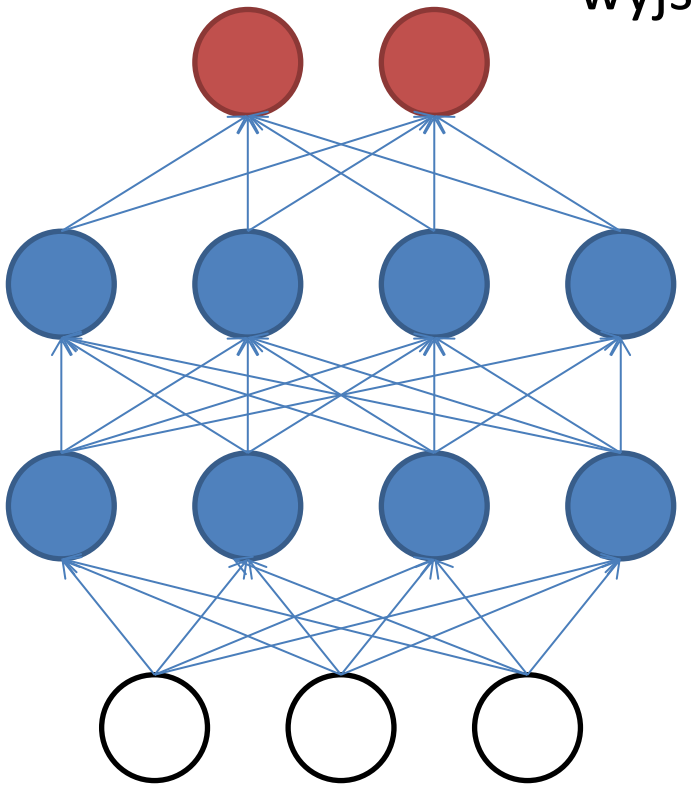




Sieci neuronowe

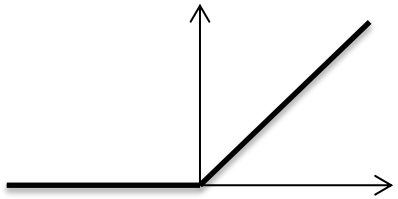
- Algorytm uczenia maszynowego
- Inspiracja: próba naśladowania mózgu
- Są uniwersalne
- Mogą wykonywać bardzo złożone operacje
- Mogą uczyć się, które cechy danych wejściowych są istotne

