# Aprendizado Profundo

*Language Modeling, Machine Translation e Attention*

Professor: Lucas Silveira Kupssinskü

# Agenda

- *Language Modeling*
  - *Language Modeling* com NNs
  - *Decoding* de *Language Models*
- seq2seq
  - Arquitetura *Encoder-Decoder*
  - *Machine Translation*
  - *Attention*

# *Language Modeling*

# *Language Modeling*

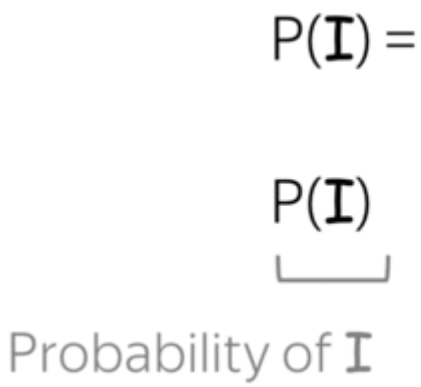
- O que significa “modelar” algum fenômeno?

# *Language Modeling*

- O que significa “modelar” algum fenômeno?
- Ter alguma capacidade preditiva sobre esse fenômeno

# *Language Modeling*

- Em Modelagem de Linguagem queremos ser capazes de computar a probabilidade de uma frase



# *Language Modeling*

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \dots P(y_n|y_1, \dots y_{n-1})$$

$$= \prod_{t=1}^{n} P(y_t|y_{<t})$$

P(I) =

P(I)

Probability of I

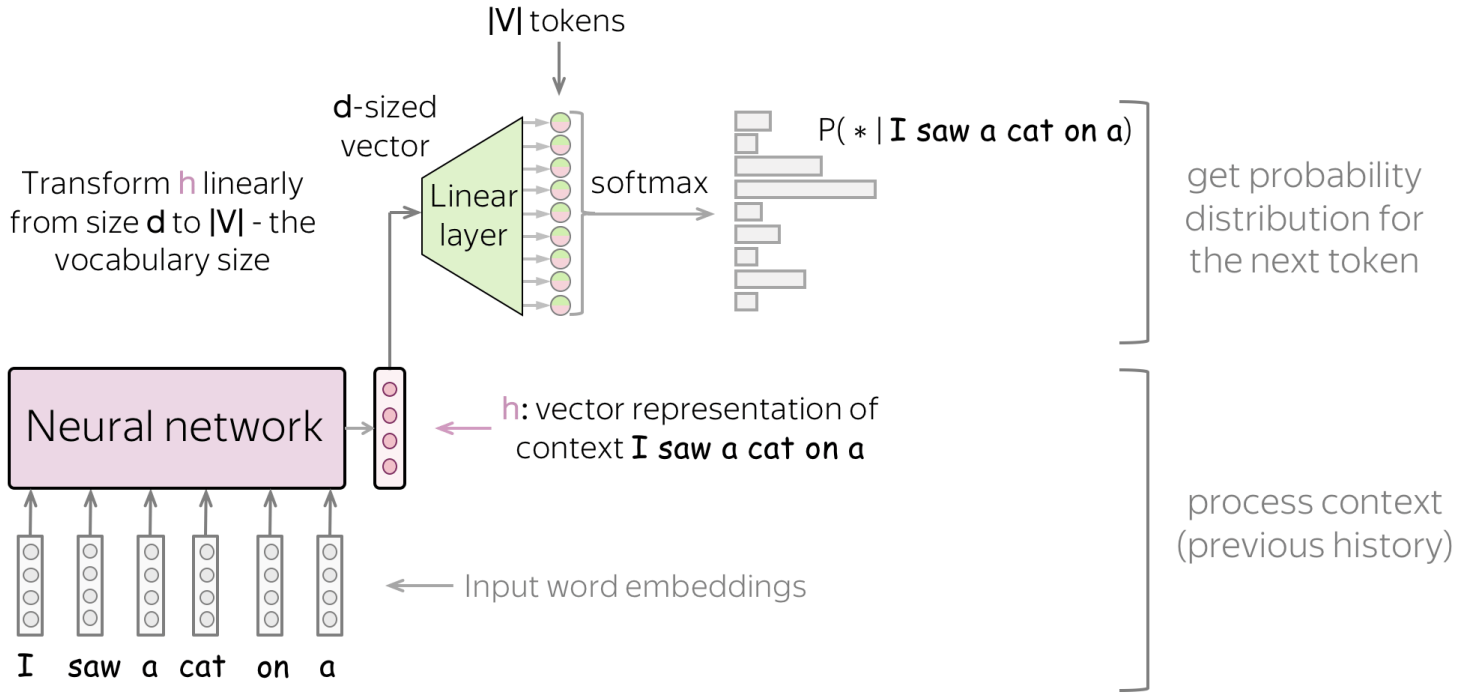
# *Language Modeling*

$$P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} P(y_t | y_{<t})$$

- Porém não temos meios práticos de computar essas probabilidades usando contagem simples
- Diversas abordagens baseadas em *n-grams* foram usadas ao longo dos anos
- O que tem funcionado melhor nos últimos anos é usar redes neurais para estimar essas probabilidades

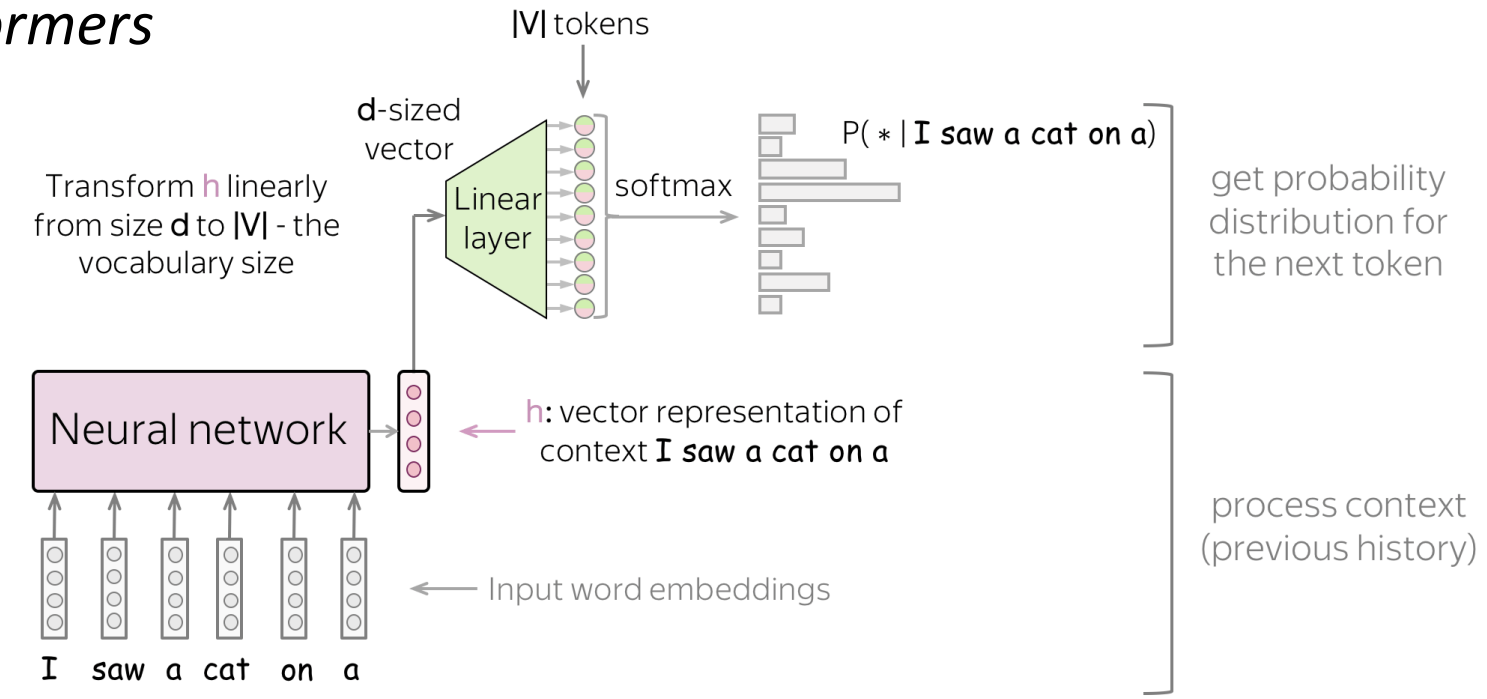
# *Language Modeling* com NNs

- *Language Model* com NN tipicamente possuem duas partes
  - Uma rede que processa o contexto
  - Uma rede que aproxima a distribuição de probabilidade do próximo *token*



