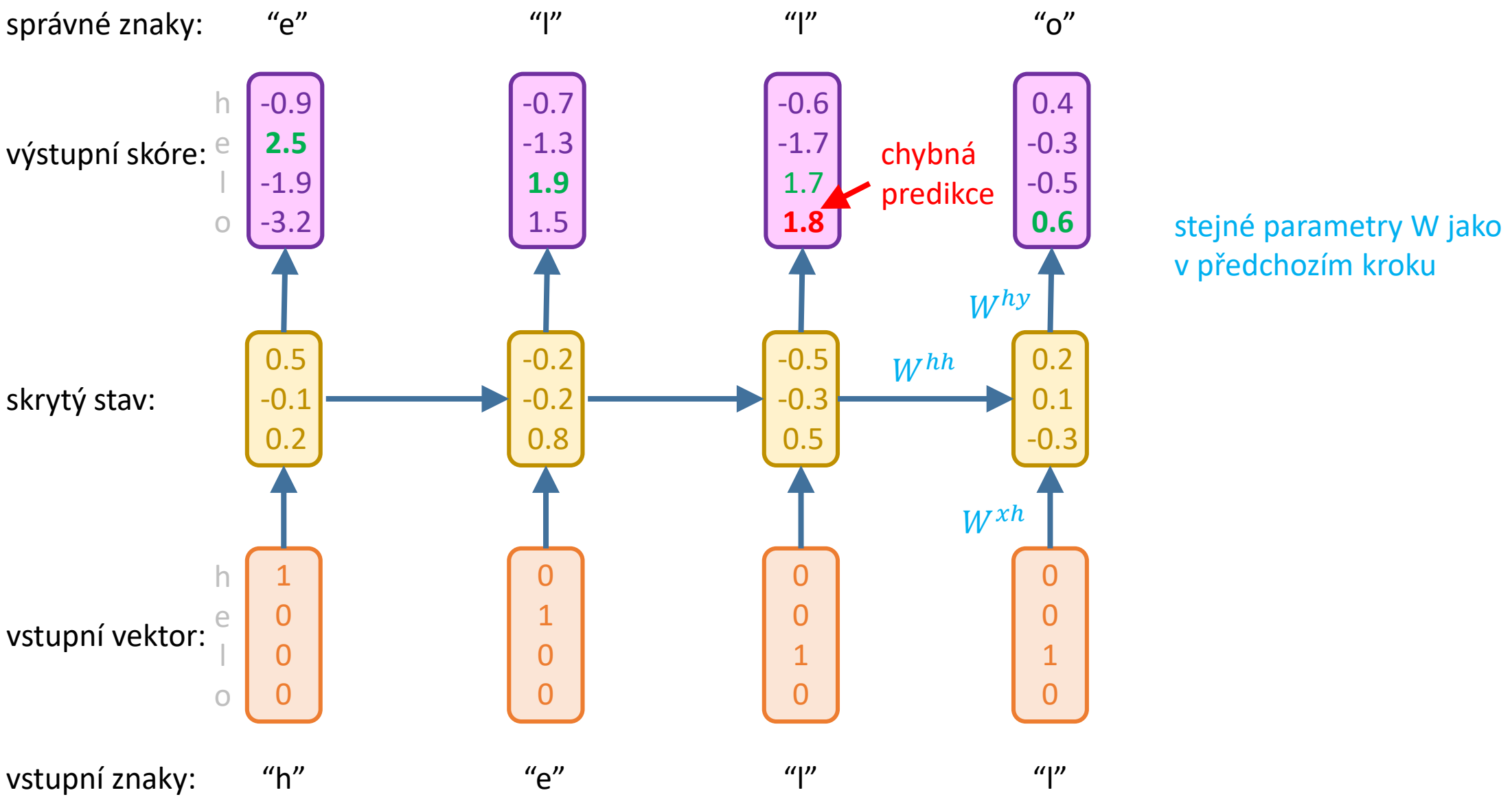
Aplikace neuronových sítí

Sequence-to-sequence

RNN znakový jazykový model



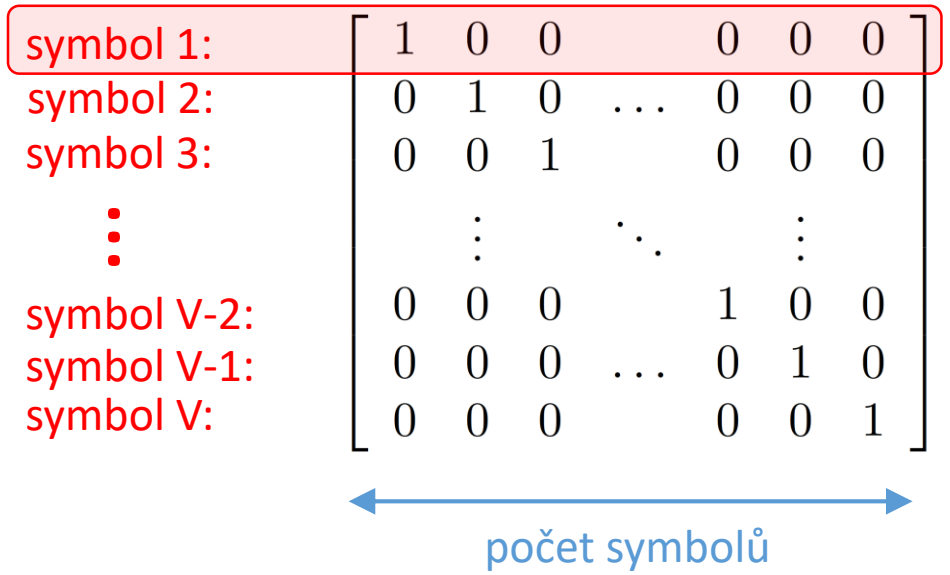
stejné parametry W jako v předchozím kroku

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 $\approx 10^4, 10^5$ → matice $10^5 \times 10^5 = 10^{10} \cdot 4B \approx 40GB$!
  - matice je řídká, pouze jednotkové vektory → nemusí být celá v paměti
  - ale: ovlivňuje i velikost matice parametrů W^{xh} → bude mít rozměr $10^5 \times H$
- Řešení: reprezentace znaků kratšími vektory → matice např. $10^5 \times 300 \approx 12MB$
  - matice W^{xh}: $300 \times H$

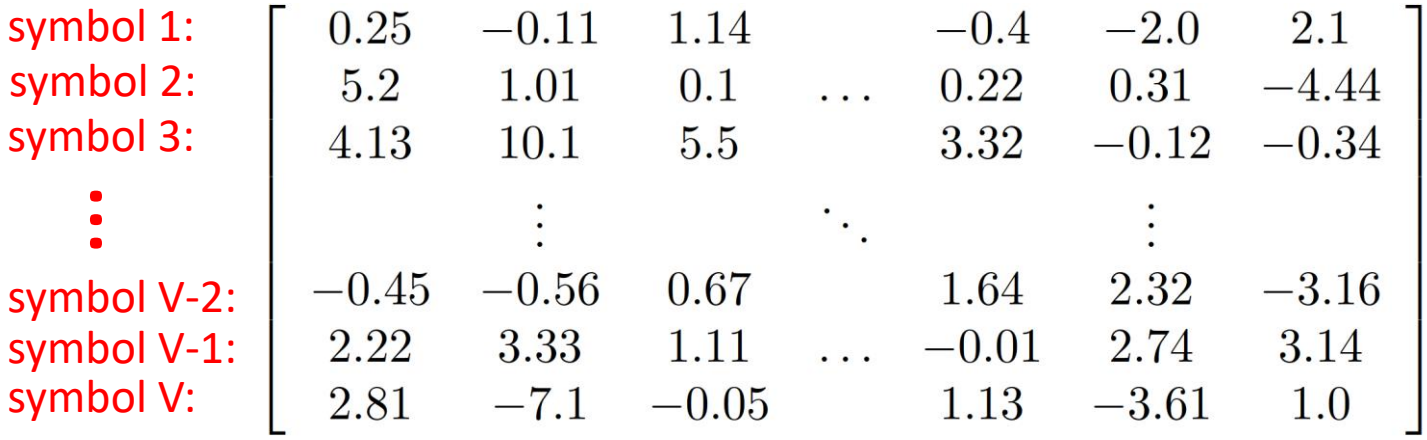
Písmena vs slova

one-hot řídká reprezentace



hustá reprezentace

libovolná dimenze

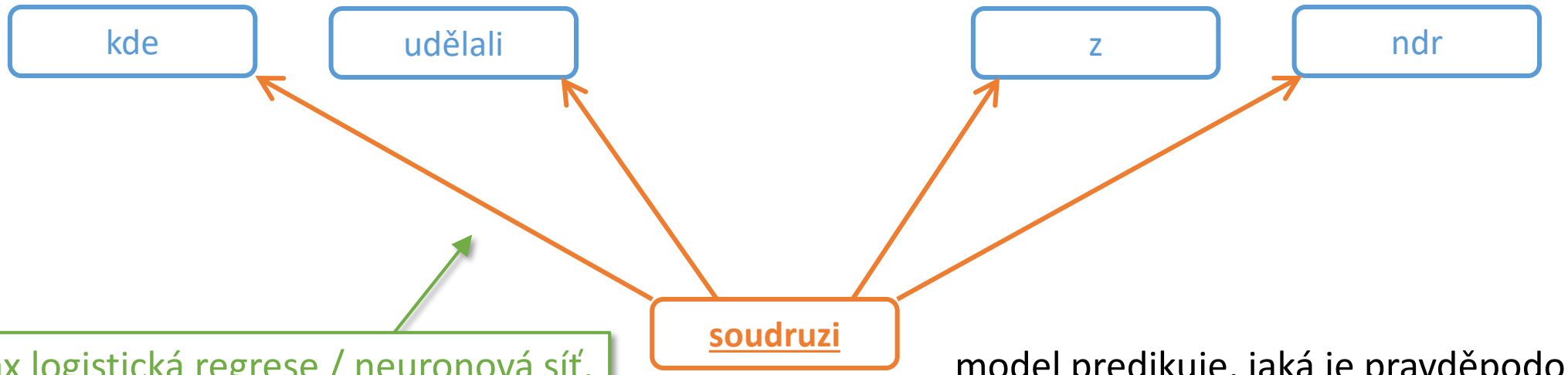


Jak ale vektory zvolit?

word2vec

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

Příklad: "Kde udělali **soudruzi** z NDR chybu?"

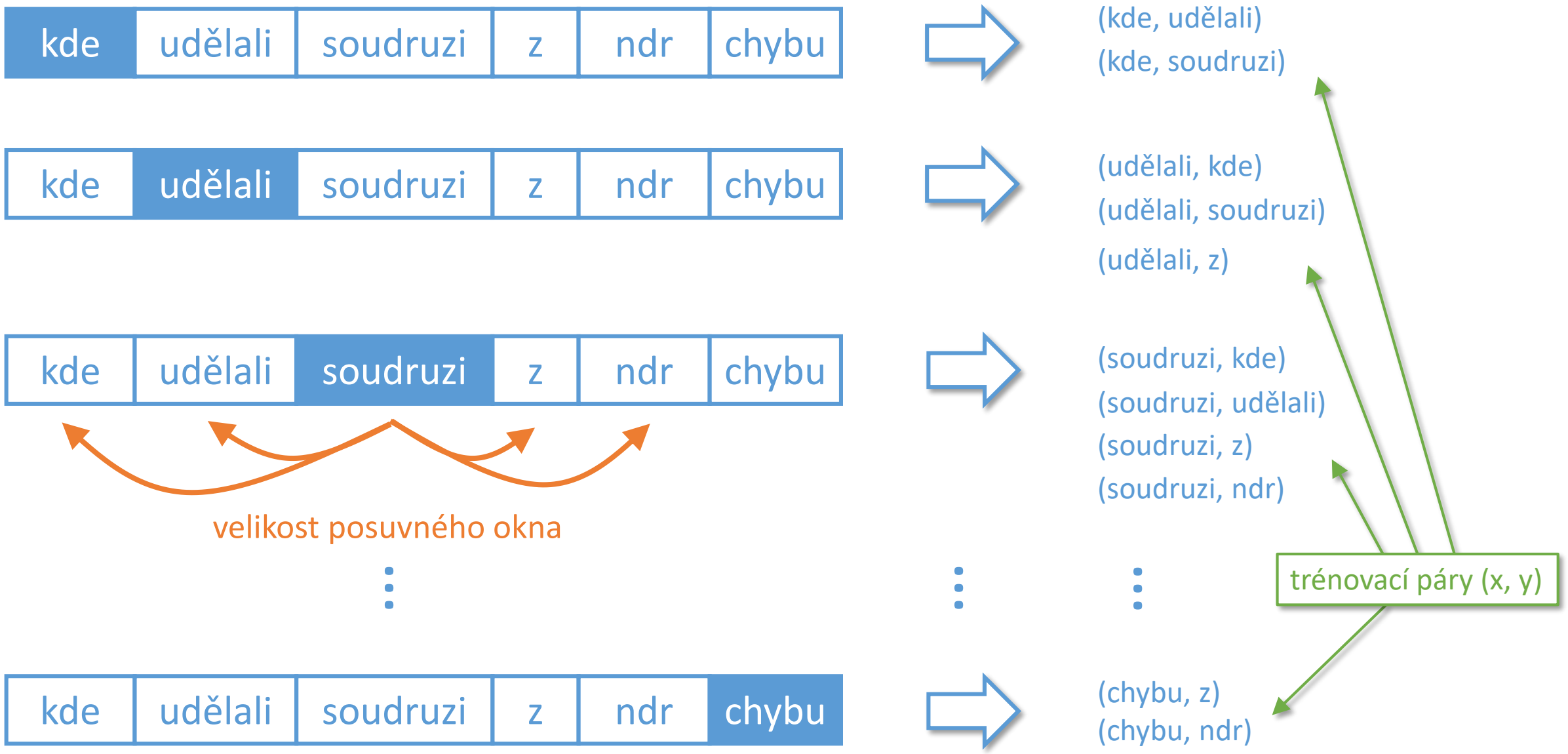


softmax logistická regrese / neuronová síť, která se učí vektory slov tak, aby maximalizovala klasifikační skóre

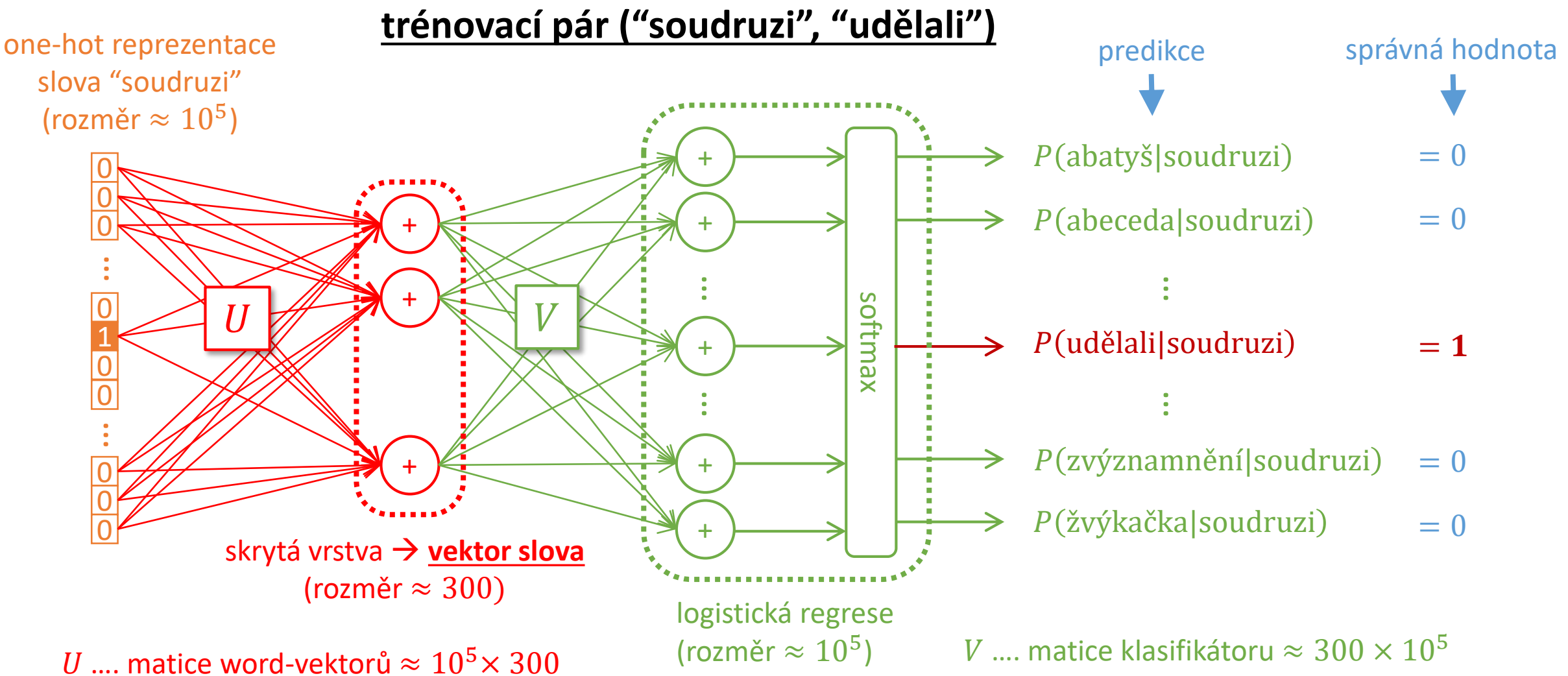
model predikuje, jaká je pravděpodobnost, že **kdekoliv** v sousedství slova "*soudruzi*" se vyskytuje např. slovo "*udělali*"