Sieci neuronowe

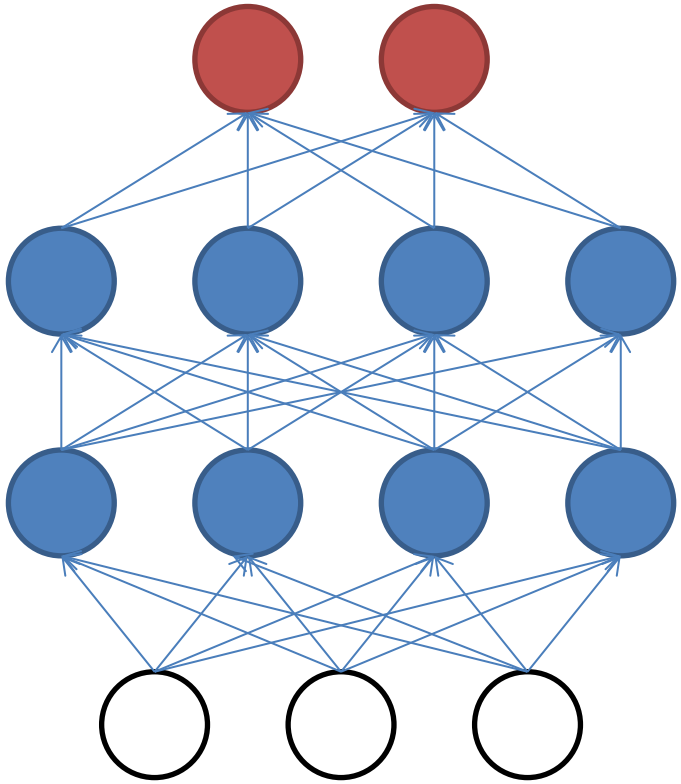


warstwy ukryte

: suma ważona wejść, dodatkowo funkcja nieliniowa na wyjściu

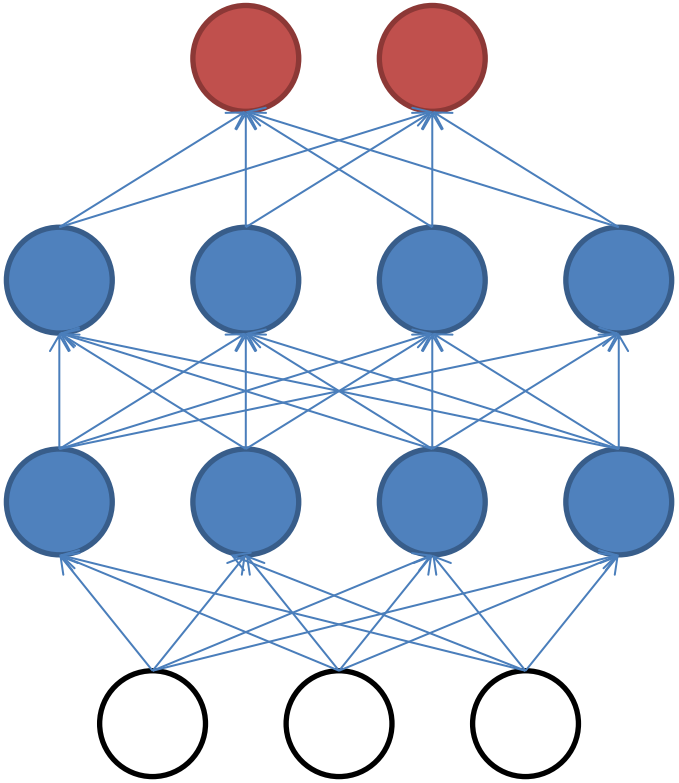


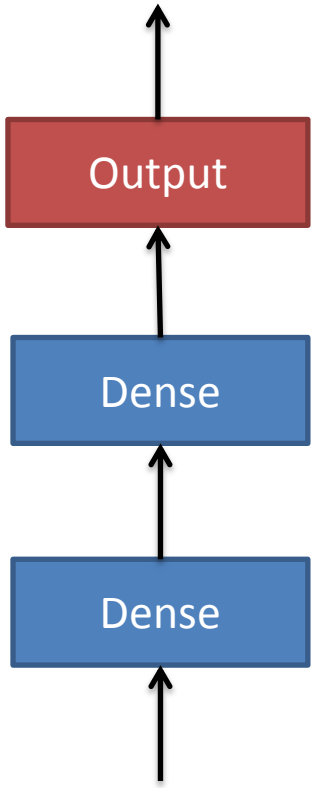
Sieci neuronowe – uczenie

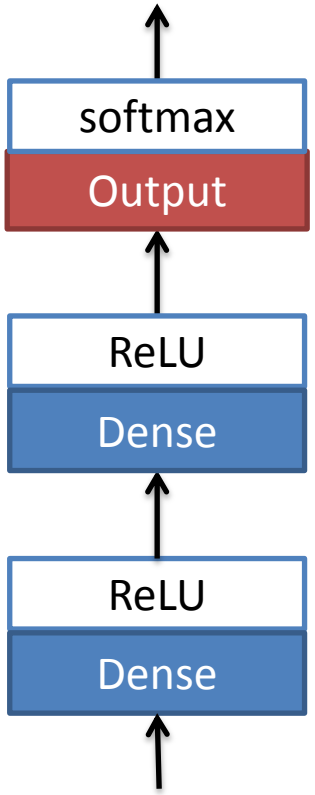


- Algorytm *propagacji wstecznej* (backpropagation)
- Inicjalizacja wag (losowa)
- Dla każdego przykładu treningowego:
  - Podajemy dane wejściowe
  - Obliczamy wynik działania sieci
  - Porównujemy z oczekiwanym wynikiem, obliczamy różnicę (błąd) na wyjściu
  - Błąd propagujemy wstecz, obliczamy o ile zmienić każdą wagę aby go zmniejszyć