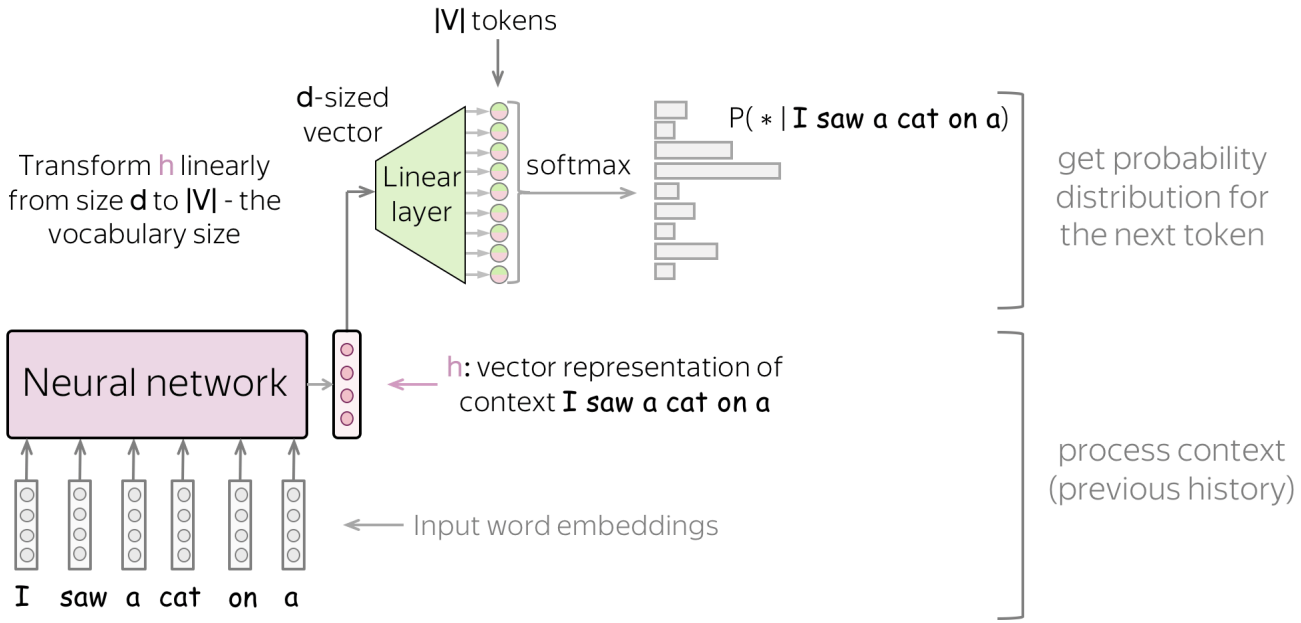
# *Language Modeling* com NNs

- Para processar contexto
  - *CNNs* (em desuso)
  - *RNNs*
  - *Transformers*



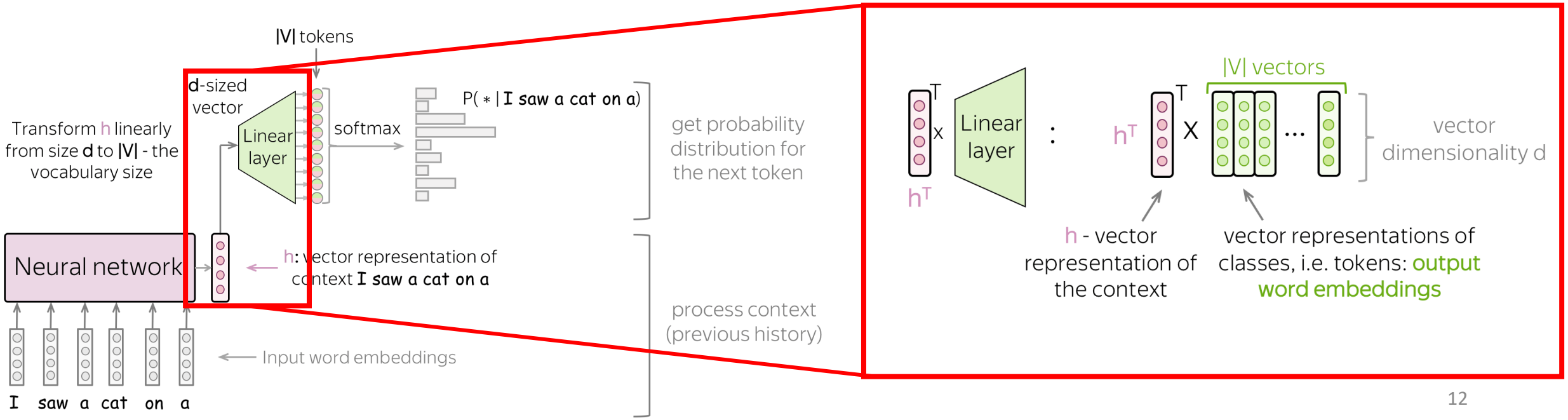
# *Language Modeling* com NNs

- Para aproximar a distribuição do próximo *token* precisamos aplicar uma transformação ao contexto
  - Um classificador
  - MLP + *Softmax* funciona bem



# *Language Modeling* com NNs

- Para aproximar a distribuição do próximo *token* precisamos aplicar uma transformação ao contexto
  - Um classificador
  - MLP + *Softmax* funciona bem



# *Language Modeling* com NNs

- Como temos um classificador basta treinar a NN como um problema de classificação
- *Loss Cross Entropy*

$$Loss(y, \hat{y}) = -\sum_{i=1}^{|V|} y_i \ln\left(p(\hat{y}_i)\right)$$

Se quisermos descrever em termos da sequência toda, temos:

$$Loss = -\ln\left(p(y_t|y_{<t})\right)$$

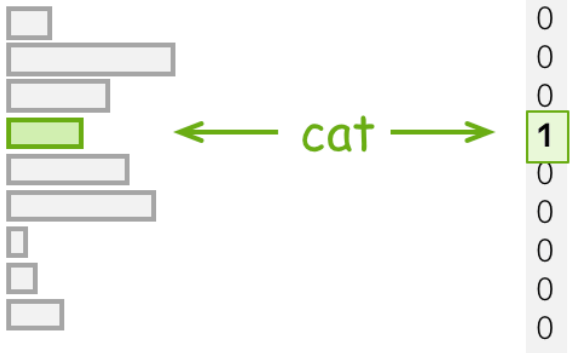
# *Language Modeling* com NNs

$$Loss = -\ln\left(p(y_t|y_{<t})\right)$$

we want the model
to predict this

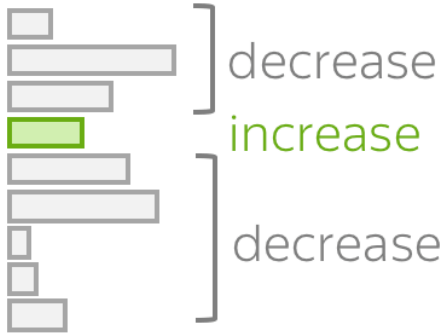
Training example: **I saw a cat** on a mat <eos>

Model prediction: p( ∗ | **I saw a**)




Target

Loss = -log (p(**cat**)) → min



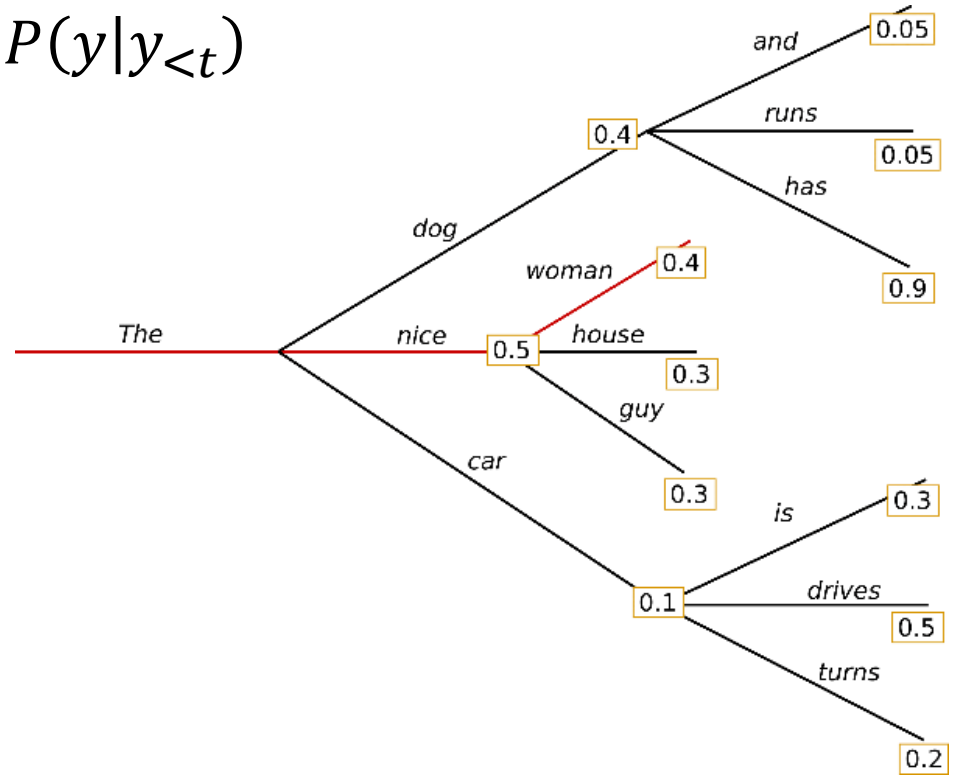

https://lena-voita.github.io/resources/lectures/lang_models/neural/rnn_lm_training_with_target.mp4

# *Language Modeling* com NNs

- Podemos gerar frases a partir do *Language Model* treinado
- Também chamado de ***decoding***
- Queremos tanto coerência quanto diversidade nas frases geradas
  - Existe um *tradeoff* entre essas características dependendo da forma de gerar os textos

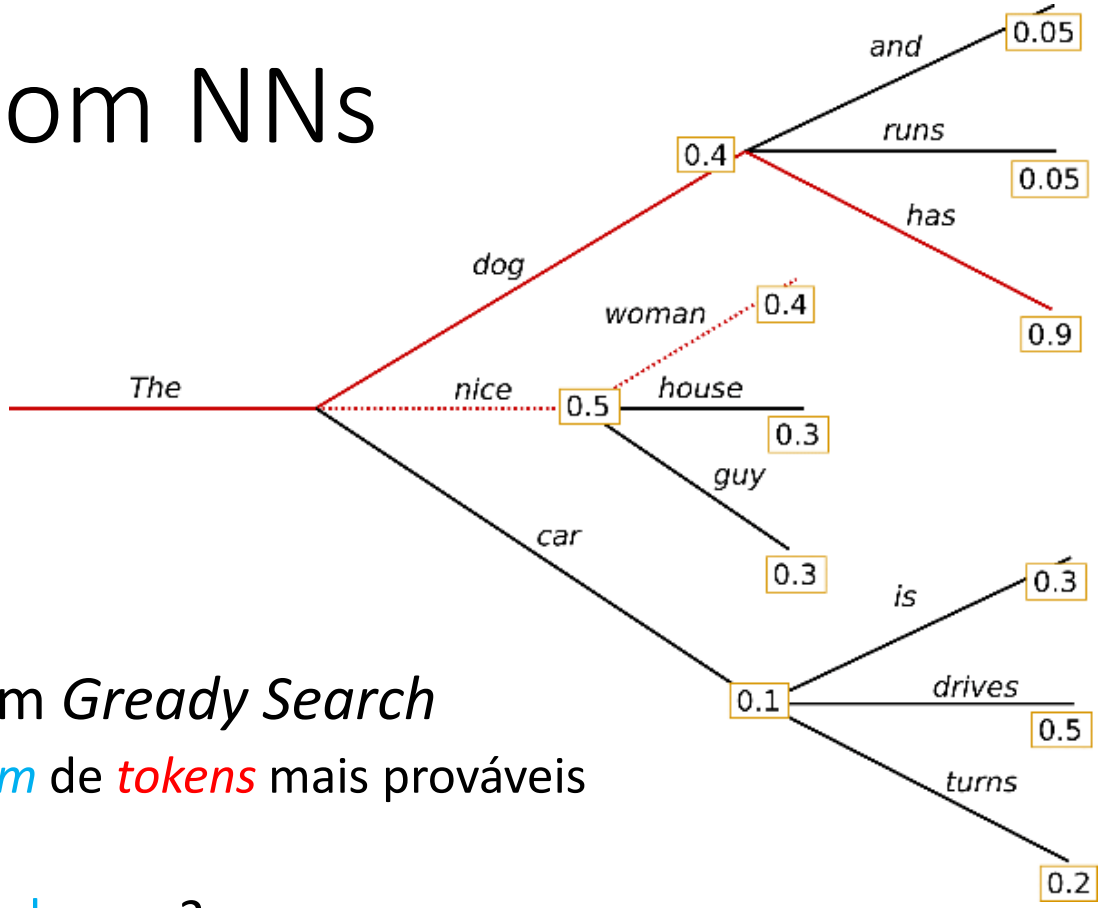
# *Language Modeling* com NNs

- Gerando frases com *Greedy Search*
- É o método mais simples
  - Consiste em considerar sempre o *token* mais provável
  - $y_t = \underset{y}{\text{argmax}}\, P(y|y_{<t})$



# *Language Modeling* com NNs

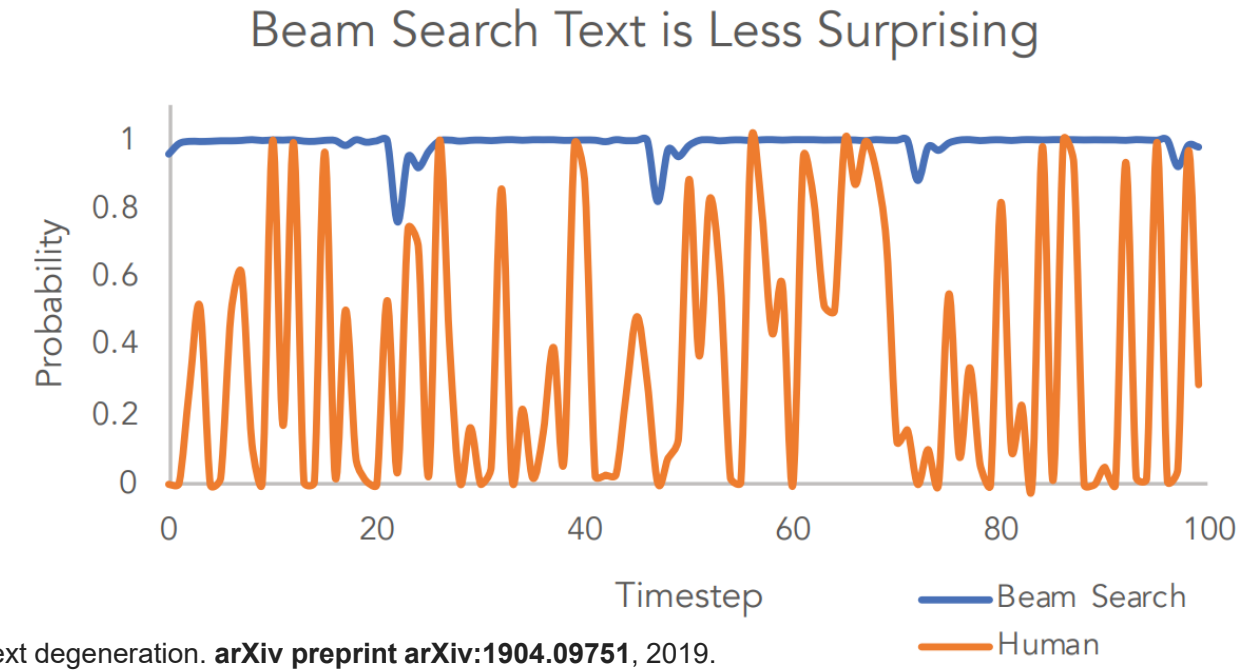
- Gerando frases com *Beam Search*




- Resultados muito melhores do que em *Gready Search*
  - Consiste em considerar sempre um *beam* de *tokens* mais prováveis
  - O tamanho do *beam* é configurável
  - No exemplo abaixo temos o tamanho do beam=2
  - Embora "*The dog*" seja menos provável que *"The nice",* como o tamanho do feixe é igual a dois ambas frases são expandidas
    - *Por consequência "The dog has" é identificado como mais provável do que "the nice"*

