

# Aplikace neuronových sítí

---

Attention mechanismus

# RNN znakový jazykový model

správné znaky: "e"

výstupní skóre:

h	-0.9
e	<b>2.5</b>
l	-1.9
o	-3.2

skrytý stav:

vstupní vektor:

h	1
e	0
l	0
o	0

vstupní znaky: "h"

"l"

h	-0.7
e	-1.3
l	<b>1.9</b>
o	1.5

"o"

"l"

h	-0.6
e	-1.7
l	<b>1.7</b>
o	<b>1.8</b>

"o"

chybná  
predikce

$W^{hy}$

0.4
-0.3
-0.5
<b>0.6</b>

$W^{hh}$

0.5
-0.3
0.2

"l"

h	-0.5
e	-0.3
l	0.5

"o"

$W^{hh}$

0.2
0.1
-0.3

$W^{xh}$

0
0
1
0

"l"

h	0
e	0
l	1
o	0

stejné parametry W jako  
v předchozím kroku

# Písmena vs slova

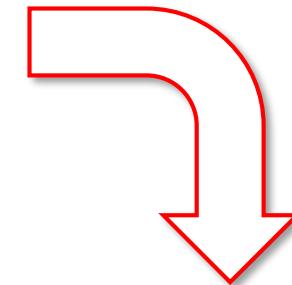
- Místo znaků modelovat závislost slov
- U znaků jsme každé písmeno reprezentovali jako jednotkový vektor (one-hot kódování)
- Matice všech vektorů tedy jednotková s velikostí = počet symbolů
- Problém: slovník řádově  $\approx 10^4, 10^5 \rightarrow$  matice  $10^5 \times 10^5 = 10^{10} \cdot 4B \approx 40GB$  !
  - matice je řídká, pouze jednotkové vektory  $\rightarrow$  nemusí být celá v paměti
  - ale: ovlivňuje i velikost matice parametrů  $W^{xh}$   $\rightarrow$  bude mít rozměr  $10^5 \times H$
- Řešení: reprezentace znaků kratšími vektory  $\rightarrow$  matice např.  $10^5 \times 300 \approx 12MB$ 
  - matice  $W^{xh}: 300 \times H$

# Písmena vs slova

one-hot řídká reprezentace

symbol 1:	[ 1 0 0 0 0 0 ]
symbol 2:	[ 0 1 0 ... 0 0 0 ]
symbol 3:	[ 0 0 1 0 0 0 ]
⋮	[ ⋮ ⋱ ⋮ ]
symbol V-2:	[ 0 0 0 1 0 0 ]
symbol V-1:	[ 0 0 0 ... 0 1 0 ]
symbol V:	[ 0 0 0 0 0 1 ]

počet symbolů



hustá reprezentace

libovolná dimenze

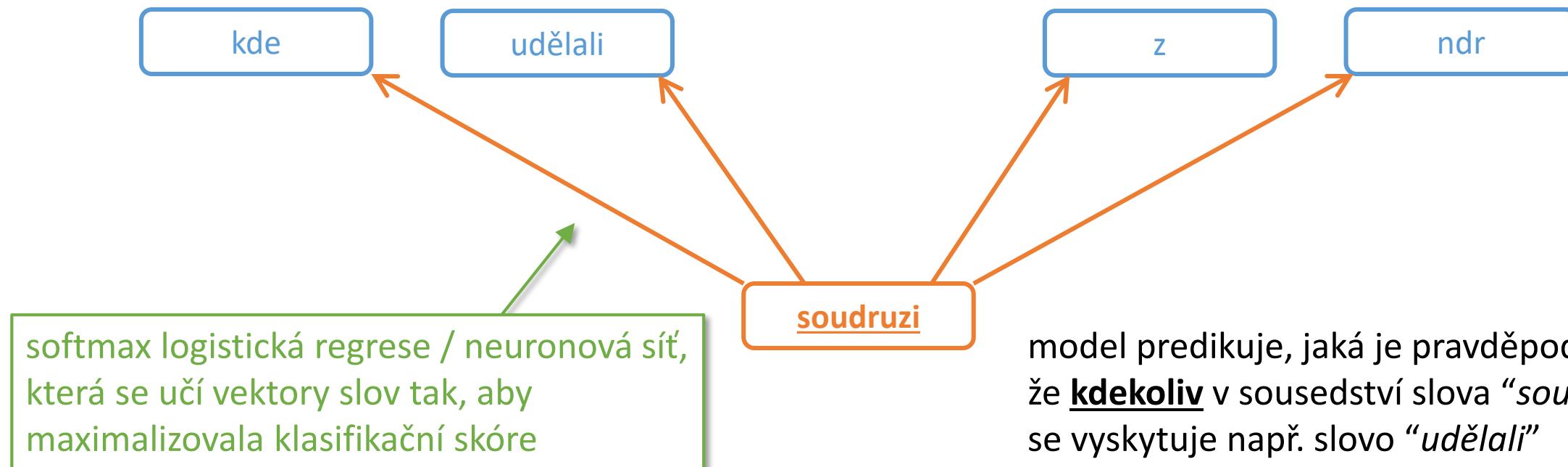
symbol 1:	0.25	-0.11	1.14	...	-0.4	-2.0	2.1
symbol 2:	5.2	1.01	0.1	...	0.22	0.31	-4.44
symbol 3:	4.13	10.1	5.5	...	3.32	-0.12	-0.34
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
symbol V-2:	-0.45	-0.56	0.67	...	1.64	2.32	-3.16
symbol V-1:	2.22	3.33	1.11	...	-0.01	2.74	3.14
symbol V:	2.81	-7.1	-0.05	...	1.13	-3.61	1.0

Jak ale vektory zvolit?

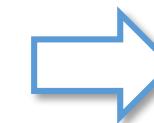
# word2vec

- Tzv. **skip-gram model** → predikce okolních slov z aktuálního (prostředního)

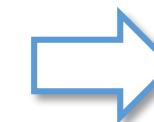
Příklad: “Kde udělali **soudruzi** z NDR chybu?”



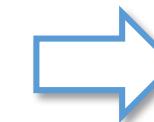
# word2vec: trénovací data



(kde, udělali)  
(kde, soudruzi)



(udělali, kde)  
(udělali, soudruzi)  
(udělali, z)



(soudruzi, kde)  
(soudruzi, udělali)  
(soudruzi, z)  
(soudruzi, ndr)



velikost posuvného okna

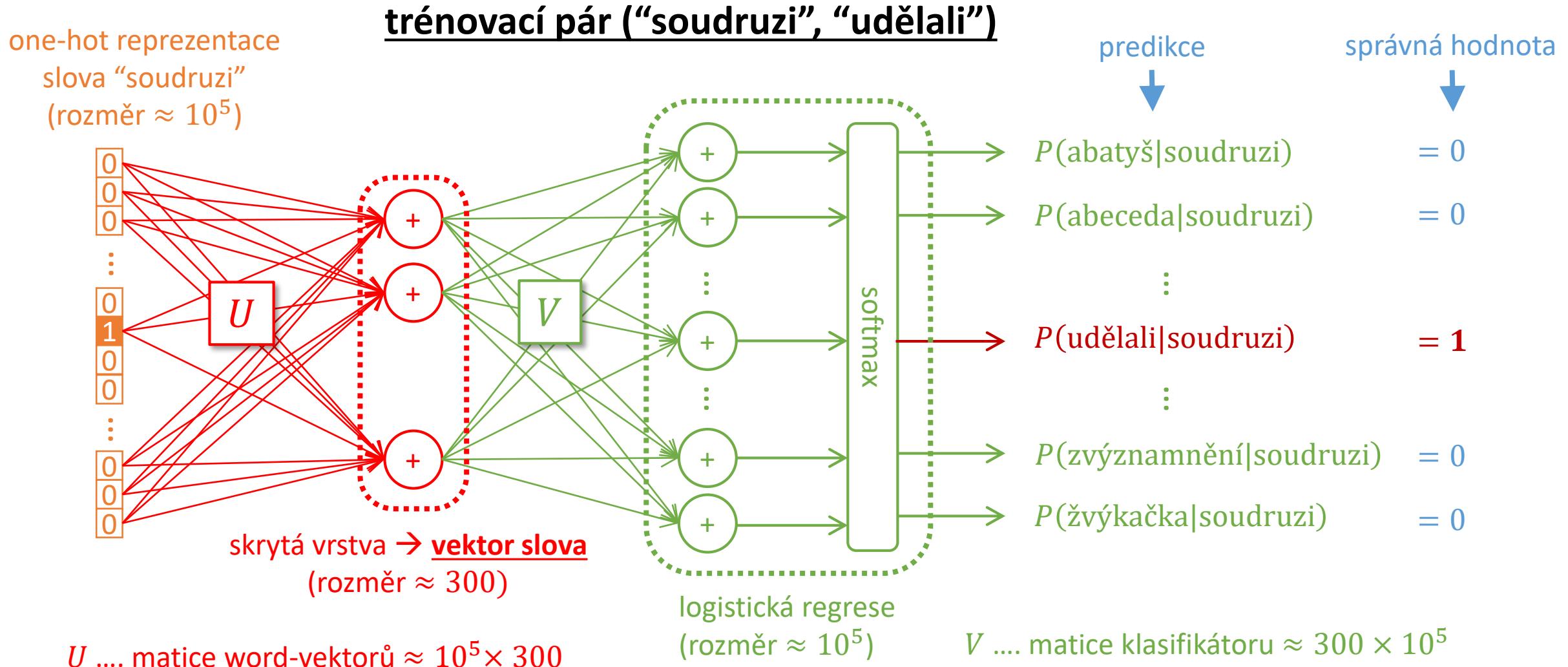
:



(chybu, z)  
(chybu, ndr)

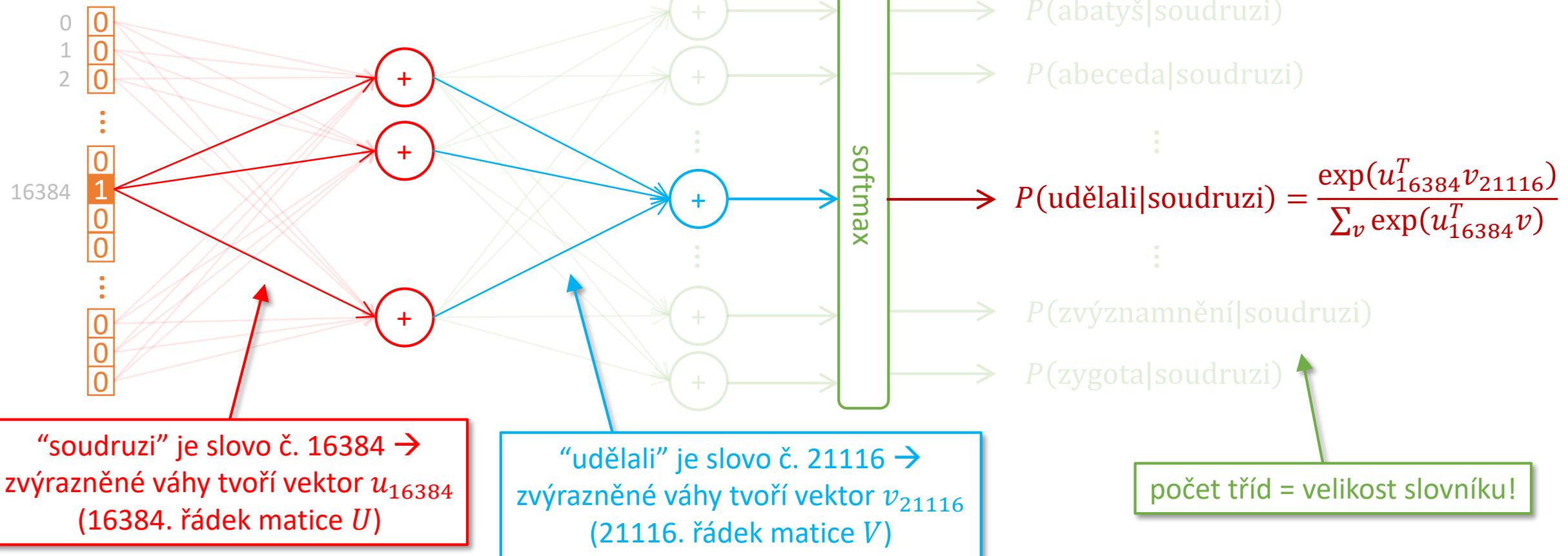
trénovací páry (x, y)

# word2vec: základní architektura sítě



# word2vec: základní architektura sítě

one-hot reprezentace  
slova "soudruzi"  
(rozměr  $\approx 10^5$ )



# word2vec: kritérium

- Maximalizujeme pravděpodobnost správného slova (třídy), tzn. pro jeden krok (jedno aktuální slovo  $u$  “v čase  $t$ ”) je kritérium

$$L_t(U, V) = \prod_{v \in \text{okolí}(u)} P(v|u) = \prod_{v \in \text{okolí}(u)} \frac{\exp(u^T v)}{\sum_{w \in \text{slovník}} \exp(u^T w)}$$

- Což je zjednodušená forma zápisu, který je např. v přednáškách [Stanford CS224D \(2017\)](#)
- word2vec je ale jen logistická regrese → prostě minimalizujeme křížovou entropii

$$L_{u,v}(U, V) = \underbrace{-u^T v}_{\text{skóre správné třídy}} + \log \sum_{w \in \text{slovník}} \exp(u^T w)$$

- a to opakujeme pro každé slovo  $u$  z trénovacího korpusu a pro každé  $v \in \text{okolí}(u)$

# word2vec: záporné vzorkování

- Problém:

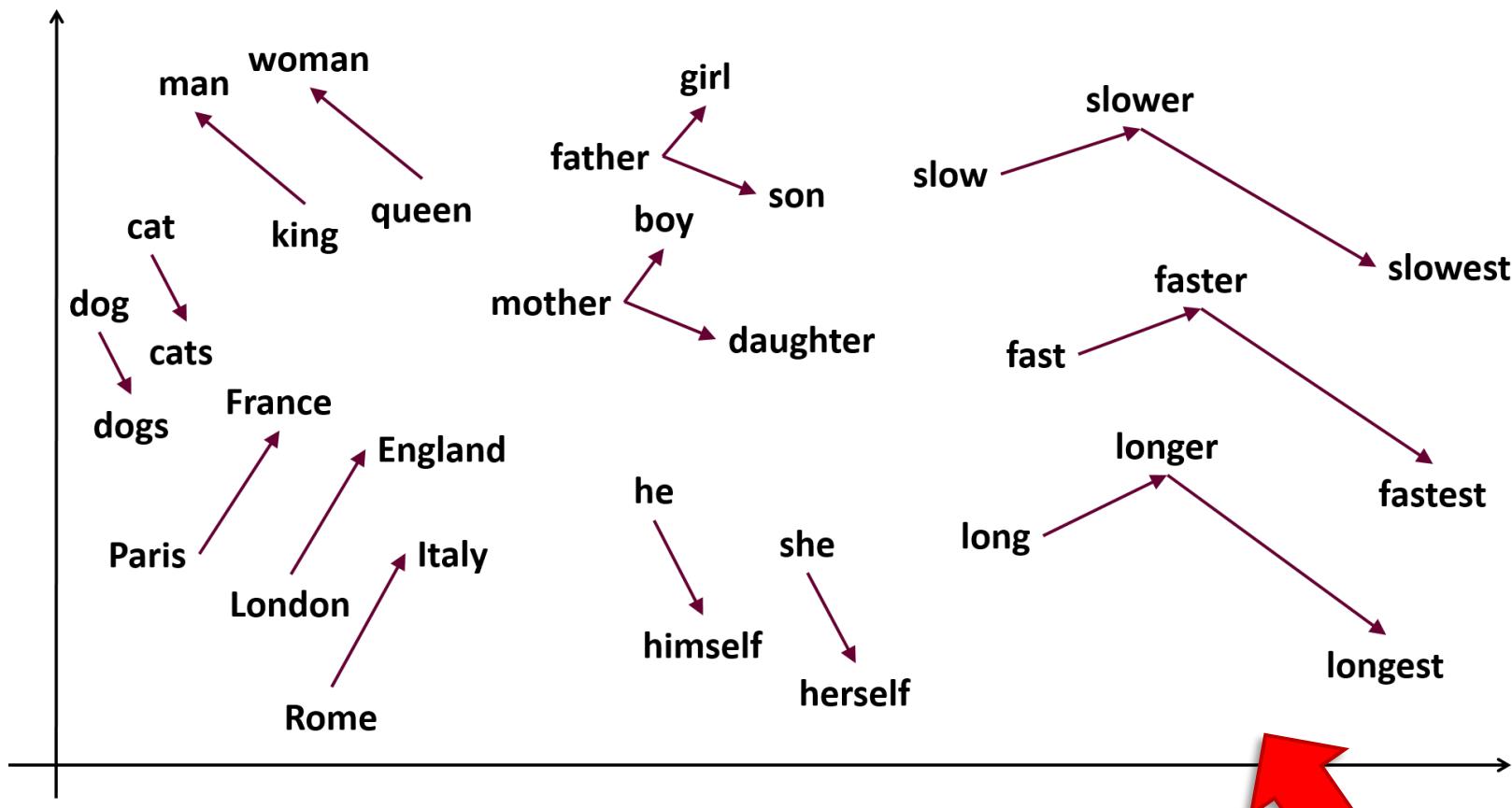
$$L_{u,v}(U, V) = -u^T v + \log \sum_{w \in \text{slovník}} \exp(u^T w)$$

- se v každém kroku vyhodnocuje pro celý slovník → velmi pomalé
- Suma představuje “záporná slova”  $w$  – která mají mít pro slovo  $u$  malou  $P(w|u)$
- Není přitom nutné vyhodnocovat pro celý slovník, např. dvojici “soudruzi” a “labrador”
- Stačí 5-20 slov pro menší datasety, 2-5 pro velké
- Suma se tedy vyhodnotí místo celého slovníku jen pro několik málo slov, tj.

$$L_{u,v}(U, V) = -u^T v + \log \sum_{w \in \text{náhodný výběr}} \exp(u^T w)$$

- Výběr není úplně náhodný, závisí na frekvenci slov apod.
  - detailly v článku či např. zde: [Stanford CS224D \(2017\)](#)

# word2vec: naučené vztahy mezi slovy



## Příklady:

$$\text{France} - \text{Paris} + \text{London} = \text{England}$$

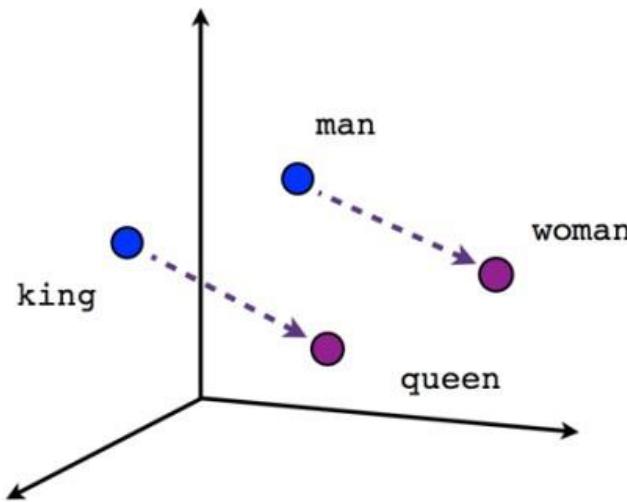
$$\text{king} - \text{man} + \text{woman} = \text{queen}$$

...

obrázek: <http://www.samyzaf.com/ML/nlp/nlp.html>

každé slovo je vektor, zde 2D

# word2vec: naučené vztahy mezi slovy

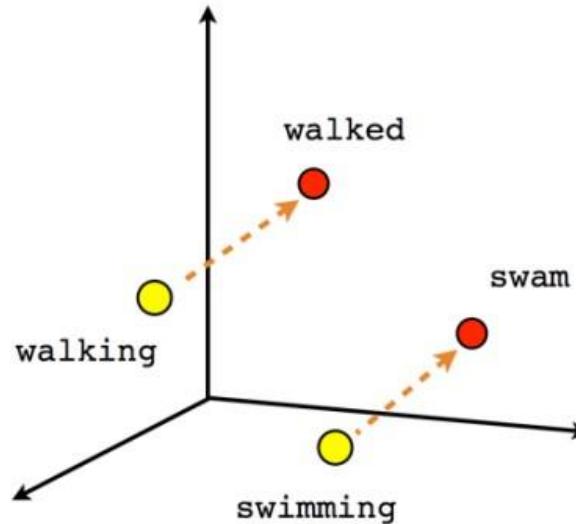


Male-Female

king – man + woman = queen



queen – king = woman - man

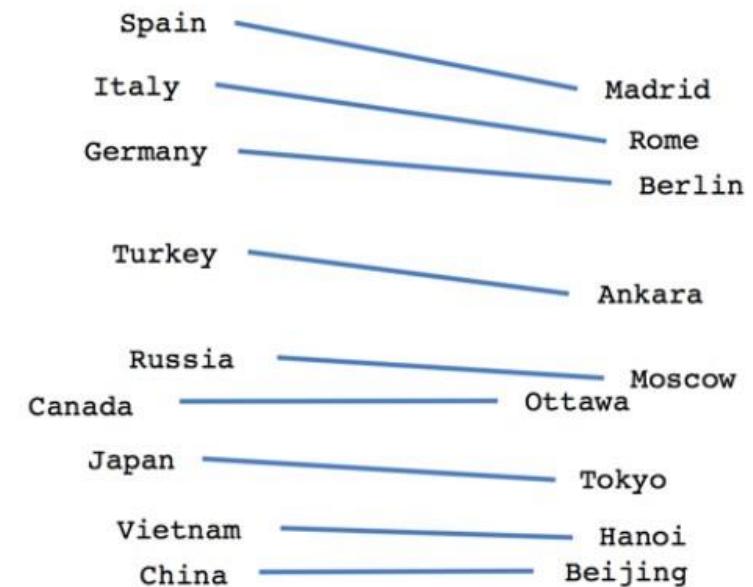


Verb tense

walking – swimming + swam = walked



walked – walking = swam – swimming



Country-Capital

podobně pak státy a města jsou od sebe vždy stejným směrem  
(liší se od sebe stejným vektorem)