Sieci neuronowe

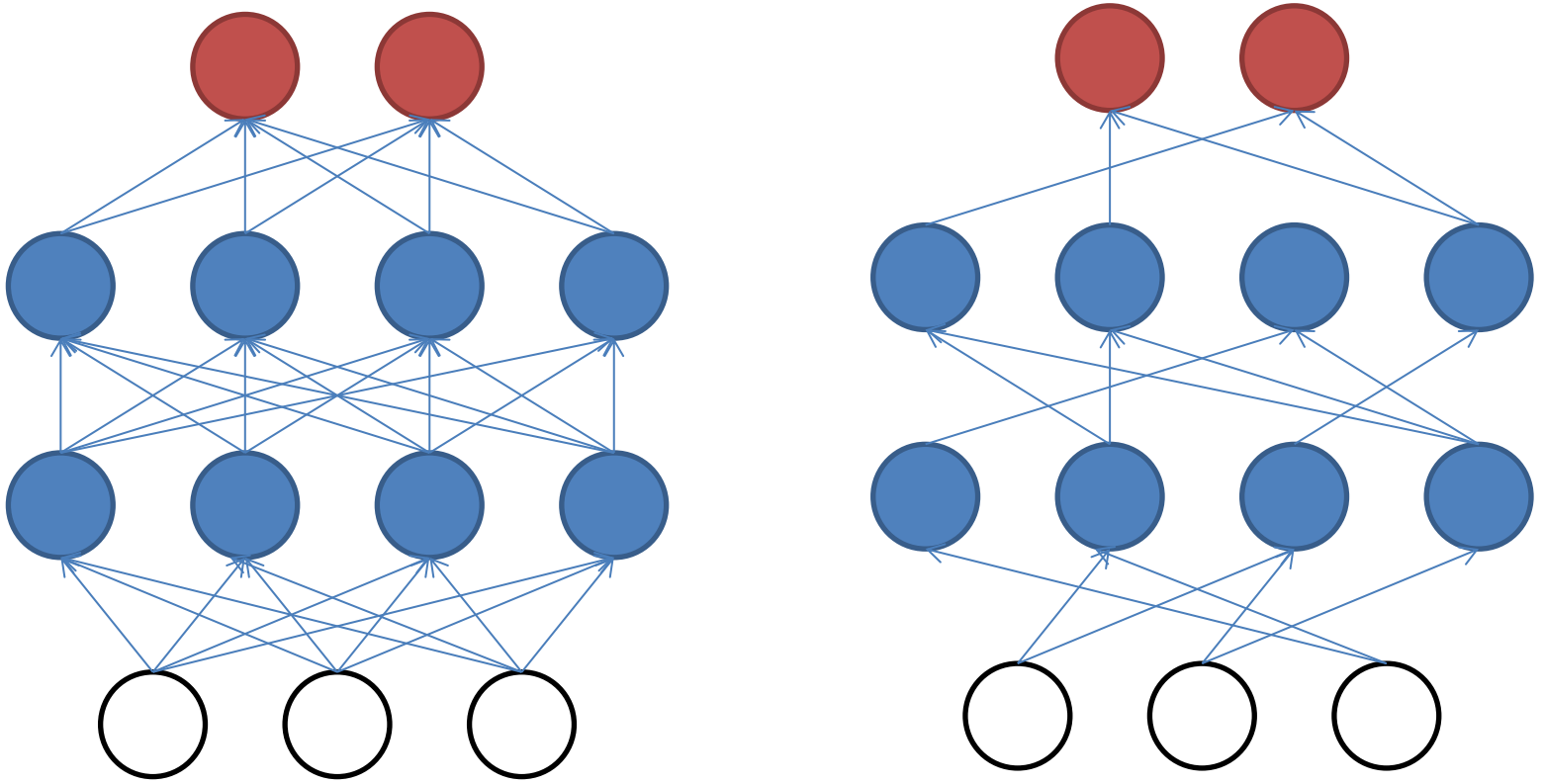




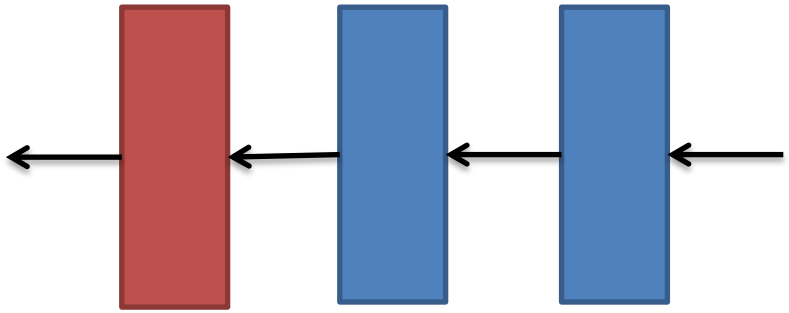


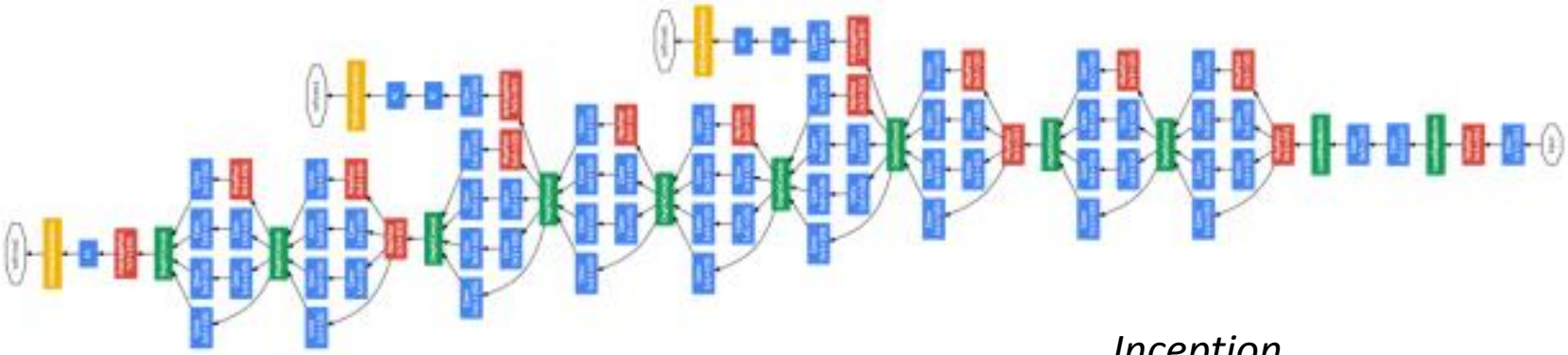
Sieć o *n* warstwach
≈ program o *n* krokach

Dropout



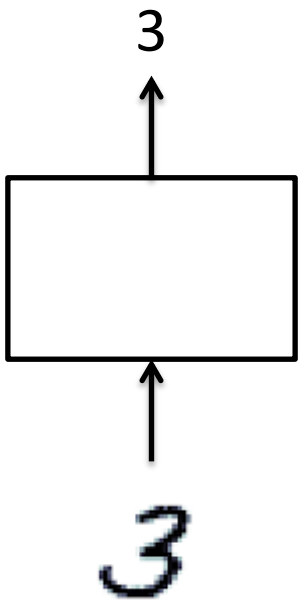