Mikolov et al.: Distributed Representations of Words and Phrases and their Compositionality

word2vec: trénovací data



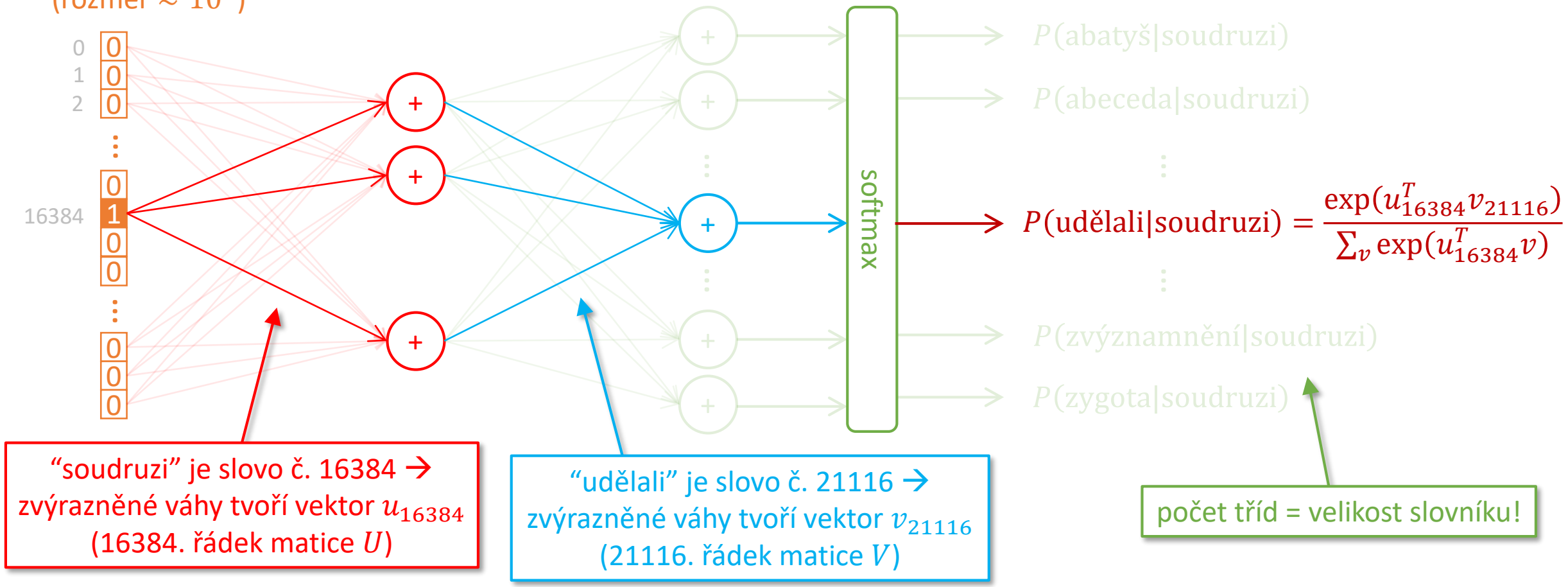
word2vec: základní architektura sítě



word2vec: základní architektura sítě

trénovací pár (“soudruzi”, “udělali”)

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



“soudruzi” je slovo č. 16384 → zvýrazněné váhy tvoří vektor u_{16384} (16384. řádek matice U)

“udělali” je slovo č. 21116 → zvýrazněné váhy tvoří vektor v_{21116} (21116. řádek matice V)

počet tříd = velikost slovníku!

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:

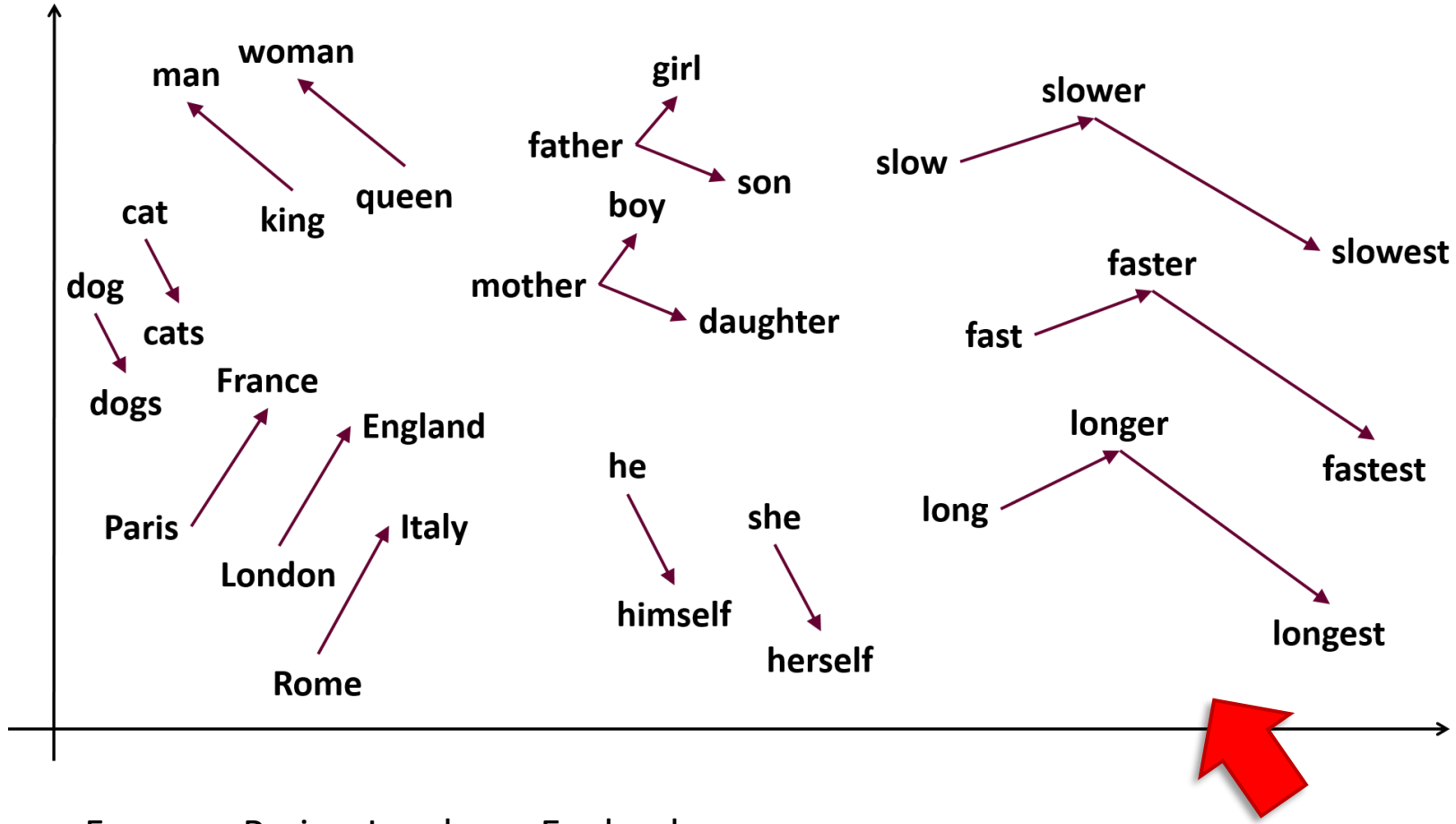
$$L_{u,v}(U,V) = -u^T v + \log \sum_{w \in \text{slovník}} \exp(u^T w)$$

- 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.
  - detaily v článku či např. zde: Stanford CS224D (2017)

word2vec: naučené vztahy mezi slovy



Příklady: France – Paris + London = England

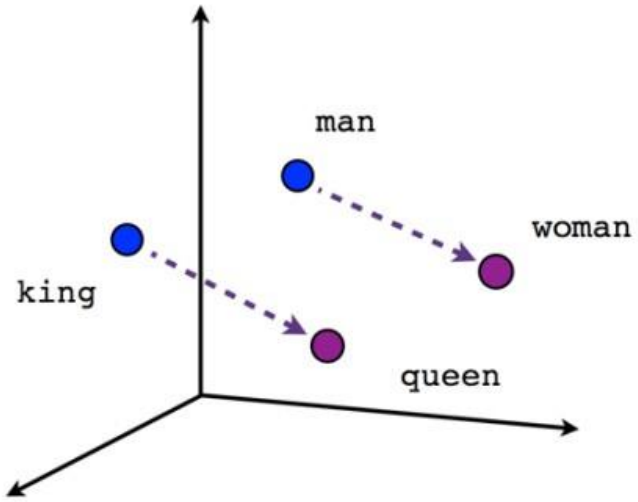
king - man + woman = queen

…

každé slovo je vektor, zde 2D

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

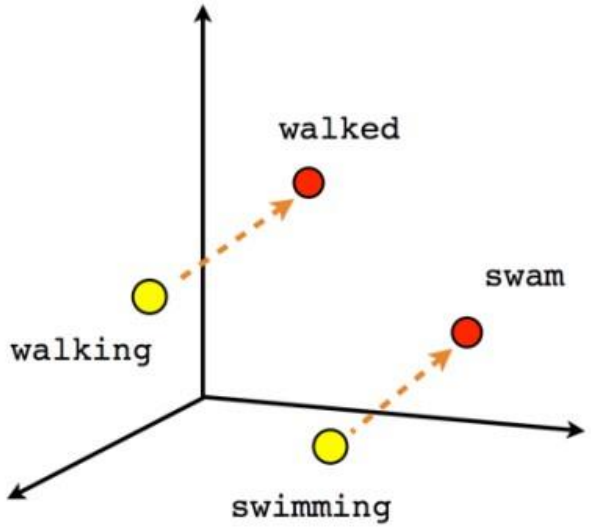
word2vec: naučené vztahy mezi slovy



Male-Female

king – man + woman = queen

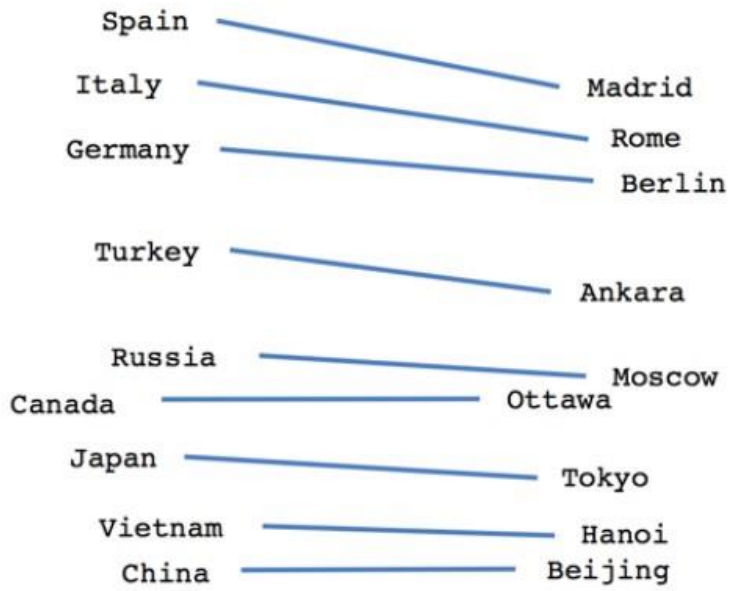
queen – king = woman - man



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)

obrázek: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

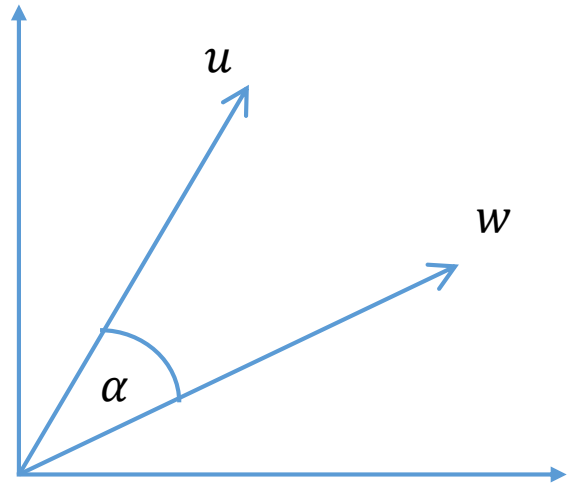
word2vec: nejbližší soused a kosinová vzdálenost

- Výsledkem operace s vektory je vektor

$$u = r - s + v$$

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

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

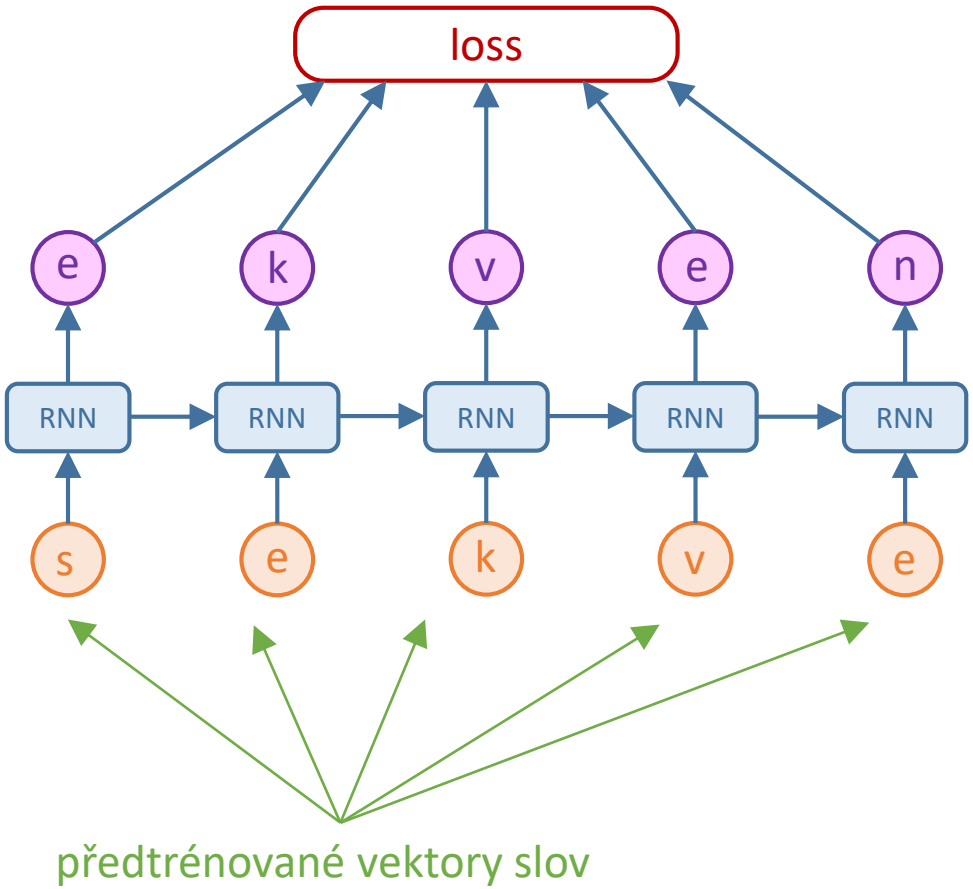


- Nezohledňuje velikost
- Zákonem velkých čísel bývají vektory ve vysokodim. prostorech na vrcholech hyperkvádru (jedna ze souřadnic >> ostatní) → úhly vystihují lépe než vzdálenost

word2vec: poznámky

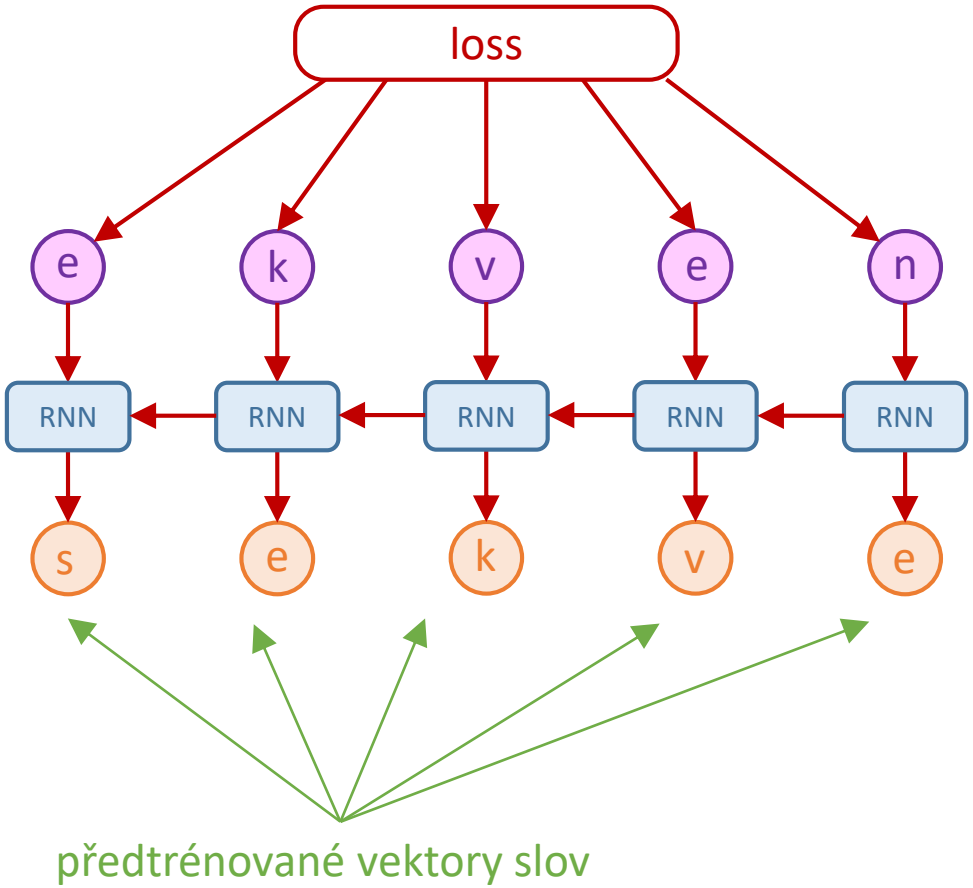
- V příkladech z praxe obvykle velikost okna = 5 (tj. 5 nalevo + 5 napravo)
- Dimenze vektorů 300
- Kromě záporného vzorkování se optimalizuje navíc omezováním příliš častých slov nebo vyhledáváním častých frází, které se pak považují za jedno slovo (např. “Dobrý den”)
- Pro různé jazyky existují předtrénované 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