# word2vec: nejbližší soused a kosinová vzdálenost

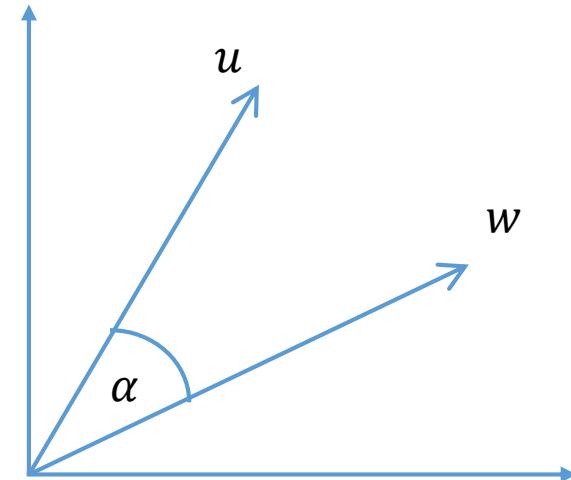
- Výsledkem operace s vektory je vektor

$$u = r - s + v$$

- Jakému slovu  $u$  odpovídá?
- Mezi všemi slovy  $w \in$  slovník najdeme takový, který se mu nejvíce podobá
- Používá se kosinová vzdálenost

$$d(u, w) = \cos \alpha = \frac{u^T w}{\|u\| \cdot \|w\|}$$

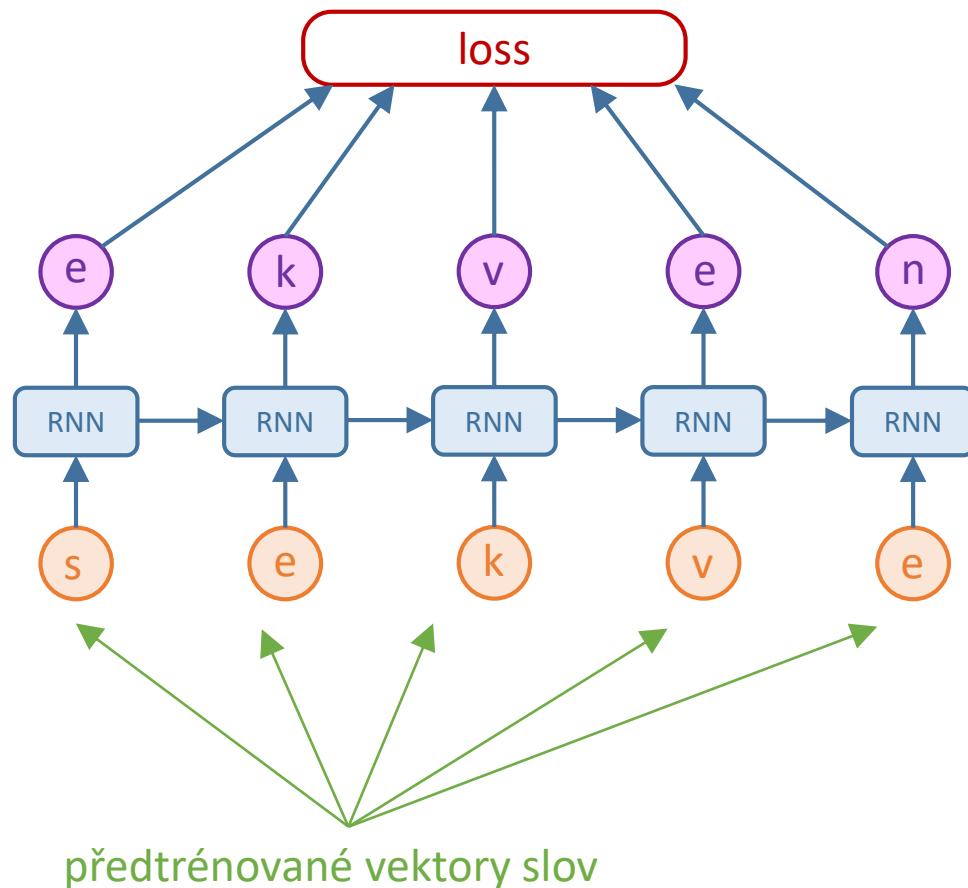
- Nezohledňuje velikost
- Zákonem velkých čísel bývají vektory ve vysokodim. prostorech na vrcholech hyperkvádru (jedna ze souřadnic >> ostatní)  $\rightarrow$  úhly vystihují lépe než vzdálenost



# word2vec: poznámky

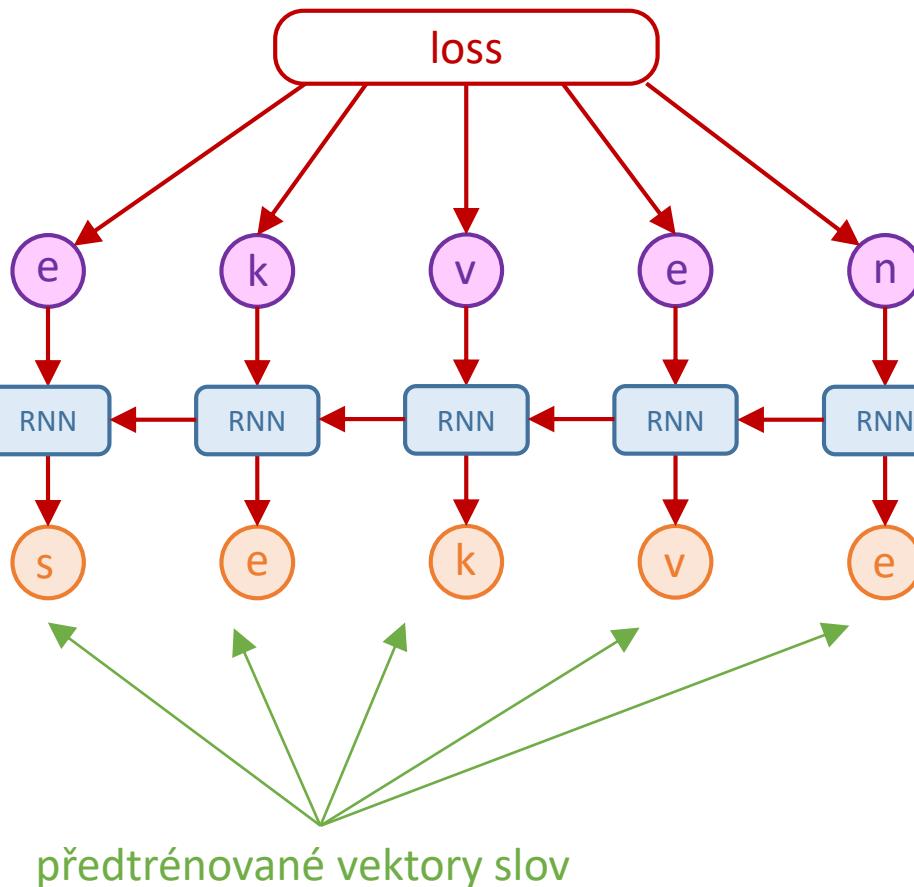
- V příkladech z praxe obvykle velikost okna = 5 (tj. 5 nalevo + 5 napravo)
- Dimenze vektorů 300
- Kromě záporného vzorkování se optimalizuje navíc omezováním příliš častých slov nebo vyhledáváním častých frází, které se pak považují za jedno slovo (např. "Dobrý den")
- Pro různé jazyky existují předtrénované word-vektory na velkých korpusech podobně jako u konvolučních sítí → není nutné trénovat
  - např. zde: <https://fasttext.cc/docs/en/pretrained-vectors.html>
- Existují i další známé varianty:
  - CBOW namísto skip-gram: predikce prostředního slova ze sumy okolních
  - GloVe: kombinuje word2vec se singulárním rozkladem SVD
  - byť to autoři alt. metod rádi tvrdí, žádná z variant ve skutečnosti nefunguje lépe než word2vec

# word2vec: finetuning



- podobný princip jako u předtrénovaných konvoučních sítí
- stáhneme předtrénované vektory a použijeme ve vlastní úloze

# word2vec: finetuning



- podobný princip jako u předtrénovaných konvoučních sítí
- stáhneme předtrénované vektory a použijeme ve vlastní úloze
- můžeme i finetunit

gradient lze propagovat i do vstupu →  
**sít se bude zároveň učit i reprezentaci dat!**

# RNN znakový jazykový model

správné znaky: "e"

výstupní skóre:

h	-0.9
e	<b>2.5</b>
l	-1.9
o	-3.2

skrytý stav:

0.5
-0.1
0.2

vstupní vektor:

1
0
0
0

vstupní znaky: "h"

"l"

-0.7
-1.3
<b>1.9</b>
1.5

"l"

-0.6
-1.7
<b>1.7</b>
<b>1.8</b>

"l"

"o"

0.4
-0.3
-0.5
<b>0.6</b>

chybná  
predikce

stejné parametry W jako  
v předchozím kroku

0.2
0.1
-0.3

$W^{hy}$

$W^{xh}$

**znaky nyní můžeme  
nahradit word-vektory  
a sestavit tak slovní  
jazykový model!**



# RNN slovní jazykový model

správné slova: “udělali”

“soudruzi”

“z”

“ndr”

výstupní skóre:

-0.9  
**2.5**  
-1.9  
-3.2

-0.7  
-1.3  
**1.9**  
1.5

-0.6  
-1.7  
**1.7**  
**1.8**

0.4  
-0.3  
-0.5  
**0.6**

skrytý stav:

0.5  
-0.1  
0.2

-0.2  
-0.2  
0.8

-0.5  
-0.3  
0.5

0.2  
0.1  
-0.3

vstupní vektor:

-0.5  
0.4  
2.1  
3.14

-1.6  
1.4  
-4.5  
2.4

0.5  
0.2  
3.3  
2.2

1.1  
4.4  
-0.6  
0.11

vstupní slova:

“kde”

“udělali”

“soudruzi”

“z”

stejné parametry W jako  
v předchozím kroku

**znaky nyní můžeme  
nahradit word-vektory  
a sestavit tak slovní  
jazykový model!**

# Analýza sentimentu

- Monitorování médií, kampaně, sledování značek, recenzí, telefonních rozhovorů, ...

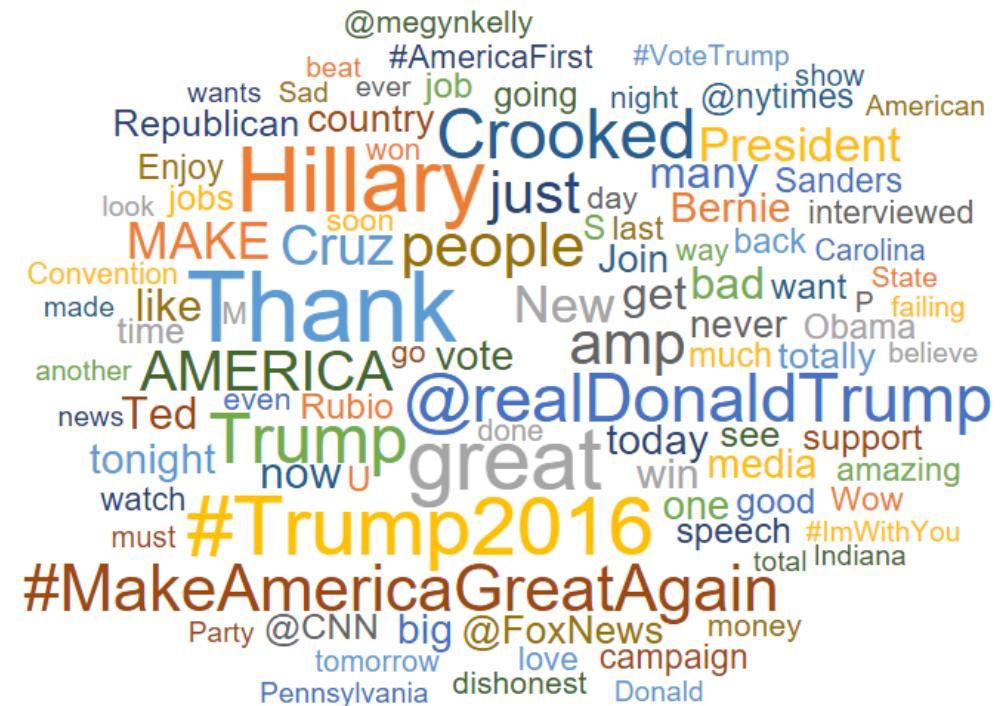
## příklad:

film je plný akce, speciálních efektů a humoru, řemeslné zpracování na jedničku, děj sice dětinský, ale zábavný

vs

přemíra akce, speciálních efektů a humoru, kvalitní zpracování, sice vcelku zábavný, ale dětinský

zpříkladu je zřejmé, že pouhé počítání kladných a záporných přívlastků stačit nebude

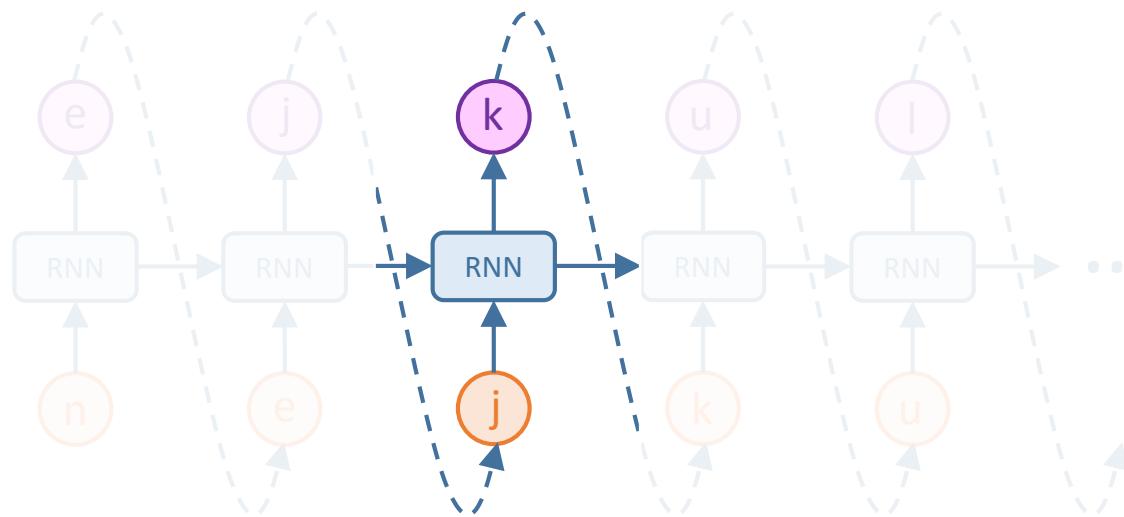


# RNN jazykový model

v každém okamžiku  $t$  síť přijme jeden symbol a ihned predikuje následující

- vstup: jeden znak / slovo
- výstup: jeden znak / slovo

label síť generuje v každém kroku



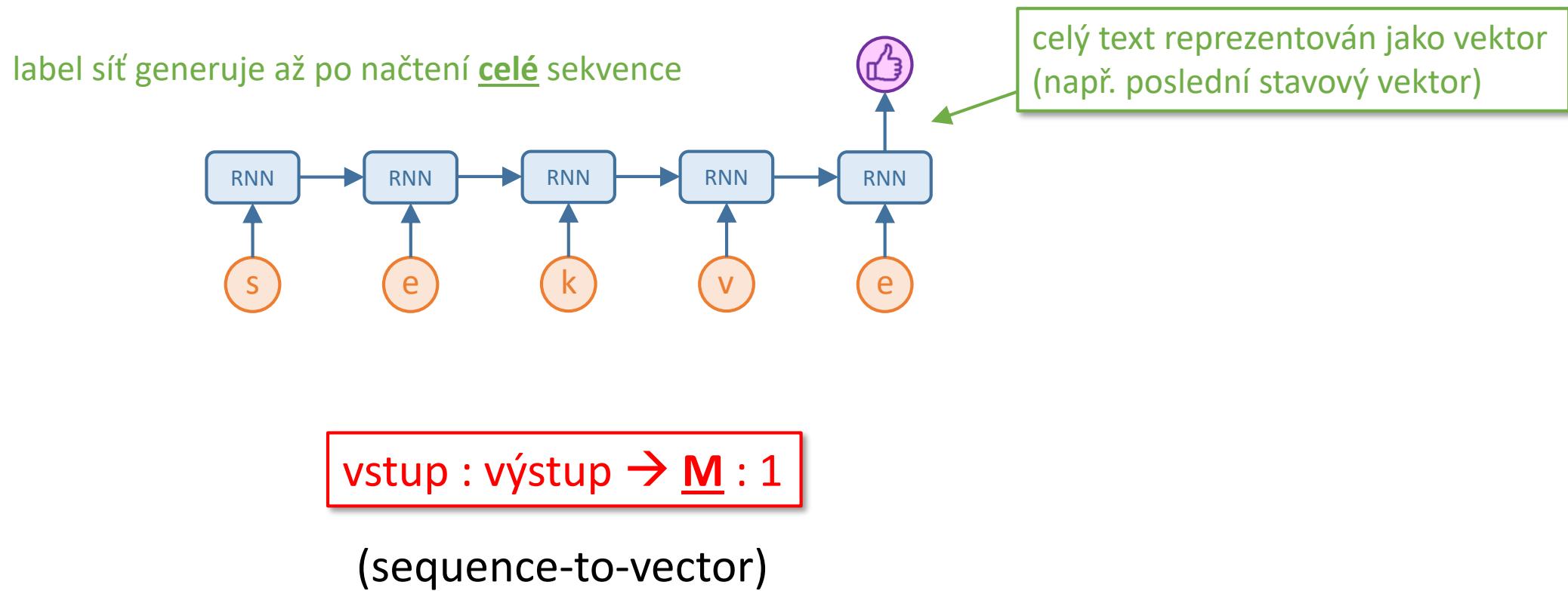
vstup : výstup → 1 : 1

(vector-to-vector)

# Klasifikace textu

např. analýza sentimentu: klasifikace celého textu do jedné z kategorií, např. pozitivní vs negativní komentář, tj. nejprve načte celý text, pak teprve predikuje výstup

- vstup: sekvence znaků / slov
- výstup: třída



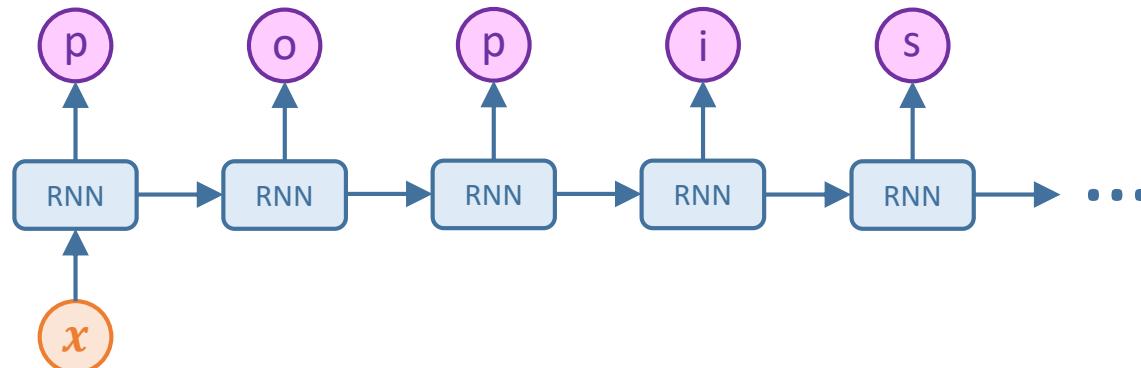
# Sentiment analysis: state of the art



# Generování textu, tagování obrázků

síť převeze vstup pouze jednou jako inicializaci, poté v každém okamžiku  $t$  síť predikuje výstup (následující znak)