Głębokie sieci neuronowe

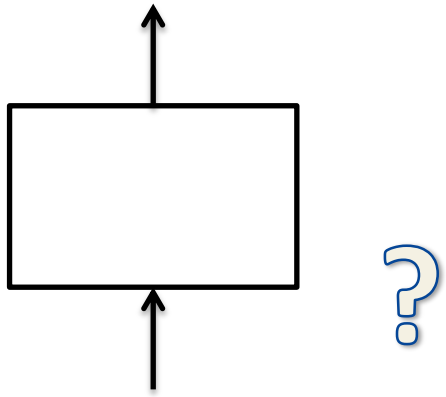




Inception

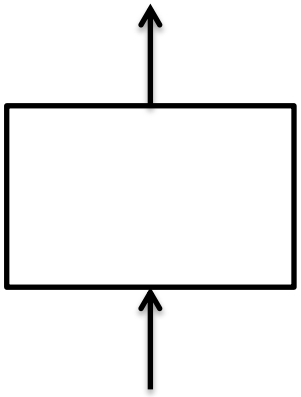
Sieci rekurencyjne



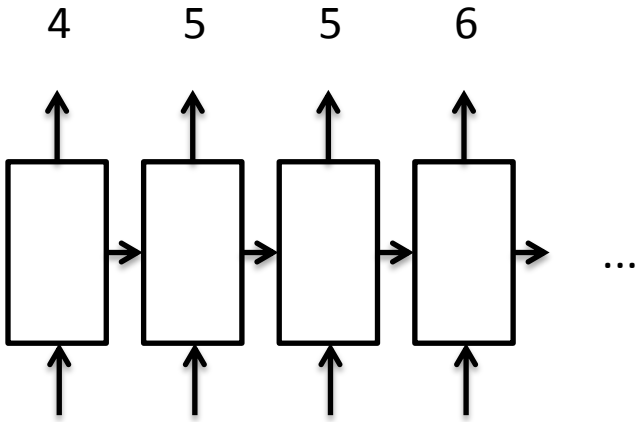


3, 4, 5, 5, 6, 4, 3, 0, 1, 2, 3, 3, 4, ...