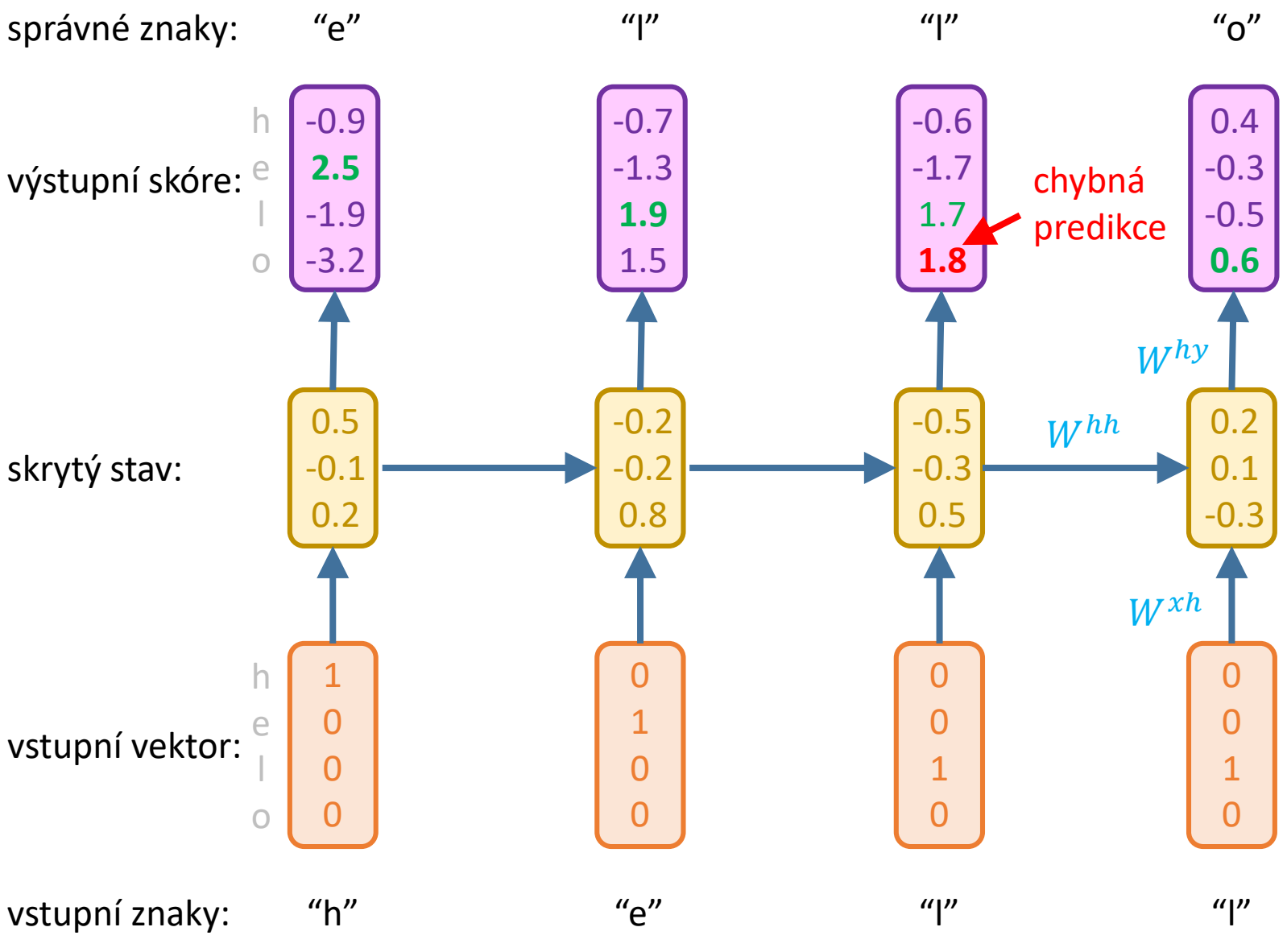
word2vec: finetuning



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

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

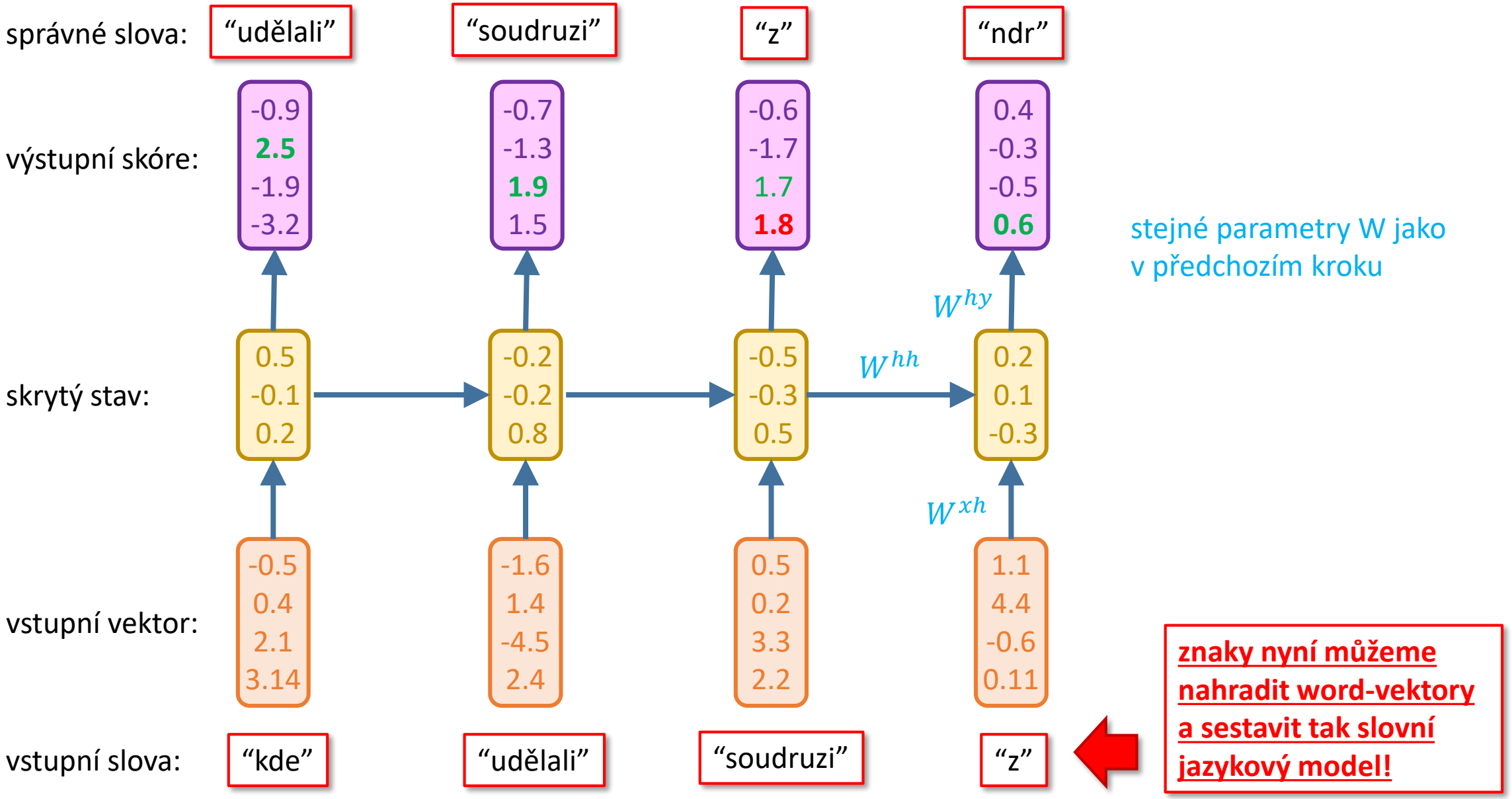
RNN znakový jazykový model



stejné parametry W jako v předchozím kroku

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

RNN znakový jazykový model



stejné parametry W jako v předchozím kroku

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

Analýza sentimentu

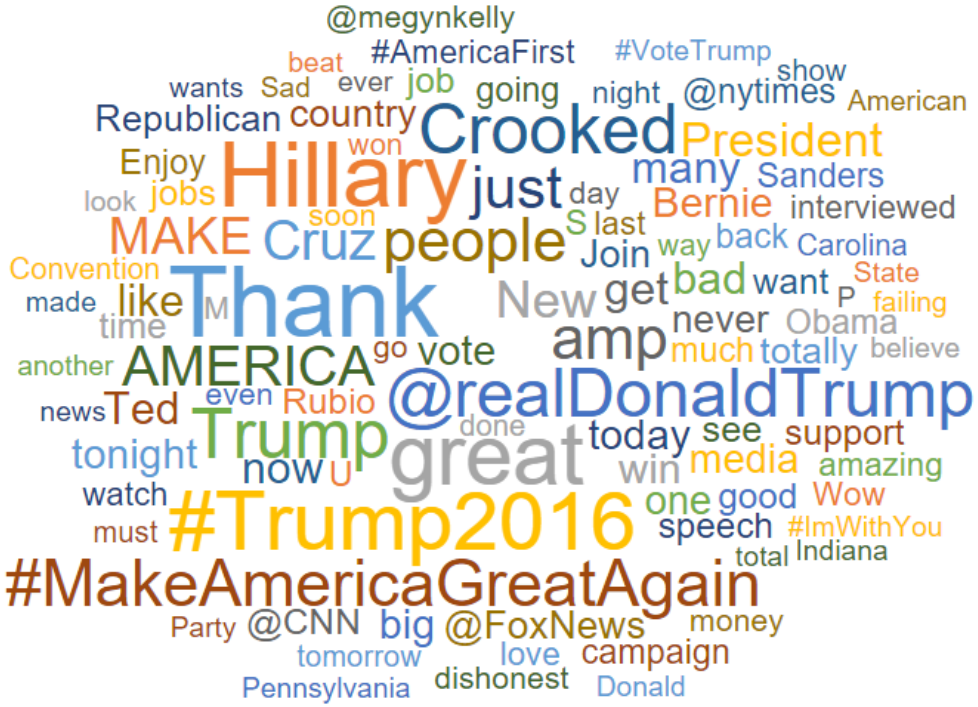
- Monitorování médií, kampaně, sledování značek, recenzí, telefonních rozhovorů, …

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



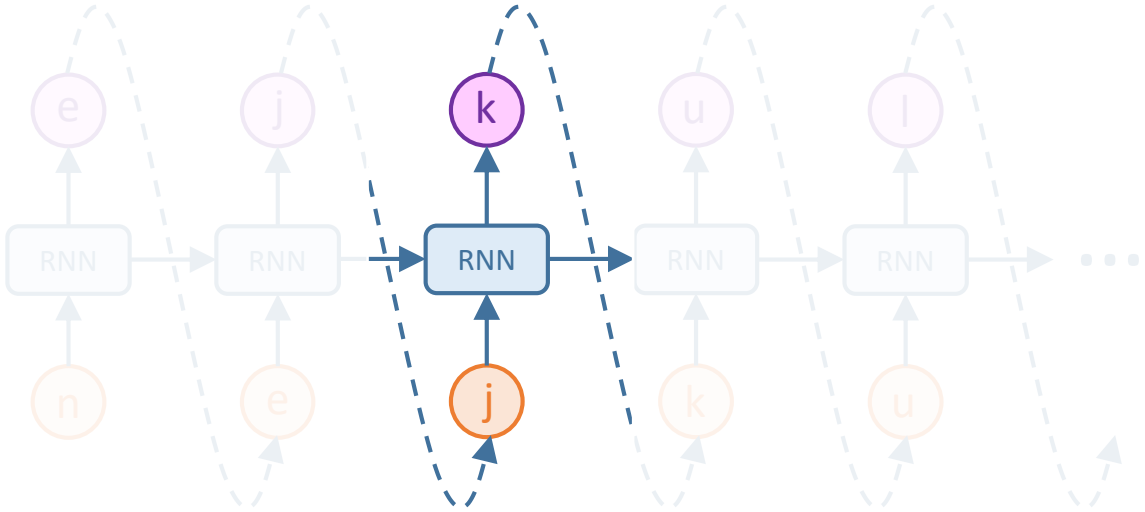
zpříkladu je zřejmé, že pouhé počítání kladných a záporných přívlastků stačit nebude

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**



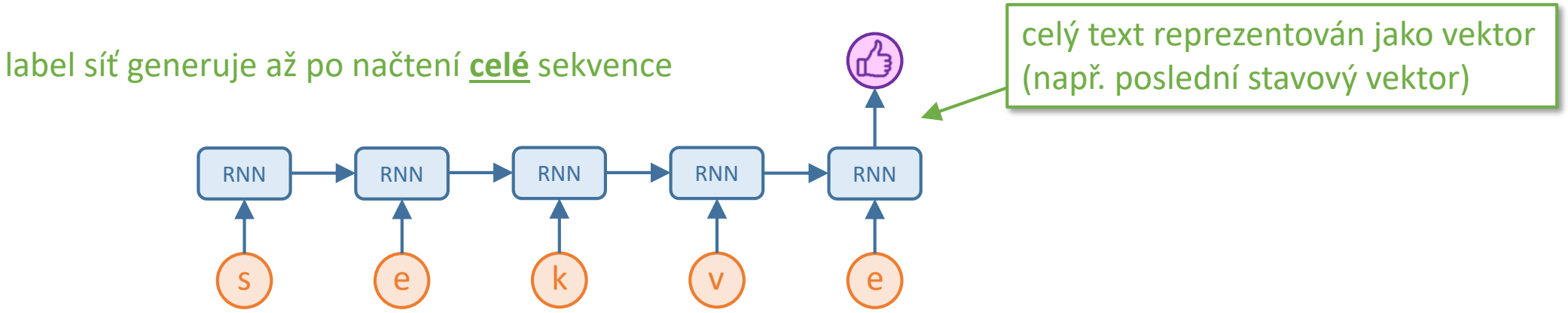
vstup : výstup → 1 : 1

(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



vstup : výstup → **M** : 1

(sequence-to-vector)

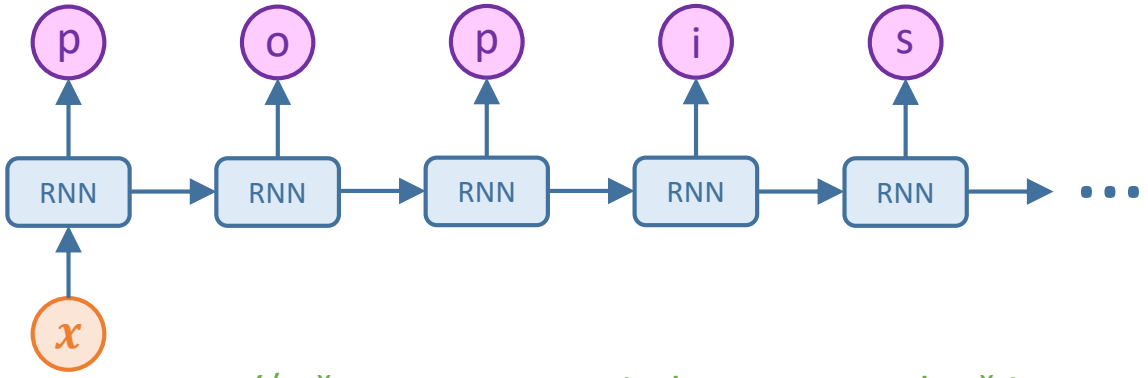
Sentiment analysis: state of the art

obrázek: http://blog.paralleldots.com/data-science/breakthrough-research-papers-and-models-for-sentiment-analysis/

Generování textu, tagování obrázků

síť převezme 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 převezme 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

FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input

baseball player is <END>

y_0 y_1 y_2 y_8

h_0 h_1 h_2 ... h_8

x_0 x_1 x_2 x_8

<START>

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."

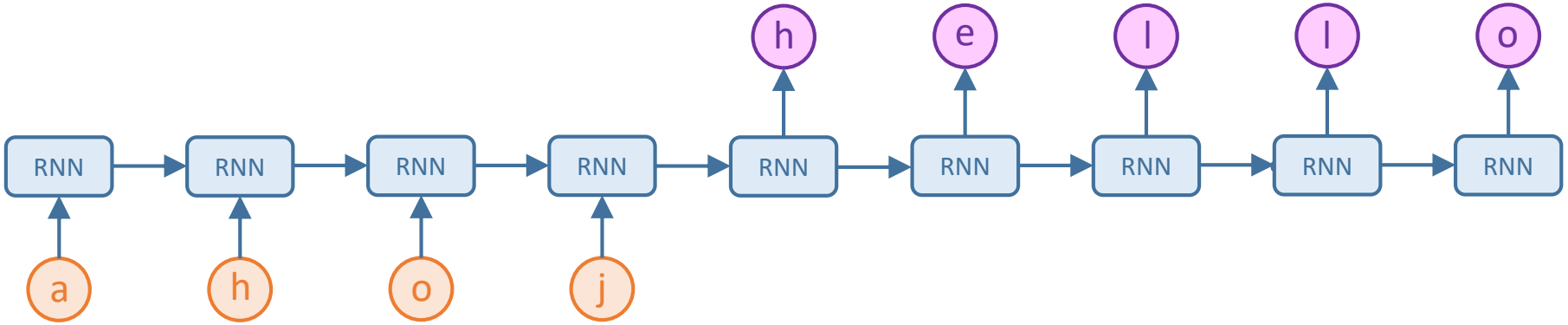
"black cat is sitting on top of suitcase."