# *Language Modeling* com NNs

- *Beam Search* funciona bem para tarefas nas quais o tamanho da sequência é relativamente conhecido
- Porém, gera saídas repetitivas e textos que não parecem gerados por humanos
  - Repare na visualização abaixo



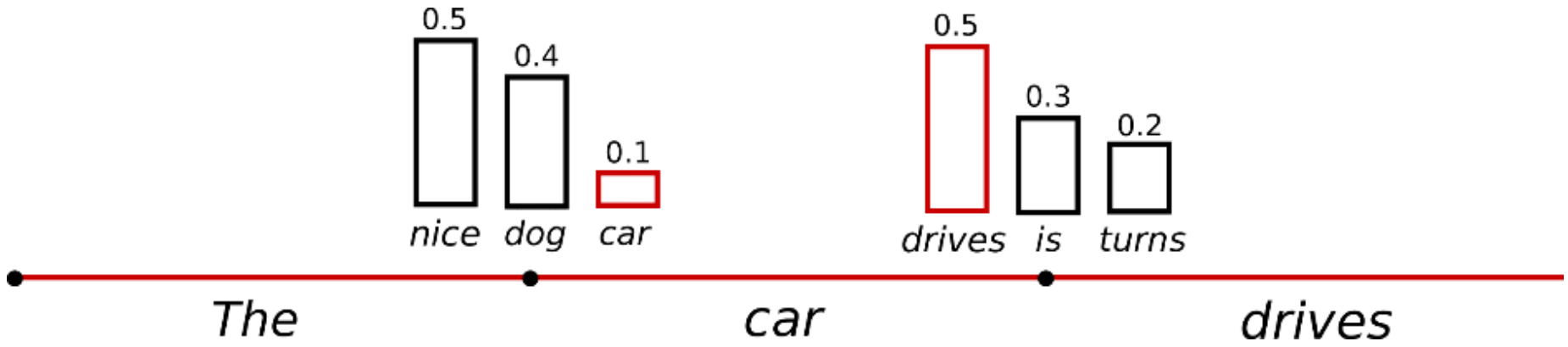



HOLTZMAN, Ari et al. The curious case of neural text degeneration. **arXiv preprint arXiv:1904.09751**, 2019.

# *Language Modeling* com NNs

- Podemos amostrar da distribuição de probabilidade até encontrar o token especial <eos>
  - Usando esse tipo de método a geração de frases deixa de ser determinística

$$y_t \sim P(y|y_{<t})$$



# *Language Modeling* com NNs

- Para aumentar a diversidade das frases geradas podemos alterar a *temperatura* da *softmax*

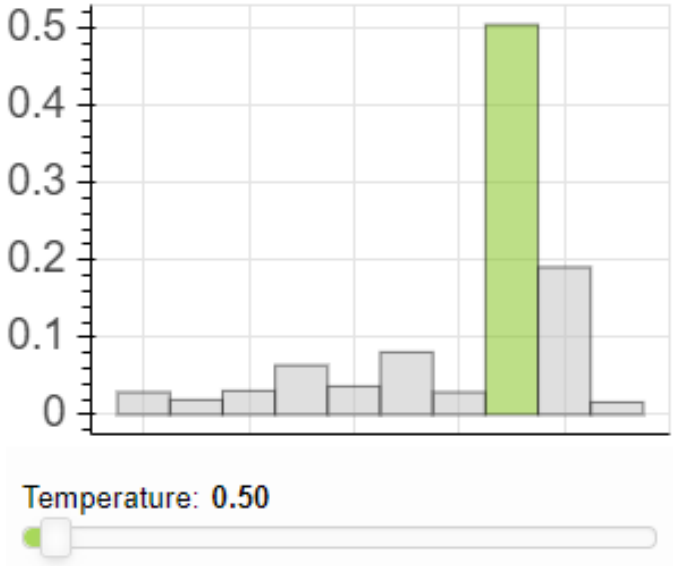
$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w}{\tau}\right)}$$

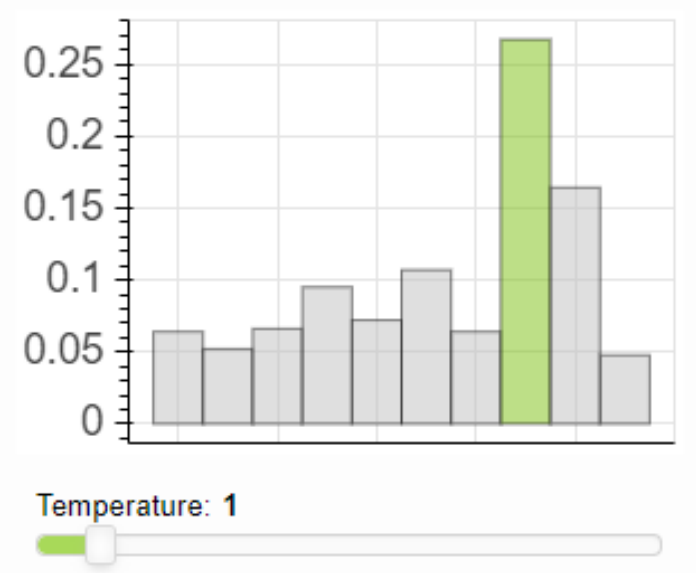
Qual o efeito da temperatura na *softmax*?

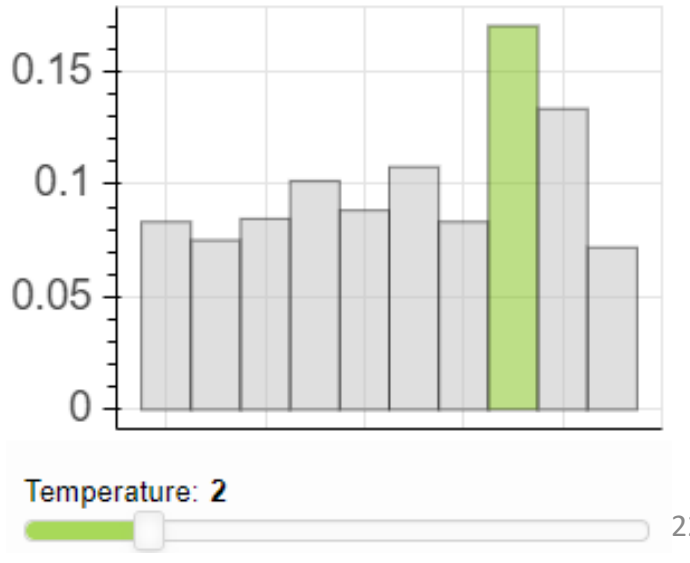
# *Language Modeling* com NNs

- Para aumentar a diversidade das frases geradas podemos alterar a *temperatura* da *softmax*