Sieci rekurencyjne

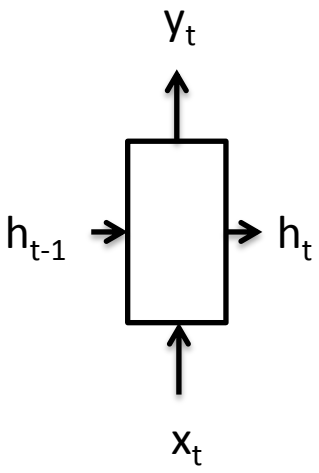


3, 4, 5, 5, 6, 4, 3, 0, 1, 2, 3, 3, 4, ...



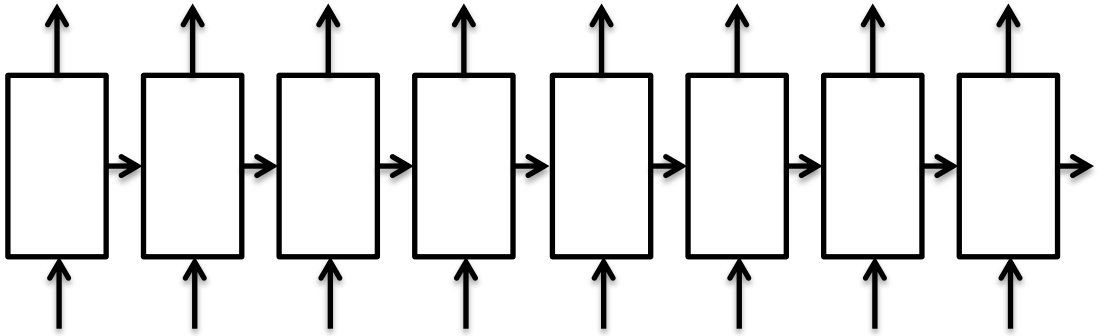
3 4 5 5

Sieci rekurencyjne

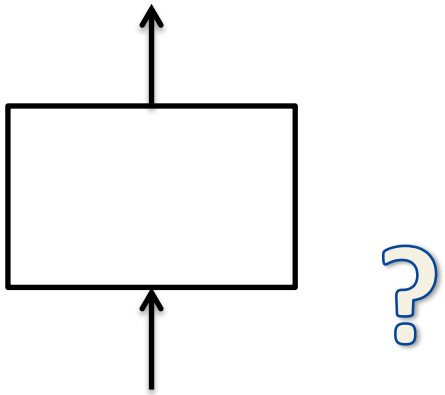


$h_t = f(\mathbf{W} \cdot [h_{t-1}, x_t] + b_W)$
$y_t = g(\mathbf{V} \cdot h_t + b_V)$

W, V – macierze wag
b_W, b_V – wektory
f, g – funkcje nieliniowe,
np. f = tanh, g = softmax