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

Strojový překlad

síť nejprve “spolkne” celou vstupní sekvenci (větu), pak teprve začne generovat překlad

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



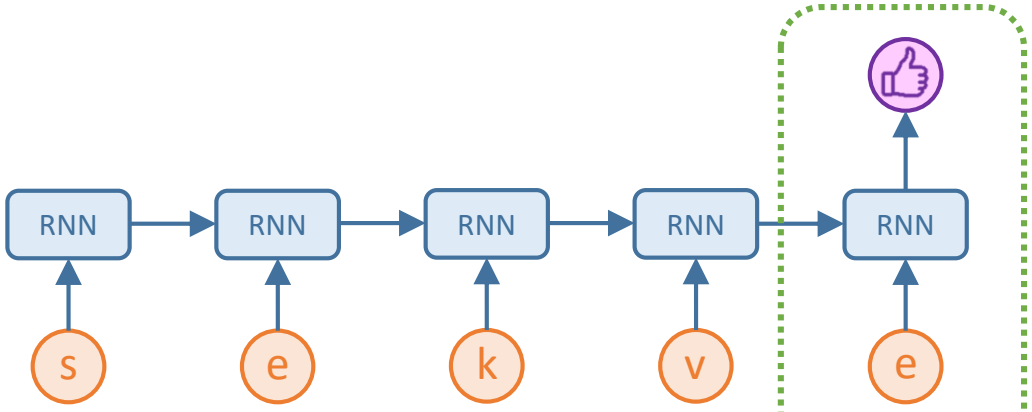
vstup : výstup → **M** : **N**

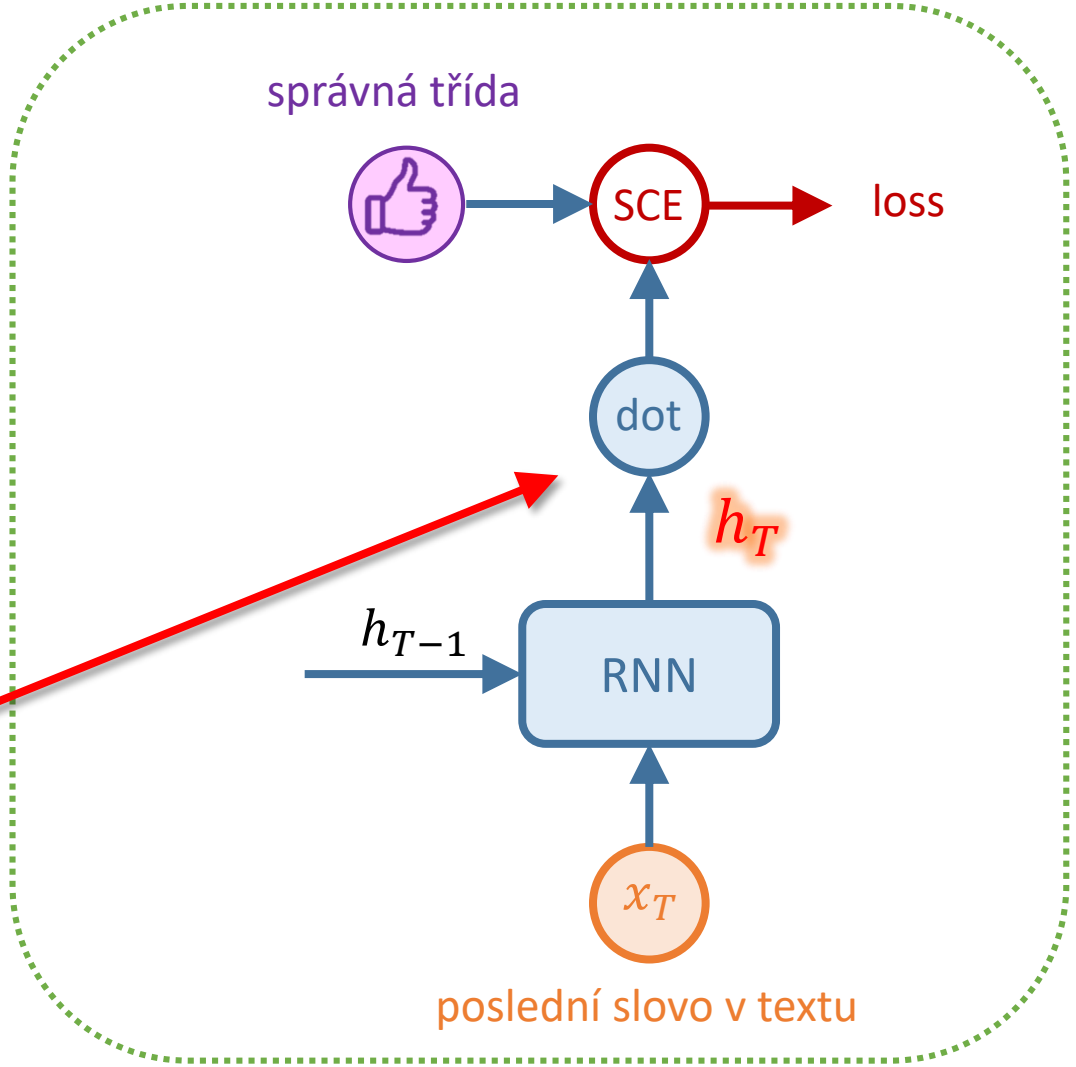
(sequence-to-sequence)

Klasifikace textu

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

- vstup: sekvence znaků / slov
- výstup: třída



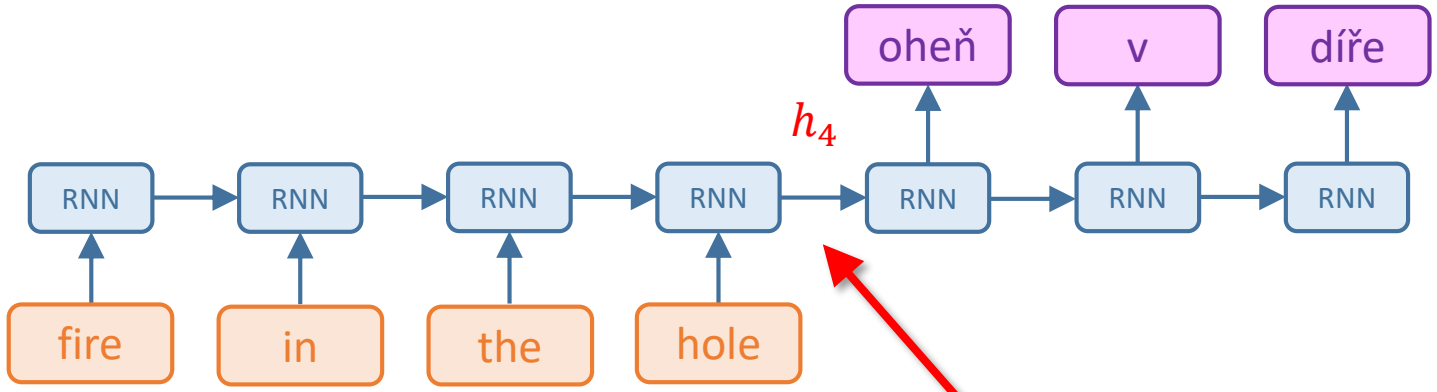


problém: stavový vektor h_T musí nějak obsahovat informaci z celé věty, protože dále už následuje pouze lineární klasifikace na základě tohoto vektoru

Strojový překlad

síť nejprve “spolkne” celou vstupní sekvenci (větu), pak teprve začne generovat překlad

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

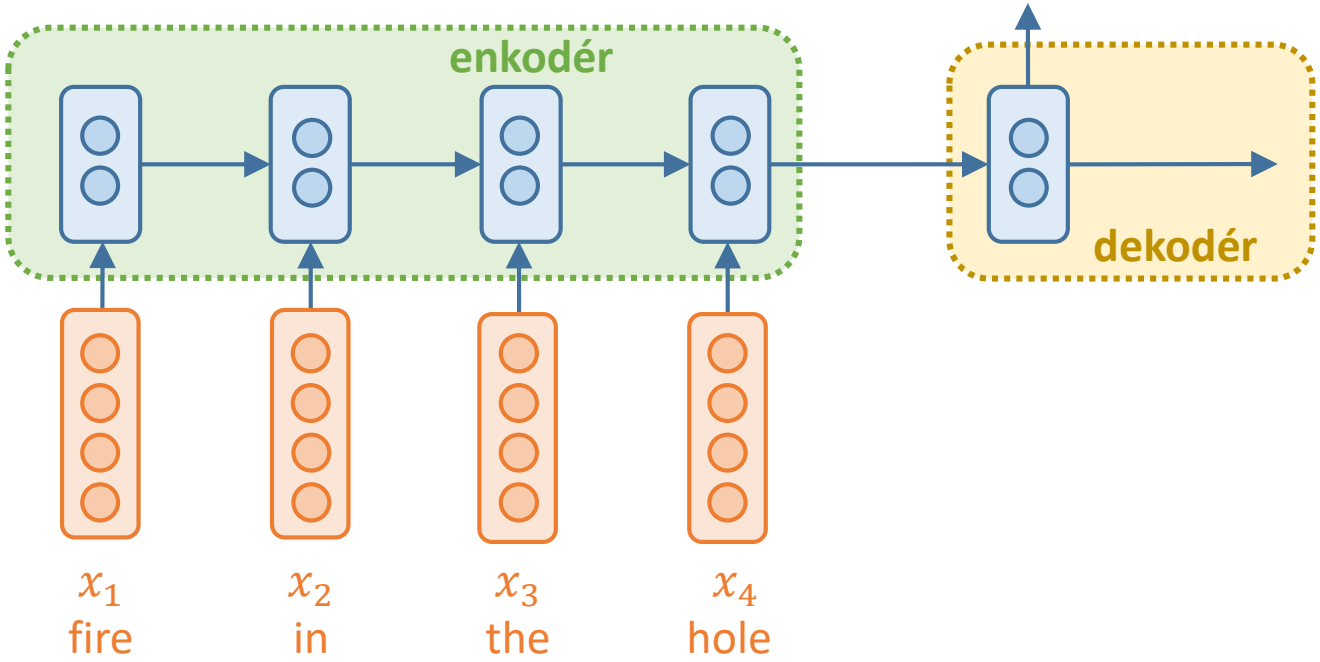


<u>podobný problém i zde:</u> na to, aby síť vygenerovala celý překlad, jí musí stačit skrytý stavový vektor h_4, slova “fire” a “oheň” jsou však od sebe vzdálena potenciálně neomezeně → problém s gradientem