- vstup: jeden vektor, např. FC7 příznaky z VGG
- výstup: sekvence znaků / slov



síť převeze pouze jeden vstup, pak už jen generuje

vstup : výstup → 1 : N

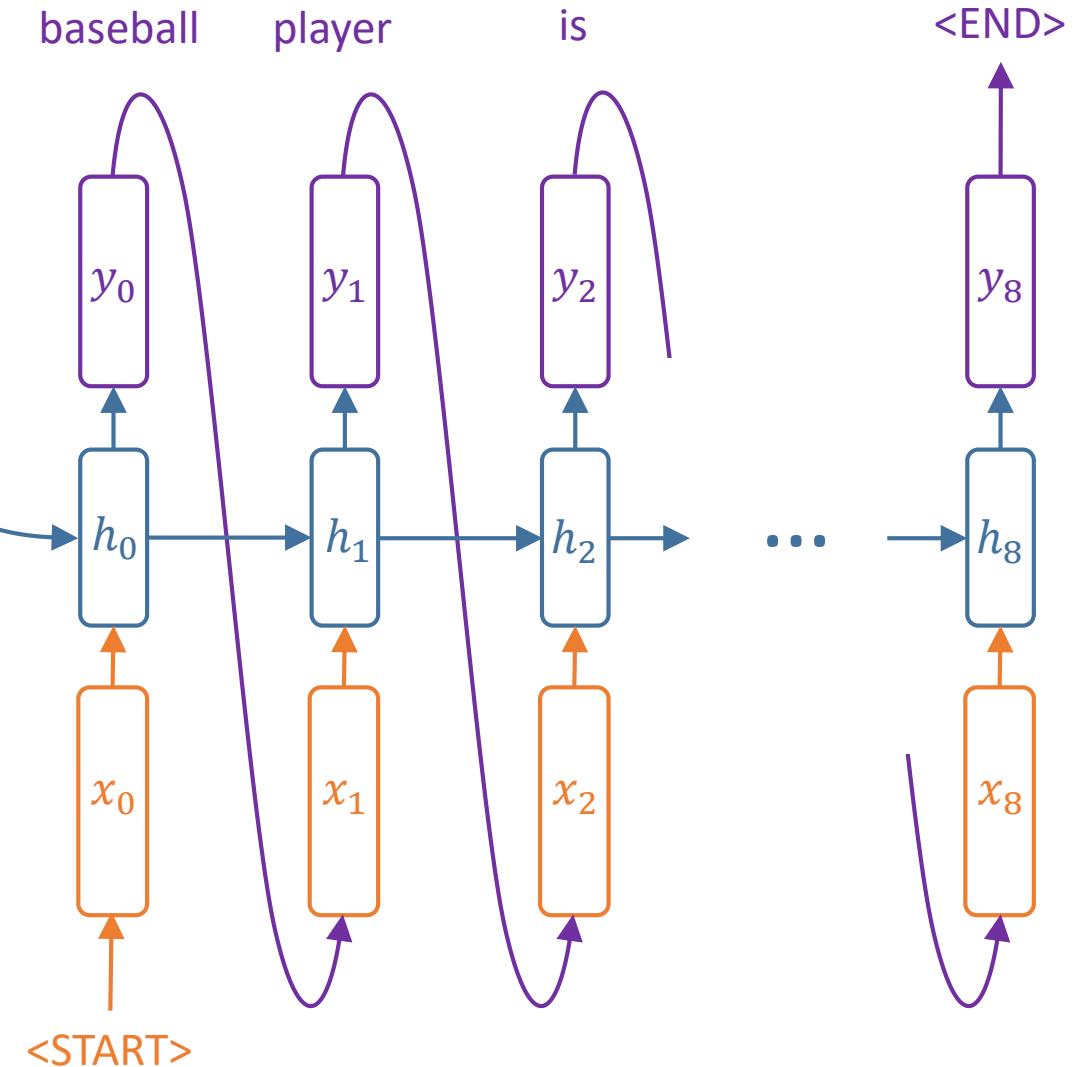
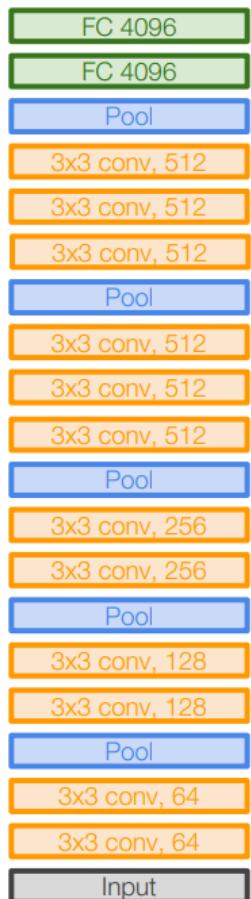
(vector-to-sequence)

# Tagování obrázků: VGG schéma



"baseball player is throwing ball  
in game."

VGG-16



# Tagování obrázků



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."



"little girl is eating piece of cake."



"baseball player is throwing ball in game."



"woman is holding bunch of bananas."



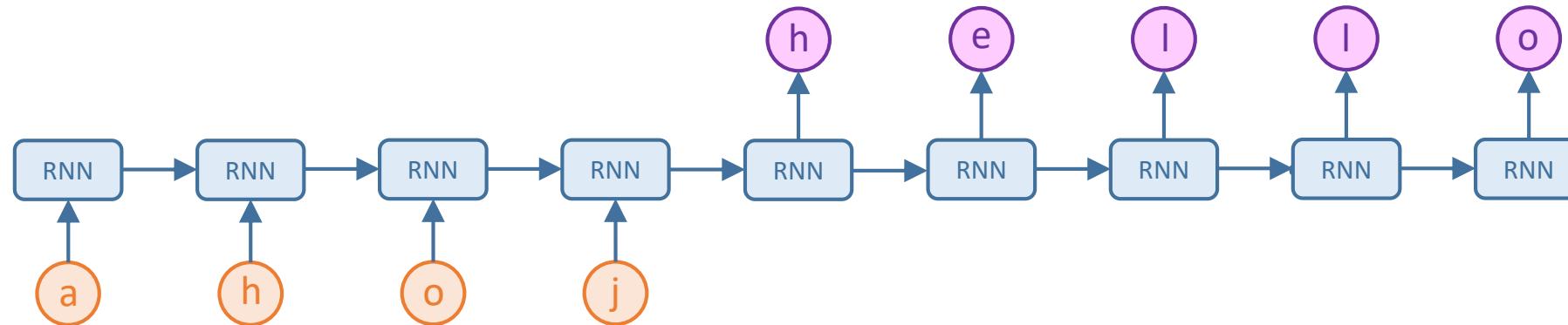
"black cat is sitting on top of suitcase."

obrázek: <https://cs.stanford.edu/people/karpathy/deepimagesent/>

# Strojový překlad

sítí nejprve "spolkne" celou vstupní sekvenci (větu), pak teprve začne generovat překlad

- vstup: sekvence znaků / slov
- výstup: sekvence znaků / slov



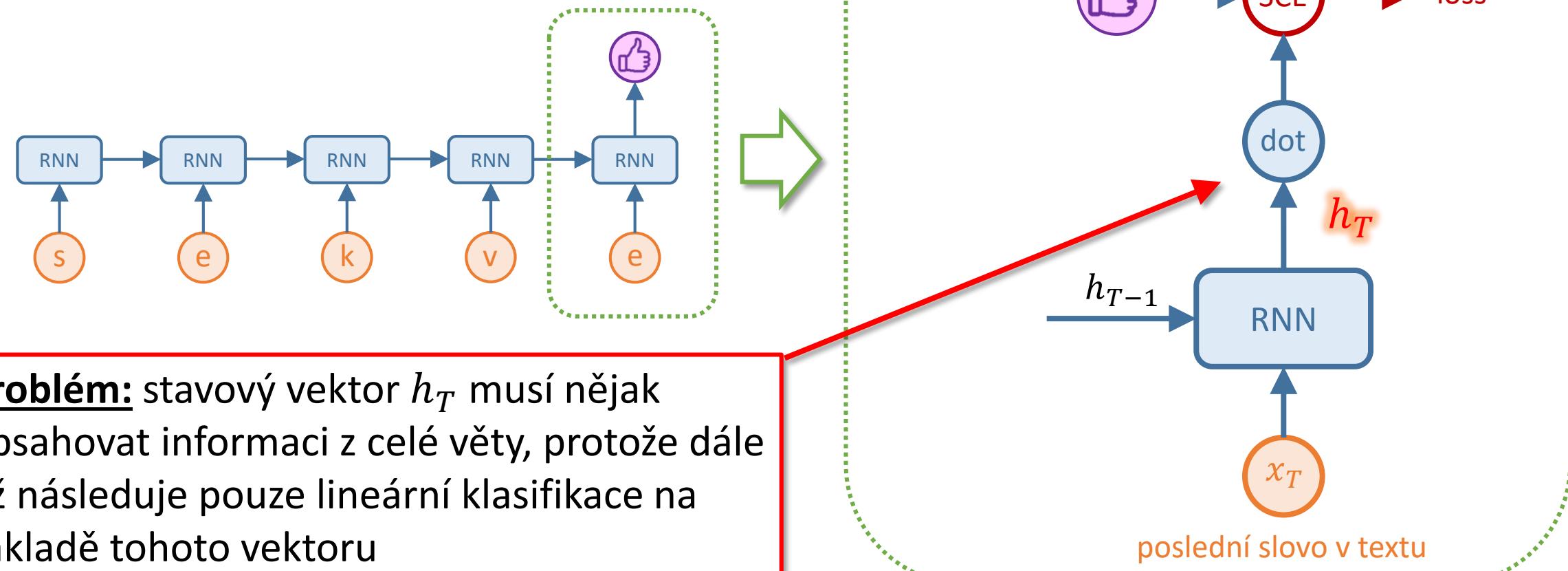
vstup : výstup → M : N

(sequence-to-sequence)

# Klasifikace textu

např. analýza sentimentu: klasifikace celého textu do jedné z kategorií, např. pozitivní vs negativní komentář

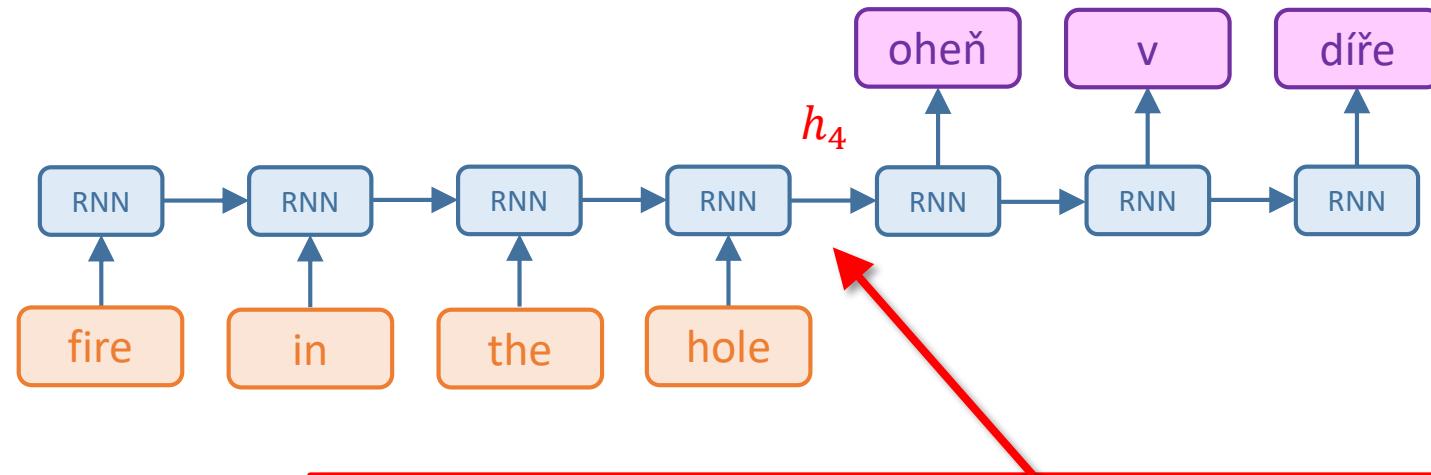
- vstup: sekvence znaků / slov
- výstup: třída



# Strojový překlad

sítí nejprve "spolkne" celou vstupní sekvenci (větu), pak teprve začne generovat překlad

- vstup: sekvence znaků / slov
- výstup: sekvence znaků / slov

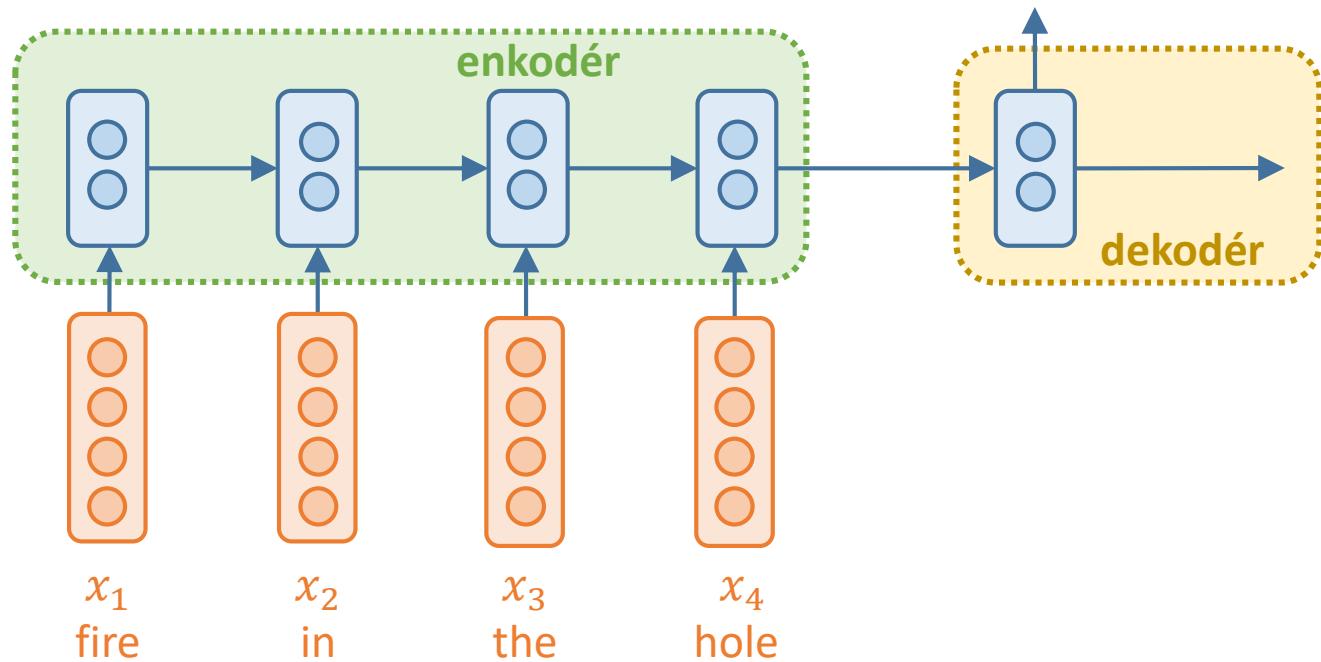


**podobný problém i zde:** na to, aby síť vygenerovala celý překlad, jí musí stačit skrytý stavový vektor  $h_4$ , slova "fire" a "oheň" jsou však od sebe vzdálena potenciálně neomezeně → problém s gradientem

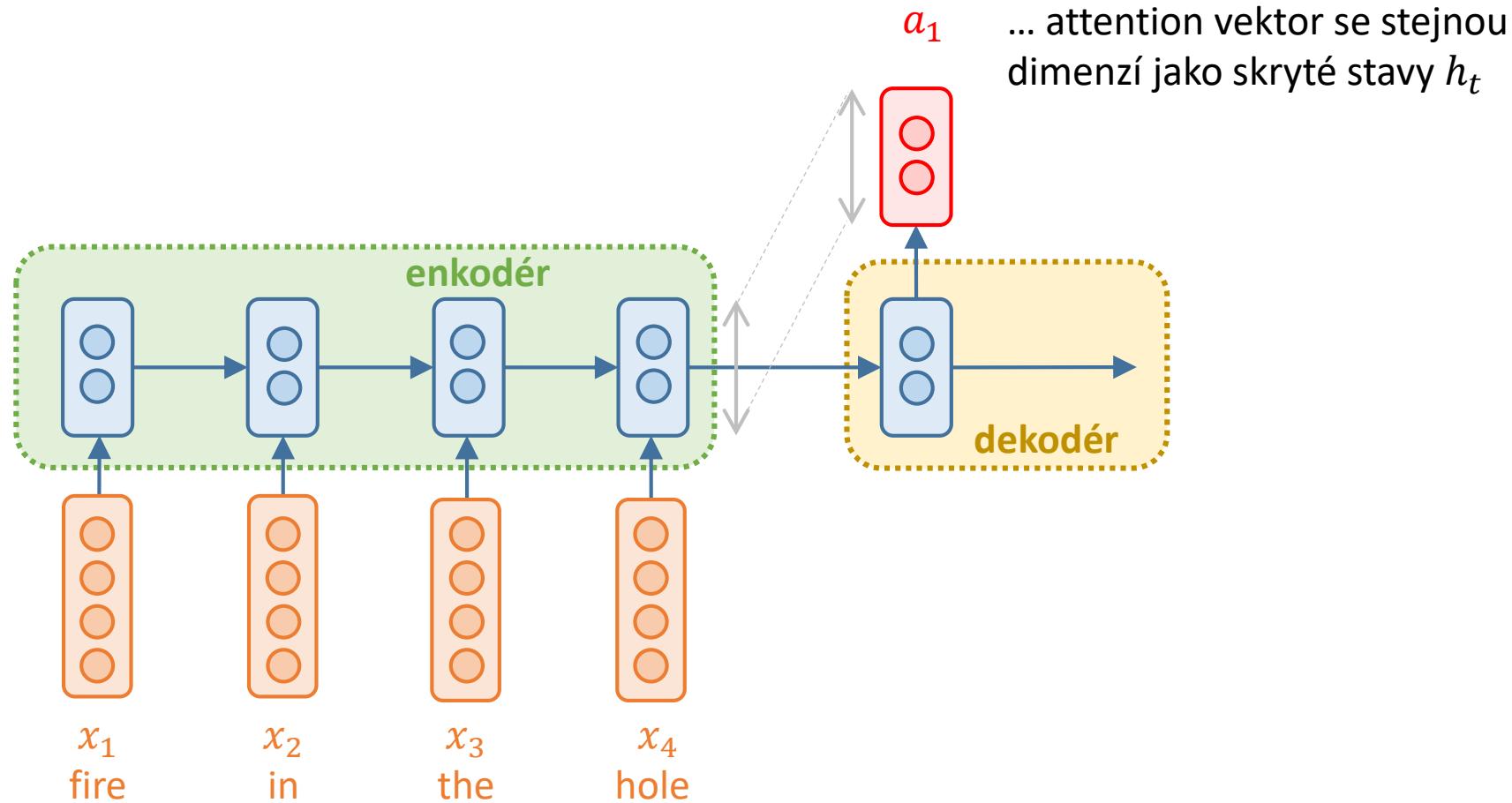
# Attention

- Problémy:
  - stavový vektor  $h_t$  musí nějak obsahovat celou větu, protože dale už následuje pouze lineární klasifikace na základě tohoto vektoru → sentence embedding
  - nezávisle na délce vstupní věty, embedding vektor má vždy stejnou dimenzi
  - navíc řešíme potenciálně velmi dlouhé závislosti, na které je v praxi i LSTM krátká
- Pokusy o řešení:
  - především v oblasti strojového překladu zadat vstupní sekvenci v obráceném pořadí
  - nechat projít vstupní sekvenci rekurentní sítí 2x
- Řešení:
  - mechanismus paměti/soustředěnosti (attention)
  - zapamatovat a při rozhodování **váhově zohlednit všechny skryté stavy**, ne jen ten poslední

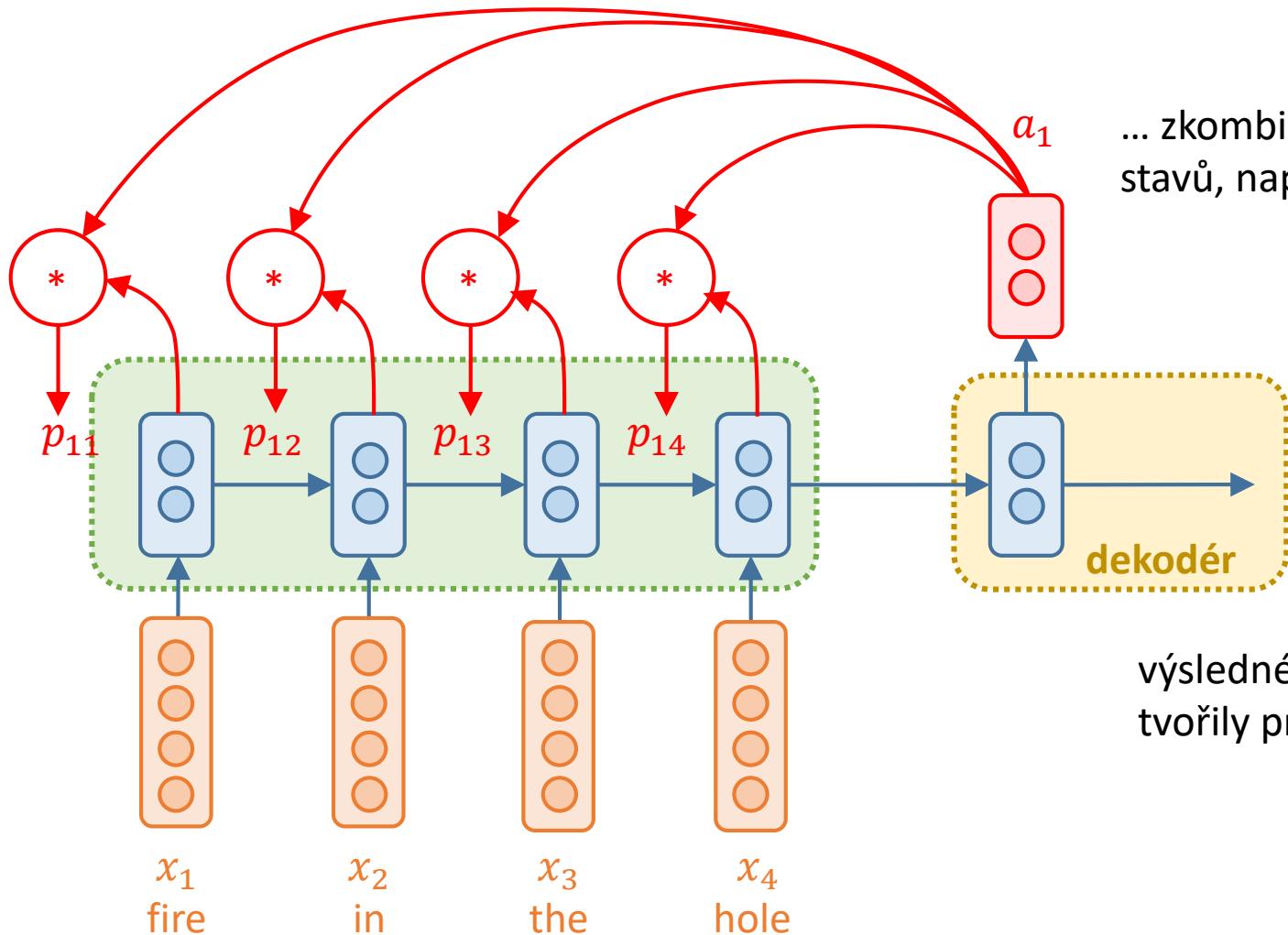
# Princip attention pro strojový překlad