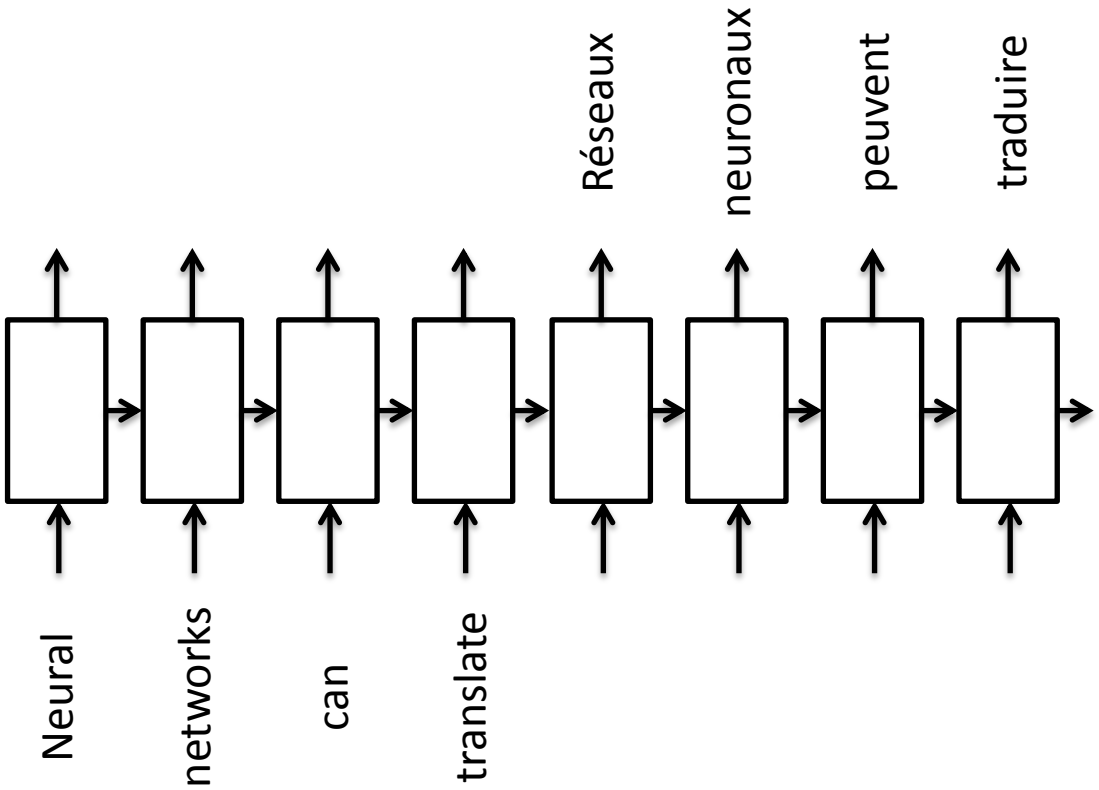


Sieci rekurencyjne

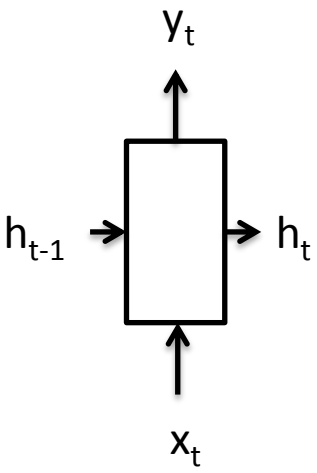


Neural networks can translate text from English to French.

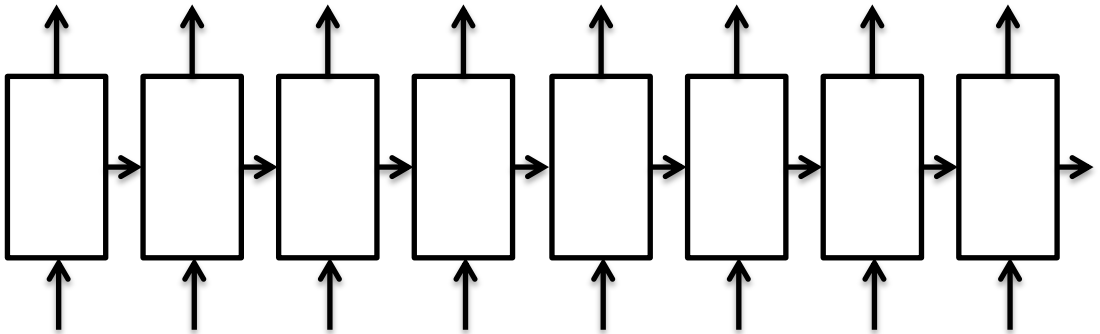
Sieci rekurencyjne



Sieci rekurencyjne



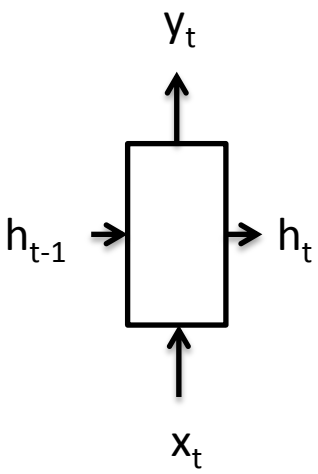
$$h_t = f(\mathbf{W} \cdot [h_{t-1}, x_t] + b_W)$$
$$y_t = g(\mathbf{V} \cdot h_t + b_V)$$



Problem 1: nawet gdy x = 0, stan h zmienia się w czasie.
Problem 2: informację o błędzie trudno przenieść o wiele kroków wstecz.