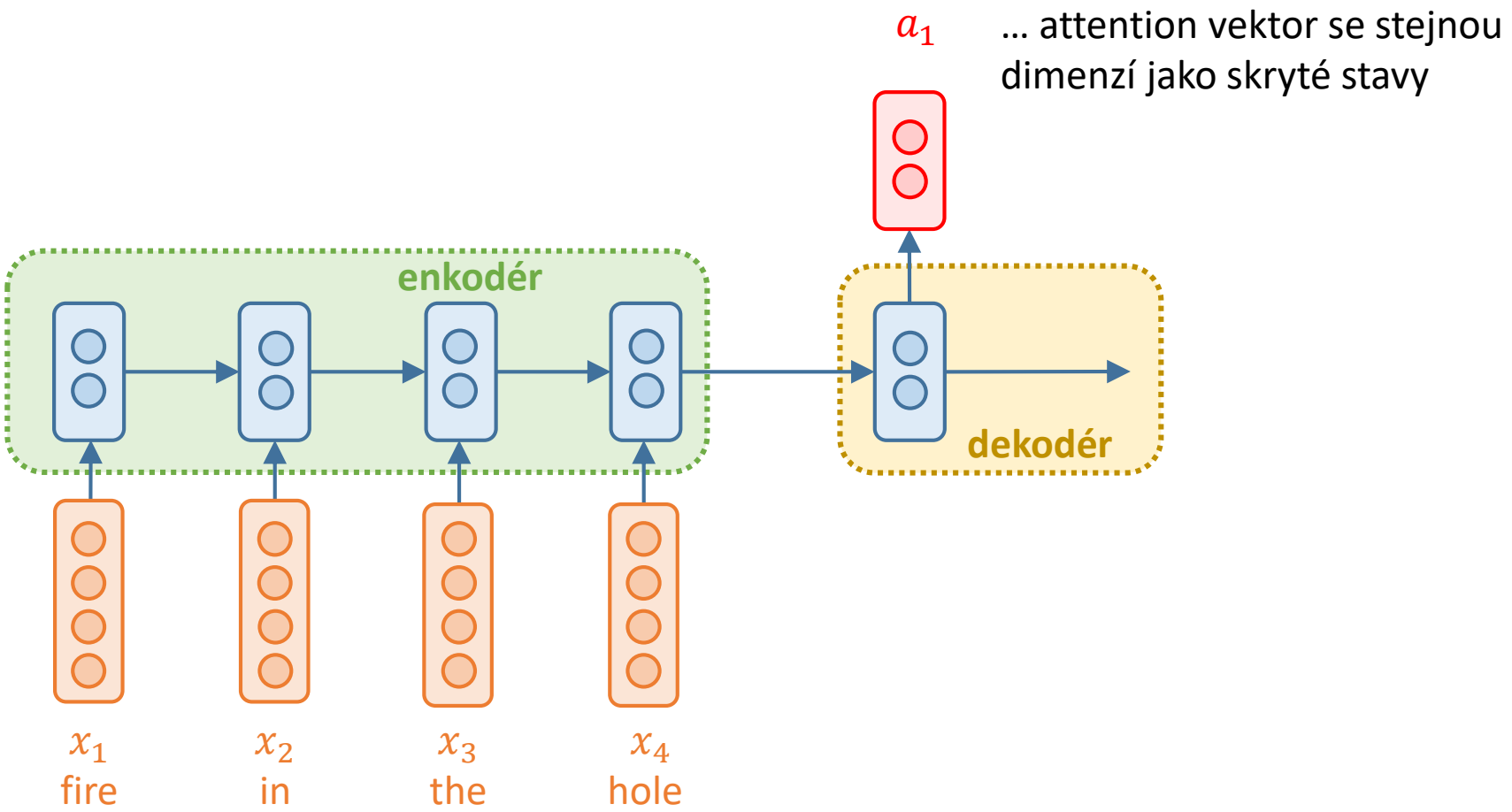
Attention

- 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

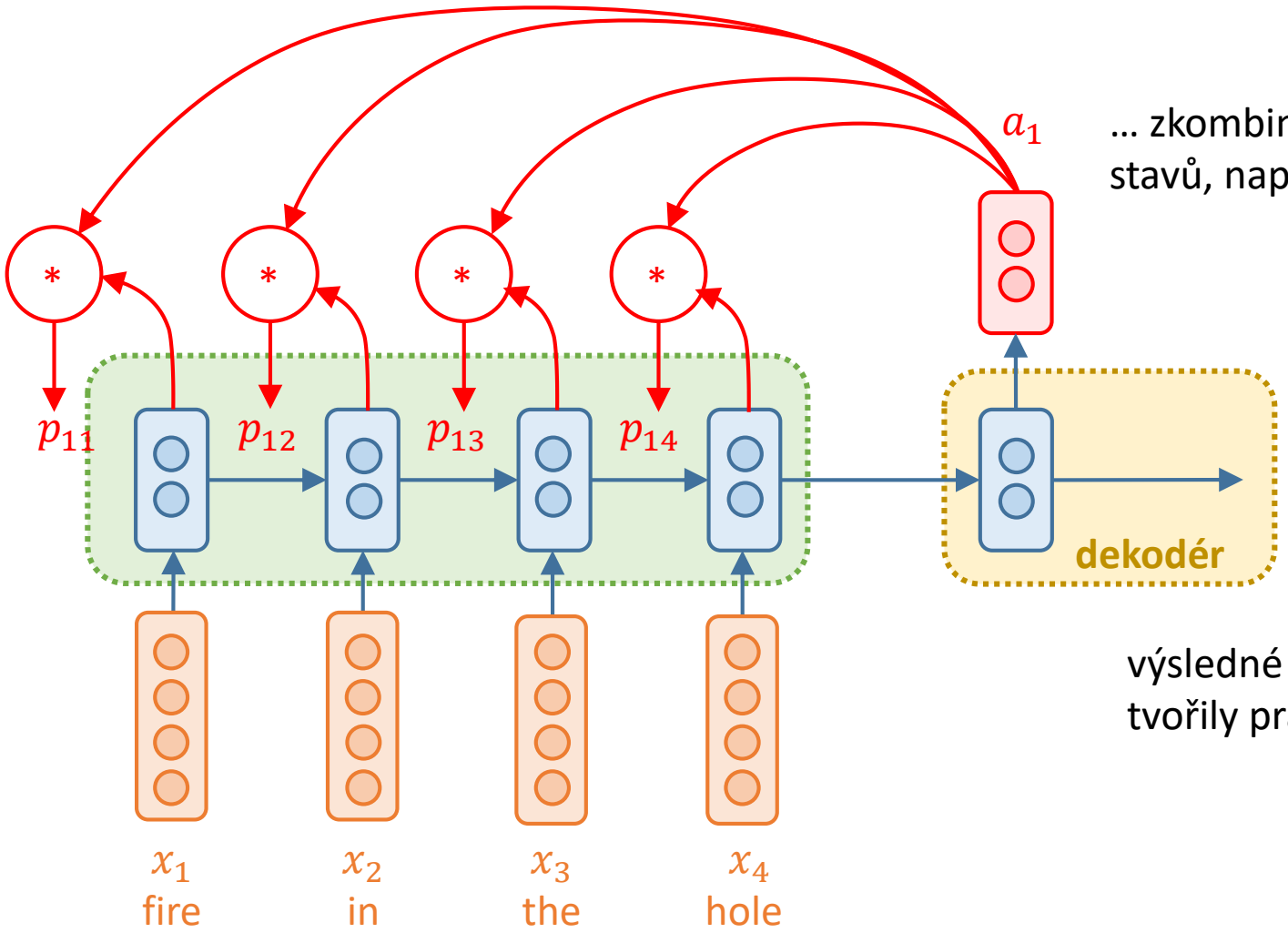
Princip attention pro strojový překlad



Princip attention pro strojový překlad



Princip attention pro strojový překlad



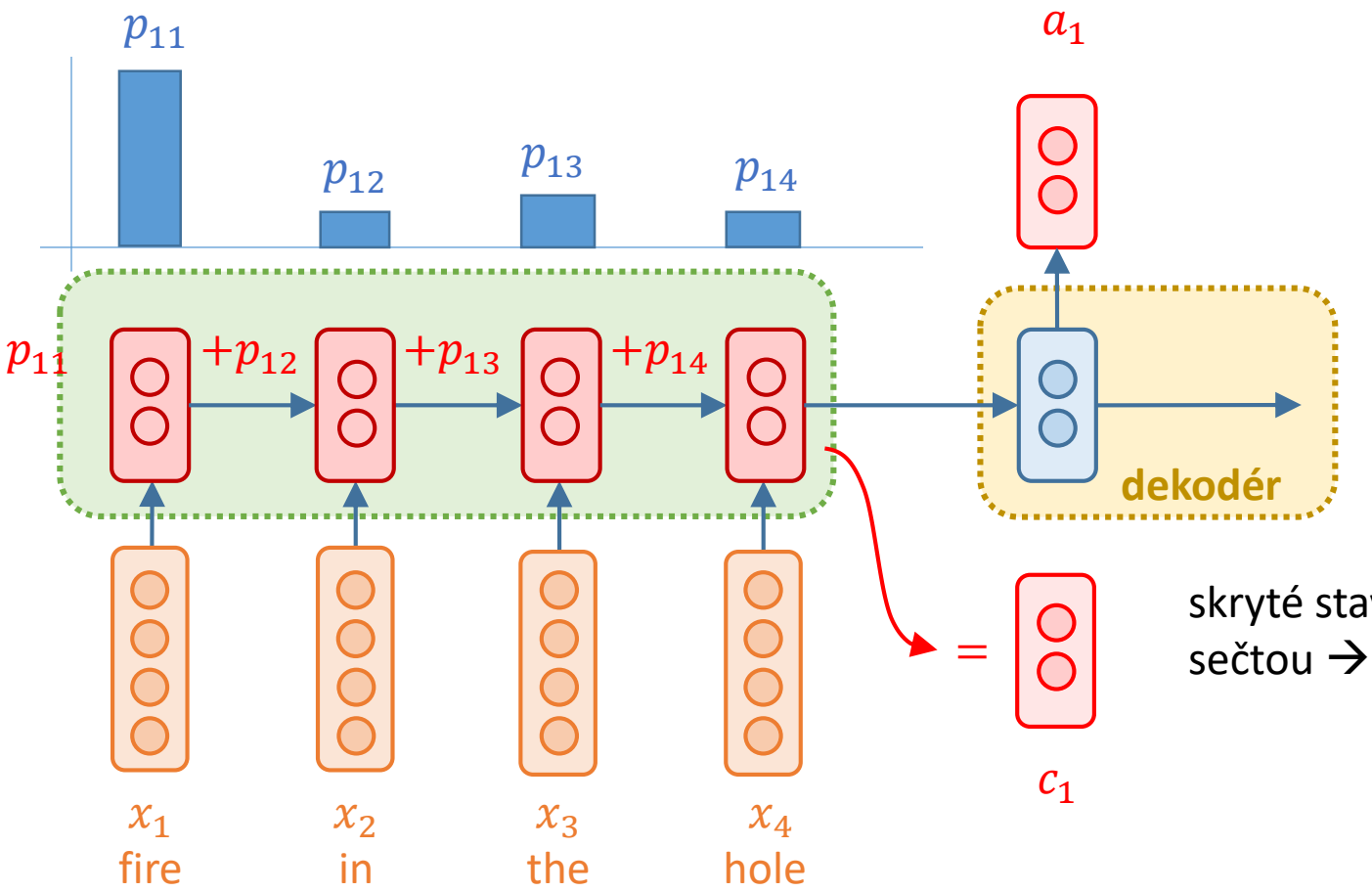
… zkombinuje se s vektory skrytých stavů, např. jako skalární součin

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

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

Princip attention pro strojový překlad

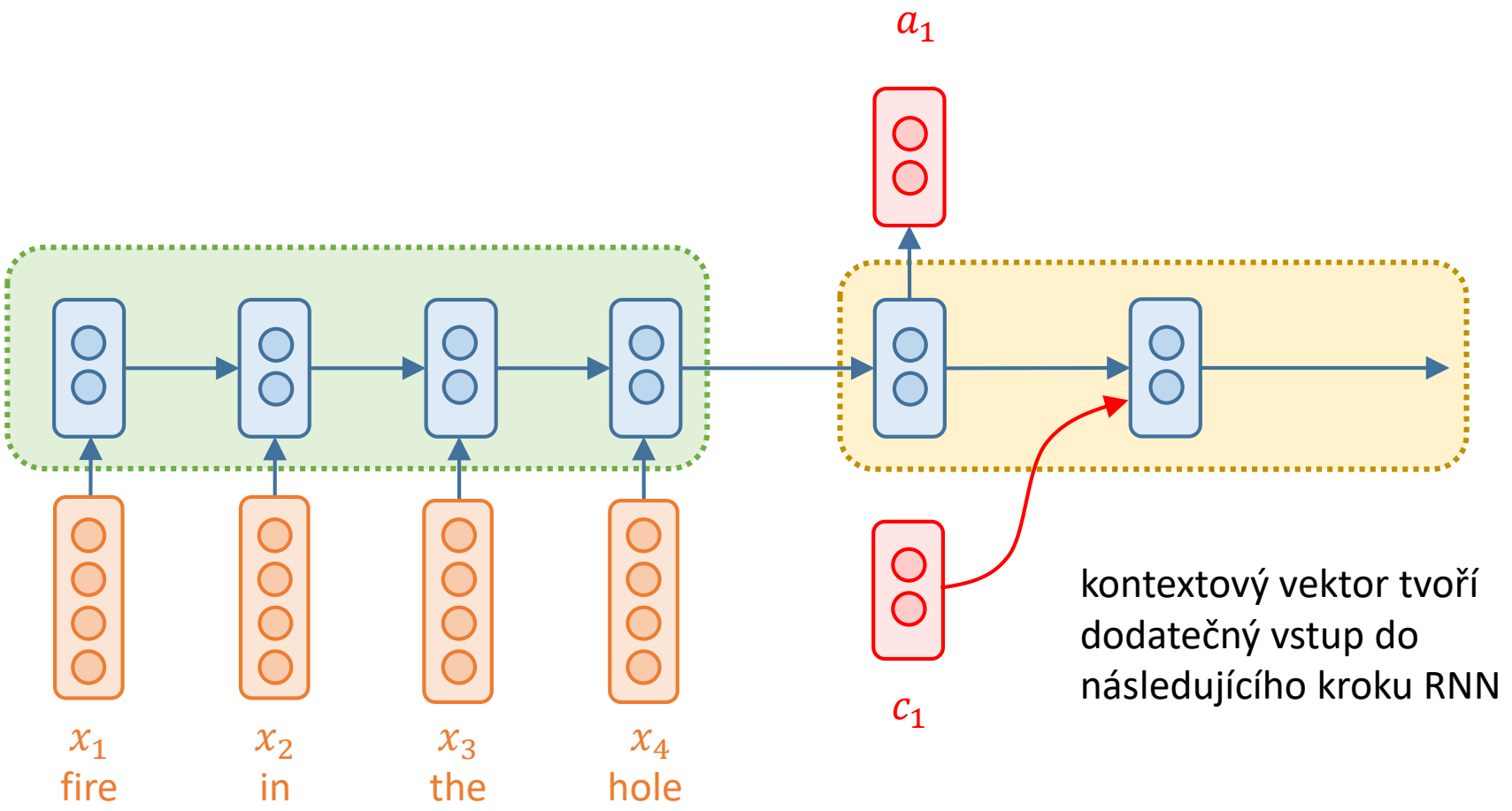
tam, kde jsou si attention vektor a_1 a skrytý stav h_t podobné → velké p_{1t} = "soustředění pozornosti"



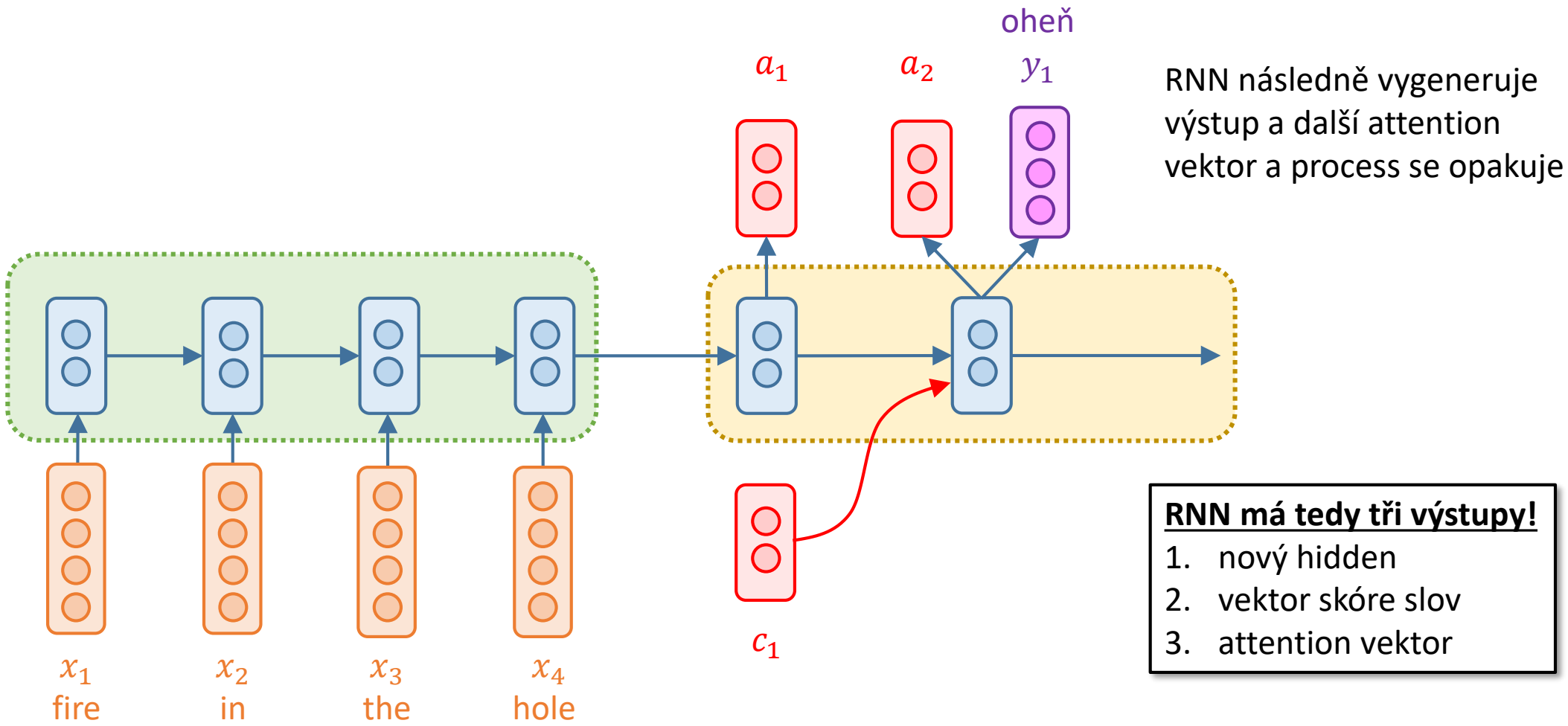
skryté stavy h_t se pronásobí koeficienty p_{1t} a sečtou → vznikne tzv. kontextový vektor

$$c_1 = \sum_{t=1}^{4} p_{1t} h_t$$

Princip attention pro strojový překlad



Princip attention pro strojový překlad

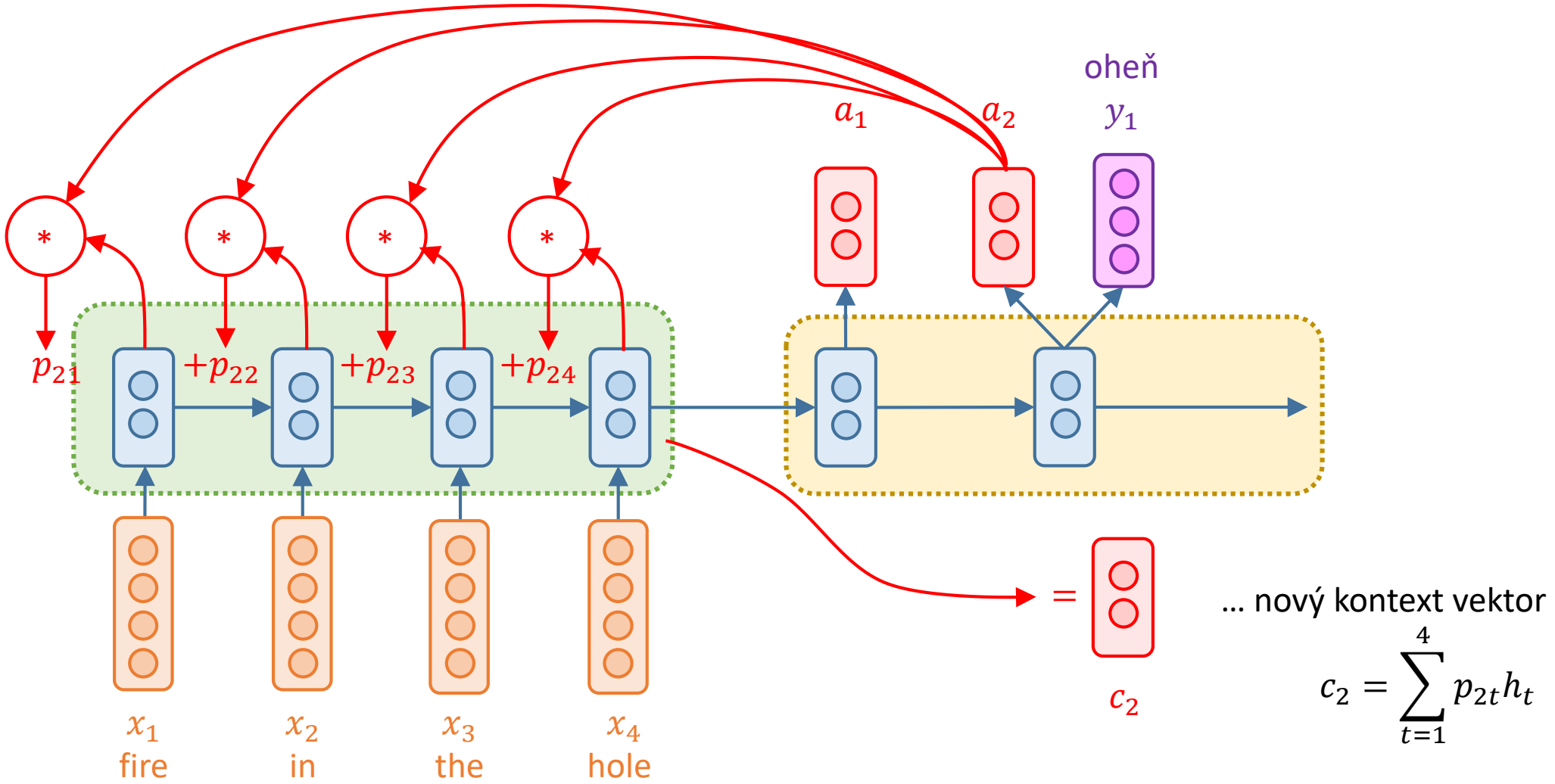


RNN následně vygeneruje výstup a další attention vektor a process se opakuje

RNN má tedy tři výstupy!