# Princip attention pro strojový překlad



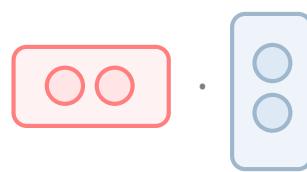
# Princip attention pro strojový překlad



... zkombinuje se s vektory skrytých stavů, např. jako skalární součin

$$s_{11} = a_1^T \cdot h_1$$

$$\dots$$
  
$$s_{14} = a_1^T \cdot h_4$$



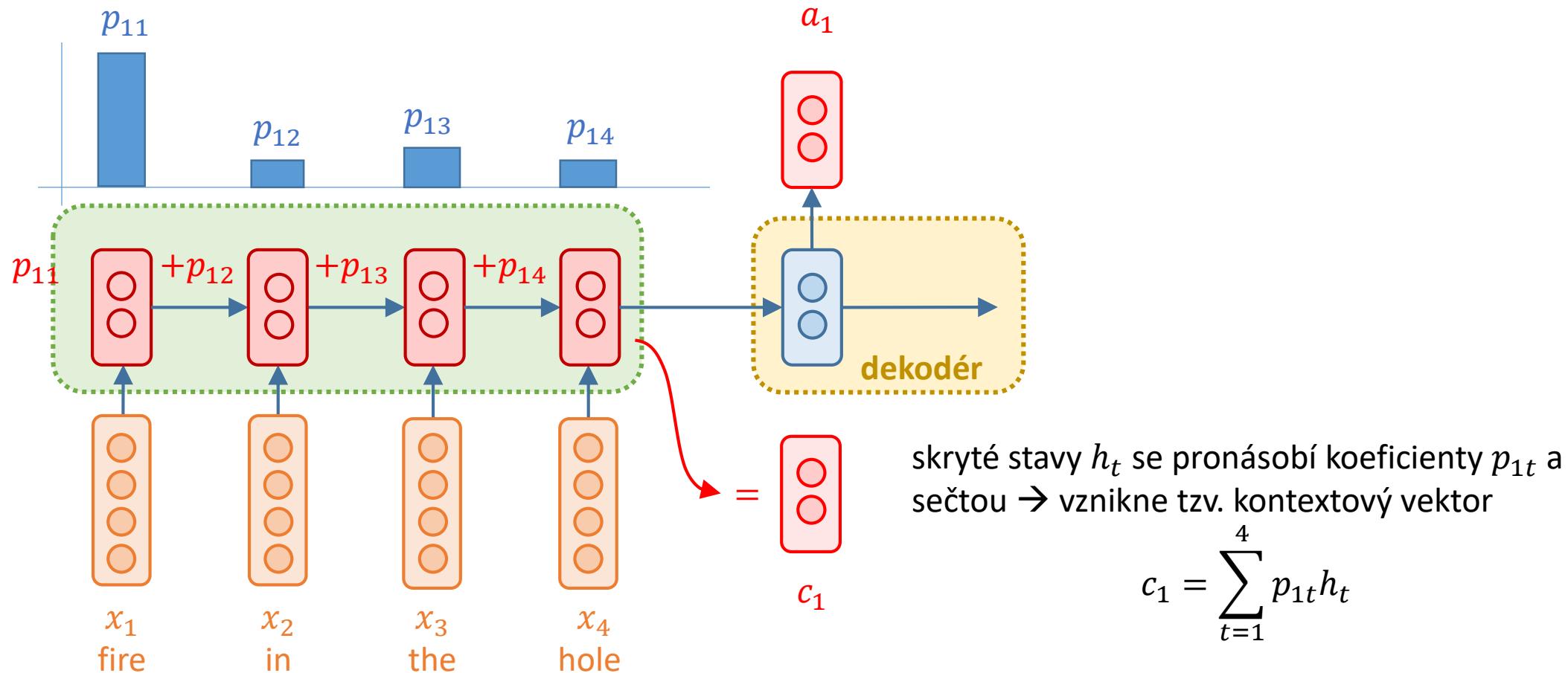
výsledné koeficienty  $p_{1t}$  se znormalizují tak, aby tvořily pravděpodobnosotní distribuci a tedy

$$p_{1t} = \text{Softmax}(s_{11}, s_{12}, s_{13}, s_{14})$$

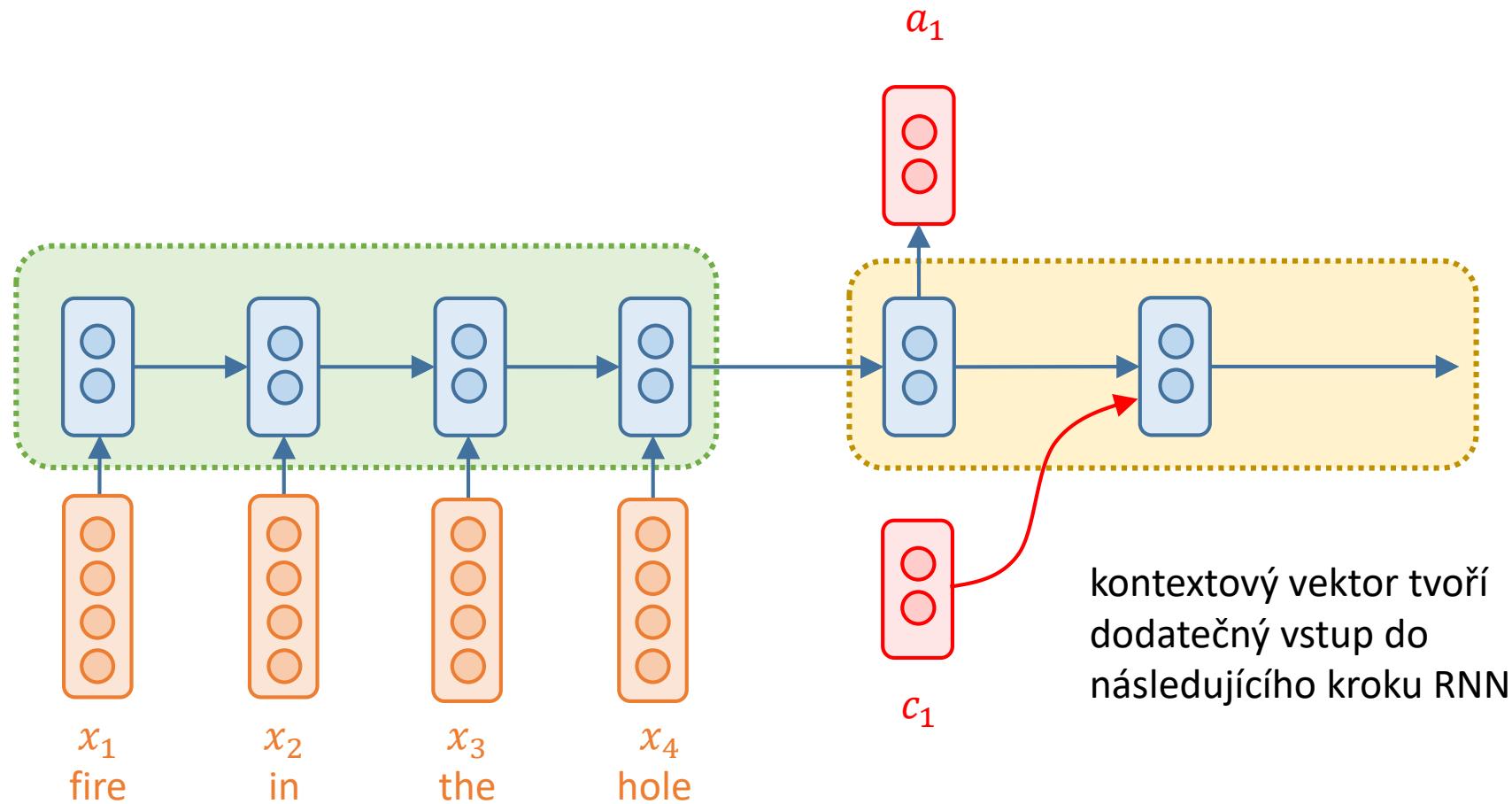
$$\sum_{t=1}^4 p_{1t} = 1$$

# Princip attention pro strojový překlad

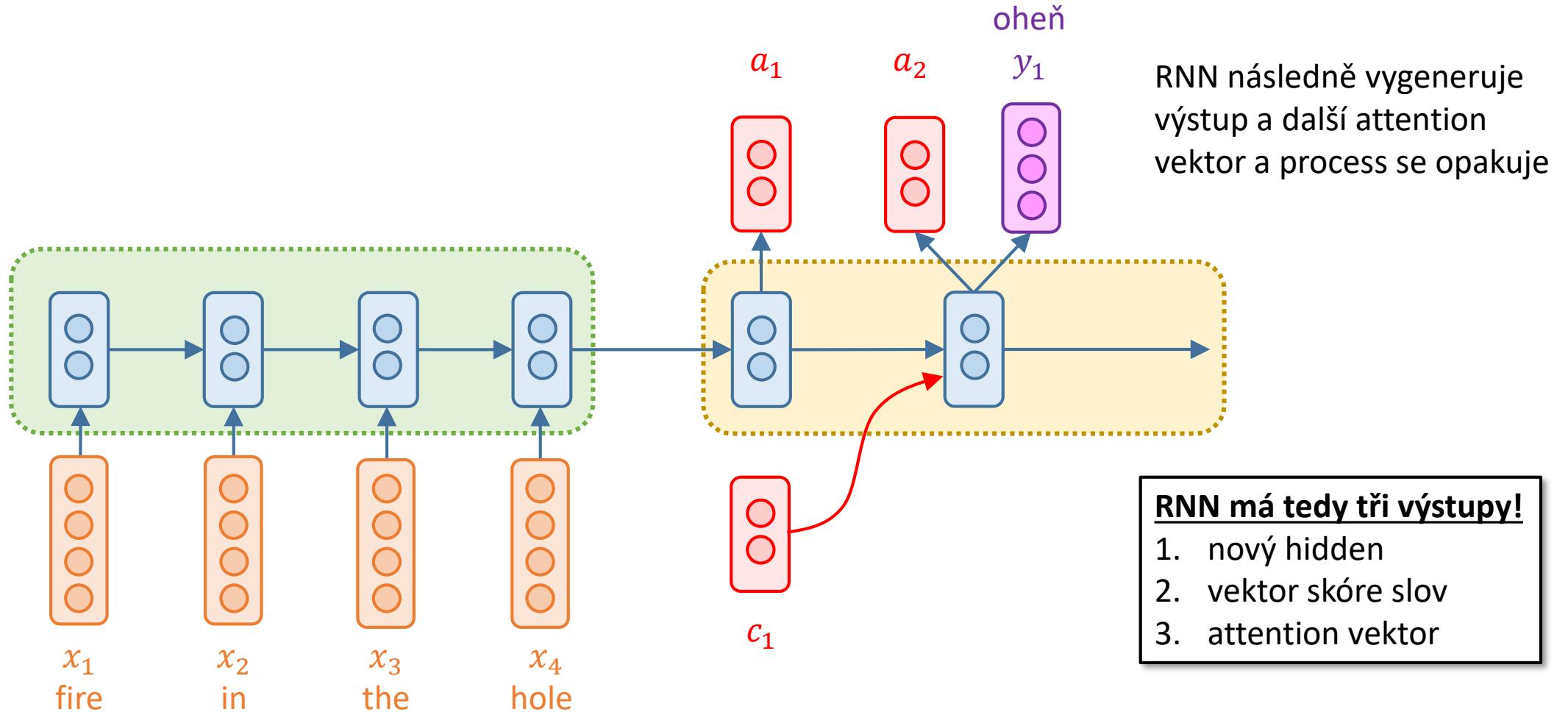
tam, kde jsou si attention vektor  $a_1$  a skrytý stav  $h_t$   
podobné → velké  $p_{1t}$  = "soustředění pozornosti"



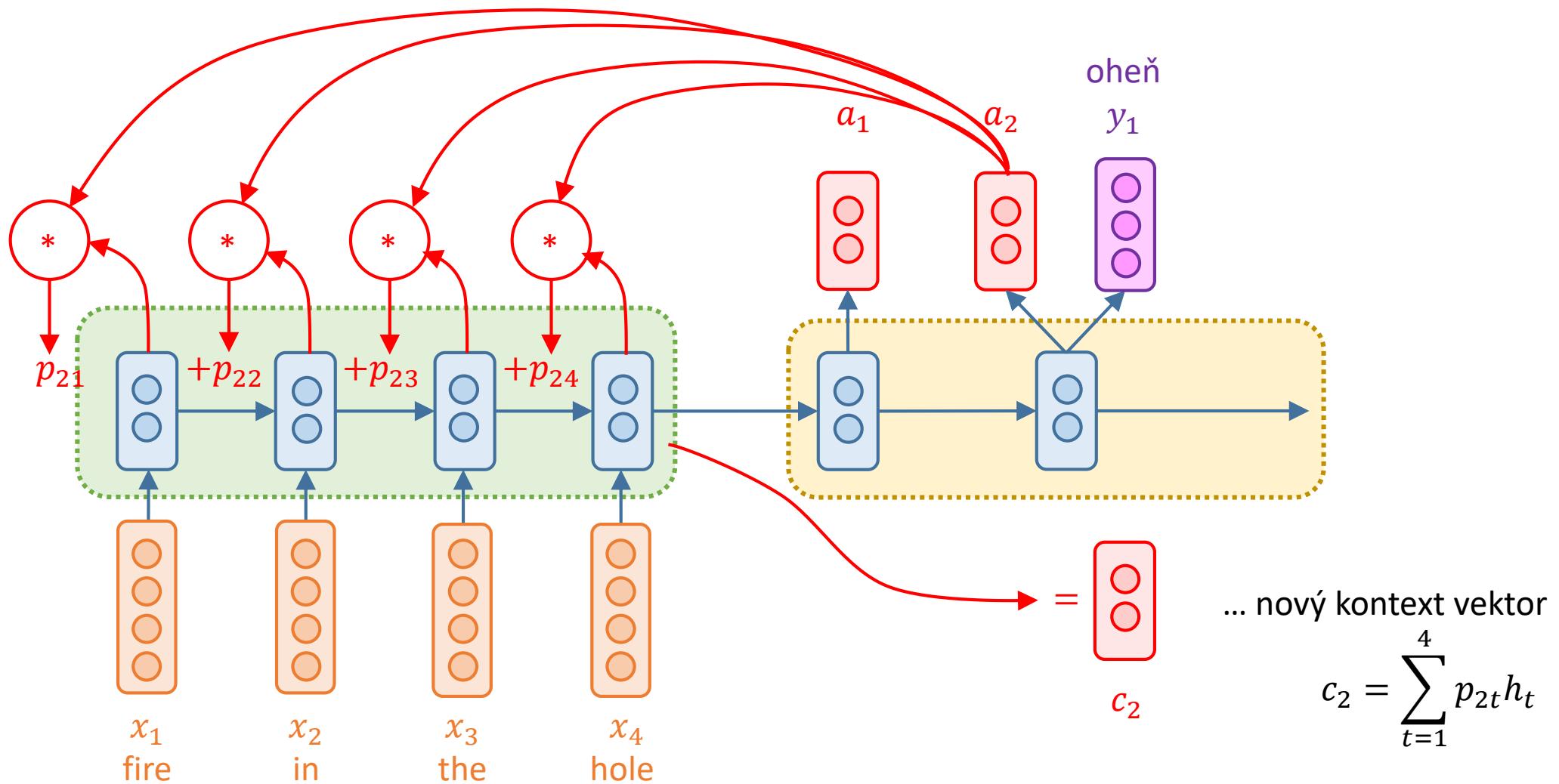
# Princip attention pro strojový překlad



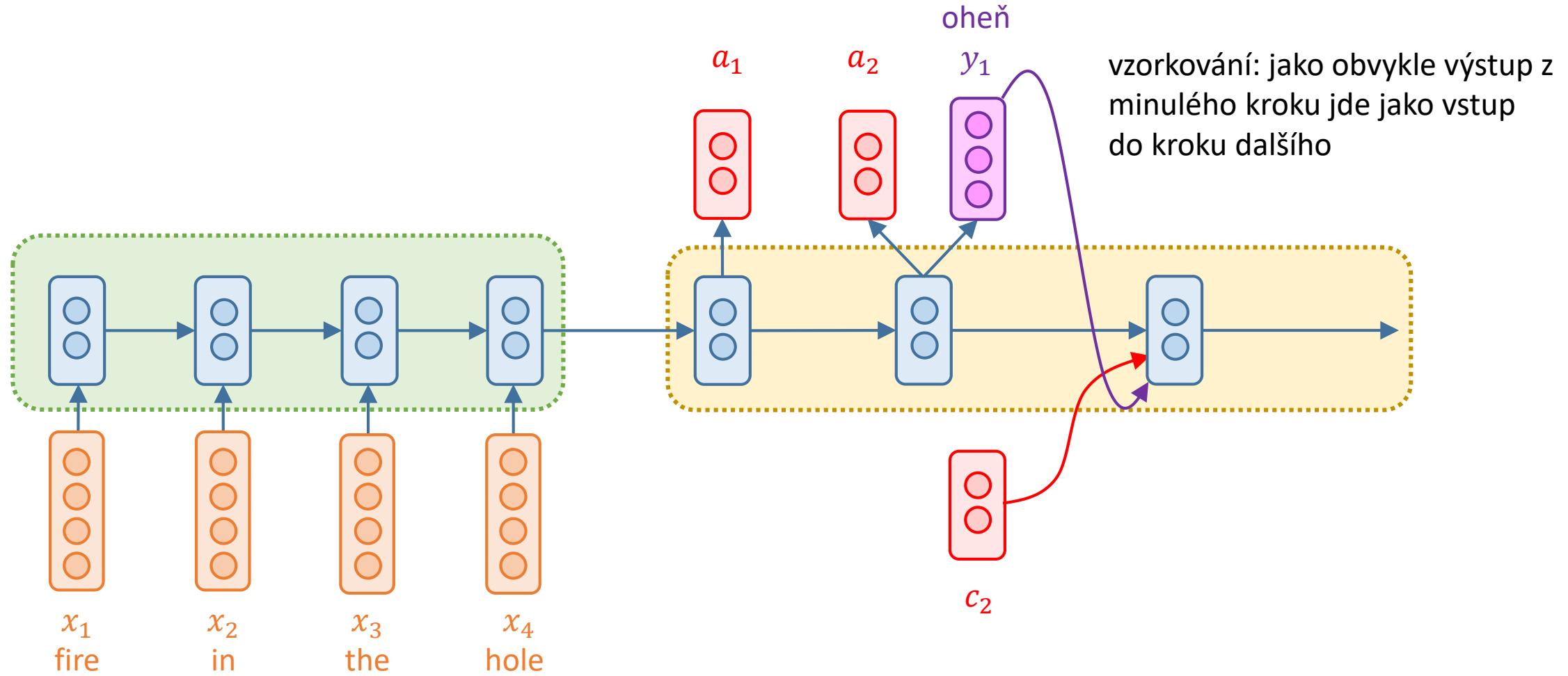
# Princip attention pro strojový překlad