$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w}{\tau}\right)}$$

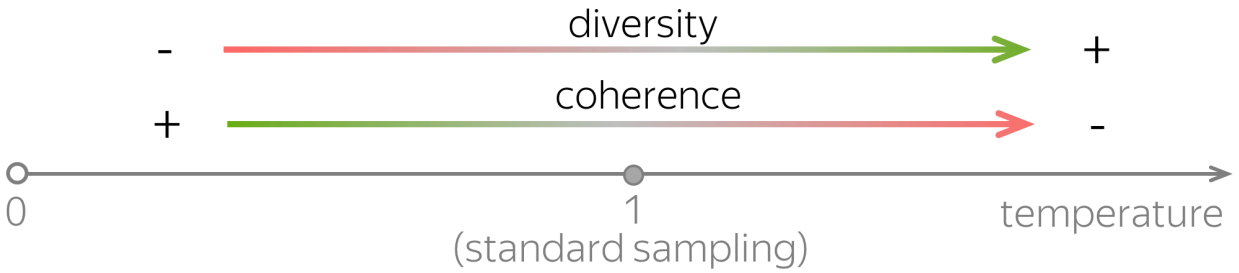


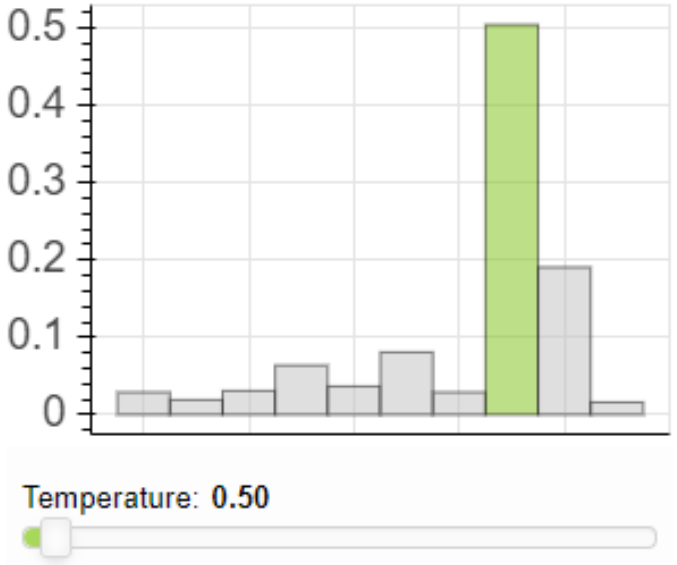



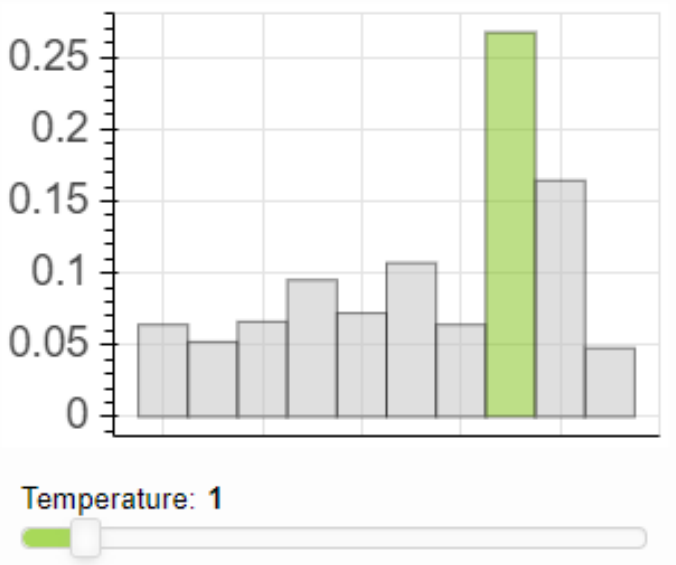



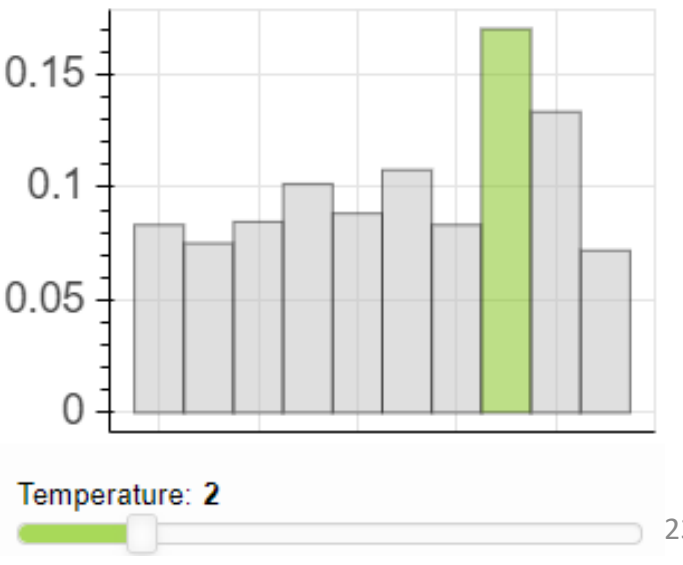
# *Language Modeling* com NNs

- $\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w}{\tau}\right)}$



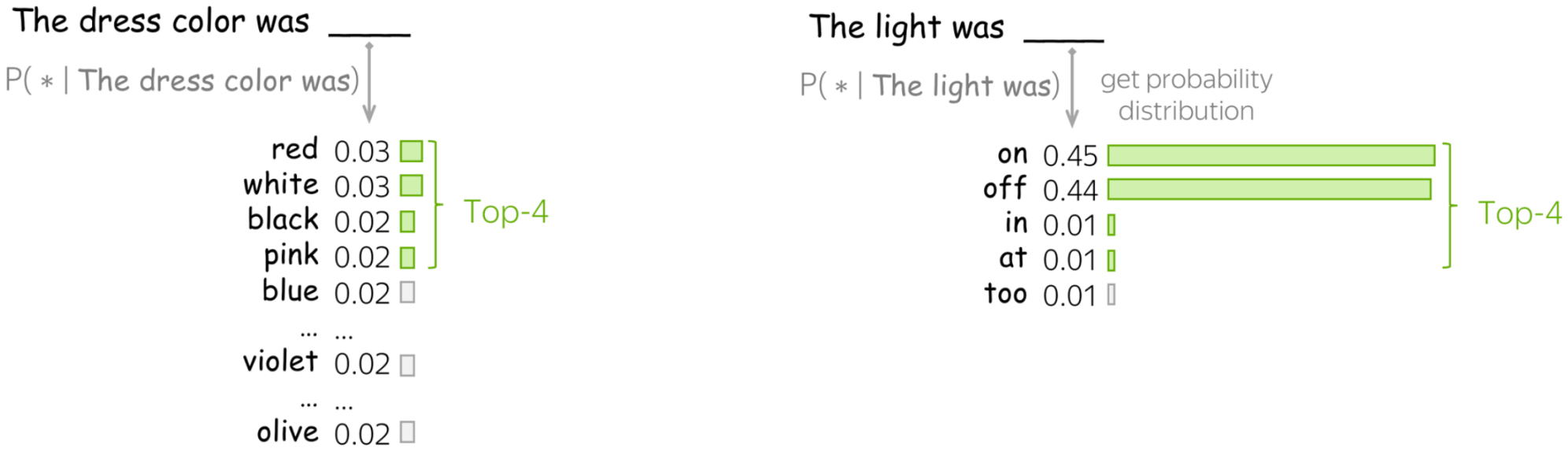



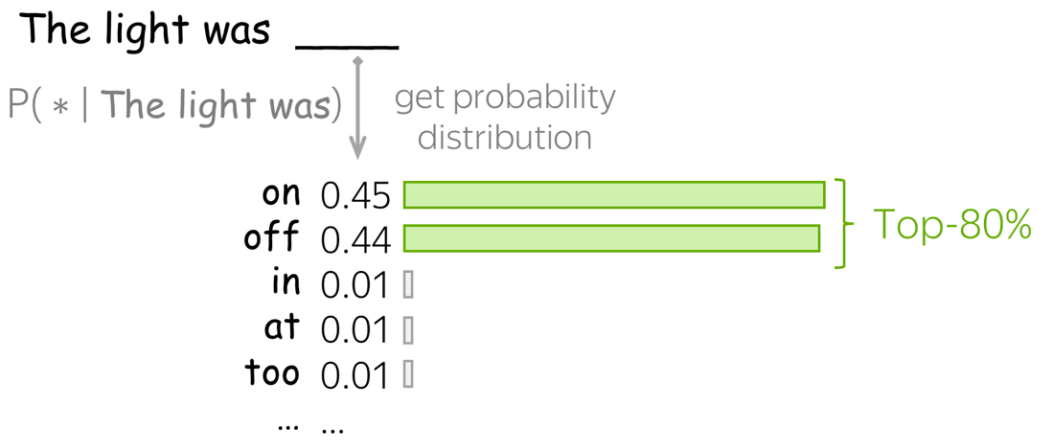
# *Language Modeling* com NNs

- *Top-K Sampling*
  - É uma heurística aplicada ao processo de amostragem descrito anteriormente
  - Sempre amostra apenas dos K *tokens* mais prováveis
  - Evita que *tokens* pouco prováveis sejam incluídos na amostra



# *Language Modeling* com NNs

- *Top-P Sampling (Nucleus)*
  - Como fixar um número k é difícil, definimos uma massa de probabilidade p
  - Sempre amostra apenas os *tokens* mais prováveis cuja probabilidade acumulada chega a p

# *Language Modeling* com NNs

- Qual melhor método de *decoding* para LM?
  - Não há consenso
  - Tanto *Top-P* quanto *Beam Search* são utilizados

# *Language Modeling* - Avaliação

- Para avaliação o mais utilizado é a perplexidade que é obtida usando a log verossimilhança
- O modelo deve atribuir alta probabilidade a textos reais não vistos e baixa probabilidade para textos que não fazem sentido
- A função de custo entropia cruzada
  - $Loss(y_{1:m}) = -\sum_{t=1}^{m} ln\big(p(y_t|y_{<t})\big)$
- A log verossimilhança
  - $L(y_{1:m}) = \sum_{t=1}^{m} \log_2\big(p(y_t|y_{<t})\big)$

# *Language Modeling* - Avaliação

- Para avaliação o mais utilizado é a perplexidade que é obtida usando a log verossimilhança
- A log verossimilhança
  - $L(y_{1:m}) = \sum_{t=1}^{m} \log_2(p(y_t|y_{<t}))$
- $\text{Perplexidade}(y_{1:m}) = 2^{-\frac{1}{m}L(y_{1:m})}$
- Bons modelos tem log verossimilhança alta e perplexidade baixa
  - $Perplexidade \in [1, |V|]$

# *seq2seq*

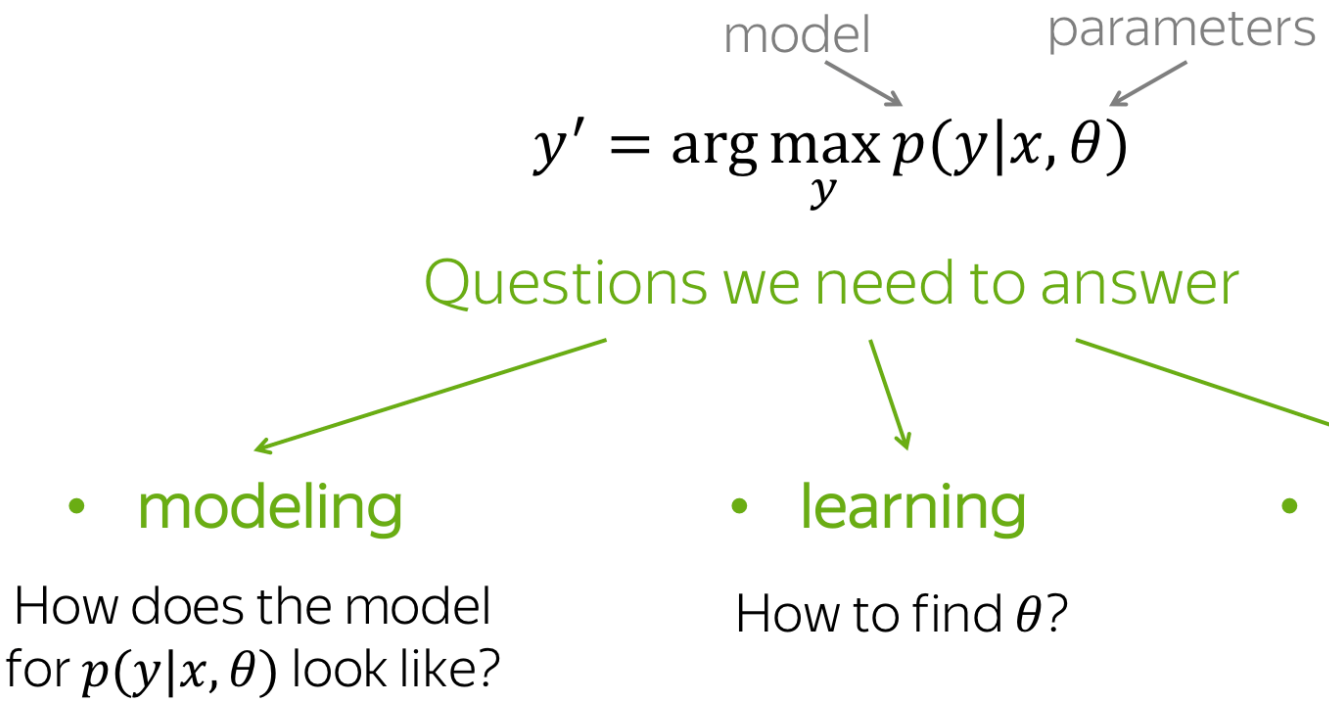
# *Language Models Condicionados*

$$P(y_1, y_2, \ldots, y_n | x) = \prod_{t=1}^{n} P(y_t | y_{<t}, x)$$

- Semelhante a tarefa de modelagem de linguagem, agora adicionamos também um condicionante $x$
- Essa formalização serve para diversas tarefas seq2seq bem como para *image captioning* e outras