1. nový hidden
2. vektor skóre slov
3. attention vektor

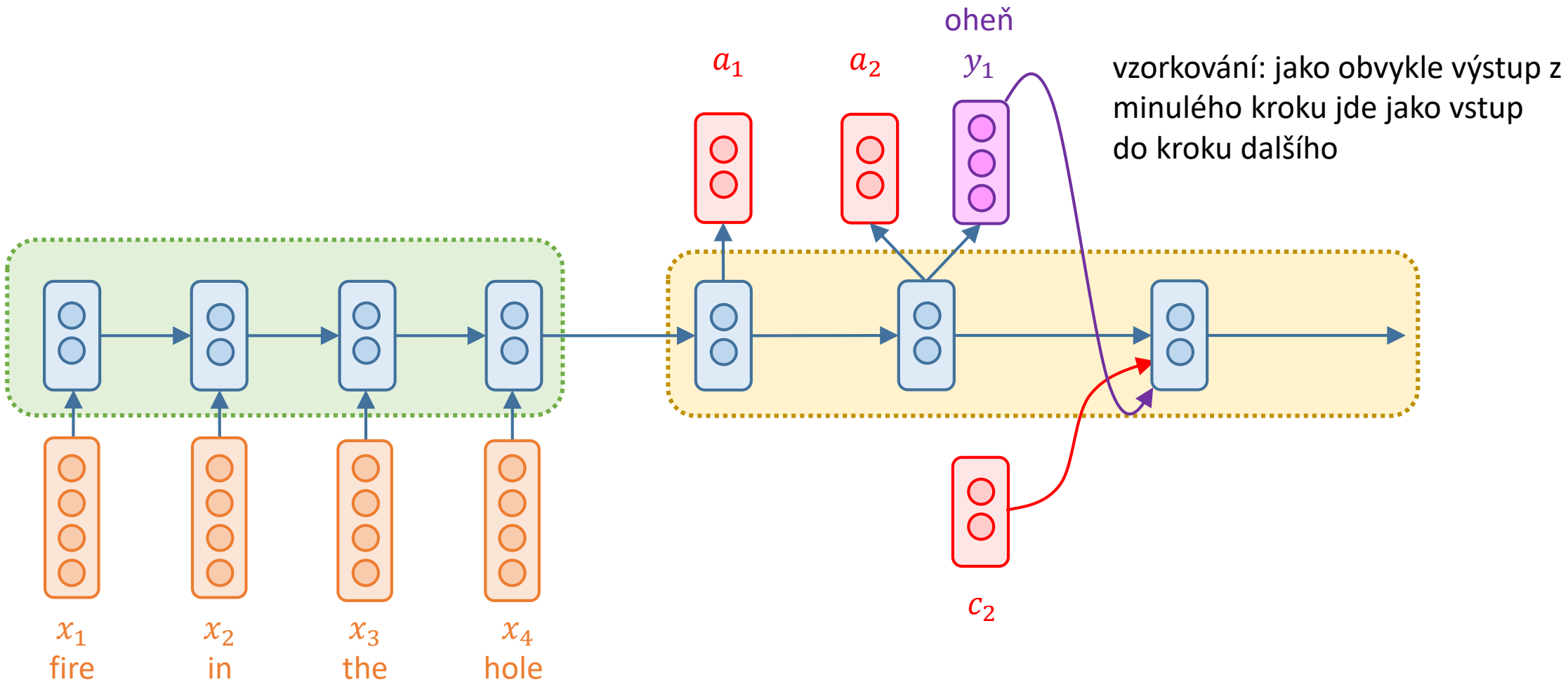
Princip attention pro strojový překlad



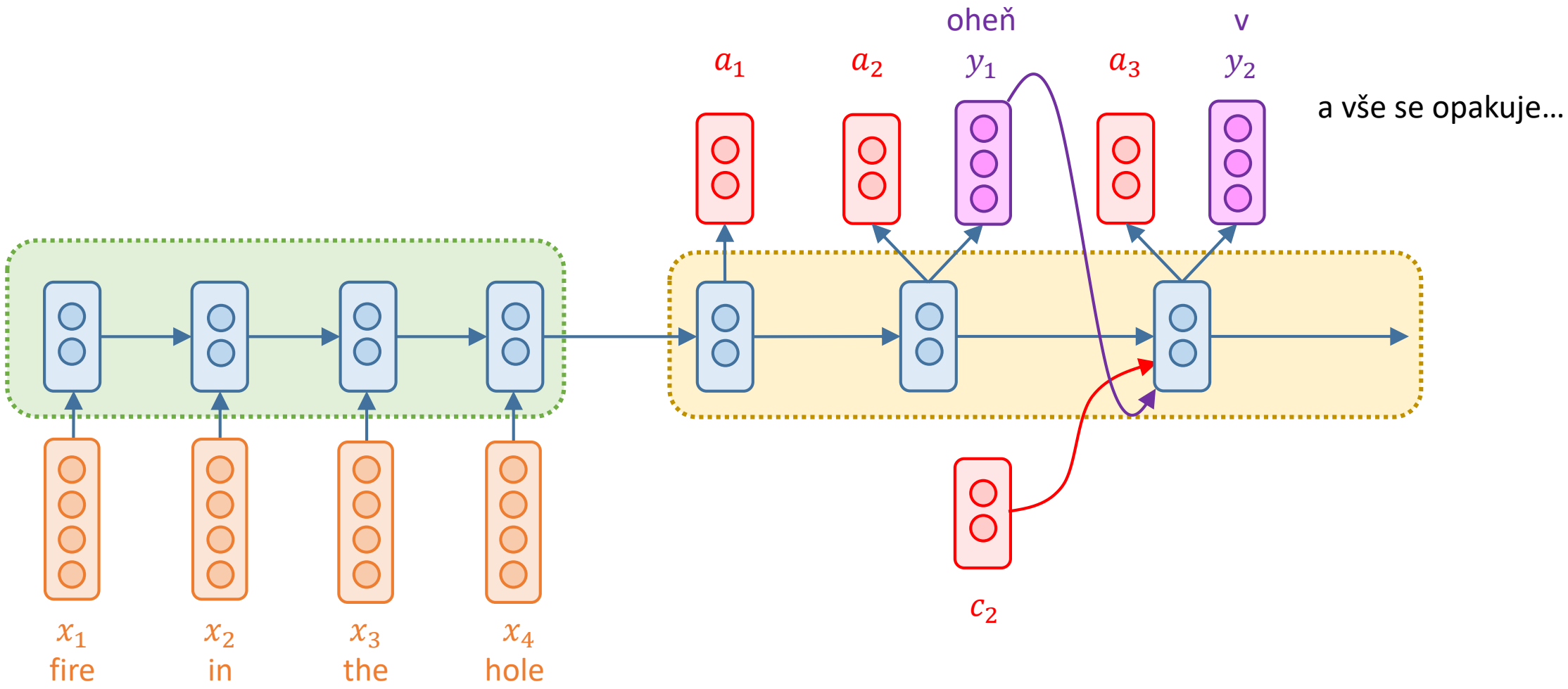
… nový kontext vektor

$$c_2 = \sum_{t=1}^{4} p_{2t} h_t$$

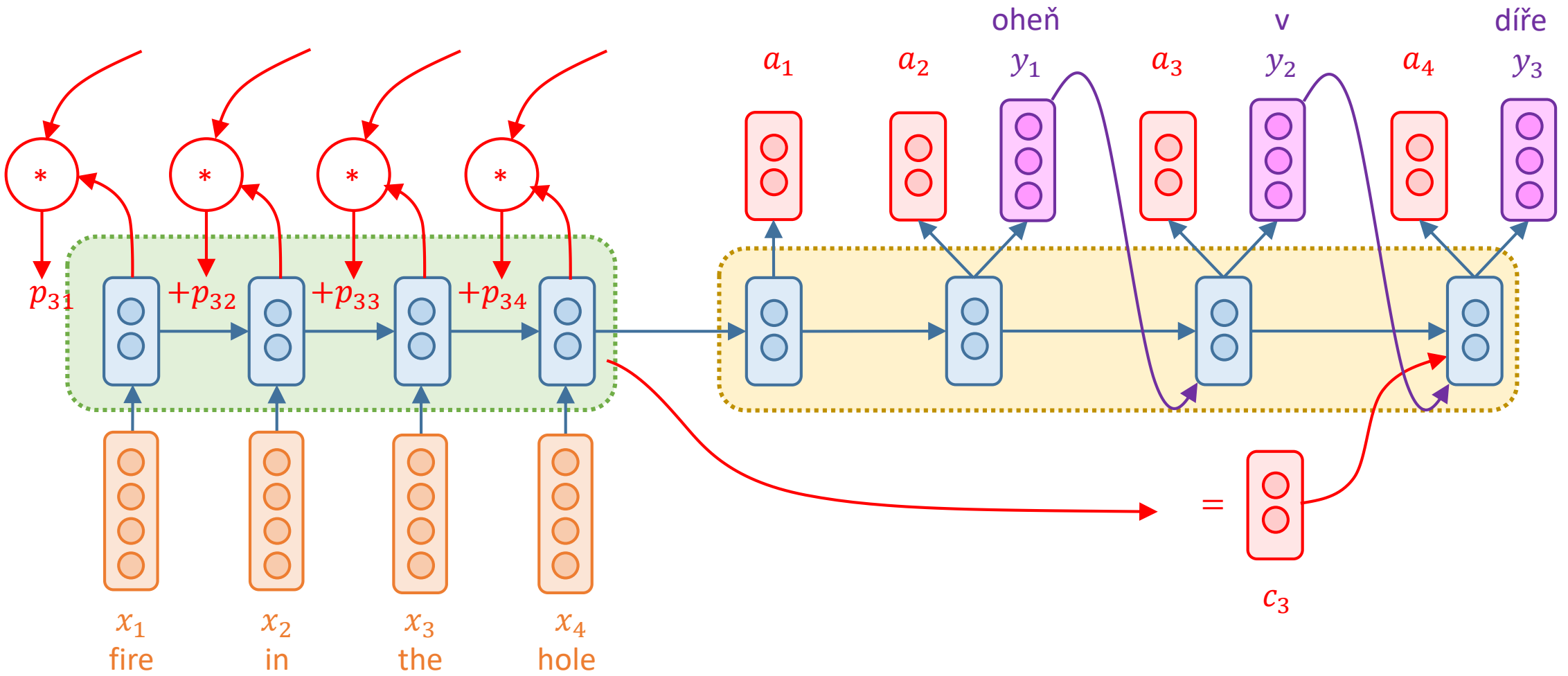
Princip attention pro strojový překlad



Princip attention pro strojový překlad

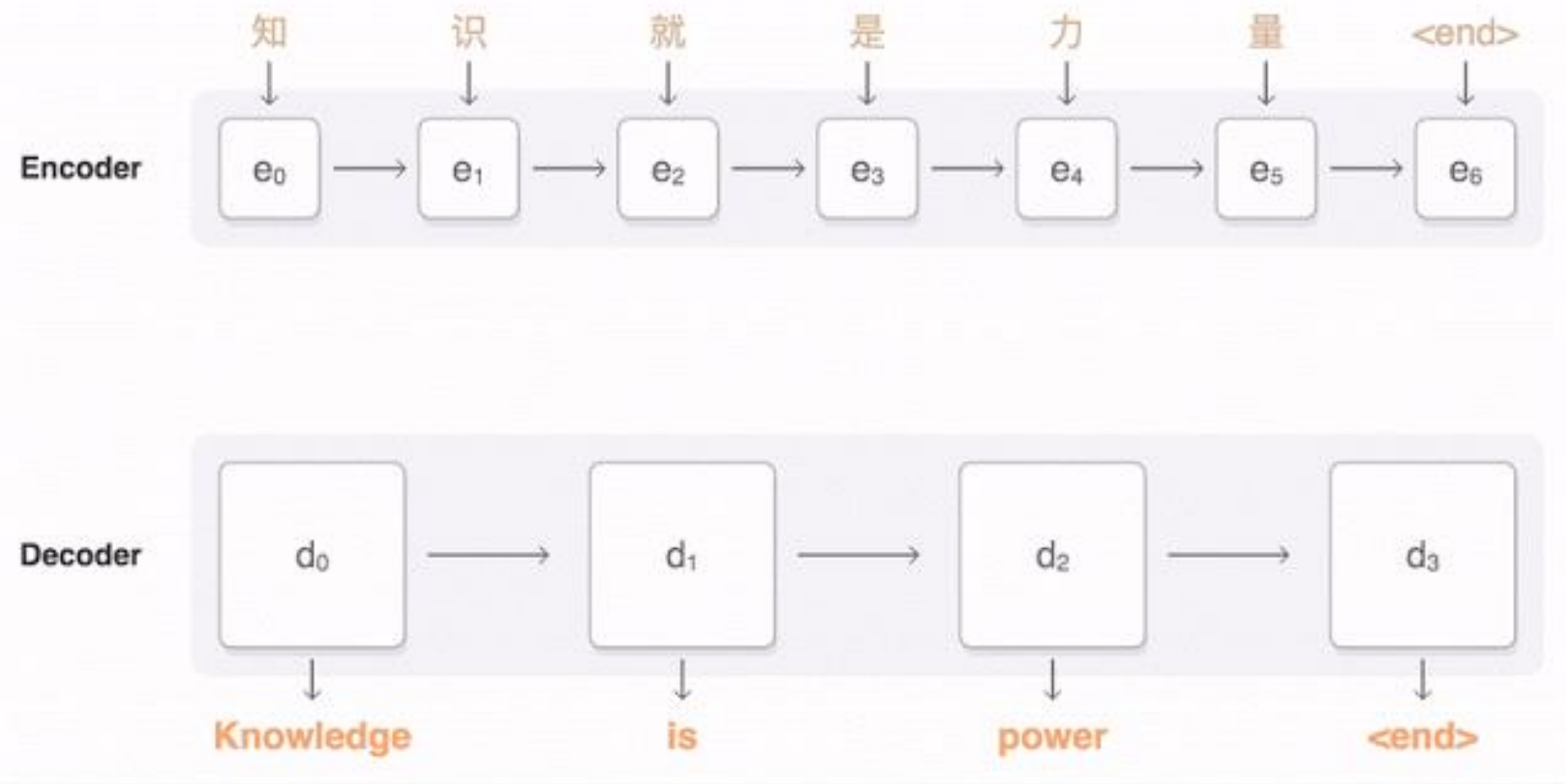


Princip attention pro strojový překlad



Attention pro strojový překlad: animace

(ne)průhlednost čar znázorňuje váhu = výsledné koeficienty p_t

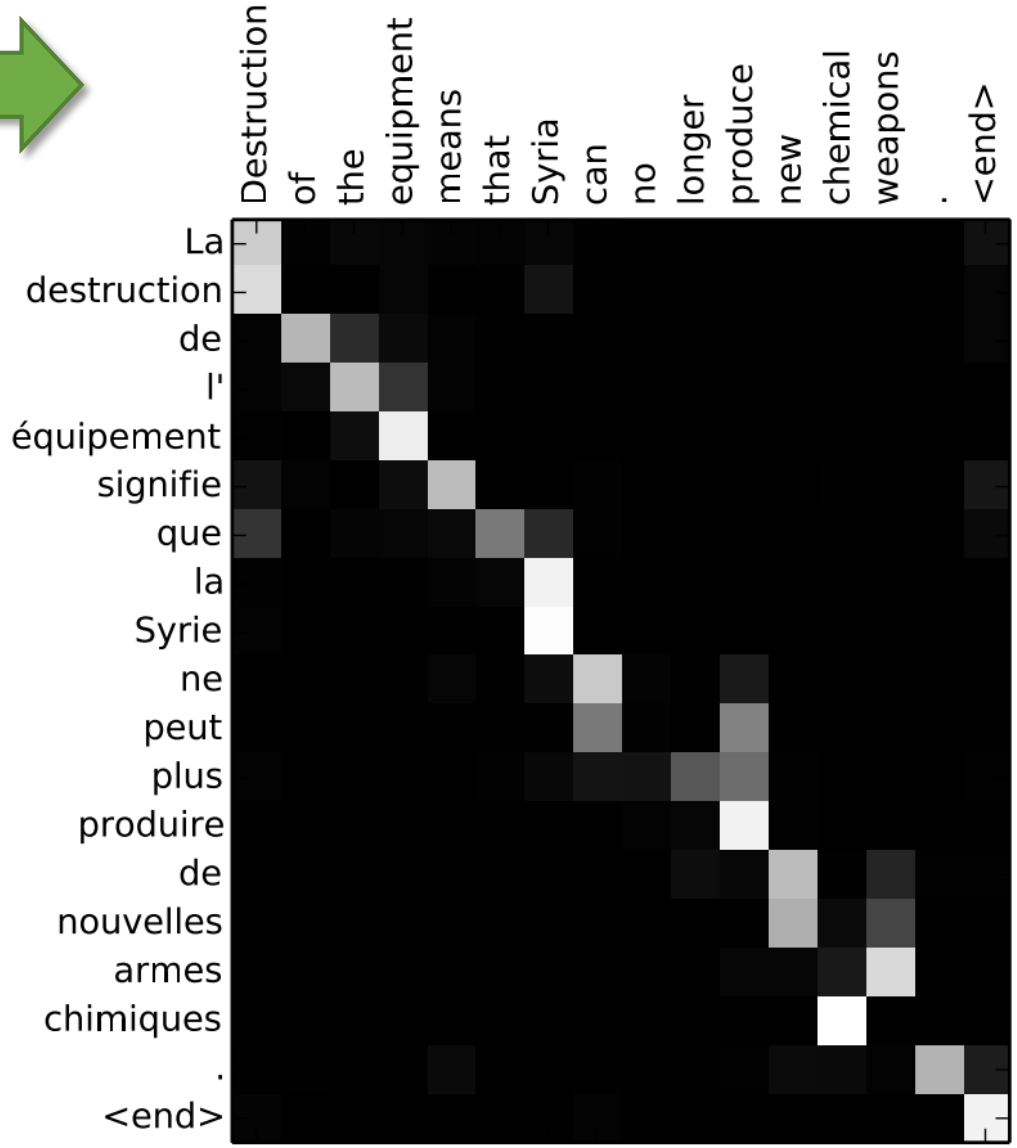


animace: https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

Vizualizace attention vah pro strojový překlad

překlad

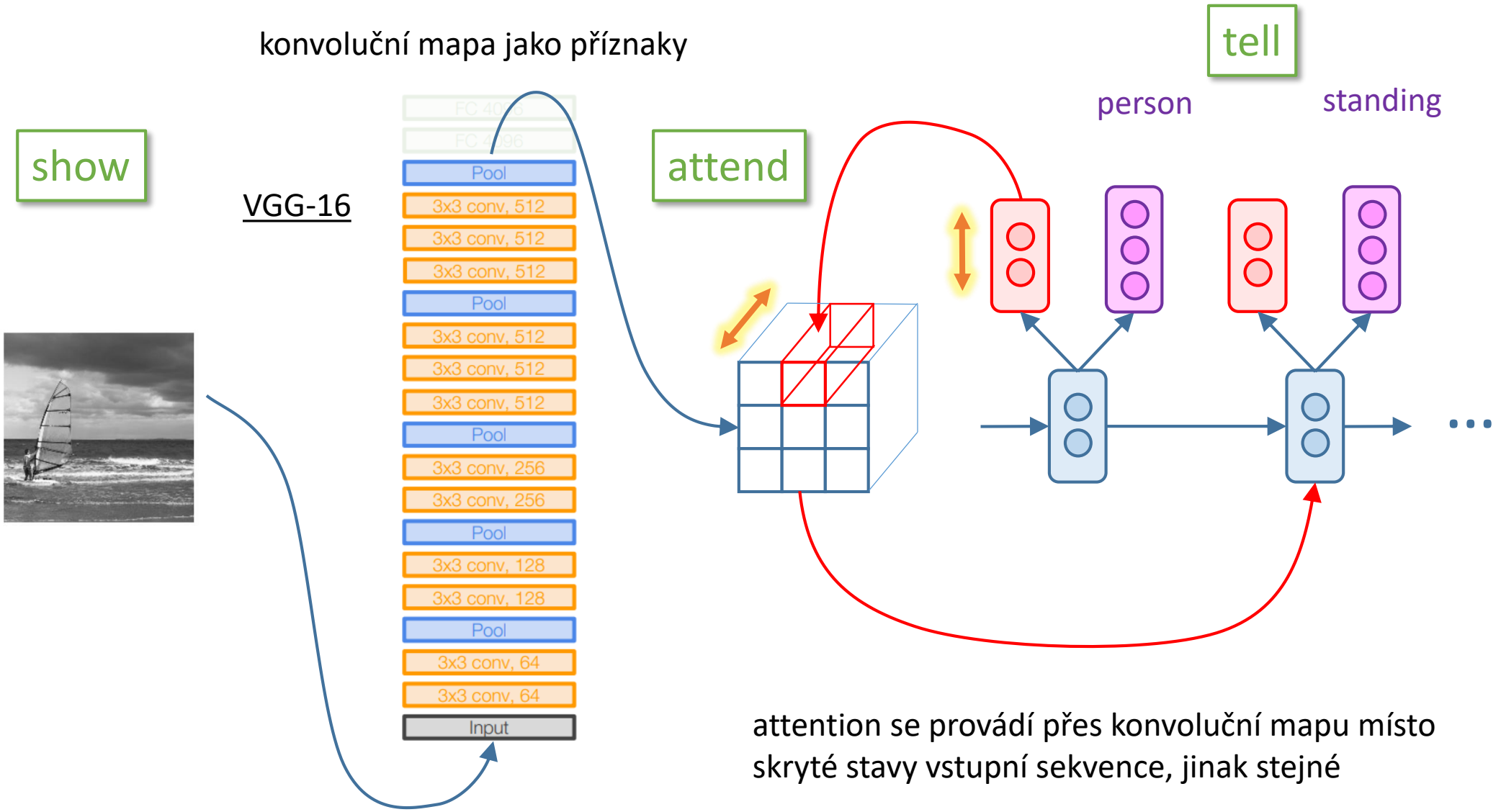
vstupní věta



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

obrázek: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

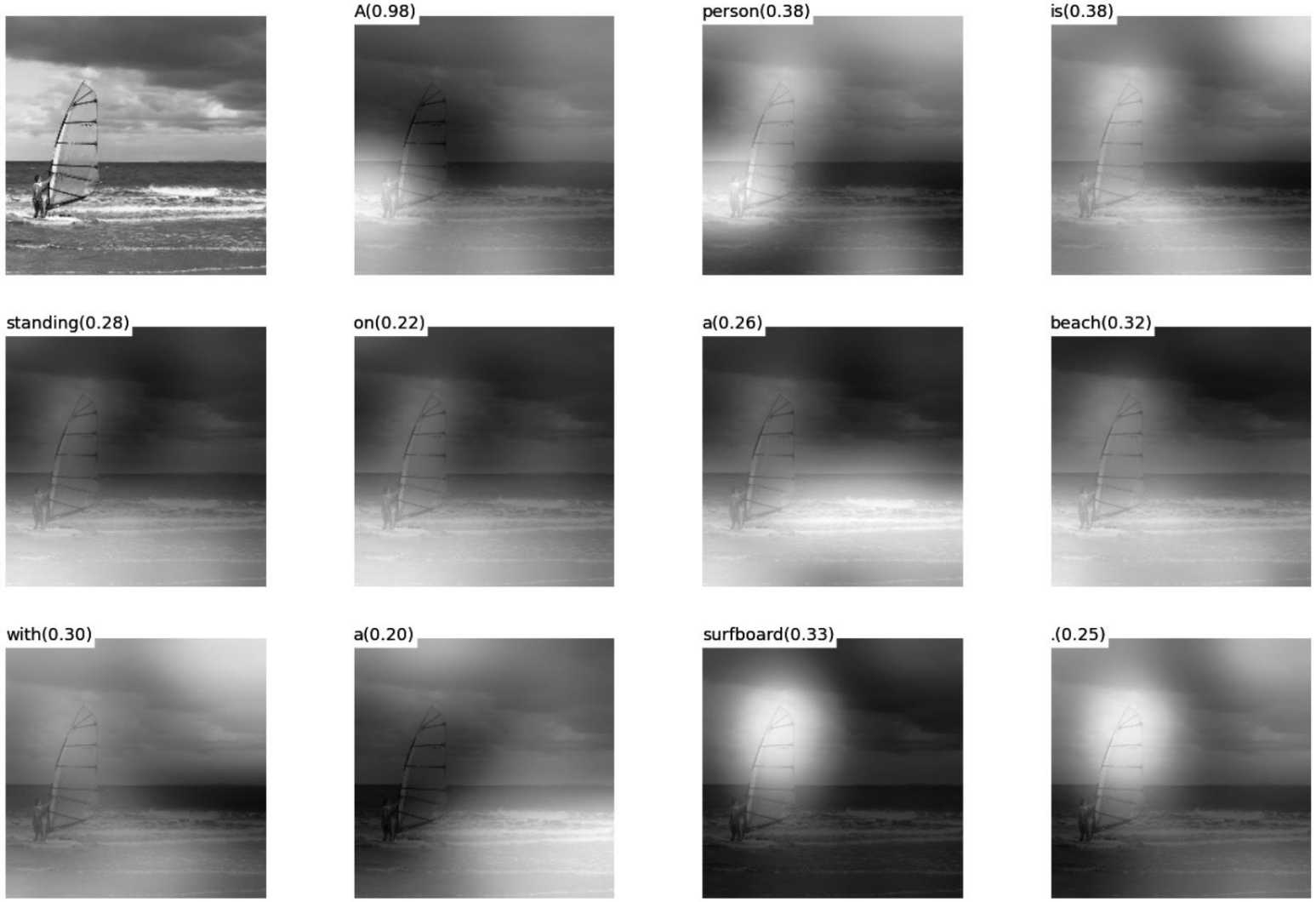
Attention pro obrazová data: show, attend and tell



attention se provádí přes konvoluční mapu místo skryté stavy vstupní sekvence, jinak stejné

např. článek: Xu et al.: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Vizualizace attention vah pro obrazová data



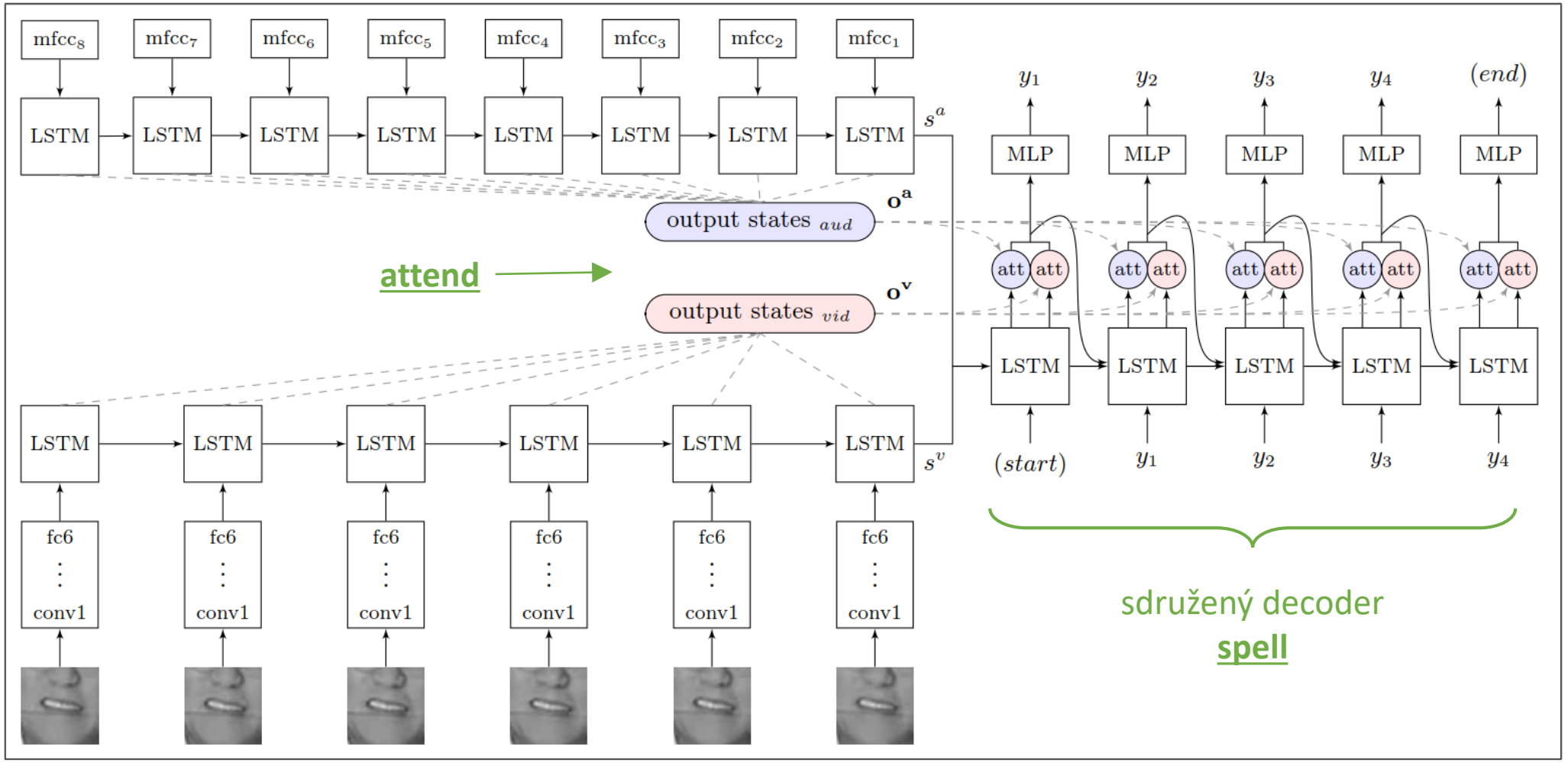
