Neural Attention Mechanisms

Ben Peters

Acknowledgments

Vlad Niculae made most of these slides.

(Bahdanau et al., 2015)

Sequence-to-Sequence With Attention

United Nations elections

```
words = [21, 79, 14]   # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)   # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)   # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]   # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)   # attn proba
  # p = [.10, .05, .85]

  a = p @ H   # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

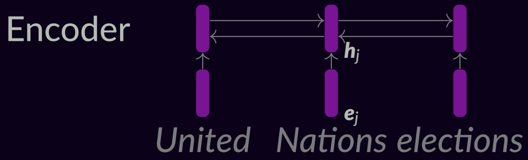
Sequence-to-Sequence With Attention

Encoder


```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

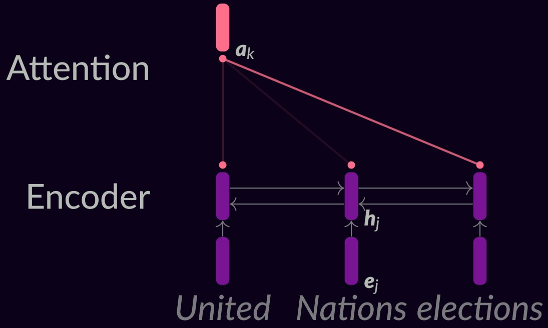
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

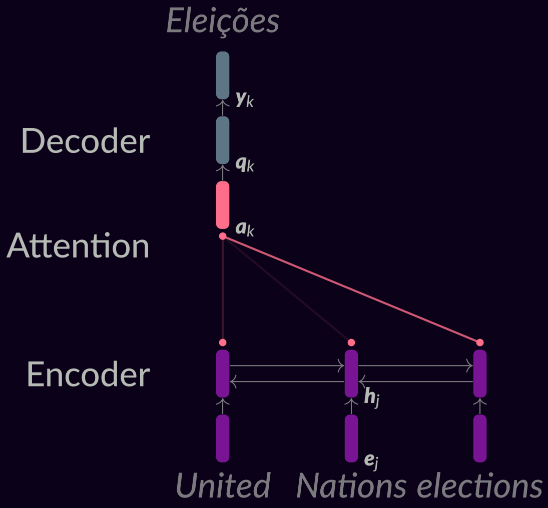
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

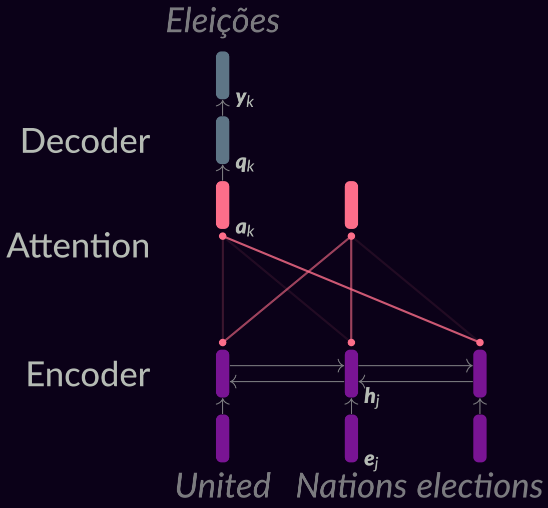
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]   # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)   # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)   # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]   # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)   # attn proba
  # p = [.10, .05, .85]

  a = p @ H   # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

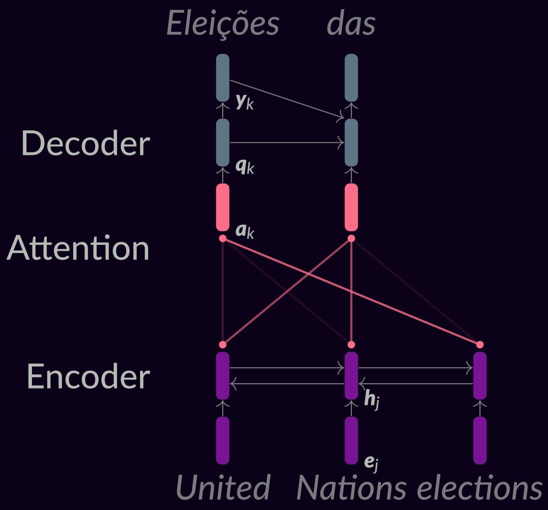
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