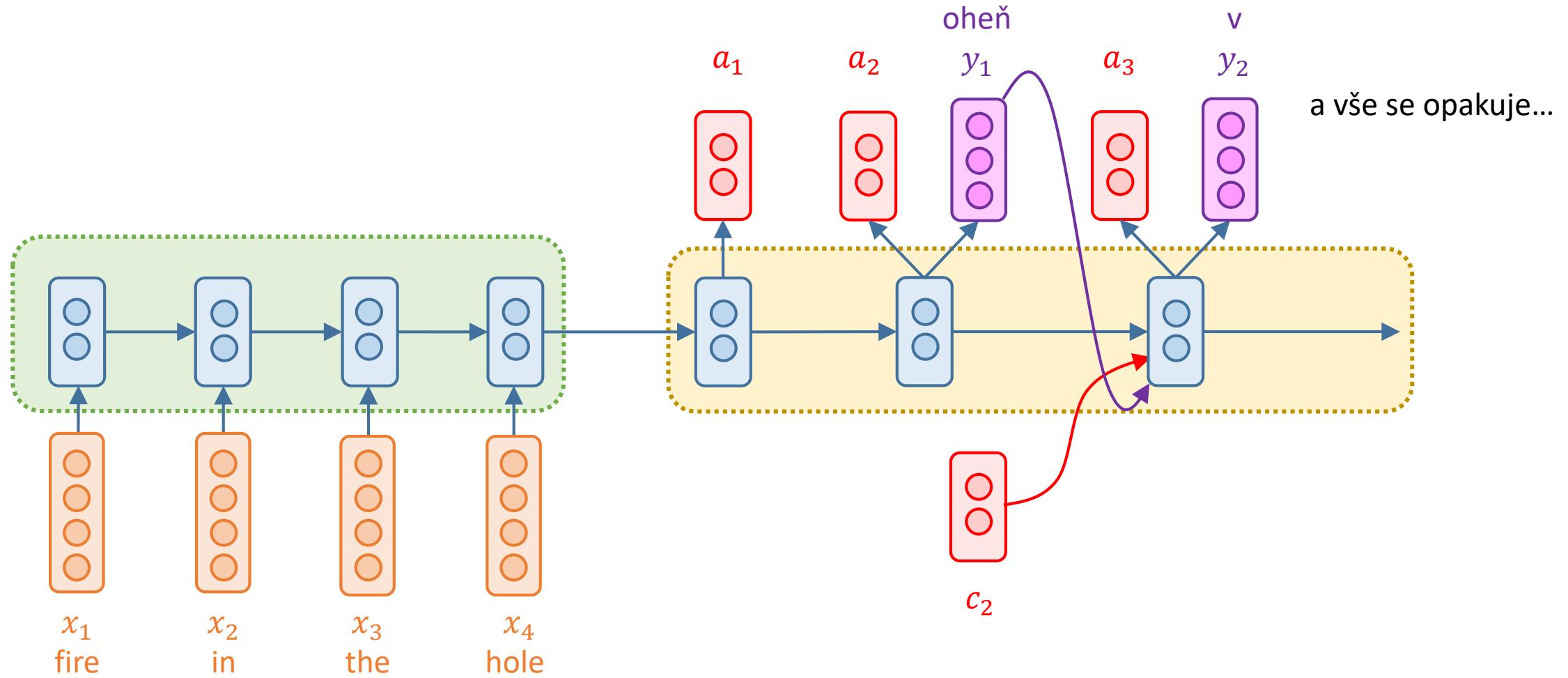
# Princip attention pro strojový překlad



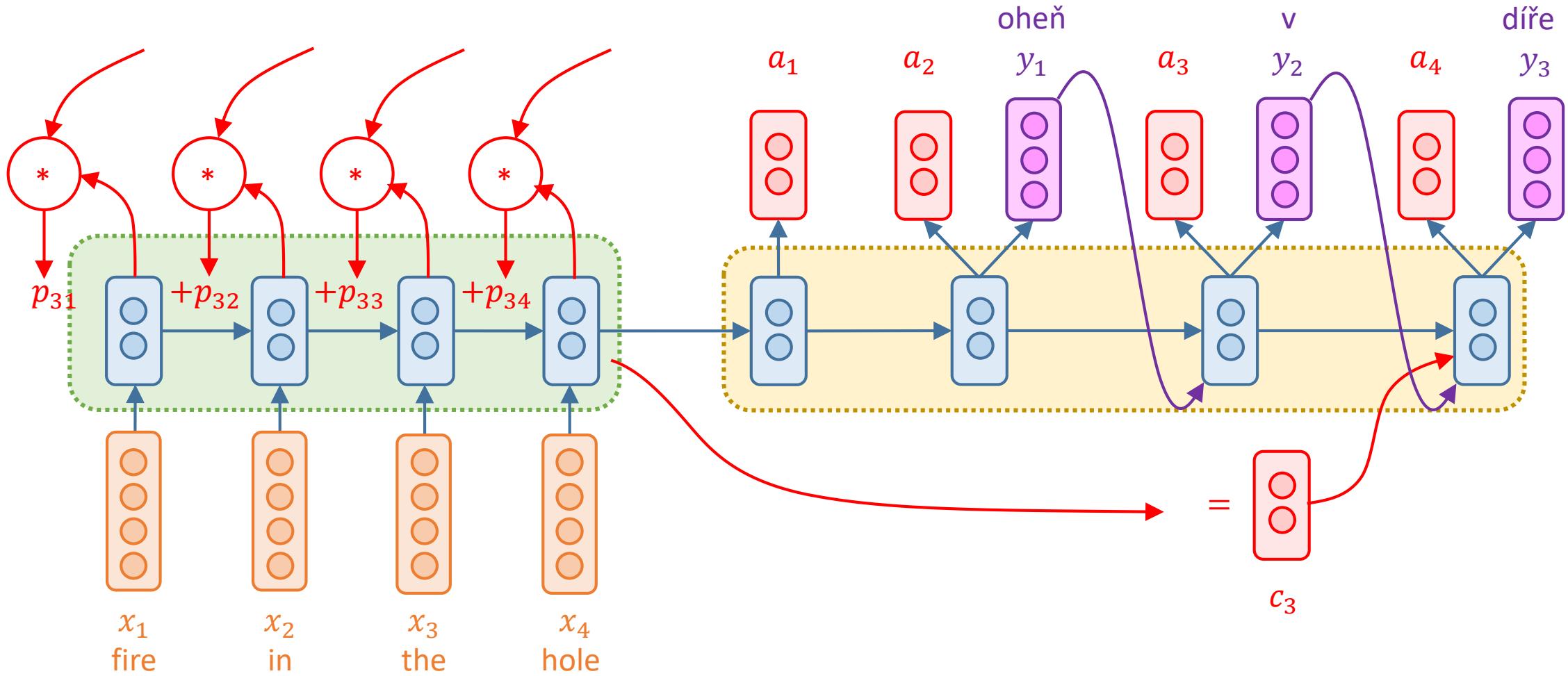
# Princip attention pro strojový překlad



# Princip attention pro strojový překlad

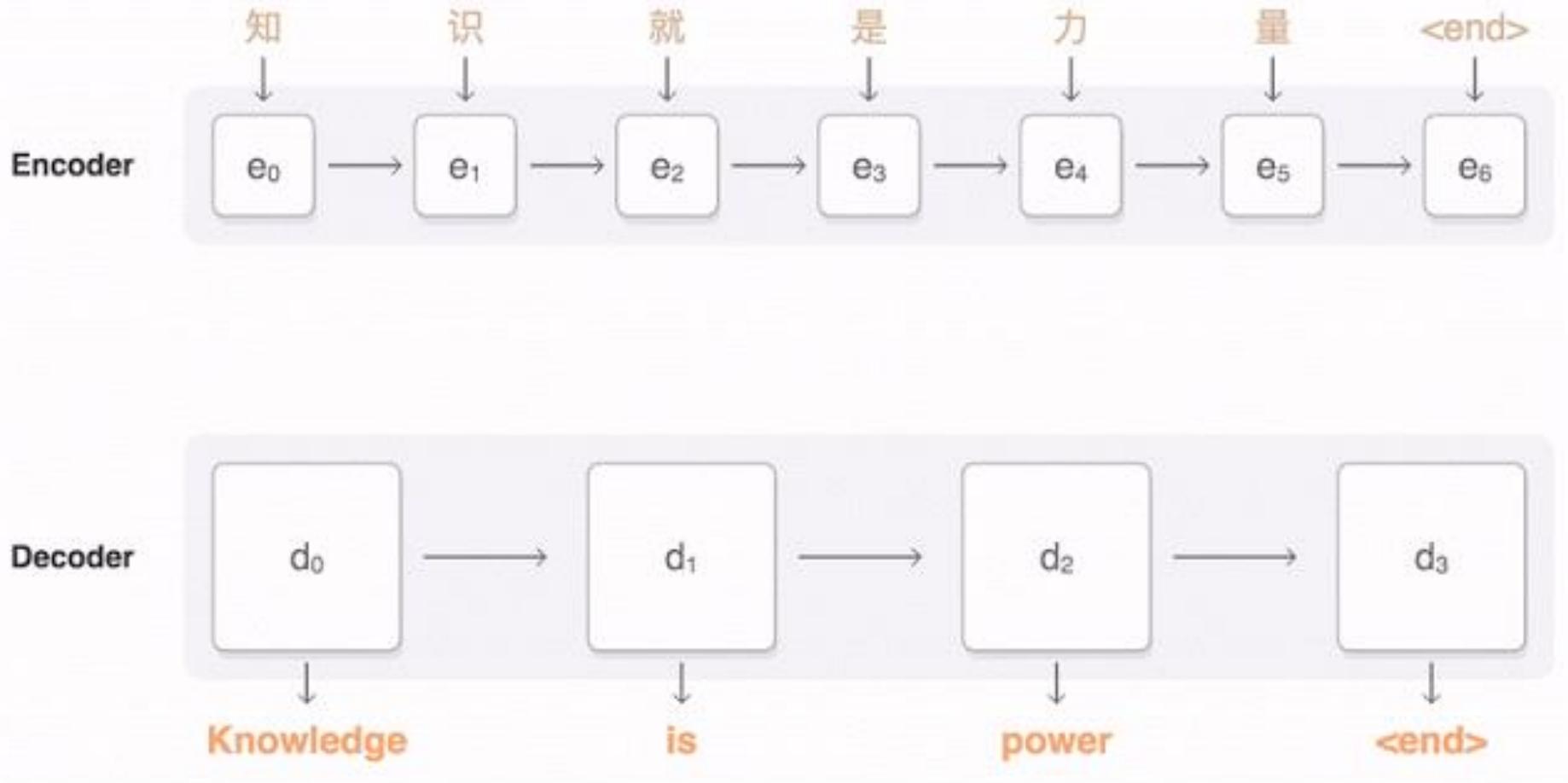


# Princip attention pro strojový překlad



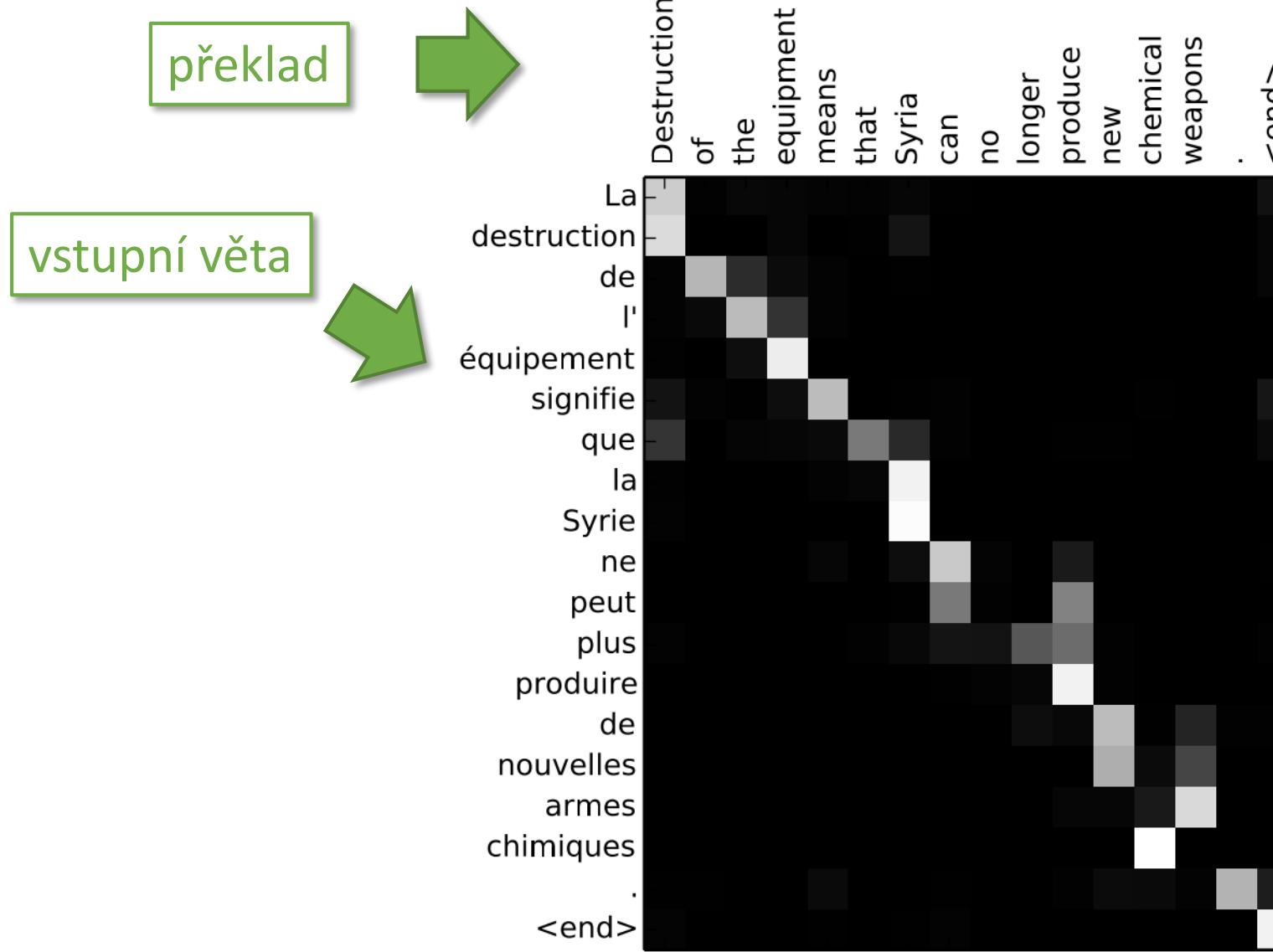
# Attention pro strojový překlad: animace

(ne)průhlednost čar znázorňuje váhu = výsledné koeficienty  $p_t$



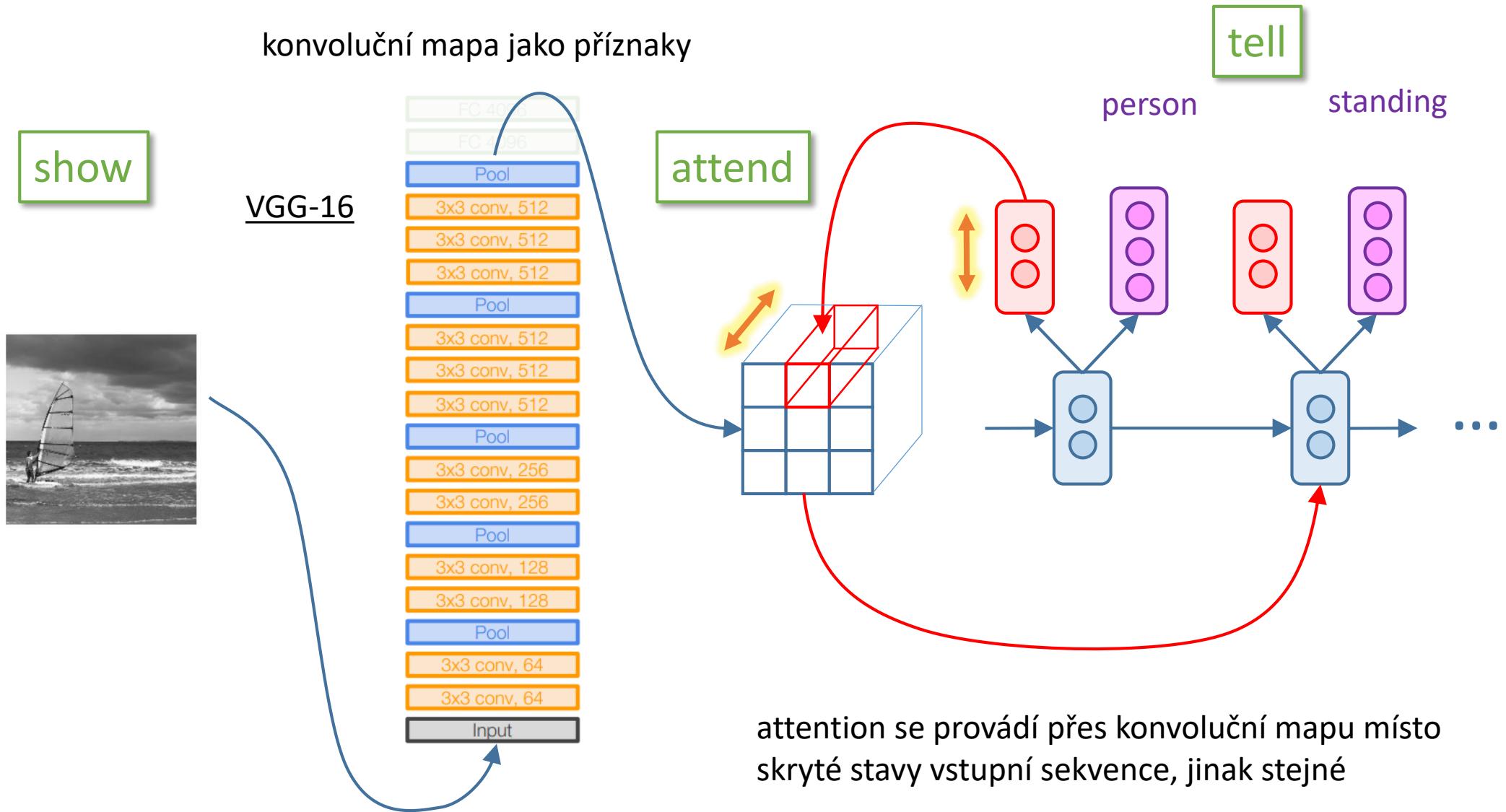
animace: <https://research.googleblog.com/2016/09/a-neural-network-for-machine.html>

# Vizualizace attention vah pro strojový překlad

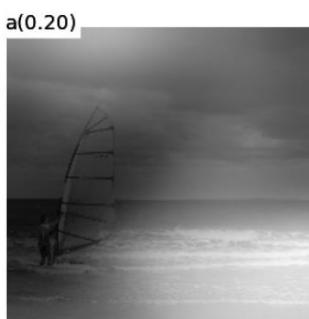
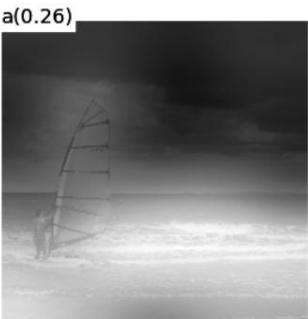


- vysoké váhy znázorňují, na která slova překládací síť zaměřuje pozornost
- na mechanismus lze nahlížet také jako na “soft” asociativní paměť
- attention vektor  $a$  je query, kontext vektor  $c$  pak navrácená hodnota
- s rozdílem, že  $c$  je složeninou celé paměti danou vahami  $p$

# Attention pro obrazová data: show, attend and tell



# Vizualizace attention vah pro obrazová data



- váhy jsou přes konvoluční mapy
- menší rozměr než vstupní obrázek
- z vrchní vrstvy zpětně dopočteno do obrázku
- obrázky znázorňují, na jaké části obrázku se síť při generování jednotlivých slov soustředí

(b) A person is standing on a beach with a surfboard.