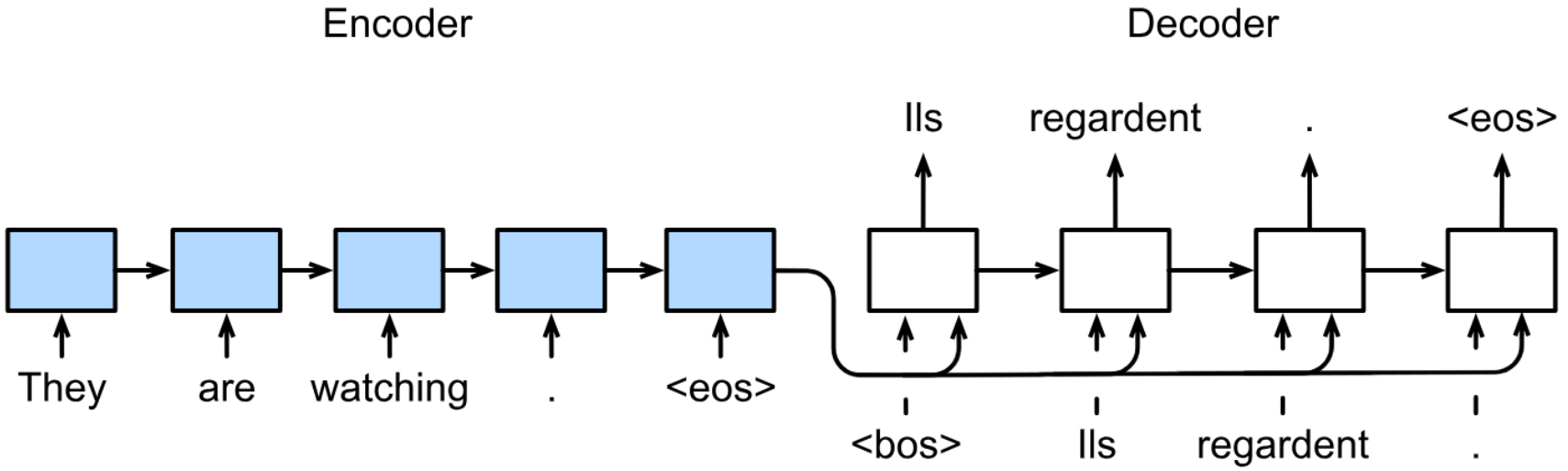
# seq2seq

- Dentre as tarefas que envolvem sequências, *machine translation* é uma das mais importantes
  - Dada uma sequência em um idioma queremos descobrir uma sequência em outro idioma



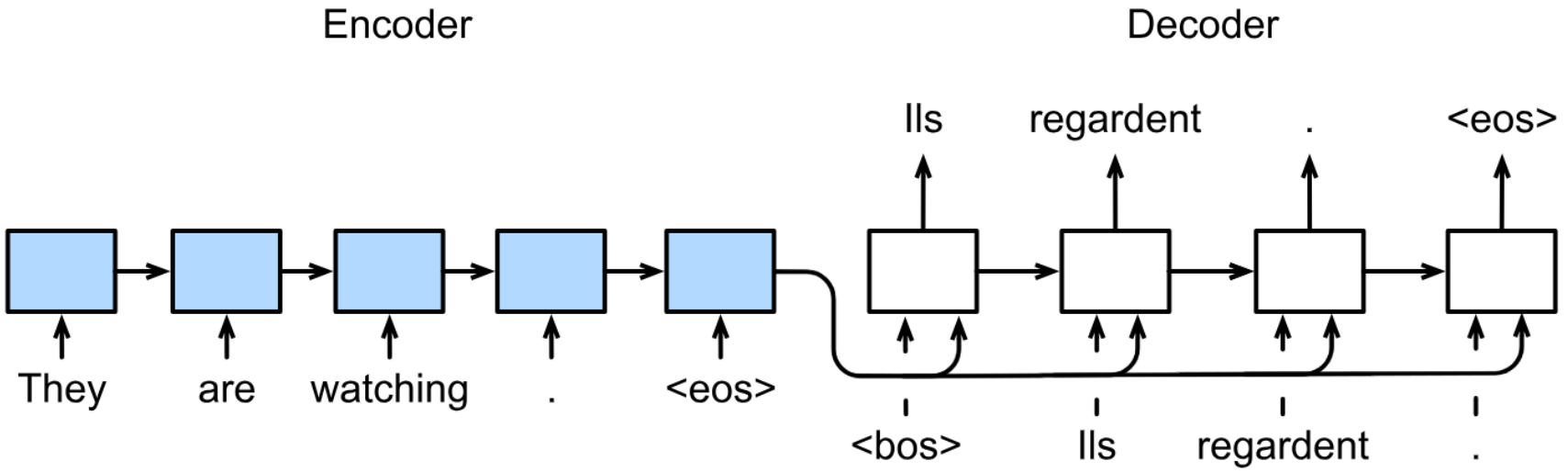
# Arquitetura *Encoder-Decoder*

- *Encoder-Decoder* são muito utilizados em redes recorrentes
  - O *Encoder* transforma a entrada de tamanho variável em um estado oculto
  - O *Decoder* transforma o estado oculto em uma sequência
  - Os *tokens* especiais <bos> e <eos> são usados para indicar o começo e o final da sentença



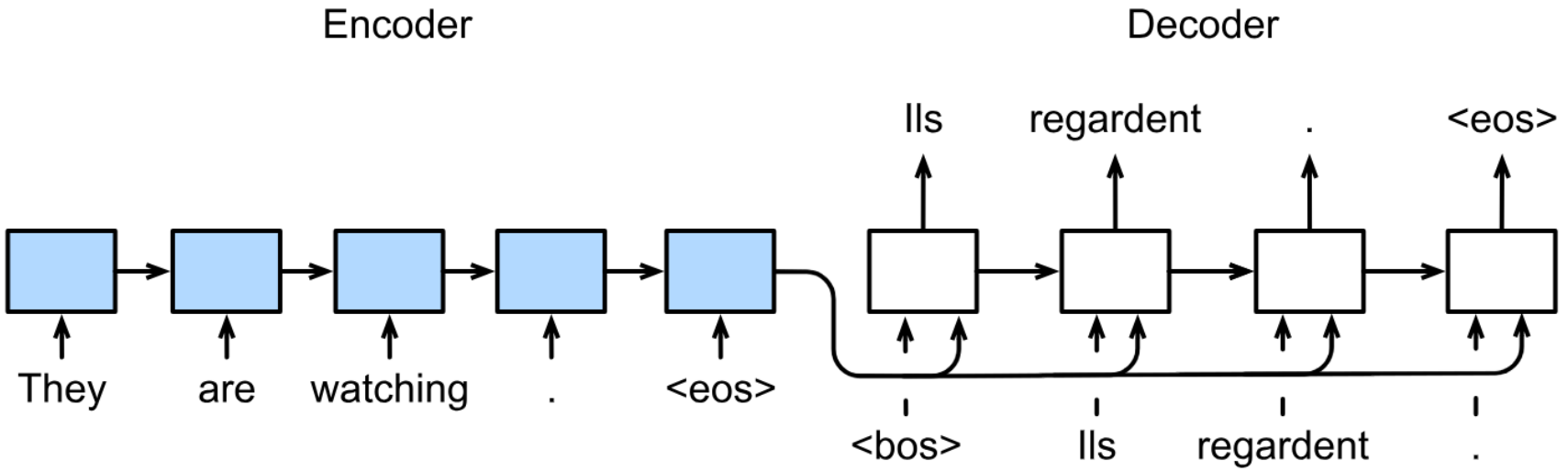
# Arquitetura *Encoder-Decoder*

- Repare que, entrar com os dados de treinamento no *encoder* é trivial
- Contudo, existem diversas abordagens para passar os dados para o *decoder*
- Uma das abordagens para o *decoder* consiste em:
  - Em treino: condicionar no estado oculto do *encoder* e nos tokens precedentes do *ground truth*
  - Em teste: condicionar no estado oculto do *encoder* e nos *tokens* que já foram preditos



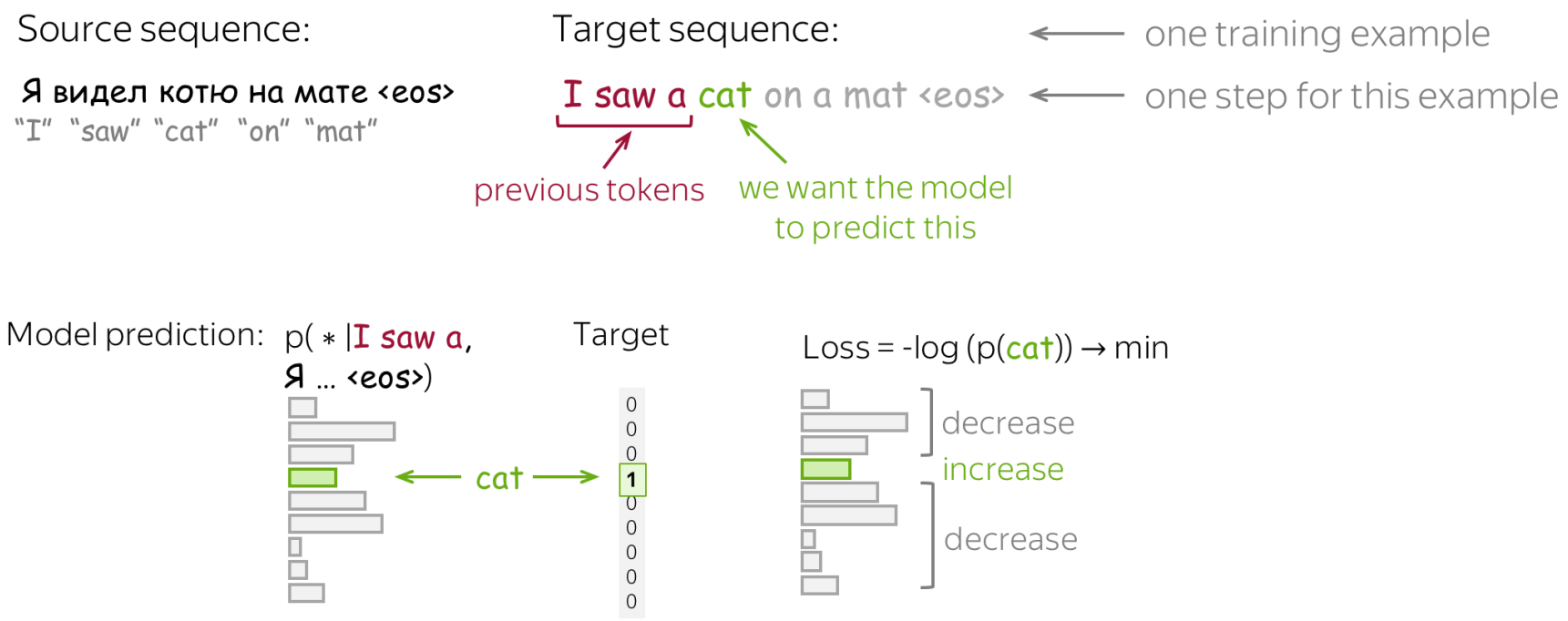
# Arquitetura *Encoder-Decoder*

- Sobre o *Decoder*
  - Repare que o *decoder* funciona como um *Language Model*, com a diferença que ele está condicionado também no estado oculto do *encoder*



# *Machine Translation*

- O processo de treinamento e amostragem seguem os mesmos
  - *Cross-Entropy Loss*
  - *Beam Search*

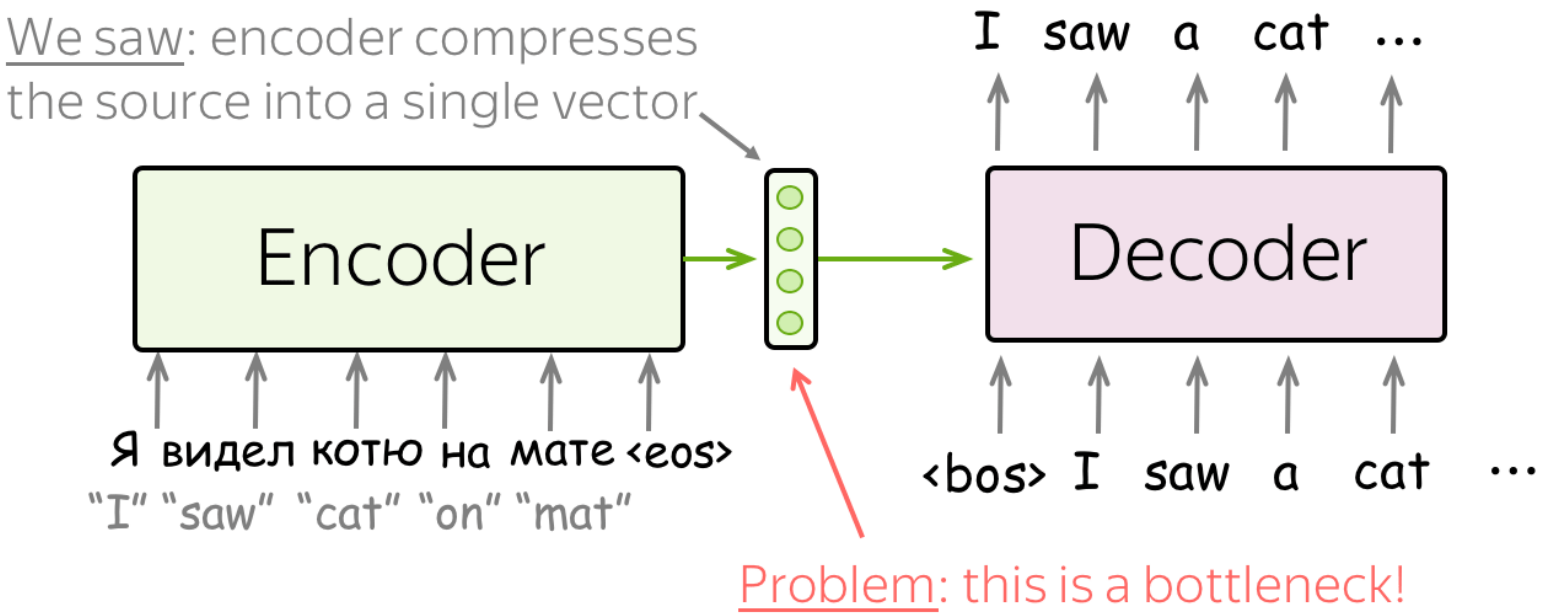





SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, v. 27, 2014.