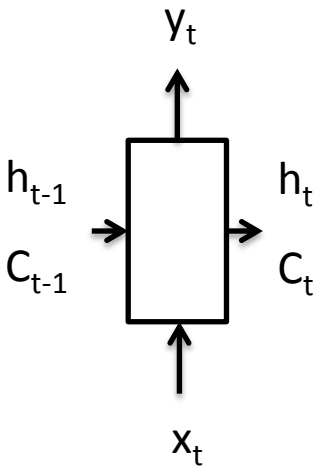
Rozwiązanie: zmodyfikowane komórki RNN, takie jak:
LSTM – Long Short-Term Memory
GRU – Gated Recurrent Unit

Sieci rekurencyjne LSTM



Stan:

$h_t = \tanh(\mathbf{W} \cdot [h_{t-1}, x_t] + b_W)$



Stan:

$h_t = o_t \cdot \tanh(C_t)$

$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(\mathbf{W_c} \cdot [h_{t-1}, x_t] + b_C)$

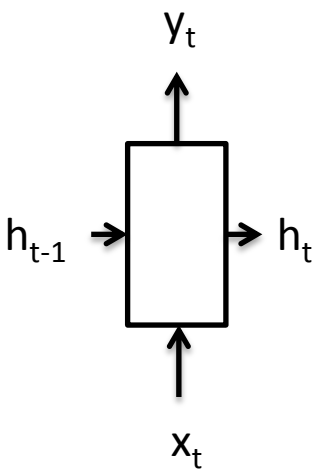
Bramki:

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

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

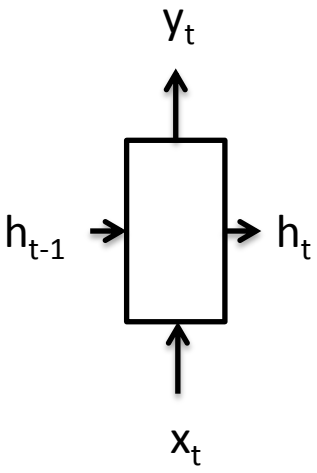
$f_t = \sigma(\mathbf{W_f} \cdot [h_{t-1}, x_t] + b_f)$

Sieci rekurencyjne GRU



Stan:

$h_t = \tanh(\mathbf{W} \cdot [h_{t-1}, x_t] + b_W)$



Stan:

$h_t = (1-z_t) \cdot h_{t-1} + z_t \cdot \tanh(\mathbf{W} \cdot [r_t \cdot h_{t-1}, x_t])$

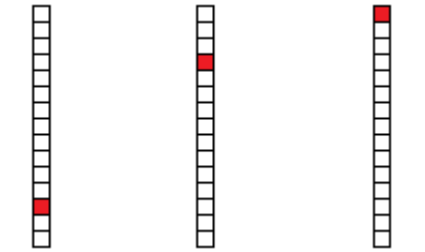
Bramki:

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

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

Słowa na wejściu sieci

- Kodowanie *one-hot*: każde słowo kodowane wektorem który ma jedną jedynkę i resztę zer



- *Zanurzenia słów*: każde słowo jest reprezentowane wektorem liczb rzeczywistych

Modele języka

- Model języka przypisuje prawdopodobieństwo ciągowi słów

  Anna poszła do kiosku i kupiła gazetę. $p = ?$

  $p = p(\text{Anna}) \cdot p(\text{poszła}|\text{Anna}) \cdot p(\text{do}|\text{Anna poszła}) \cdot ...$
  $\cdot\ p(.|\ \text{Anna poszła do kiosku i kupiła gazetę})$

- Pozwala na porównywanie alternatywnych wersji zdania:

  p (Zaczyna się wiosna) > p (Zaczyna się wiosła)

Modele języka – zastosowania

- Rozpoznawanie mowy
- Rozpoznawanie tekstu, pisma (OCR)
- Tłumaczenie maszynowe
- Korekcja błędów
- Wprowadzanie tekstu
- ...