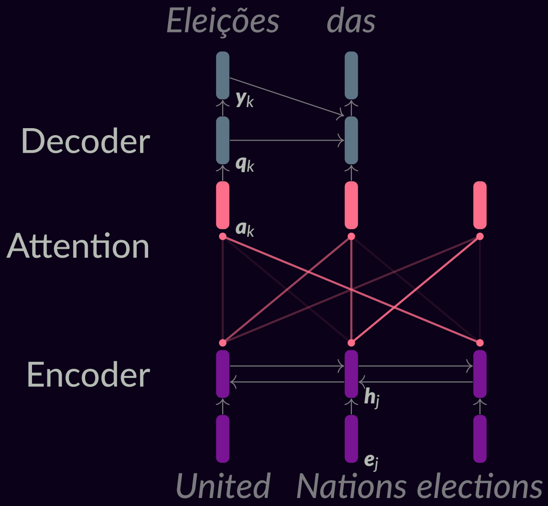
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]   # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)   # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)   # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]   # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)   # attn proba
  # p = [.10, .05, .85]

  a = p @ H   # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

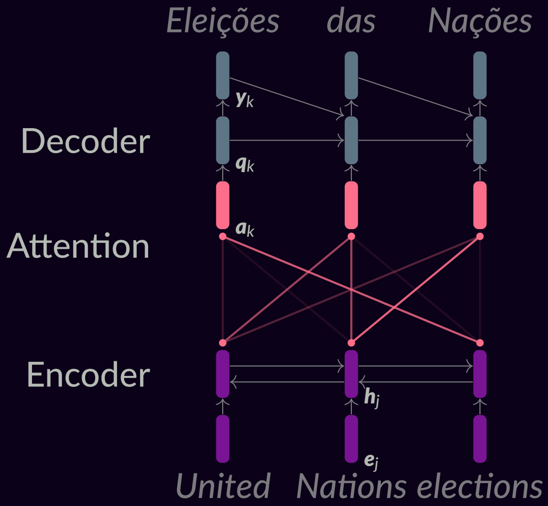
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]   # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)   # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)   # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]   # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)   # attn proba
  # p = [.10, .05, .85]

  a = p @ H   # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

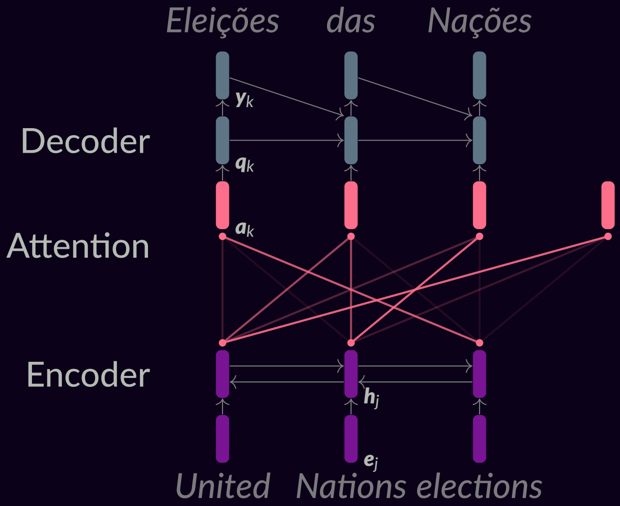
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]   # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)   # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)   # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]   # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)   # attn proba
  # p = [.10, .05, .85]

  a = p @ H   # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

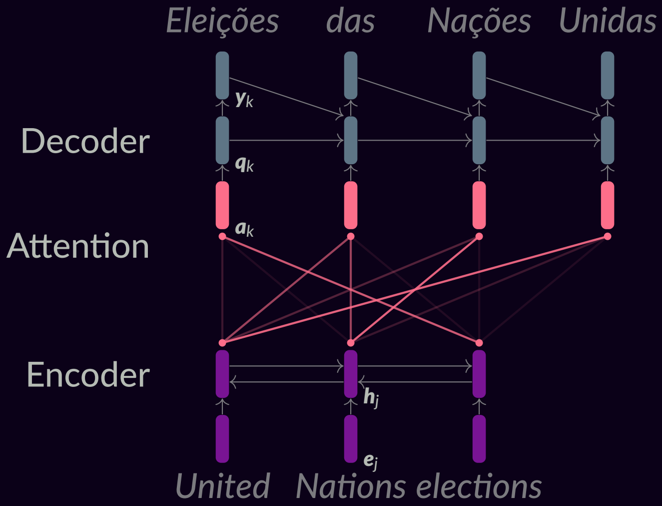
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

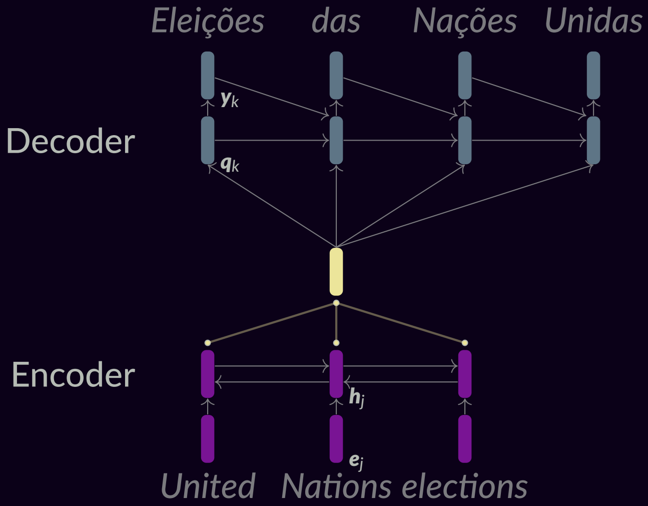
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