# *Machine Translation*

- Atenção aos problemas
  - É difícil para o *encoder* capturar todo o contexto da frase original
  - É difícil para o *decoder* gerar todos os tokens condicionado em um contexto fixo do *encoder*



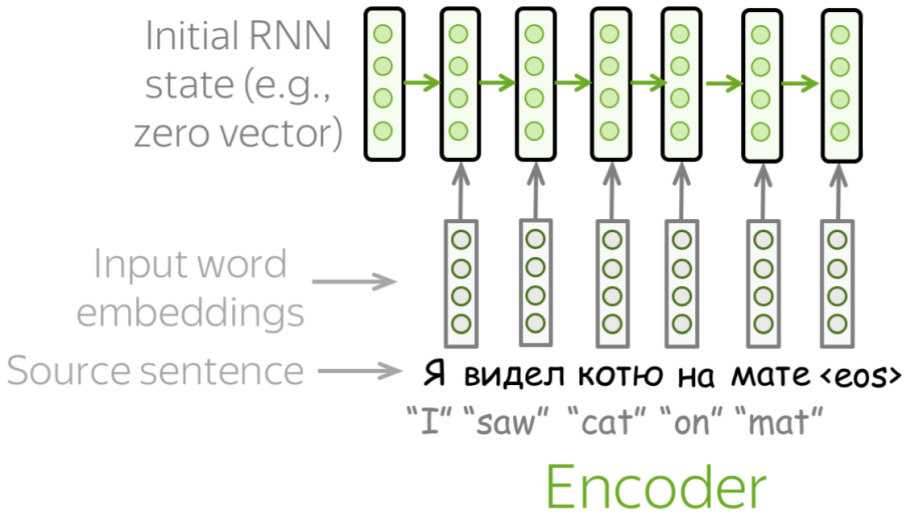
# *Attention*

- Para resolver o problema de gargalo entre o encoder e o decoder foi introduzido o mecanismo de atenção
  - A atenção é um bloco da NN que conecta o encoder e o decoder e que atribui scores a cada etapa do encoder
  - Interpretamos a atenção como uma decisão de quais partes da sequência de entrada são mais importantes
- Existem várias implementações de diferentes de atenção



BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

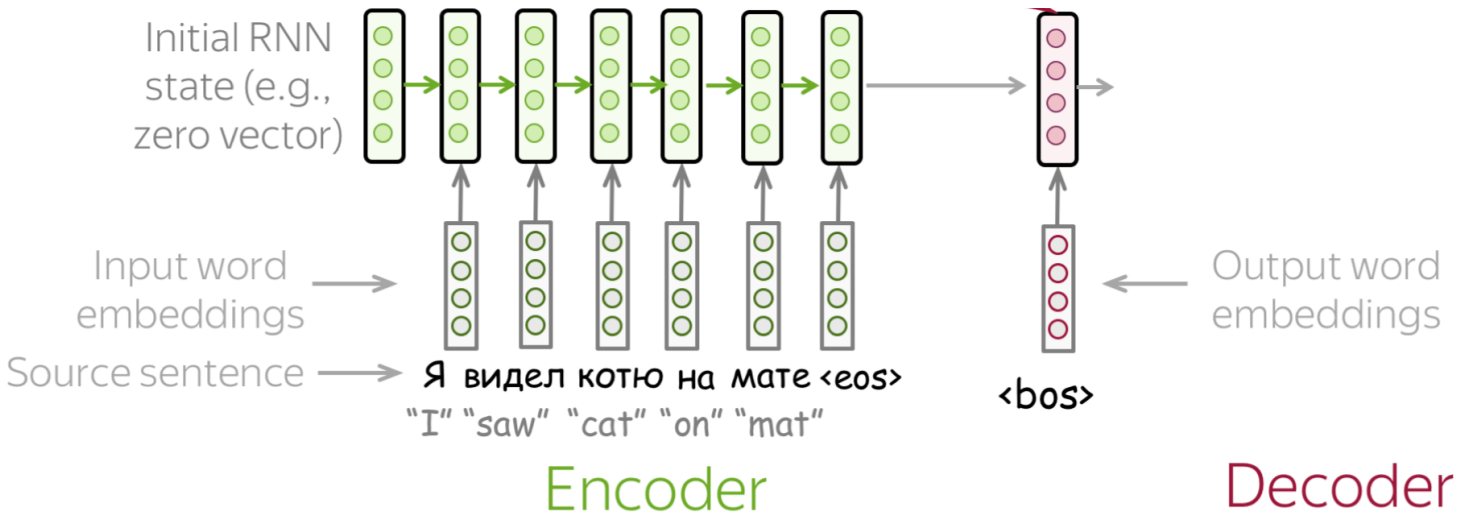
# *Attention*






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

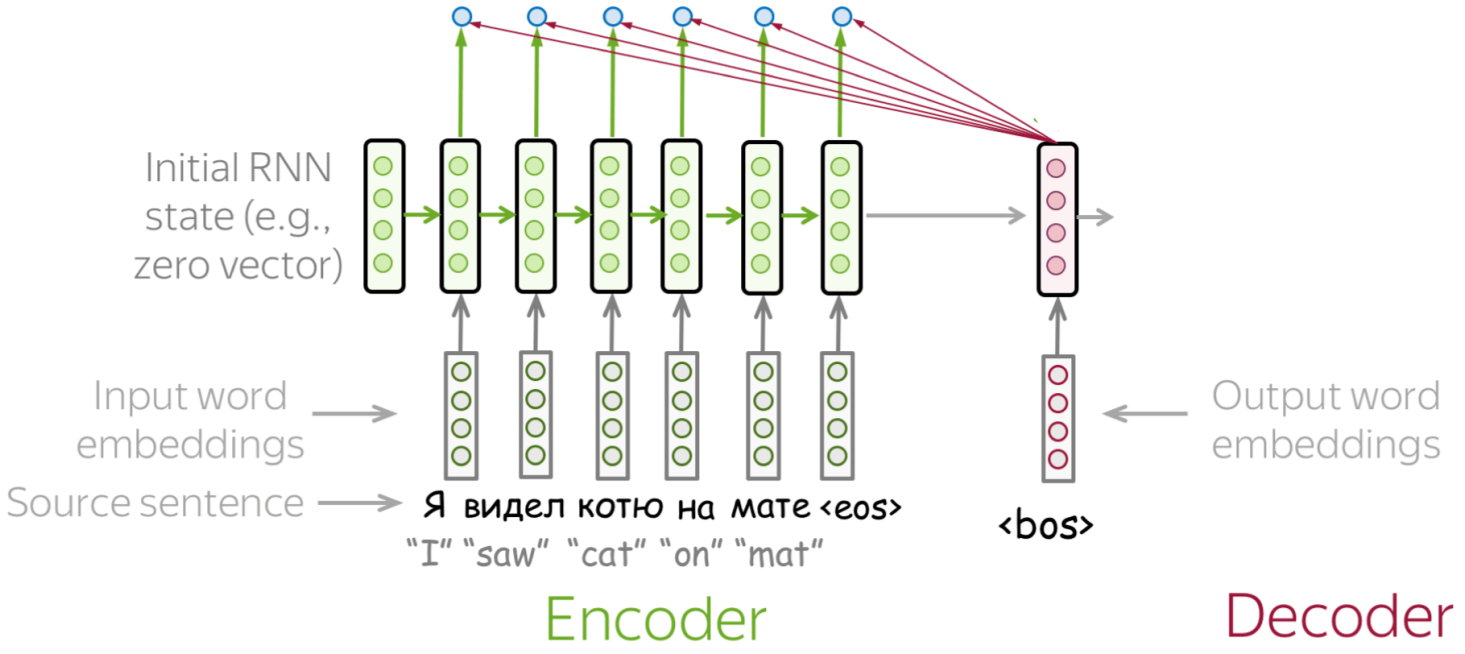
# *Attention*






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

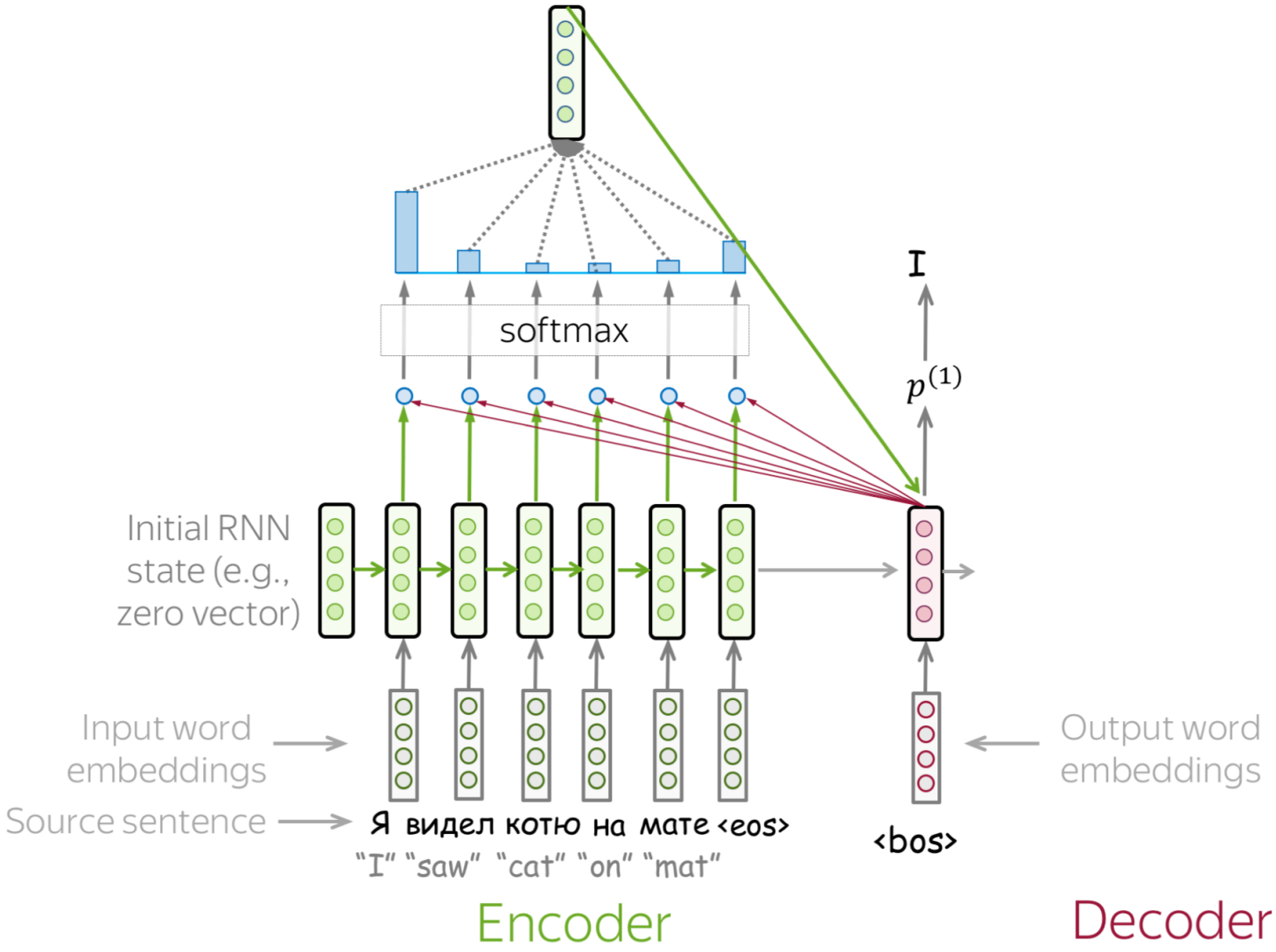
# *Attention*






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

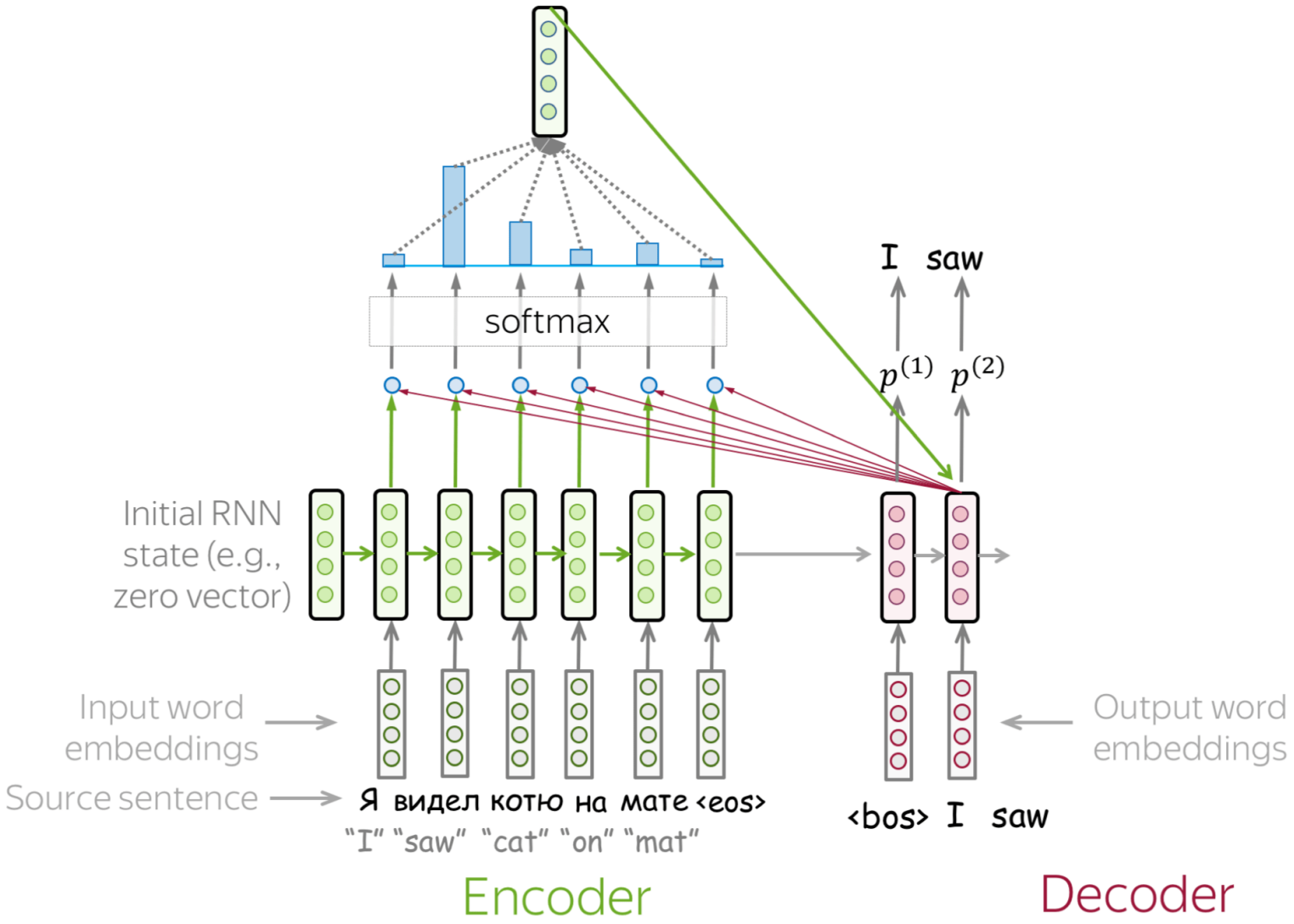
# *Attention*






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

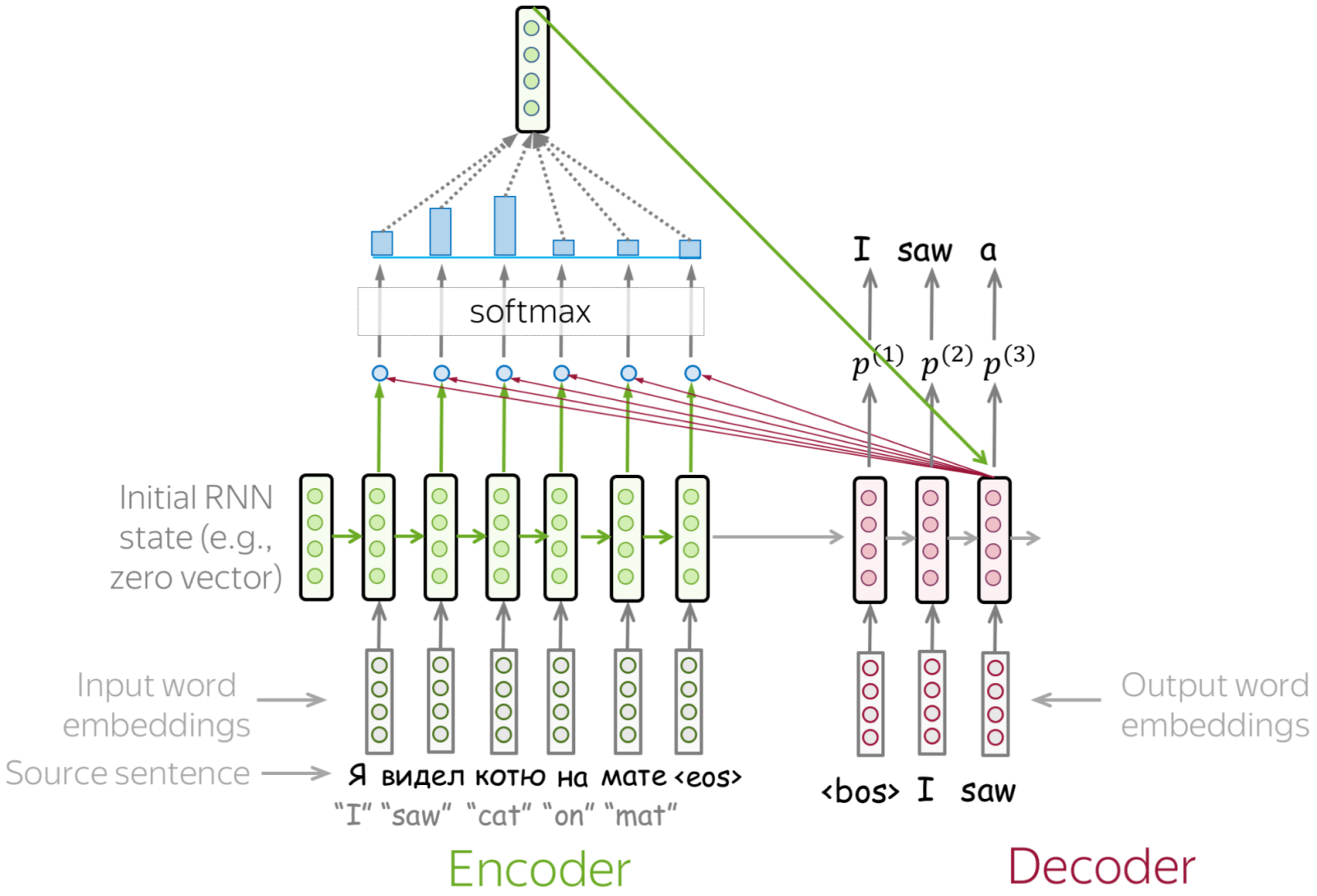
# Attention






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

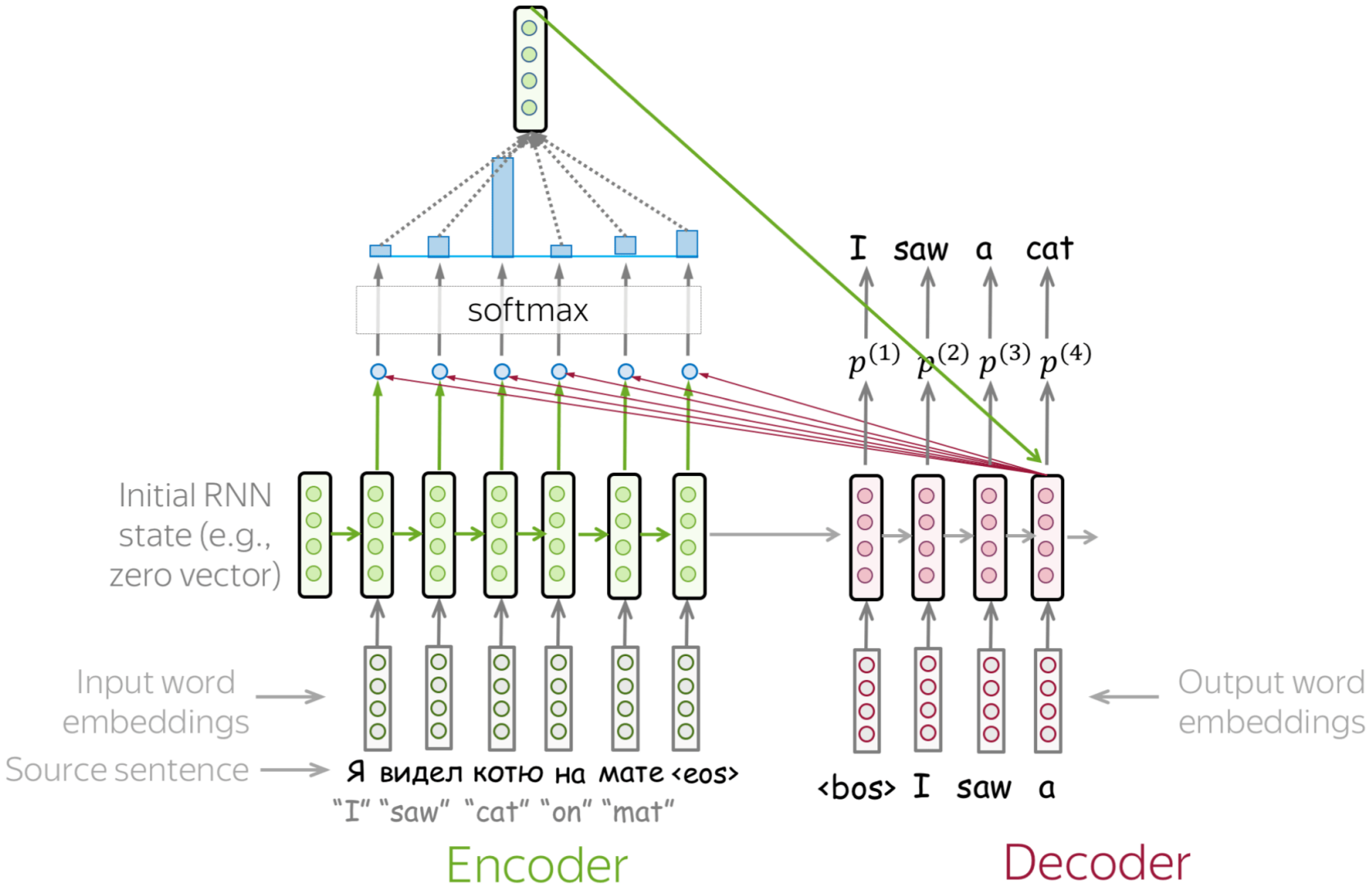
# *Attention*






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

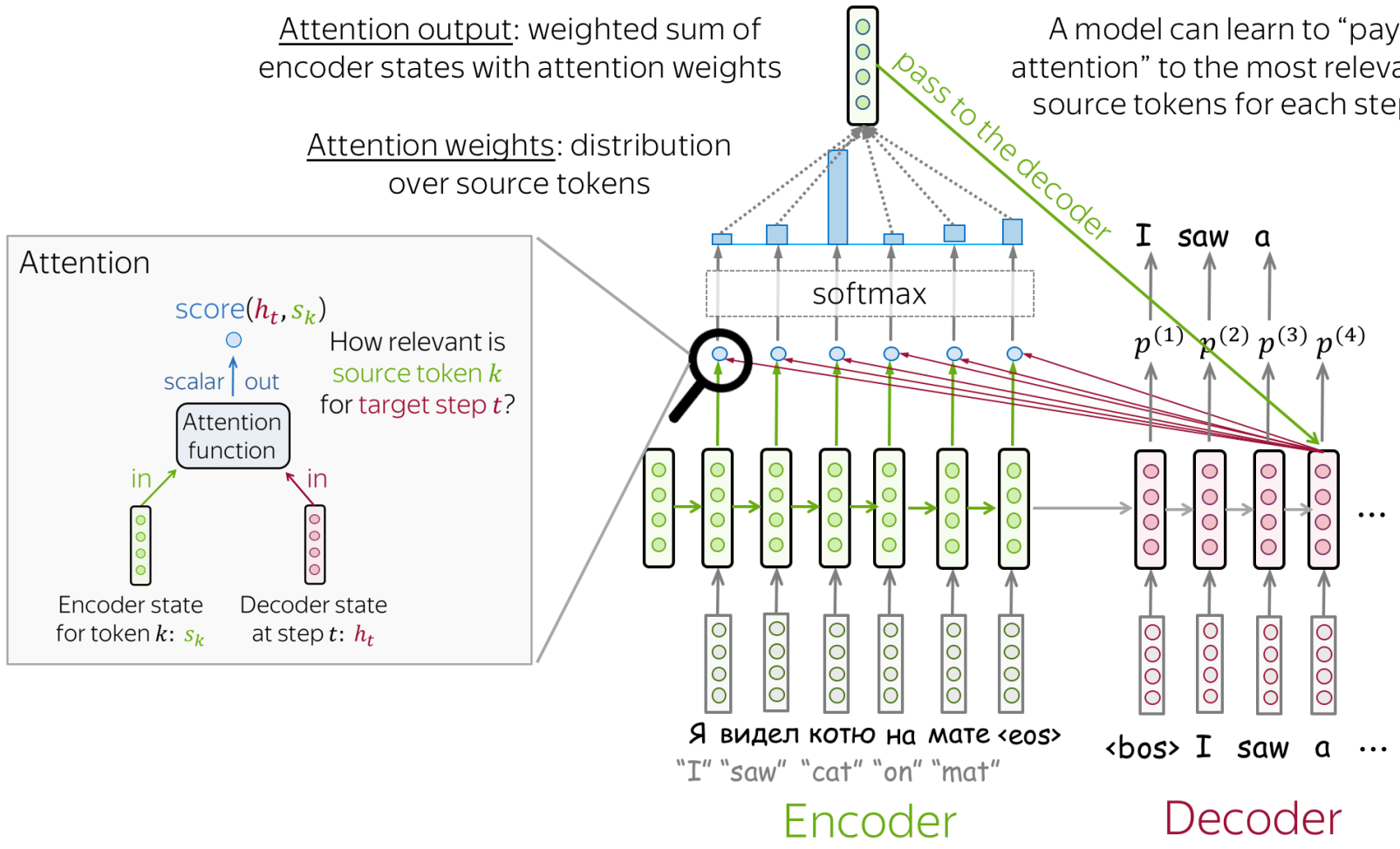
# Attention






BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

# Attention



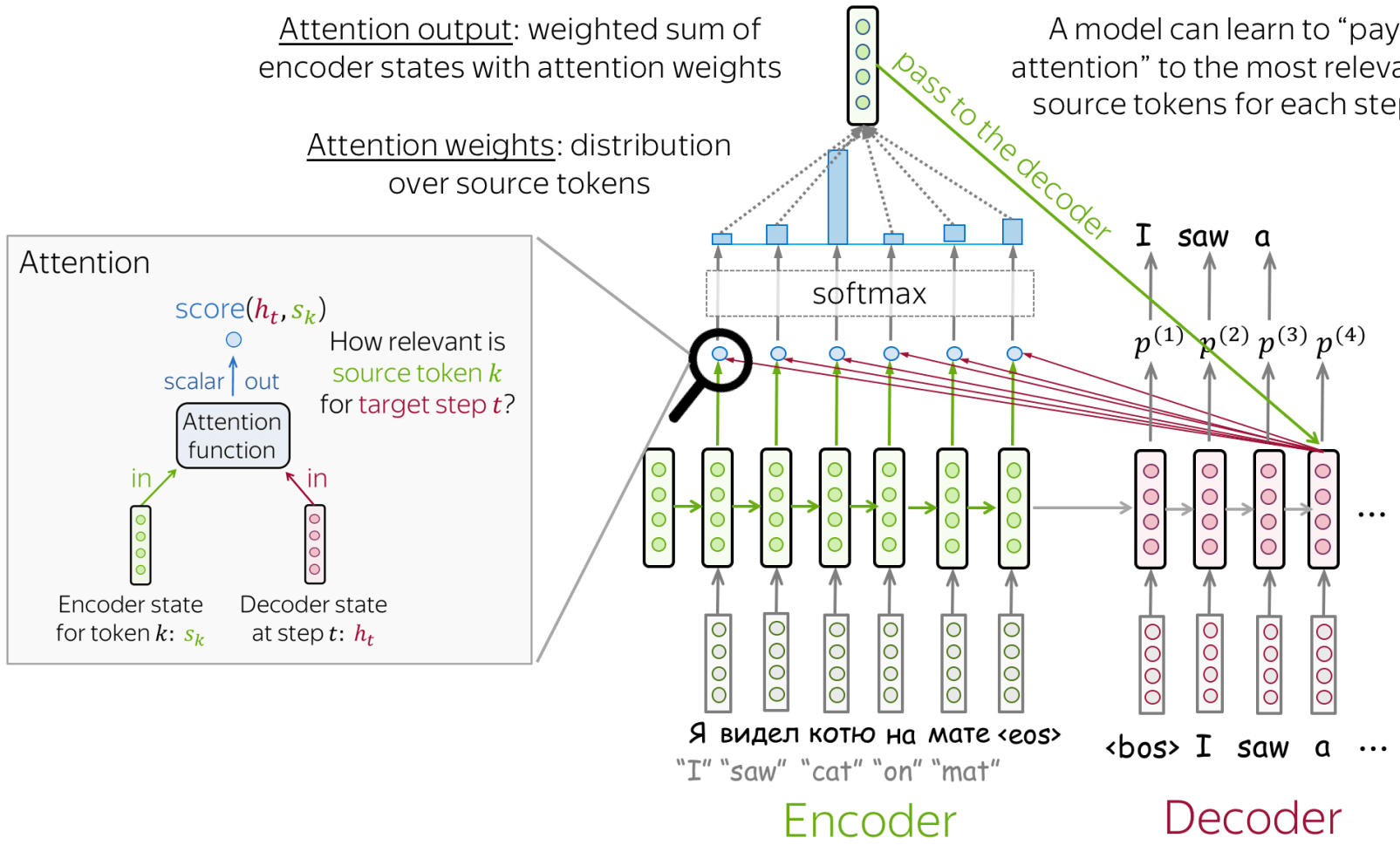

A cada passo do decoder a atenção:

- Recebe como entrada o estado oculto do decoder e todos os estados do encoder $s_k$
- Computa um score de atenção $score(h_t, s_k)$
- Computa uma *softmax* dos scores
- Computa a saída da atenção como uma soma ponderada dos estados do encoder pela softmax dos scores
- A saída da atenção é usada em conjunto do estado atual do decoder para gerar a distribuição de palavras
  - Pode também ser passada como entrada para o próximo passo



BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