(b) A person is standing on a beach with a surfboard.

- 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

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

audio: **listen**
enkodér

video: **watch**
enkodér



sdružený decoder
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

člověk expert!

trénovací data

Channel	Series name	# hours	# sent.
BBC 1 HD	News†	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. †BBC News at 1, 6 and 10.

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

Method	SNR	CER	WER	BLEU†
Lips only				
Professional‡	-	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
Audio only				
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
Audio and lips				
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. †Unigram BLEU with brevity penalty. ‡Excluding samples that the lip reader declined to annotate. Including these, the CER rises to 78.9% and the WER to 87.6%.

zdroj: Chung, Zisserman: Lip Reading Sentences in the Wild

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).

zdroj: Chung, Zisserman: Lip Reading Sentences in the Wild

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

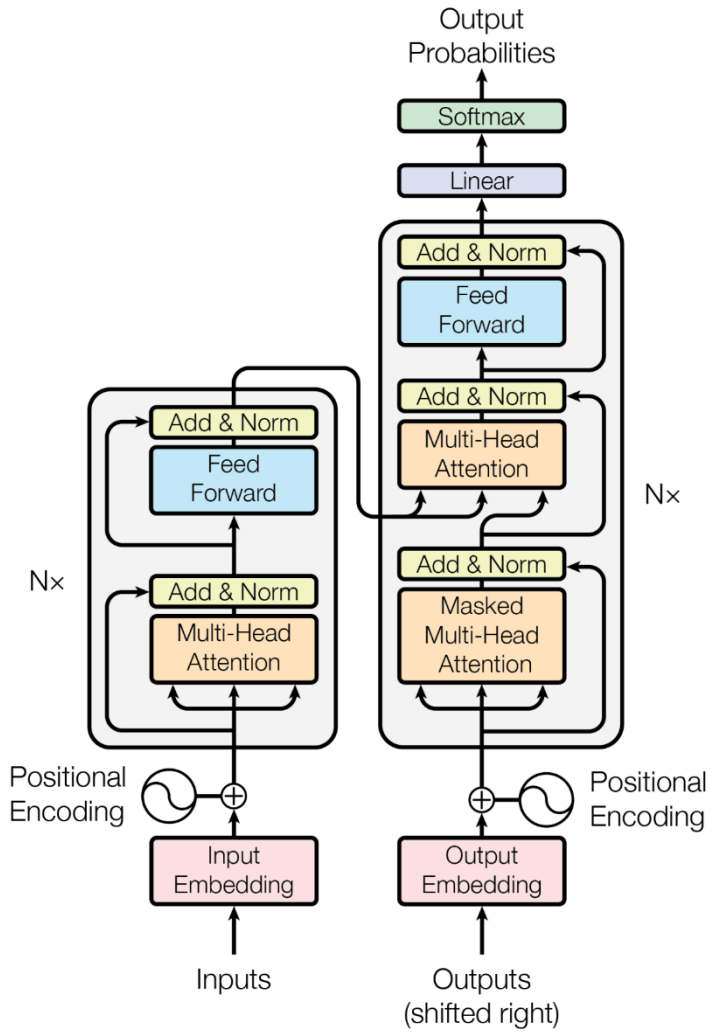
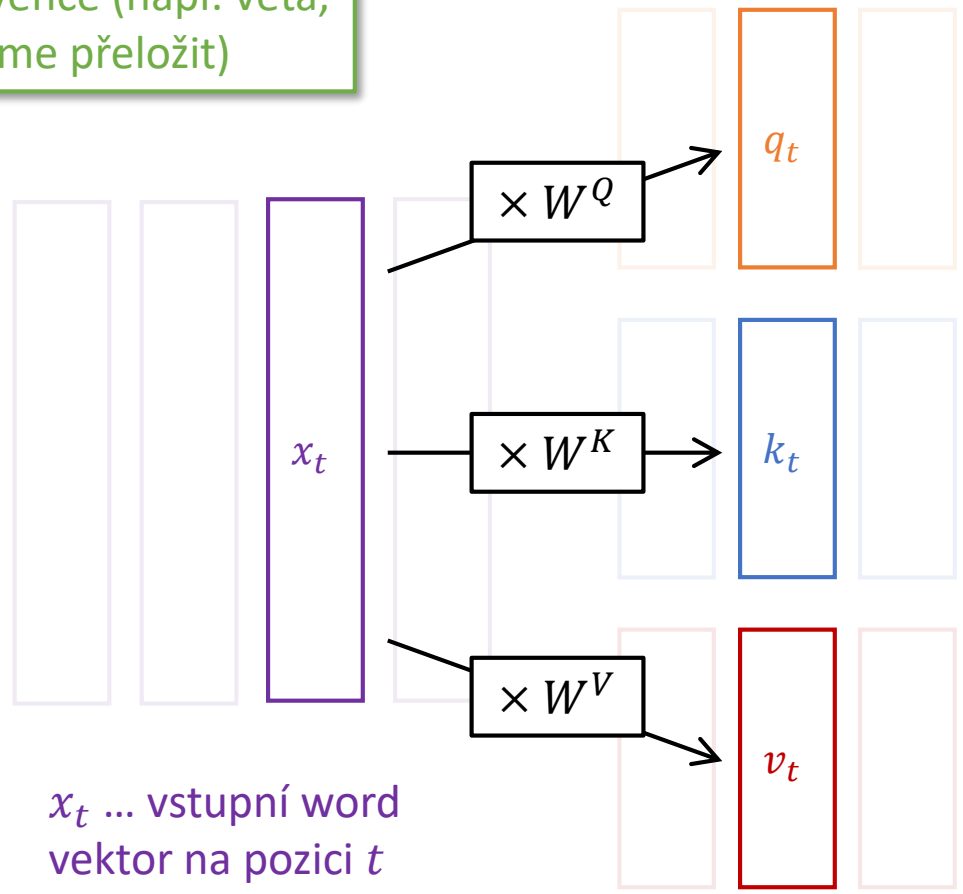


Figure 1: The Transformer - model architecture.