# Automatické odezírání ze rtů: listen/watch, attend and spell

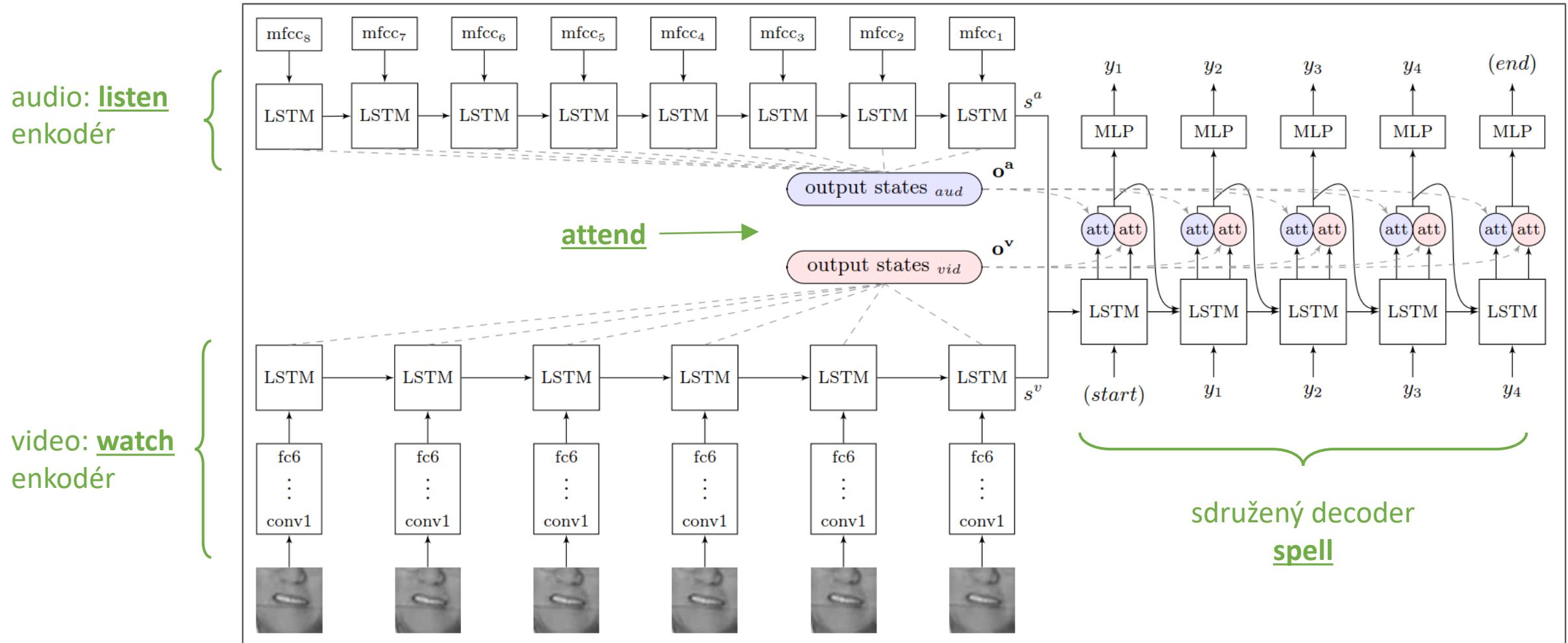


Figure 1. *Watch, Listen, Attend and Spell* architecture. At each time step, the decoder outputs a character  $y_i$ , as well as two attention vectors. The attention vectors are used to select the appropriate period of the input visual and audio sequences.

obrázek: [Chung, Zisserman: Lip Reading Sentences in the Wild](#)

# Automatické odezírání ze rtů

## trénovací data

Channel	Series name	# hours	# sent.
BBC 1 HD	News <sup>†</sup>	1,584	50,493
BBC 1 HD	Breakfast	1,997	29,862
BBC 1 HD	Newsnight	590	17,004
BBC 2 HD	World News	194	3,504
BBC 2 HD	Question Time	323	11,695
BBC 4 HD	World Today	272	5,558
All		4,960	118,116

Table 1. Video statistics. The number of hours of the original BBC video; the number of sentences with full facetrack. <sup>†</sup>BBC News at 1, 6 and 10.

člověk expert!

výsledky: CER, WER ... chybovost (méně = lépe)

Method	SNR	CER	WER	BLEU <sup>†</sup>
<b>Lips only</b>				
Professional <sup>‡</sup>	-	58.7%	73.8%	23.8
WAS	-	59.9%	76.5%	35.6
WAS+CL	-	47.1%	61.1%	46.9
WAS+CL+SS	-	44.2%	59.2%	48.3
WAS+CL+SS+BS	-	42.1%	53.2%	53.8
<b>Audio only</b>				
LAS+CL+SS+BS	clean	16.2%	26.9%	76.7
LAS+CL+SS+BS	10dB	33.7%	48.8%	58.6
LAS+CL+SS+BS	0dB	59.0%	74.5%	38.6
<b>Audio and lips</b>				
WLAS+CL+SS+BS	clean	13.3%	22.8%	79.9
WLAS+CL+SS+BS	10dB	22.8%	35.1%	69.6
WLAS+CL+SS+BS	0dB	35.8%	50.8%	57.5

Table 5. Performance on the LRS test set. **WAS**: Watch, Attend and Spell; **LAS**: Listen, Attend and Spell; **WLAS**: Watch, Listen, Attend and Spell; **CL**: Curriculum Learning; **SS**: Scheduled Sampling; **BS**: Beam Search. <sup>†</sup>Unigram BLEU with brevity penalty.

<sup>‡</sup>Excluding samples that the lip reader declined to annotate. Including these, the CER rises to 78.9% and the WER to 87.6%.

# Automatické odezírání ze rtů

příklady rozpoznaných vět **pouze z videa**

MANY MORE PEOPLE WHO WERE INVOLVED IN THE ATTACKS
CLOSE TO THE EUROPEAN COMMISSION'S MAIN BUILDING
WEST WALES AND THE SOUTH WEST AS WELL AS WESTERN SCOTLAND
WE KNOW THERE WILL BE HUNDREDS OF JOURNALISTS HERE AS WELL
ACCORDING TO PROVISIONAL FIGURES FROM THE ELECTORAL COMMISSION
THAT'S THE LOWEST FIGURE FOR EIGHT YEARS
MANCHESTER FOOTBALL CORRESPONDENT FOR THE DAILY MIRROR
LAYING THE GROUNDS FOR A POSSIBLE SECOND REFERENDUM
ACCORDING TO THE LATEST FIGURES FROM THE OFFICE FOR NATIONAL STATISTICS
IT COMES AFTER A DAMNING REPORT BY THE HEALTH WATCHDOG

Table 6. Examples of unseen sentences that WAS correctly predicts (lips only).

# Transformer

- [Vaswani et al: Attention is all you need](#)
- RNN zpracovávají vstupy jeden po druhém → jsou hůře paralelizovatelné
- Ukazuje se, že rekurentní sítě lze úplně vynechat a vše řešit pouze attention!
- Lepší paralelizace + zkrácení délky cesty mezi závislými slovy ve větě → naučí se opravdu dlouhé závislosti
- State of the art výsledky v NLP
- Nevýhoda: seq2seq modely s LSTM transformer překonává pouze na opravdu velkých datasetech

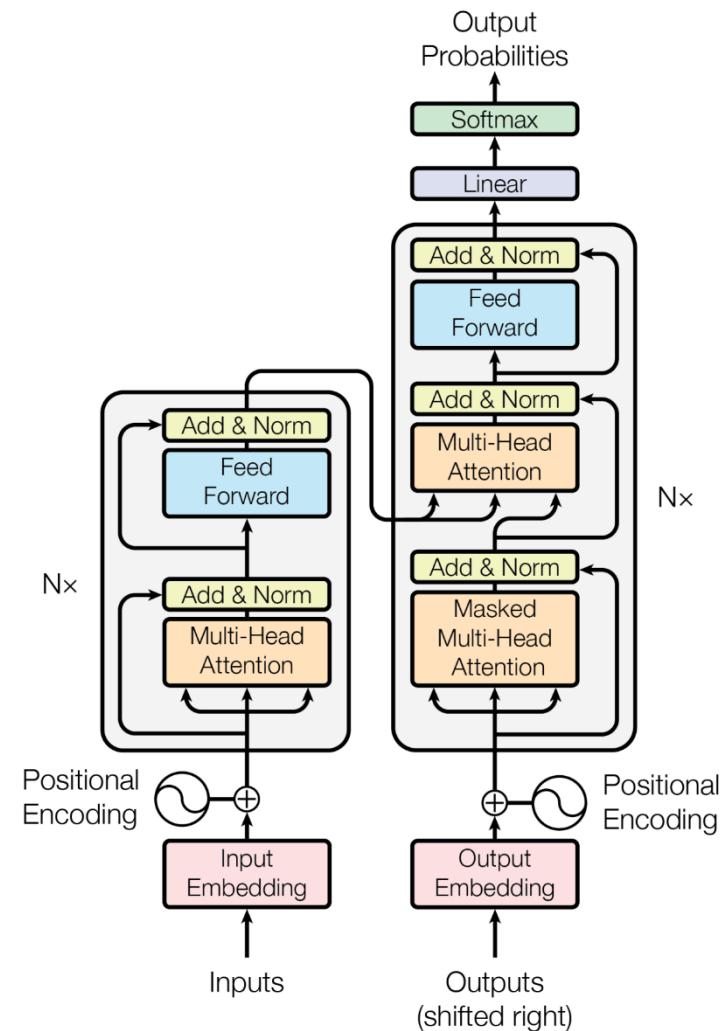
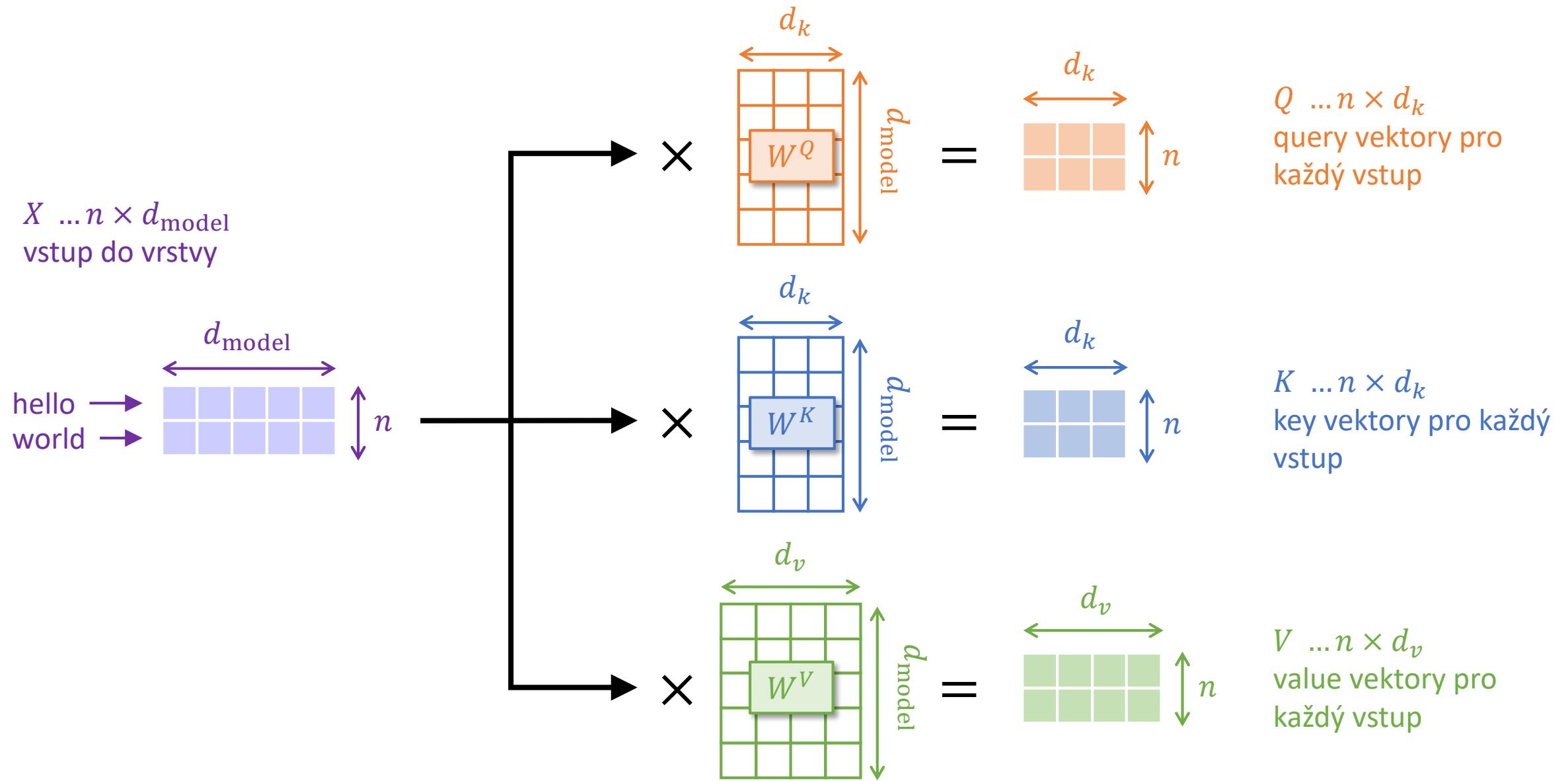
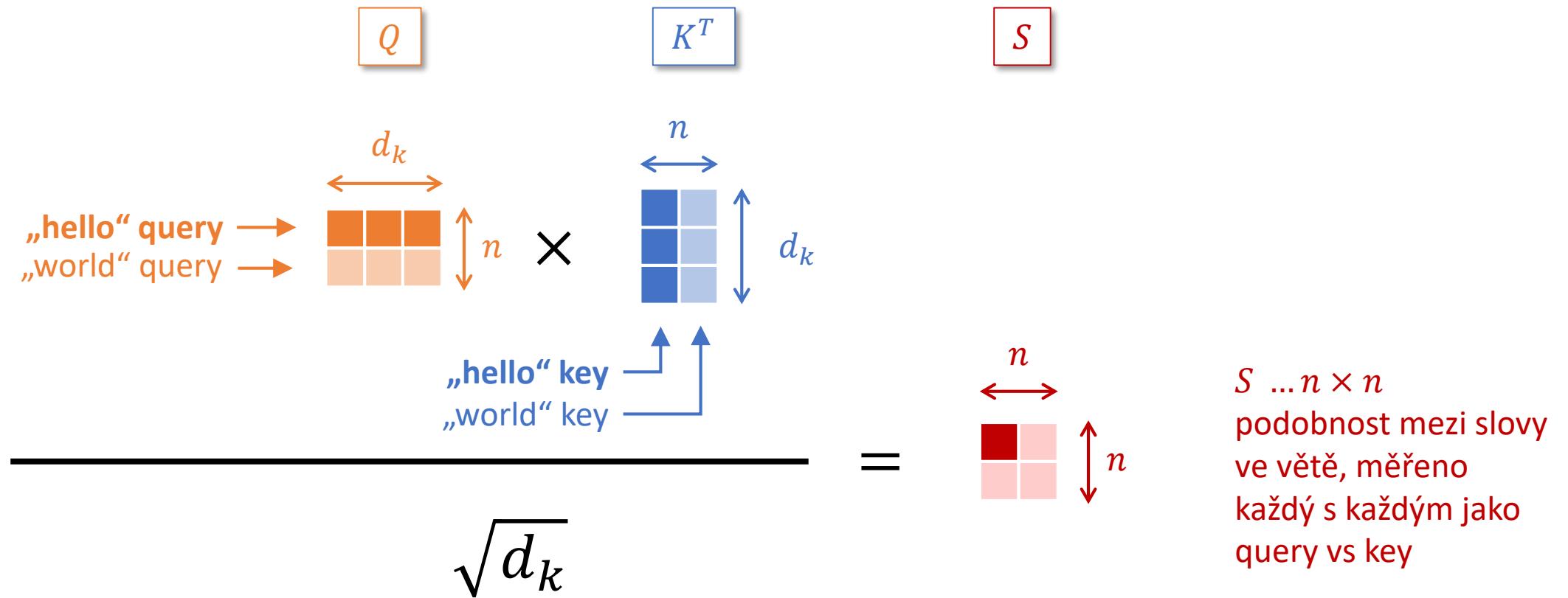


Figure 1: The Transformer - model architecture.

# Self attention

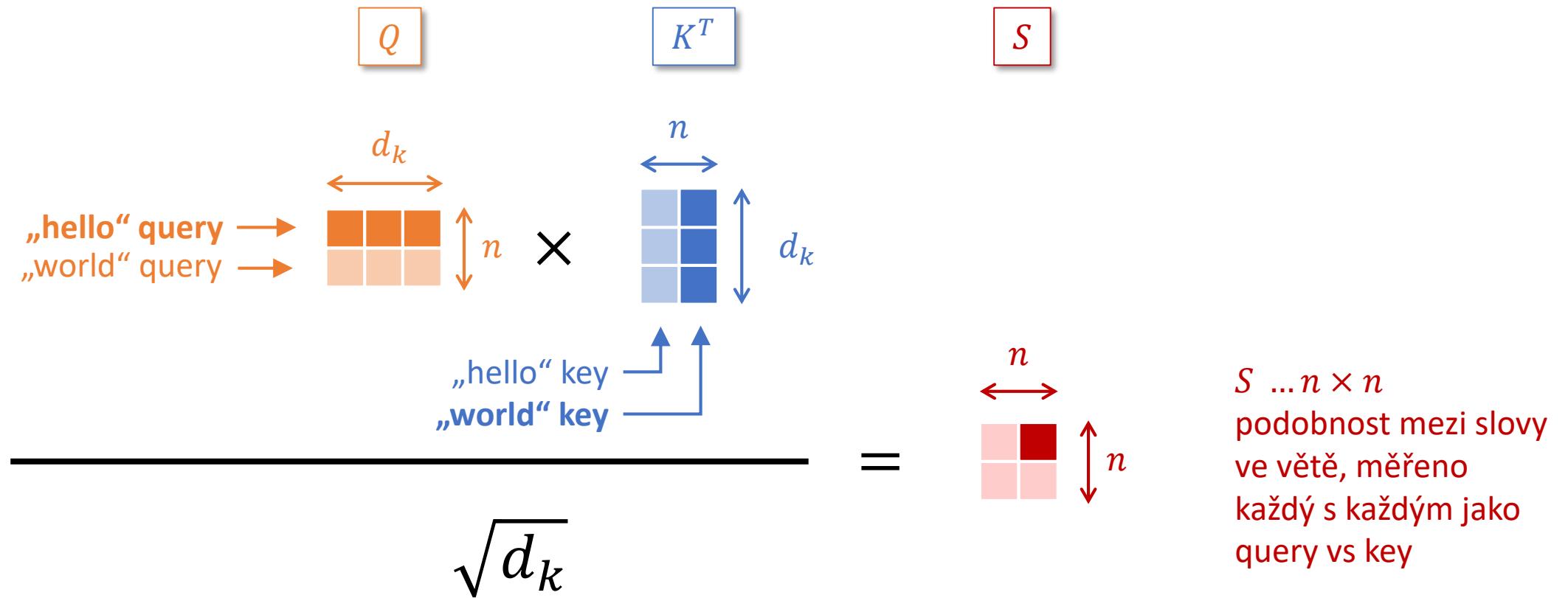


# Self attention



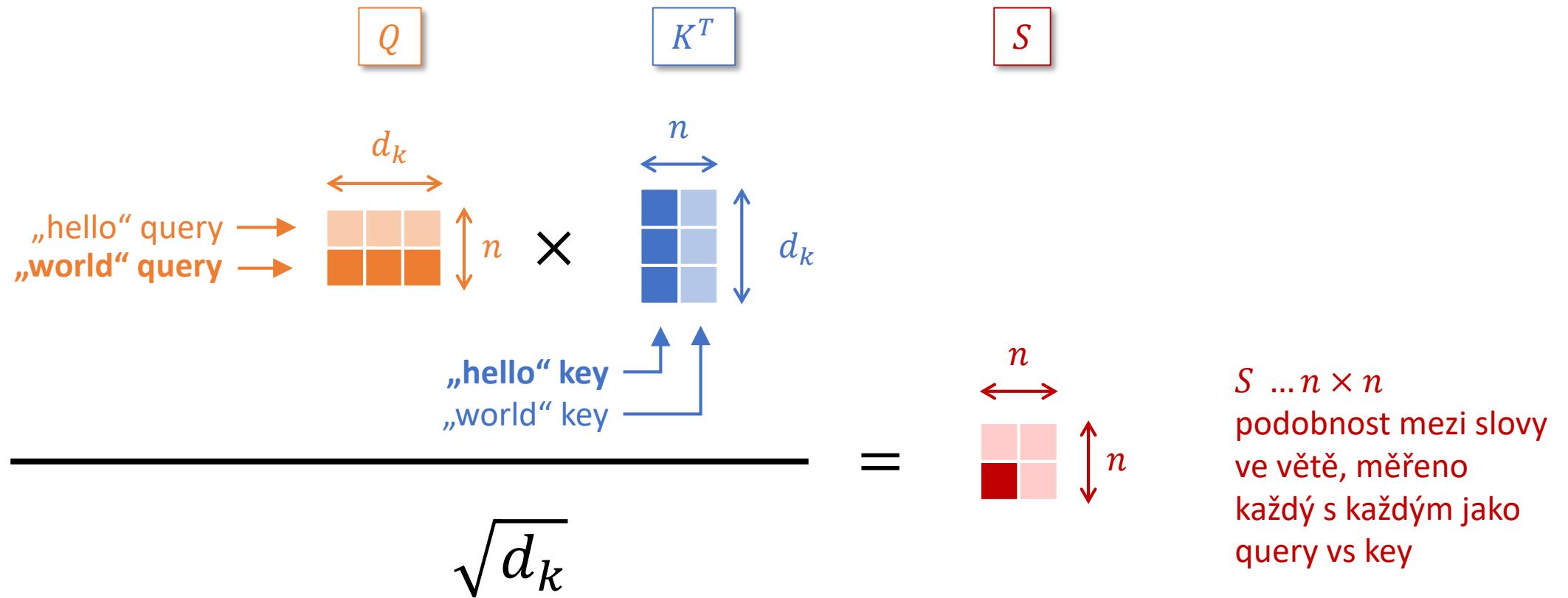
$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