Modele języka – jak zbudować

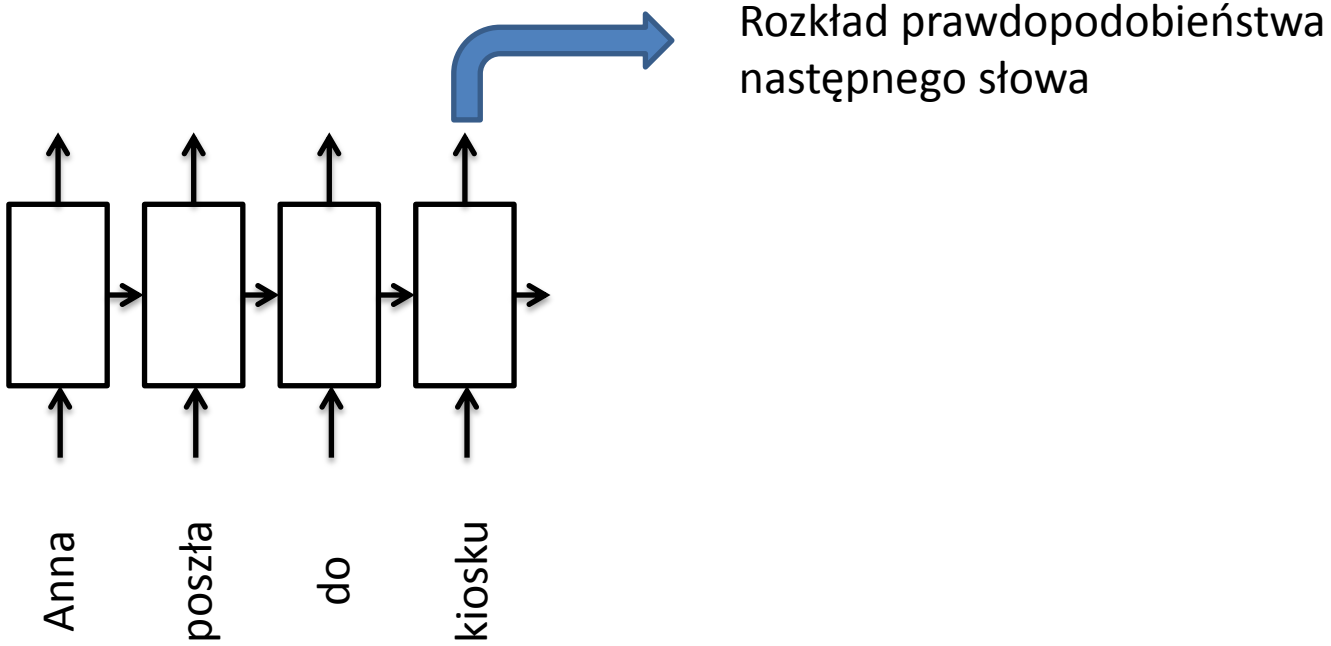
- $p(w_1, w_2, \ldots w_N) = p(w_1) \cdot p(w_2|w_1) \cdot$
  $\cdot\, p(w_3|w_2, w_1) \cdot \ldots \cdot p(w_N|w_1, w_2, \ldots w_{N-1})$
- Model unigramowy:
  $p \approx p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_2) \cdot \ldots$
  $\cdot\, p(w_N|w_{N-1})$
- Model bigramowy:
  $p(x|y_1,y_2,\ldots y_k) \approx p(x|y_1, y_2)$
- Prawdopodobieństwa szacowane z *korpusu:*
  $p(w_3|w_1, w_2) = n(w_1, w_2, w_3) / n(w_1, w_2)$

n-gramowe modele języka

- Zalety: dobre skalowanie, szybkość
- Problemy:
  - dla większych *n* wiele *n*-gramów w ogóle nie wystąpi w korpusie (krótki kontekst)
  - nie uwzględniają pokrewieństwa słów: np. poszedł ↔ pojechał; kot ↔ pies

Modele języka oparte na RNN

- Kolejne słowa są wejściem do RNN w kolejnych krokach czasowych



Modele języka oparte na RNN

- Mogą uchwycić nawet długie zależności (nie mają ograniczenia do *n* słów kontekstu)
- Mogą działać na poziomie słów, morfemów, znaków...