Self attention

vstupní sekvence (např. věta, kterou chceme přeložit)

pro každý x_t se vytvoří 3 různé vektory q_t, k_t, v_t



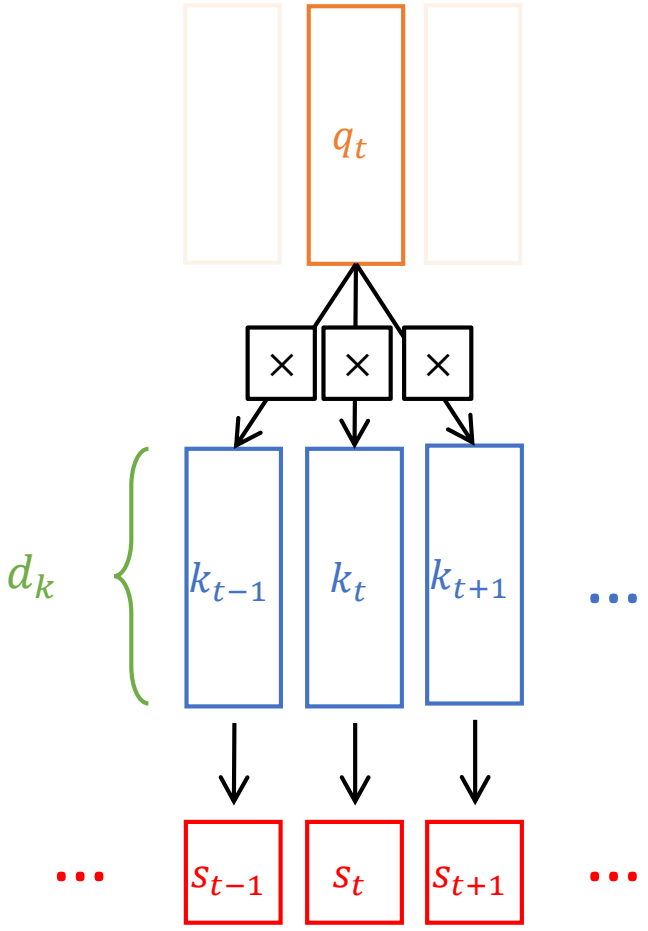
q_t … query vektor $q_t = W^Q \cdot x_t$

k_t … key vektor $k_t = W^K \cdot x_t$

v_t … value vektor $v_t = W^V \cdot x_t$

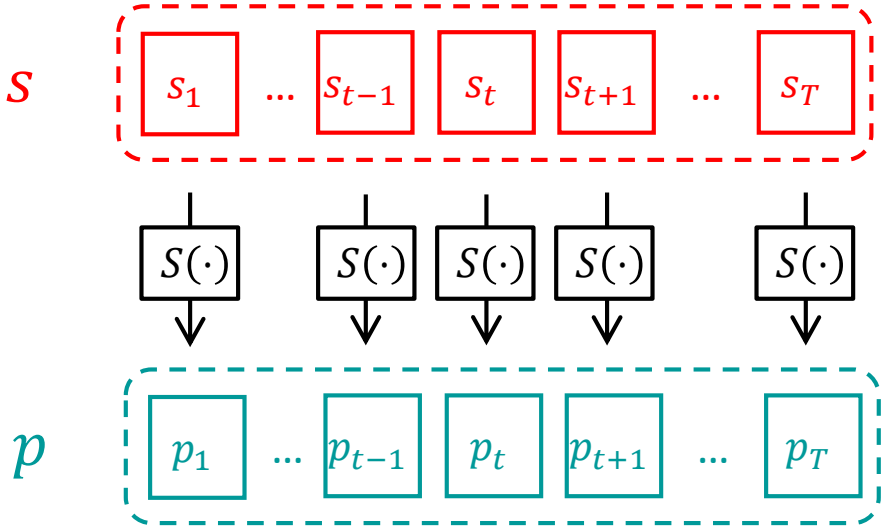
x_t … vstupní word vektor na pozici t

Self attention



$$\begin{aligned} \dots &= \dots \\ s_{t-1} &= q_t k_{t-1} \\ s_t &= q_t k_t \\ s_{t+1} &= q_t k_{t+1} \\ \dots &= \dots \end{aligned}$$

$$s = q_t^T K$$

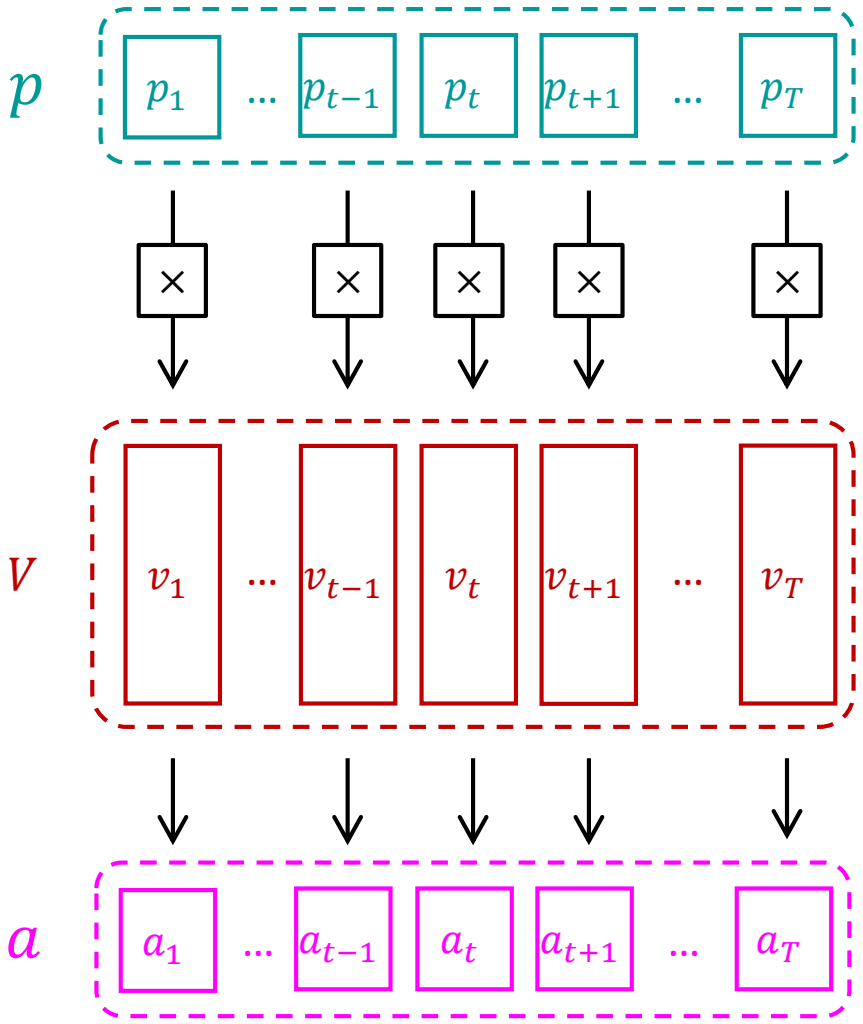


$$p = \text{softmax}\left(\frac{s}{\sqrt{d_k}}\right)$$

d_k … rozměr key vektorů

p … attention váhy

Self attention



pro všechny query vektory ve vstupu (např. všechna slova) zároveň:

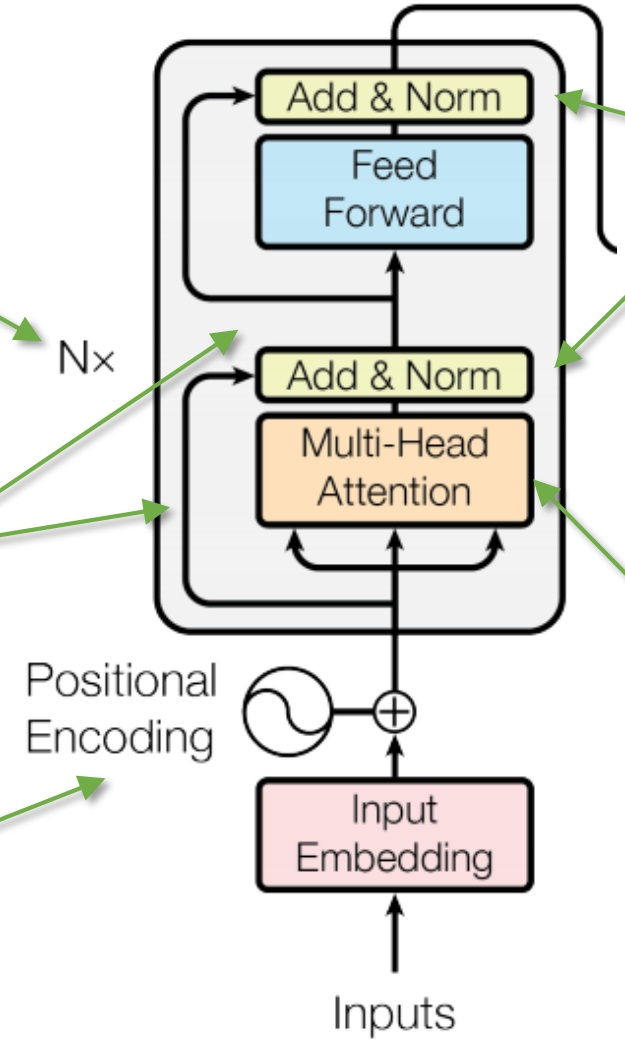
$$\mathrm{A} = \mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

což je rovnice (1) v článku; v práci uvažují řádkové vektory, proto jiné transpozice

a ... výstup self attention pro t-tý vektor

$$a = \mathrm{Attention}(q_t, K, V) = \mathrm{softmax}\left(\frac{q_t^T K}{\sqrt{d_k}}\right)V$$

Transformer: encoder



6-8 attention na sobě

residuální spoje

poziční embedding: jelikož síť není sekvenční, přičítá se ke vstupním embeddingům časově závislá funkce → slovní vektory obsahují informace o čase

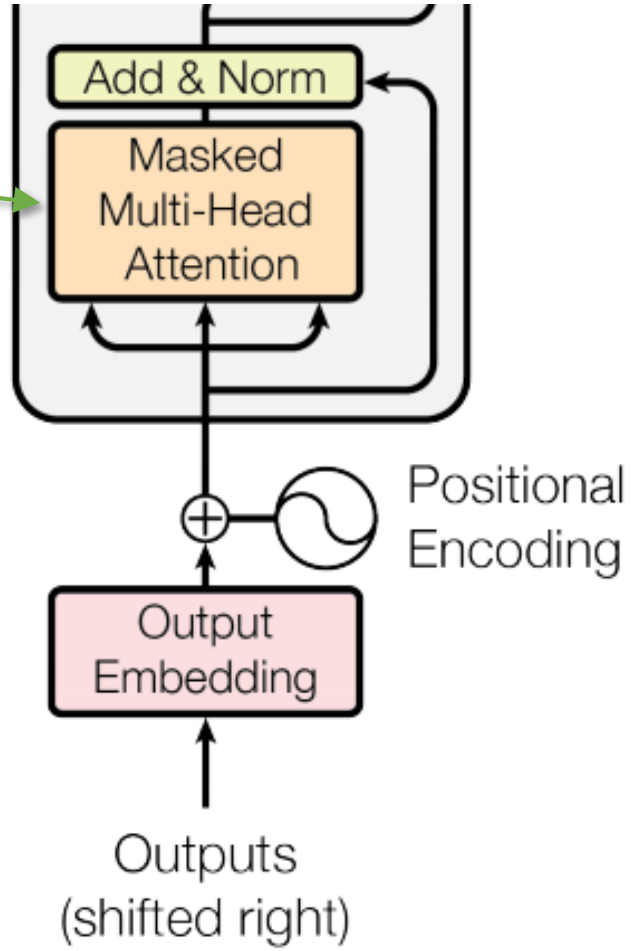
layer normalizace ("transponovaná" batch norm.)

multi-head

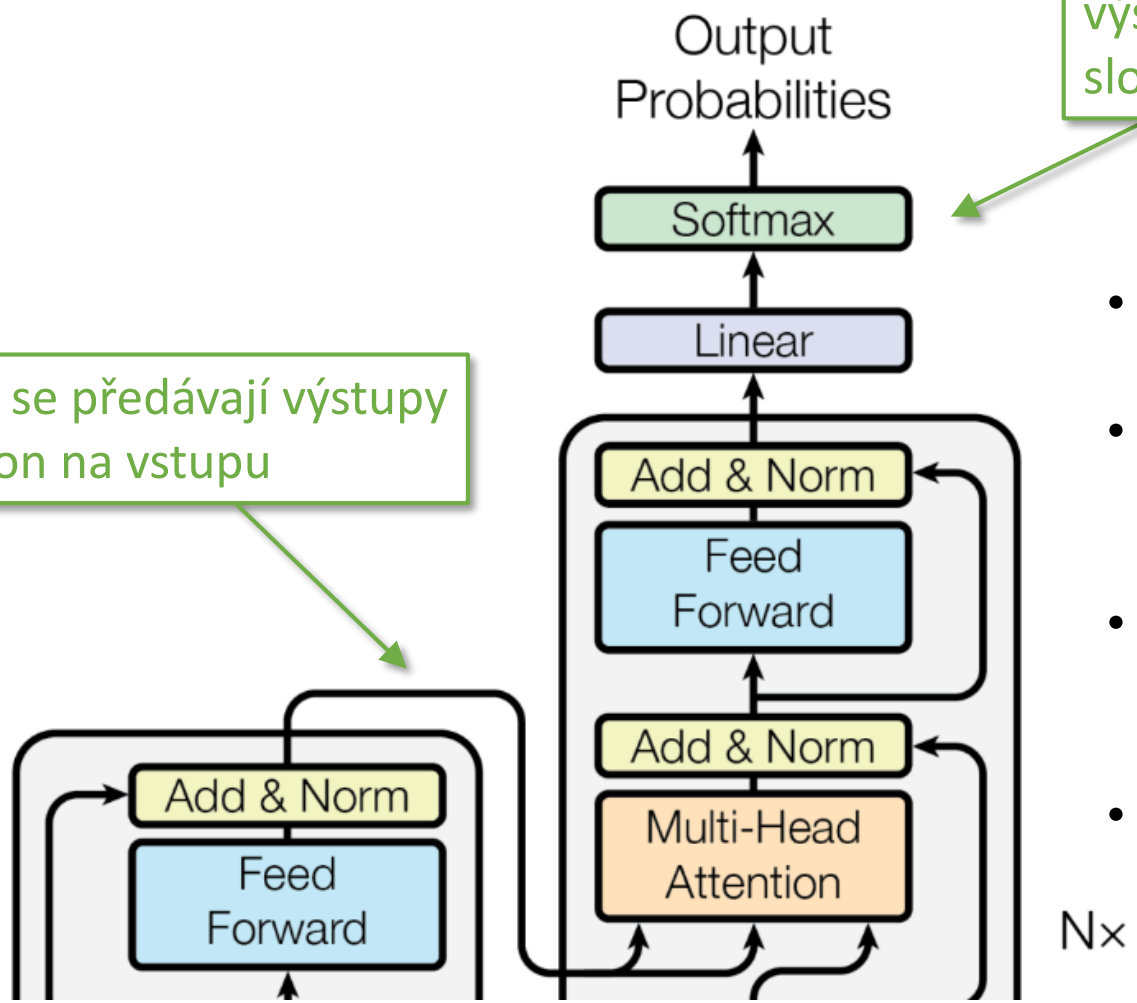
- uvedená self attention je v každé vrstvě 8x
- výstup všech je spojen do jedné matice
- výstup pak je $\mathrm{MultiHead}(Q, K, V) = [A^1, \ldots, A^8]W^O$

Transformer: decoder

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



Transformer: decoder



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

z enkodéru se předávají výstupy self attention na vstupu

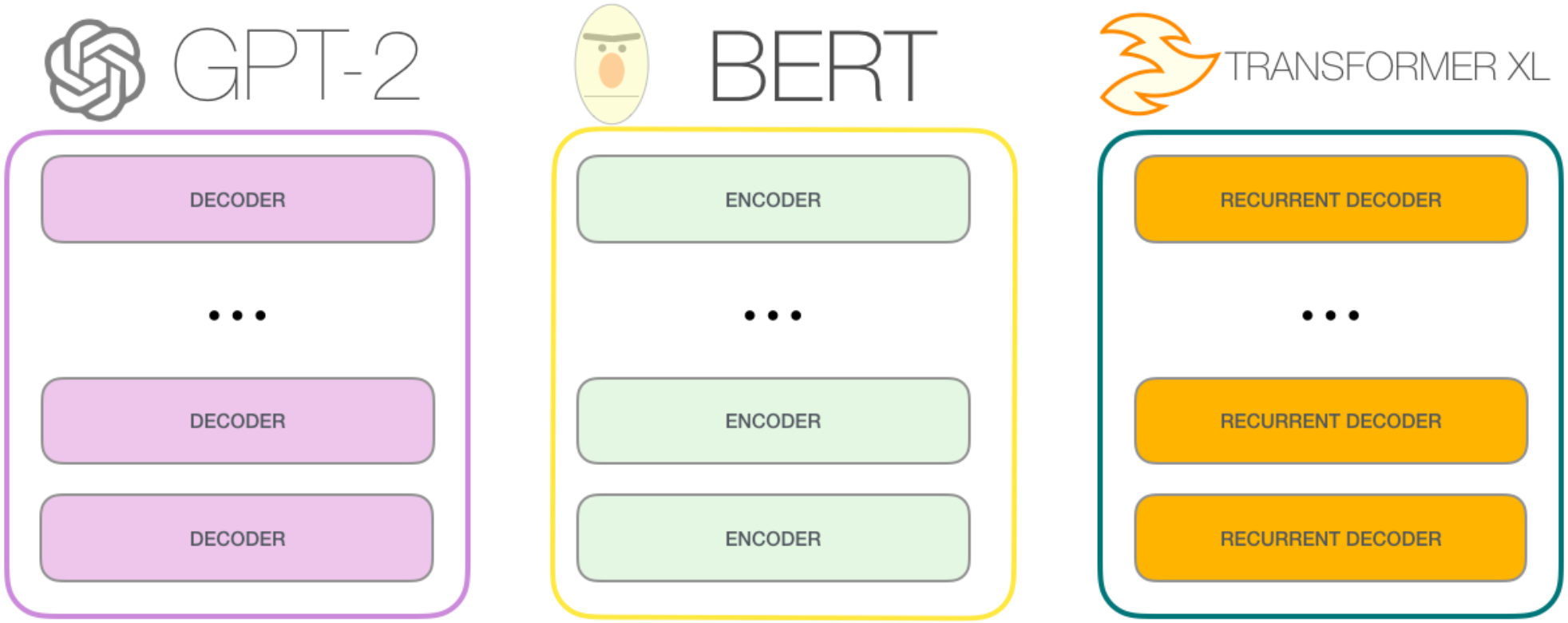
- trénování funguje kompletně paralelně, tj. všechna slova vstupu i výstupu najednou
- minimalizuje se cross entropy pro výstupní pravděpodobnosti na slova ze slovníku cílového jazyka (pokud jde o překlad)
- v inferenci dekodér produkuje slova postupně po jednom, není zde ovšem rekurence, pouze attention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

Transformer: výsledky v půdovním 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	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Další varianty



obrázek: http://jalammar.github.io/illustrated-gpt2/

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