# Self attention



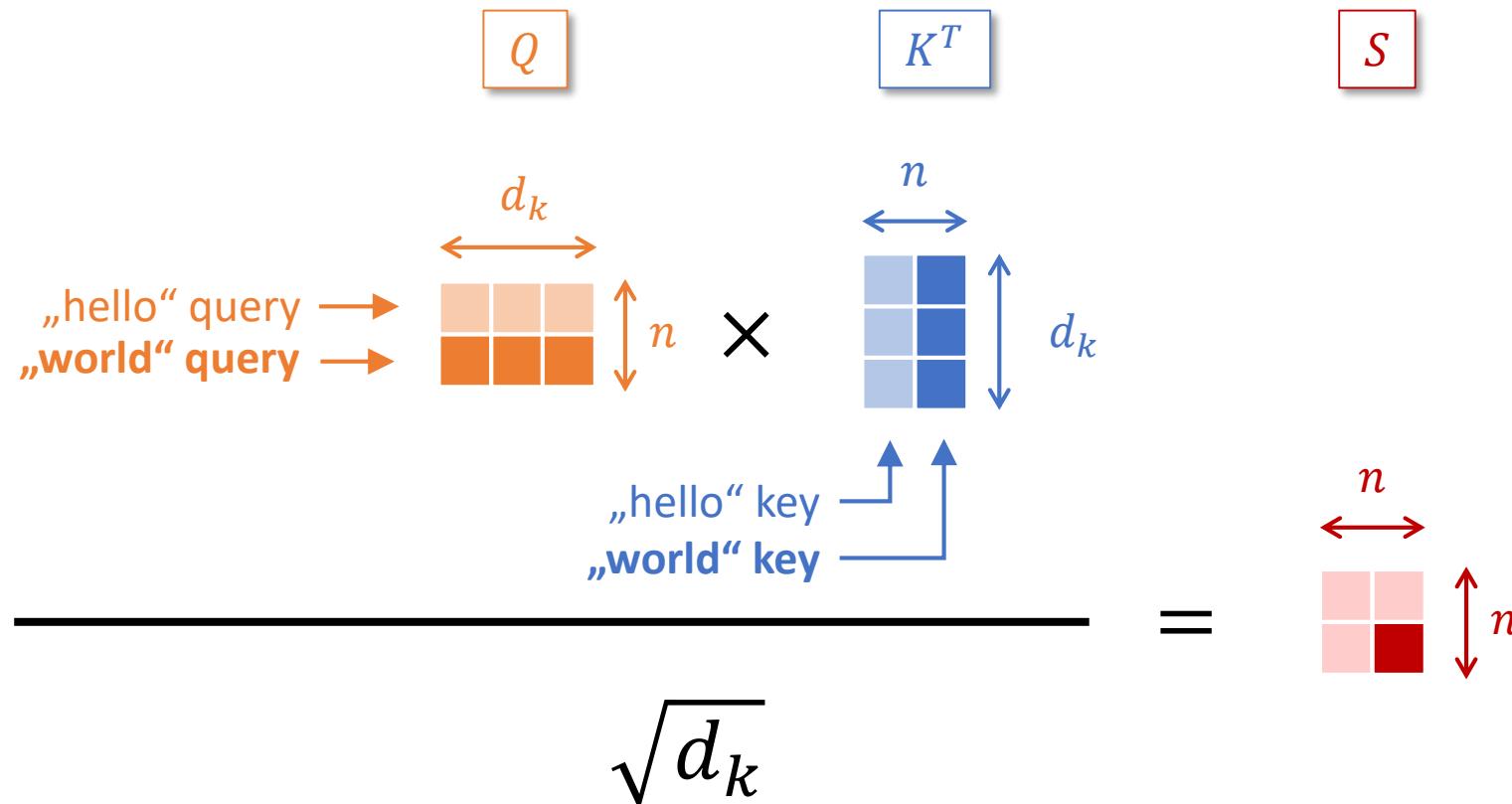
$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

# Self attention



$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

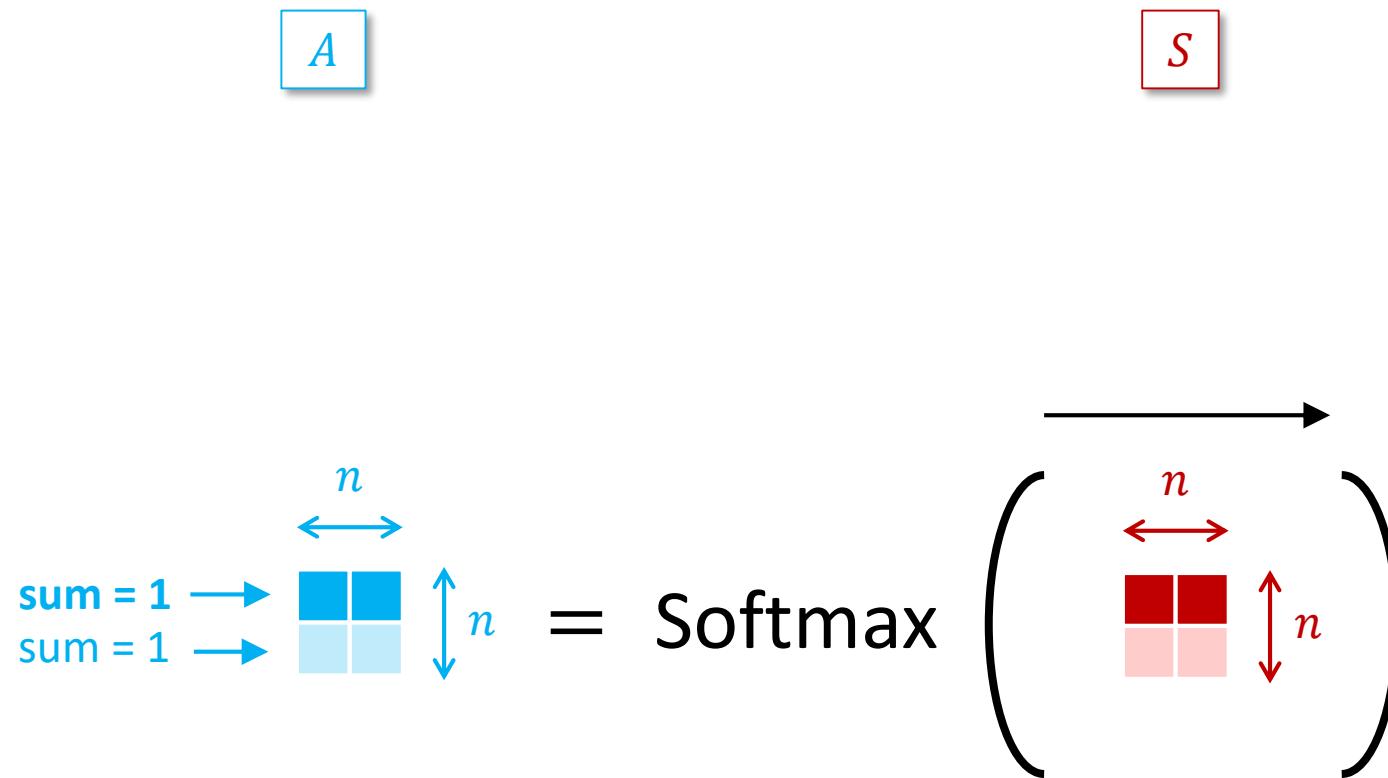
# Self attention



$S \dots n \times n$   
podobnost mezi slovy  
ve větě, měřeno  
každý s každým jako  
query vs key

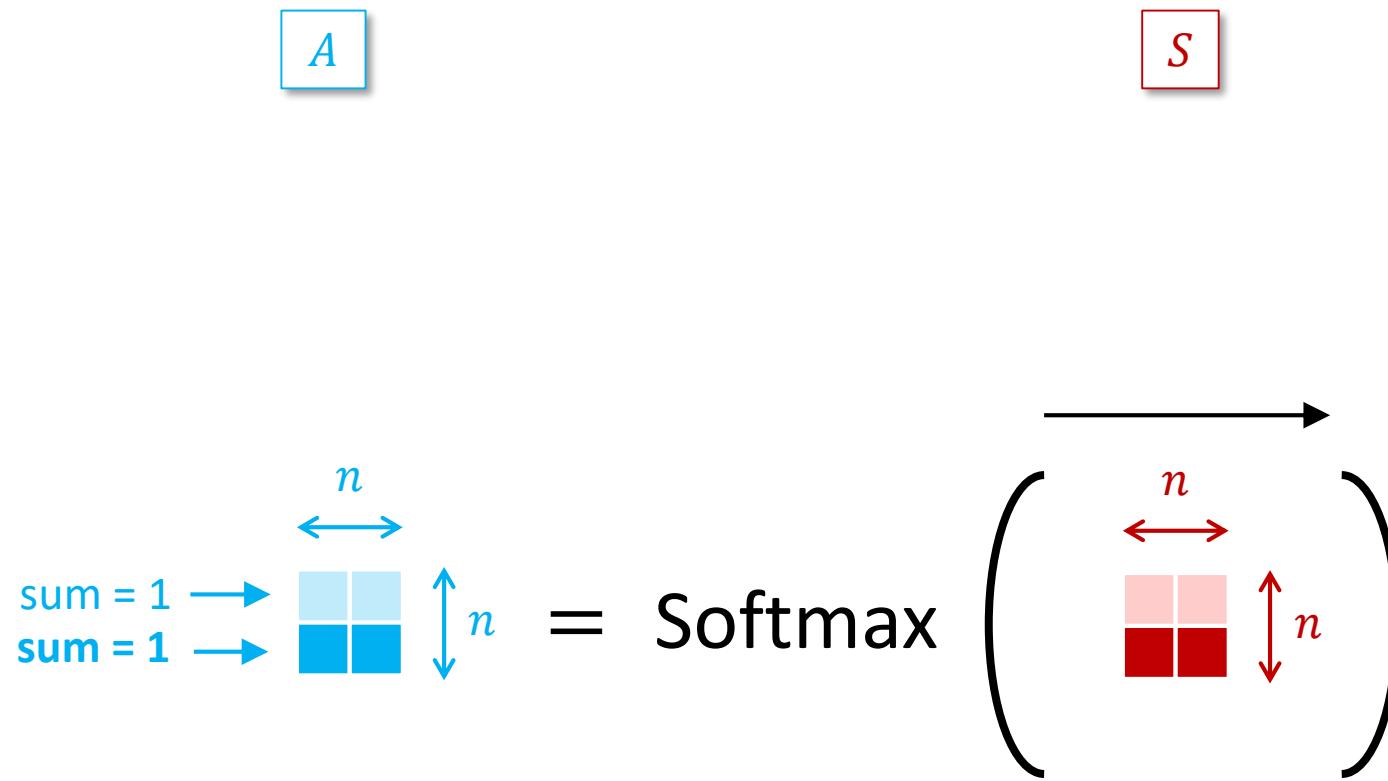
$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

# Self attention



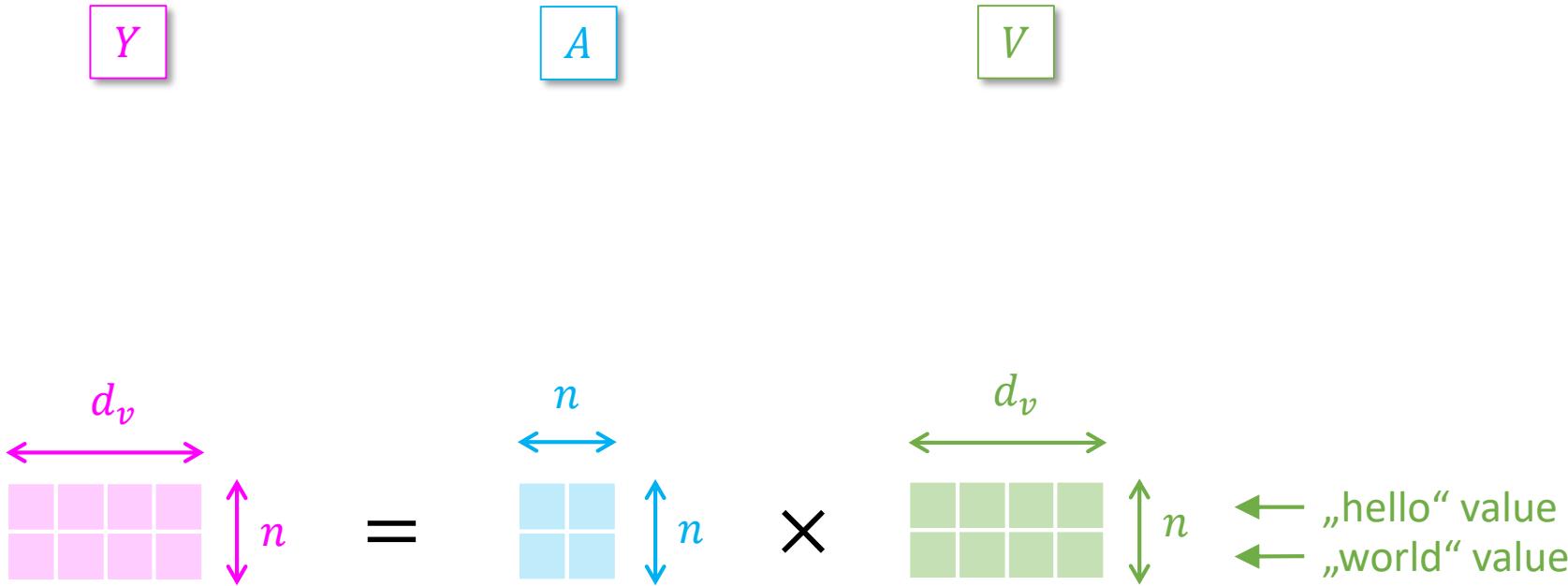
$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

# Self attention



$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

# Self attention



$$Y = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

# Self attention

$Y$

$A$

$V$

$$y_1 = \underbrace{[a_{11}v_1]}_{\text{dashed box}} + a_{12}v_2$$



$$Y = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Self attention

$Y$

$A$

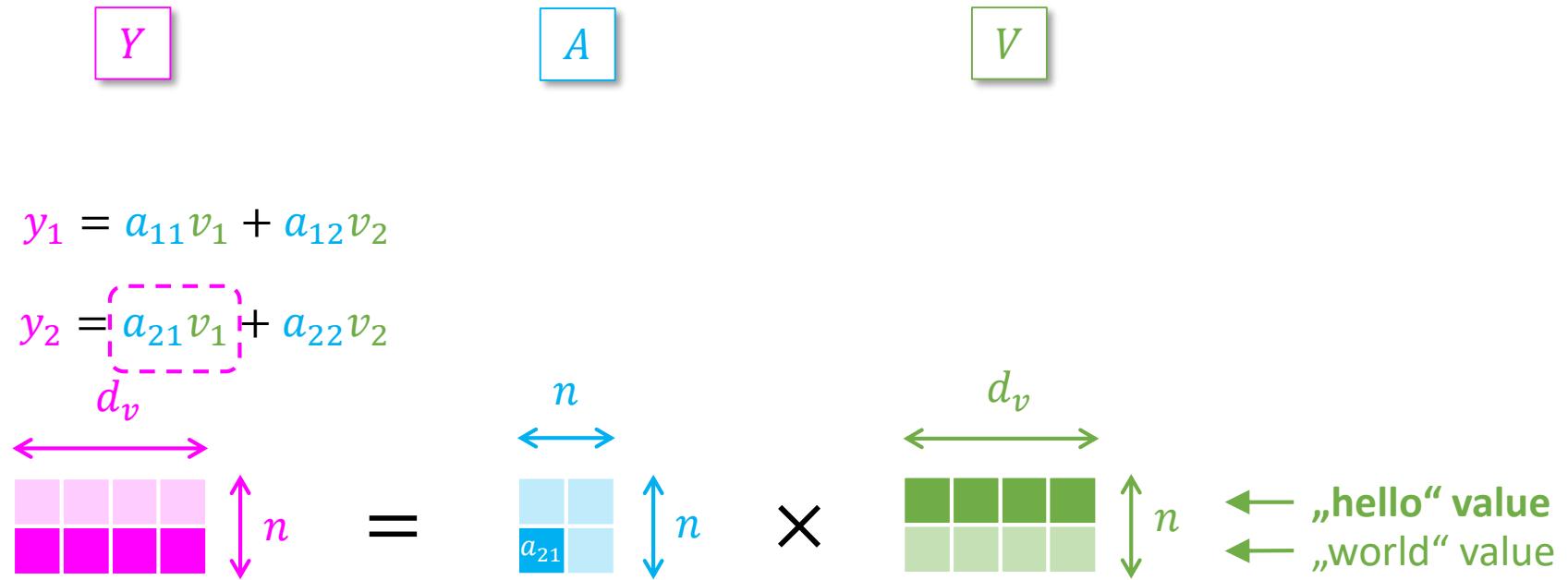
$V$

$$y_1 = a_{11}v_1 + \boxed{a_{12}v_2}$$



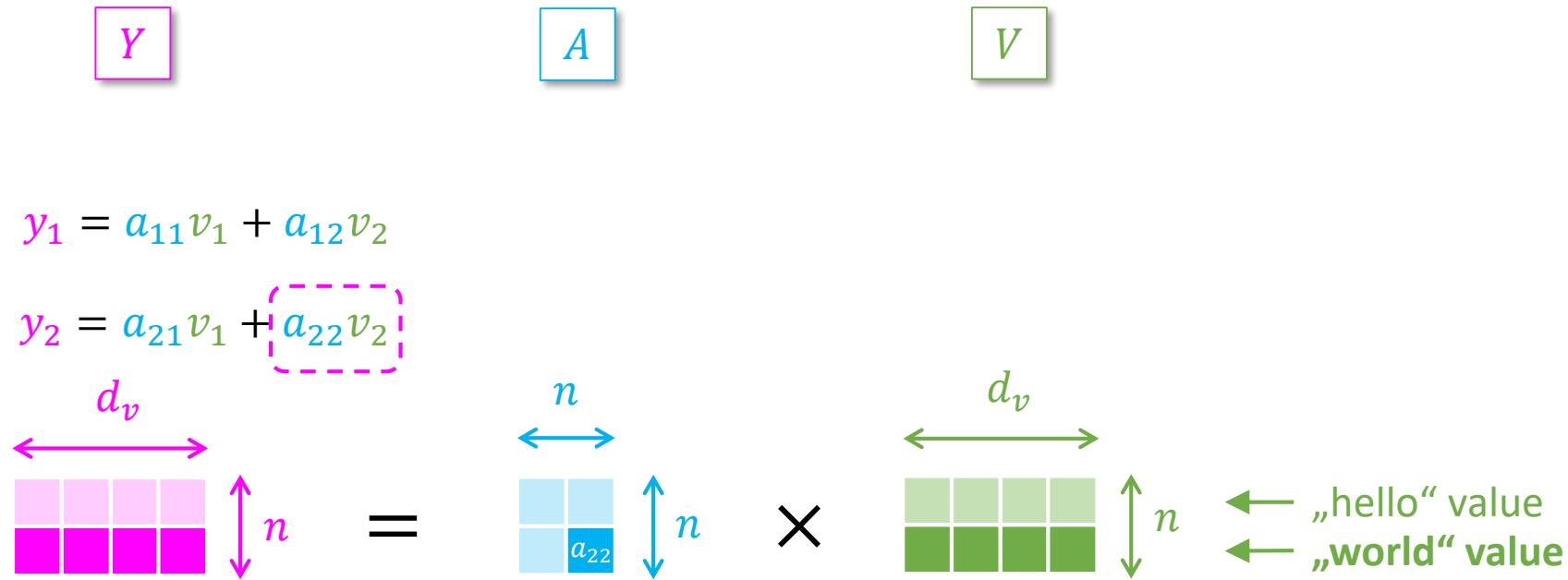
$$Y = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Self attention