# Attention




Attention output: $c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$
↑ (weighted sum)
"source context for decoder step $t$"

Attention weights: $a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$
↑ (softmax)
"attention weight for source token $k$ at decoder step $t$"

Attention scores: $\text{score}(h_t, s_k), \text{k} = 1..\text{m}$
↑
"How relevant is source token $k$ for target step $t$?"

Attention input: $s_1, s_2, \ldots, s_m$ (all encoder states); $h_t$ (one decoder state)

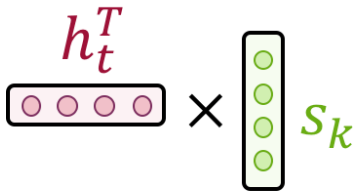


BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

# Attention

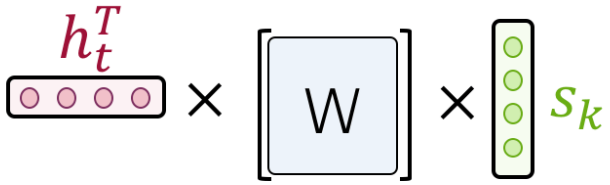
- Como calcular os *scores*

Dot-product




$$\text{score}(h_t, s_k) = h_t^T \; s_k$$

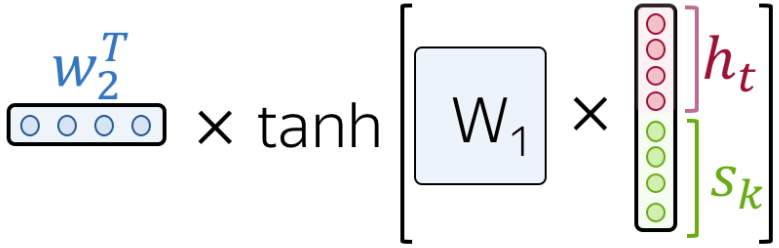
Bilinear




$$\text{score}(h_t, s_k) = h_t^T \; W s_k$$

*Luong Attention*

Multi-Layer Perceptron




$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

*Bahdanau Attention*


LUONG, Minh-Thang; PHAM, Hieu; MANNING, Christopher D. Effective approaches to attention-based neural machine translation. **arXiv preprint arXiv:1508.04025**, 2015.

BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

# Referências:

- Material baseado em:
  - Capítulo 10 - ZHANG, Aston et al. Dive into deep learning. Cambridge University Press, 2023.
  - Excelente blog post de Lena Voita: https://lena-voita.github.io/nlp_course/language_modeling.html
  - https://huggingface.co/blog/how-to-generate