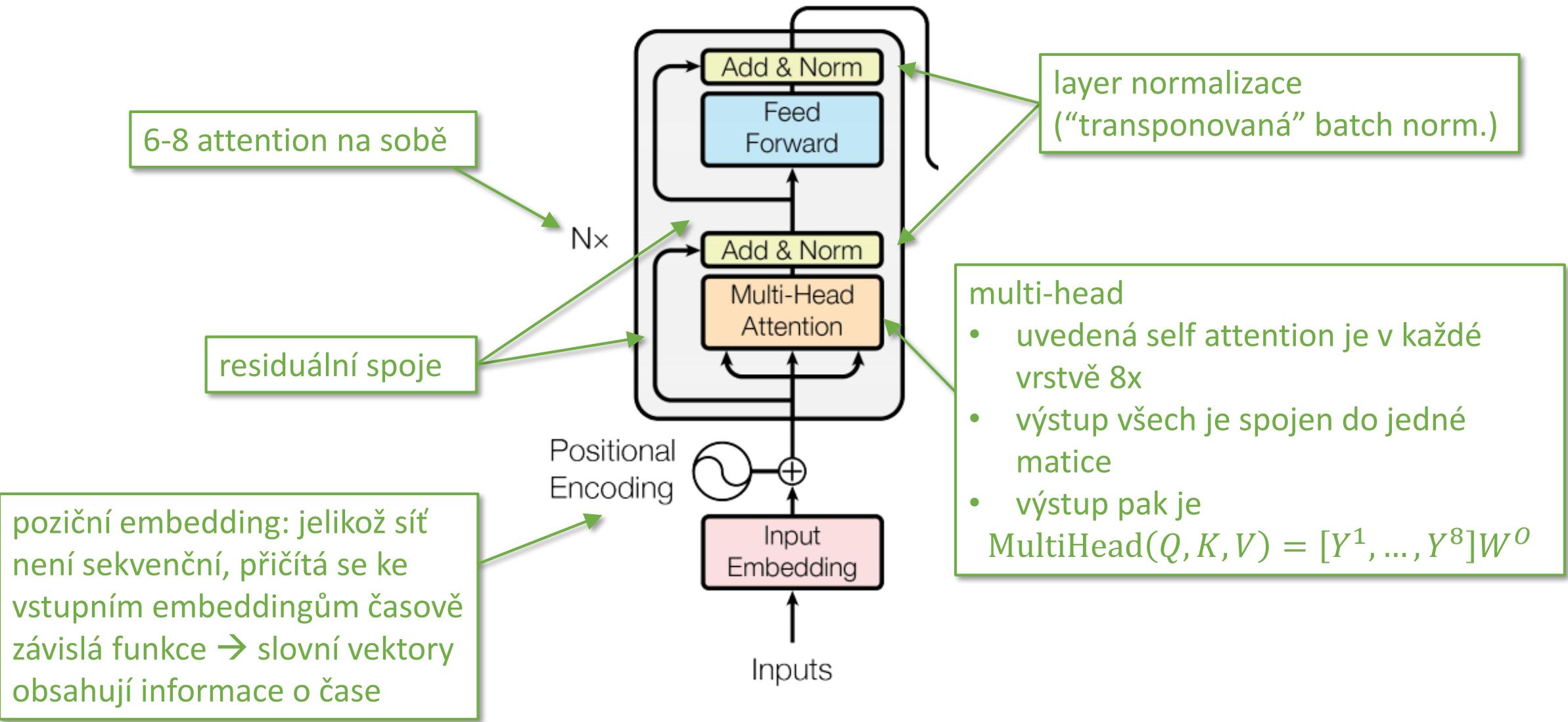
$$Y = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Self attention



$$Y = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

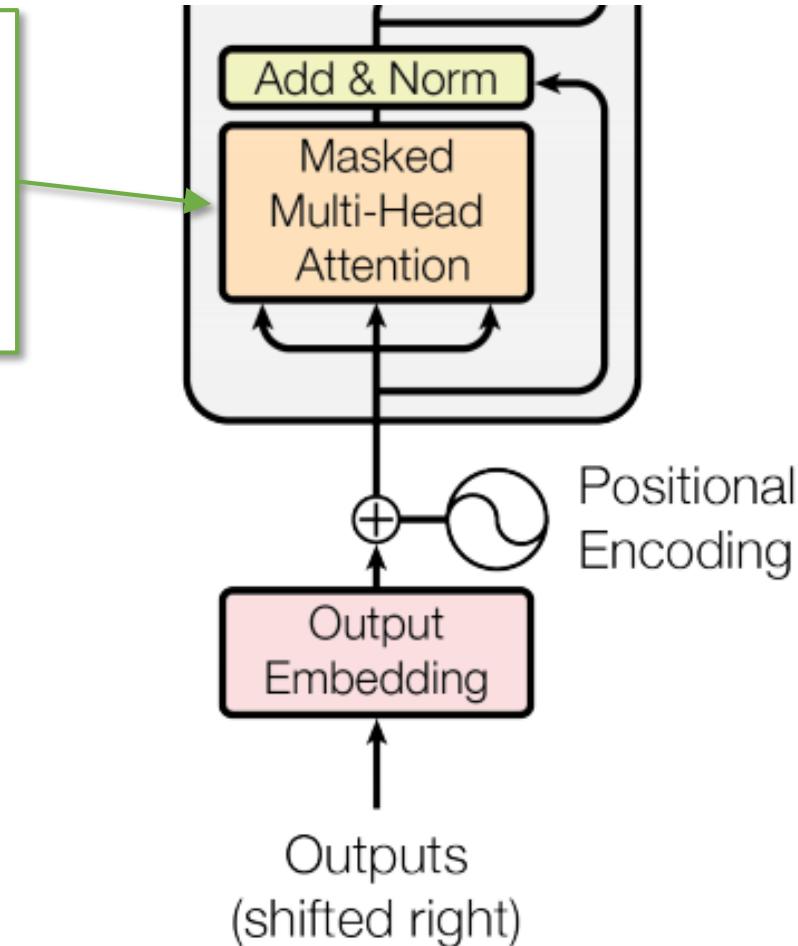
# Transformer: encoder



# Transformer: decoder

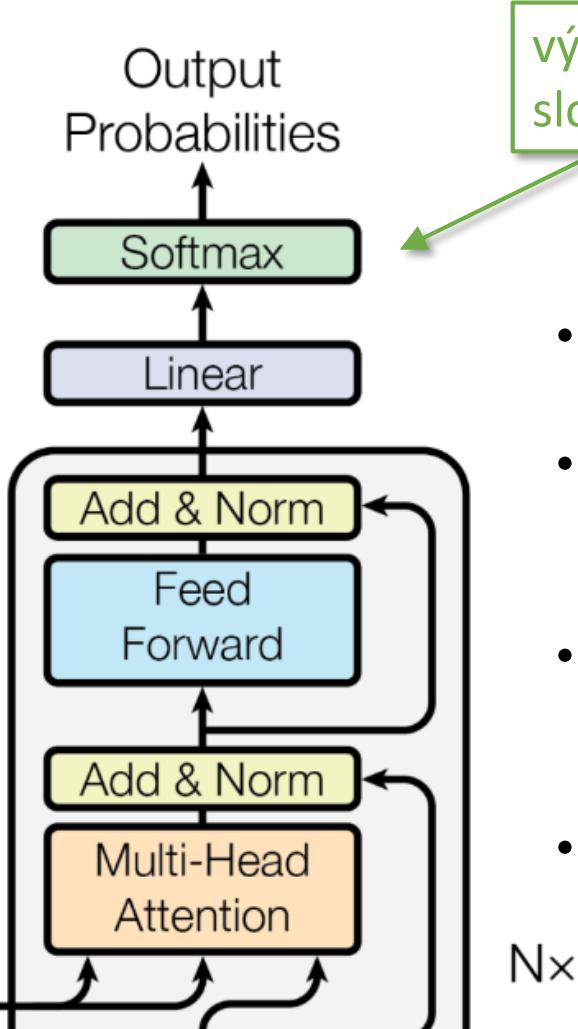
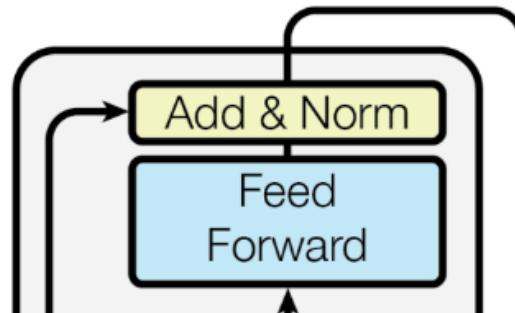
maskovaná self attention

- self = výstupní sekvence
- maska = nelze se dívat dopředu, attention povolena jen na předchozí vektory v sekvenci



# Transformer: decoder

z enkodéru se předávají výstupy self attention na vstupu



výstup je pravděpodobnost pro každé slovo ve slovníku (např.  $\approx 10^4$ )

- trénování funguje kompletně paralelně, tj. všechna slova vstupu i výstupu najednou
- minimalizuje se cross entropy pro výstupní pravděpodobnosti na slova ze slovníku cílového jazyka (pokud jde o překlad)
- v inferenci dekodér produkuje slova postupně po jednom, není zde ovšem rekurence, pouze attention
- konec se řeší speciálním tokenem (<eos>)

# Transformer: výsledky v původním článku

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		<b><math>3.3 \cdot 10^{18}</math></b>
Transformer (big)	<b>28.4</b>	<b>41.8</b>		$2.3 \cdot 10^{19}$

# Generative Pre-Training (GPT)

- **Předtrénován jako jazykový model** (predikce příštího slova z minulých), poté přizpůsoben na cílovou úlohu

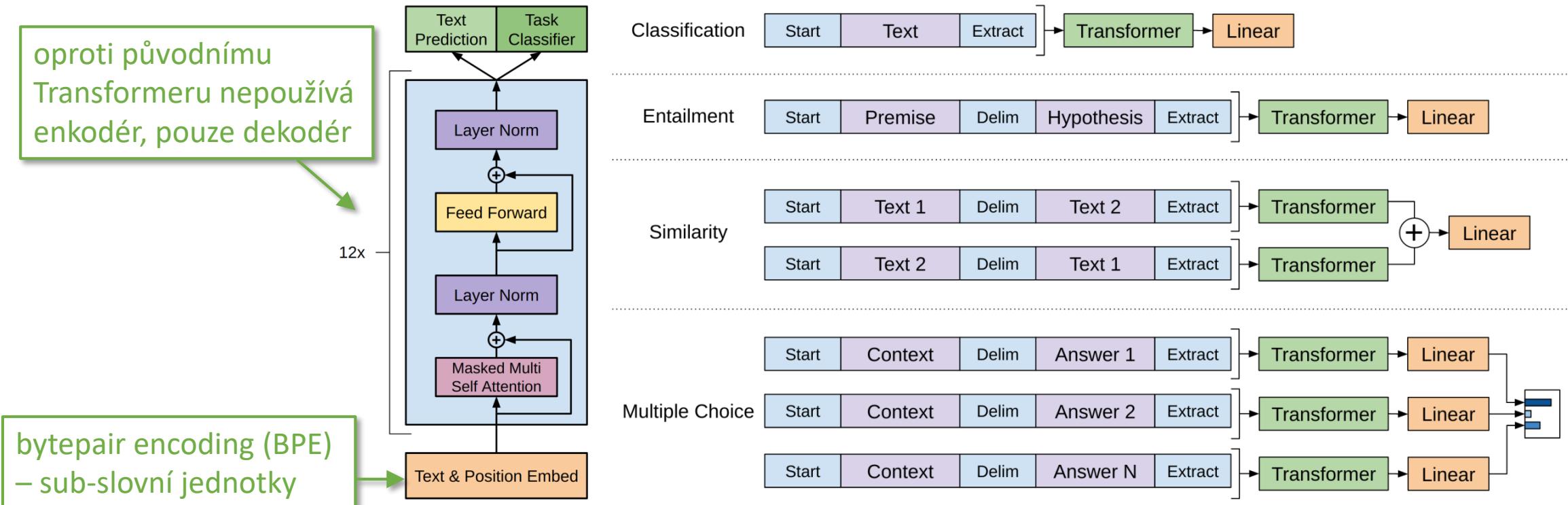


Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

# GPT-3

pro srovnání ResNet-50 má cca 25M parametrů 😊

Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 \times 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 \times 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 \times 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 \times 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2 \times 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 \times 10^{-4}$
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	$0.6 \times 10^{-4}$

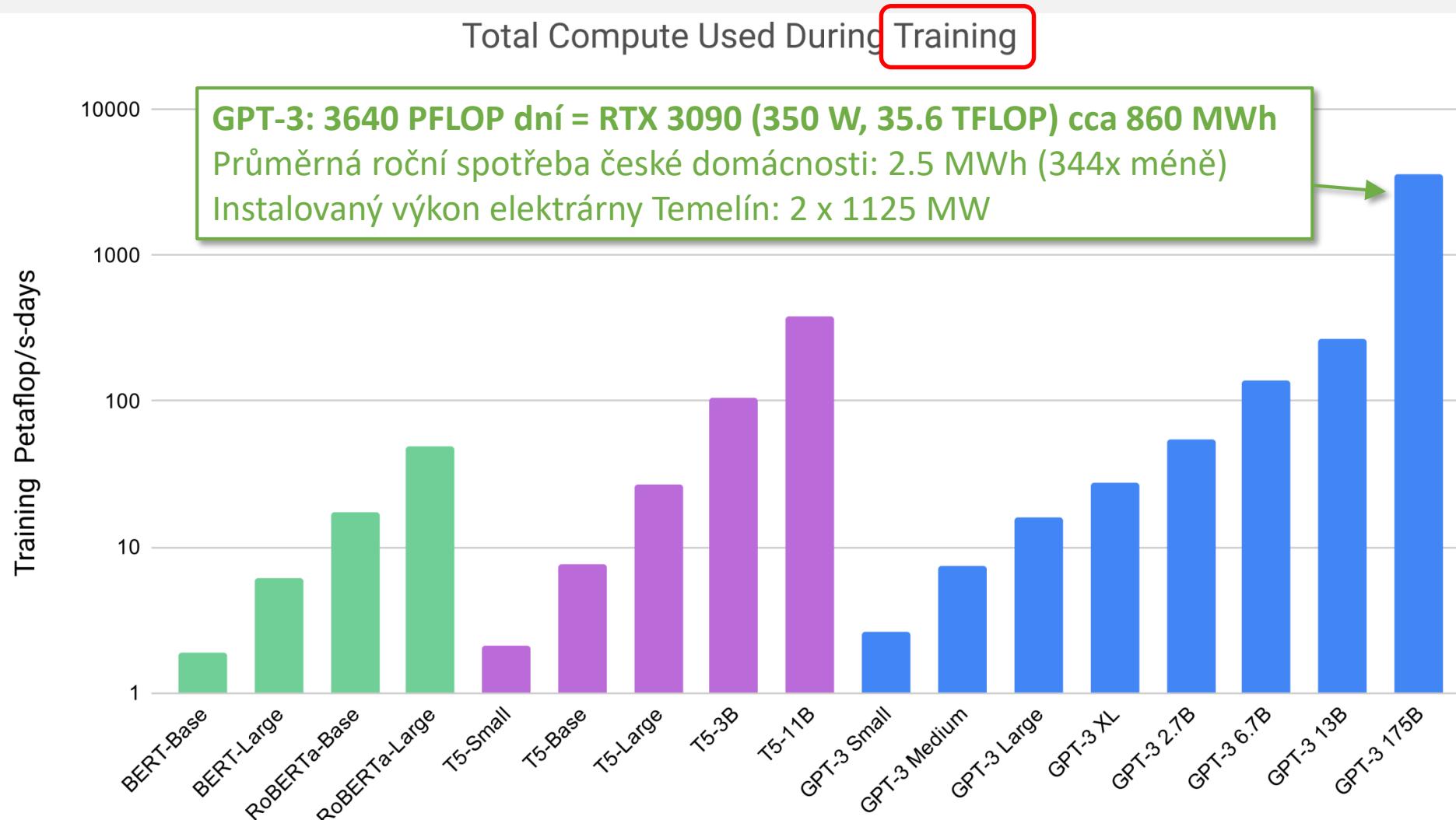
**Table 2.1:** Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

# GPT-3

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

**Table 2.2: Datasets used to train GPT-3.** “Weight in training mix” refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

# GPT-3



**Figure 2.2: Total compute used during training.** Based on the analysis in Scaling Laws For Neural Language Models [KMH<sup>+</sup>20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

# GPT-3

	Mean accuracy	95% Confidence Interval (low, hi)	<i>t</i> compared to control ( <i>p</i> -value)	“I don’t know” assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 (2e-4)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 (7e-21)	6.0%
GPT-3 Large	68%	64%–72%	7.3 (3e-11)	8.7%
GPT-3 XL	62%	59%–65%	10.7 (1e-19)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 (5e-19)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 (3e-21)	6.2%
GPT-3 13B	55%	52%–58%	15.3 (1e-32)	7.1%
GPT-3 175B	52%	49%–54%	16.9 (1e-34)	7.8%

**Table 3.11: Human accuracy in identifying whether short (~200 word) news articles are model generated.** We find that human accuracy (measured by the ratio of correct assignments to non-neutral assignments) ranges from 86% on the control model to 52% on GPT-3 175B. This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).

# GPT-3

Title: United Methodists Agree to Historic Split  
Subtitle: Those who oppose gay marriage will form their own denomination  
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.  
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

**Figure 3.14:** The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

# GPT-3

- Model je až neuvěřitelně přenosný do dalších úloh (např. strojový překlad, korekce pravopisu, odpovídání na otázky, vyhodnocování podobnosti vět, ...)
- Fine-tuning
  - Předtrénovaný model se přizpůsobí na cílovou úlohu v supervised učení na menším datasetu
- Few-shot learning
  - Pouze několik málo příkladů
  - Nedochází k učení, příklady jsou předtrénovanému modelu dodány jako kontext
  - Např. **Přelož „žába“ do angličtiny:** př. „pes“ → „dog“, „kočka“ → „cat“: ...
- One-shot learning
  - speciální případ few-shot learning s pouze jedním příkladem
- Zero-shot learning
  - bez příkladů, pouze struktura textu

# GPT-3 few-shot learning

Poor English input: I eated the purple berries.

Good English output: I ate the purple berries.

Poor English input: Thank you for picking me as your designer. I'd appreciate it.

Good English output: Thank you for choosing me as your designer. I appreciate it.

Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.

Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.

Poor English input: I'd be more than happy to work with you in another project.

**Good English output: I'd be more than happy to work with you on another project.**

---

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.

**Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.**

---

Poor English input: The patient was died.

**Good English output: The patient died.**

---

Poor English input: We think that Leslie likes ourselves.

**Good English output: We think that Leslie likes us.**

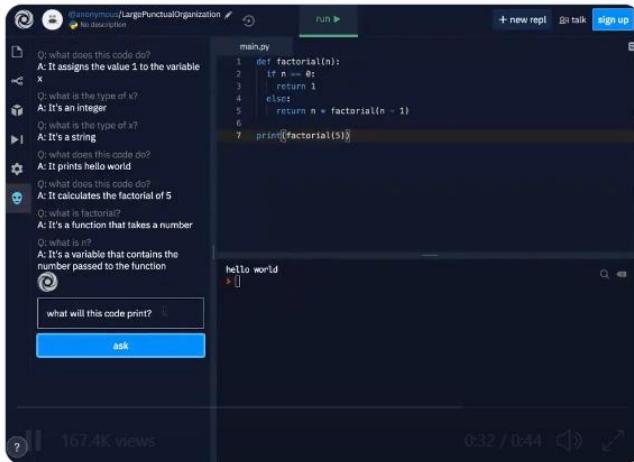
---

Poor English input: Janet broke Bill on the finger.

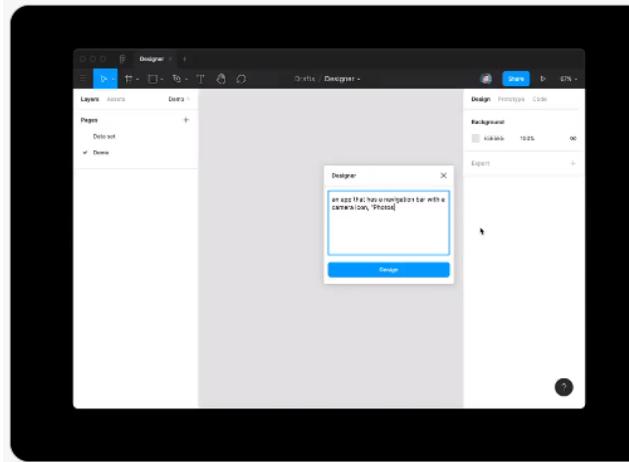
**Good English output: Janet broke Bill's finger.**

# GPT-3 aplikace

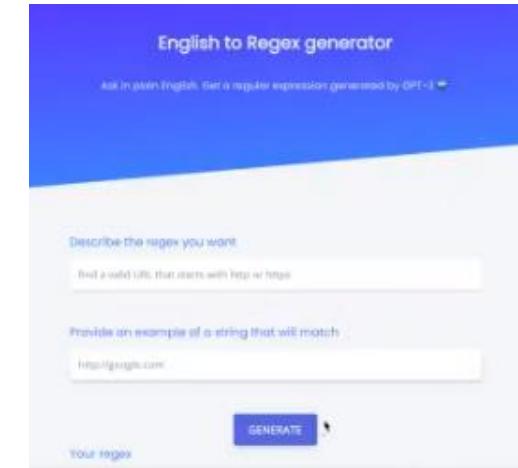
## Čtení zdrojového kódu



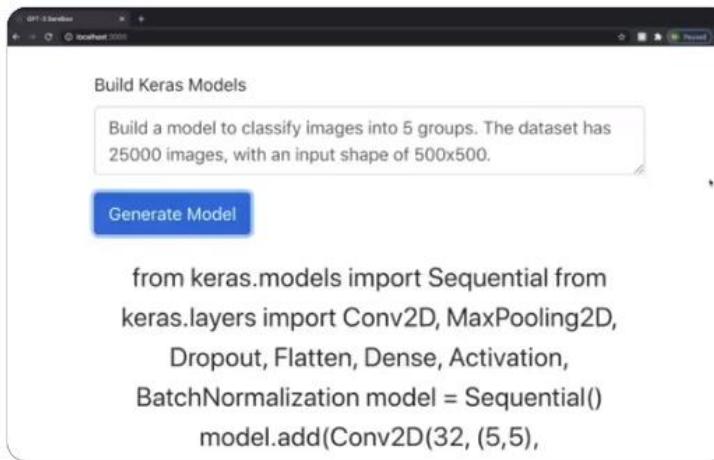
## Web design



## Regex generátor



## Generování machine learning kódu



## Generování memů



## Konverzace

Q: How many eyes does a giraffe have?  
A: A giraffe has two eyes.

Q: How many eyes does my foot have?  
A: Your foot has two eyes.

Q: How many eyes does a spider have?  
A: A spider has eight eyes.

Q: How many eyes does the sun have?  
A: The sun has one eye.

Q: How many eyes does a blade of grass have?  
A: A blade of grass has one eye.

# Shrnutí

- slova vhodné reprezentovat jako husté vektory
- nejrozšířenější metoda skipgram word2vec
- podobně jako u konv. sítí lze využít předtrénované word-vektory a příp. finetunit

aplikace a odpovídající architektura	
jazykový model	1:1
klasifikace sekvence, analýza sentimentu	M:1
popis obrázků	1:N
strojový překlad, klasifikace na úrovni snímků, chatbot	M:N

- pro fungování především M:1 a M:N systémů velmi důležitý attention mechanismus
- stav poznání v NLP jsou transformer-based sítě, které RNN (mnohdy) vůbec nevyužívají