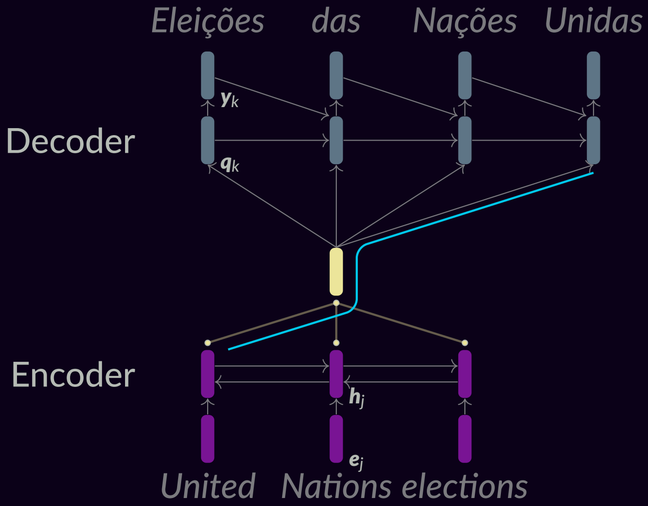
Sequence-to-Sequence With Attention




```
words = [21, 79, 14]  # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)  # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)  # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]  # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)  # attn proba
  # p = [.10, .05, .85]

  a = p @ H  # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

(Bahdanau et al., 2015)

Sequence-to-Sequence With Attention




```
words = [21, 79, 14]   # indices

embed = Embedding(vocab_sz, dim)
E = embed(words)   # (3 × dim)
enc = LSTM(dim, dim)
H = enc(E)   # (3 × dim)

dec = LSTM(2 * dim, dim)
initialize k=1, q[0], y[0]

while not done:
  s = H @ q[k - 1]   # attn scores
  # s = [-.3, -1.0, 1.8]

  p = softmax(s)   # attn proba
  # p = [.10, .05, .85]

  a = p @ H   # (1 × dim)

  q[k] = dec(a, y[k - 1], q[k - 1])
  y[k] = W_out @ q[k] + b
  k += 1
```

In a Nutshell

In a Nutshell

- You have one query vector (think tgt word)

In a Nutshell

- You have one query vector (think tgt word)
- You have a lot of key vectors (one for each src word)

In a Nutshell

- You have one query vector (think tgt word)
- You have a lot of key vectors (one for each src word)
- First: come up with a *score* between the query and each key

In a Nutshell

- You have one query vector (think tgt word)
- You have a lot of key vectors (one for each src word)
- First: come up with a *score* between the query and each key
- Use the scores to decide where to attend

Computing the scores

$$s_j = \sigma(\boldsymbol{h}_j, \boldsymbol{q})$$

name	$\sigma(\boldsymbol{h}, \boldsymbol{q})$	
additive	$\boldsymbol{v}^\top \tanh(\boldsymbol{W}_1\boldsymbol{h} + \boldsymbol{W}_2\boldsymbol{q})$	(Bahdanau et al., 2015)
dot-product	$\boldsymbol{h}^\top \boldsymbol{q}$	(Luong et al., 2015)
bilinear	$\boldsymbol{h}^\top \boldsymbol{W}\boldsymbol{q}$	(Luong et al., 2015)
scaled	$(1/\sqrt{d})\ \boldsymbol{h}^\top \boldsymbol{W}\boldsymbol{q}$	(Vaswani et al., 2017)

```
# attention scores:
s = H @ W_attn @ state
# s = [-.3, -1.0, 1.8]

p = softmax(s)
# p = [.10, .05, .85]
```

record scratch

freeze frame

1. How to select an item from a set?

How to select an item from a set?

United

Nations

Elections

How to select an item from a set?

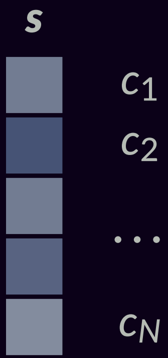
$$c_1$$

$$c_2$$

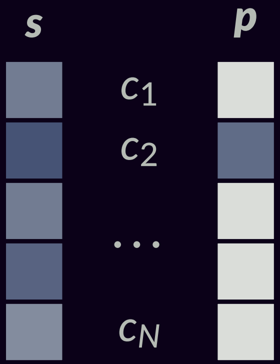
$$\cdots$$

$$c_N$$

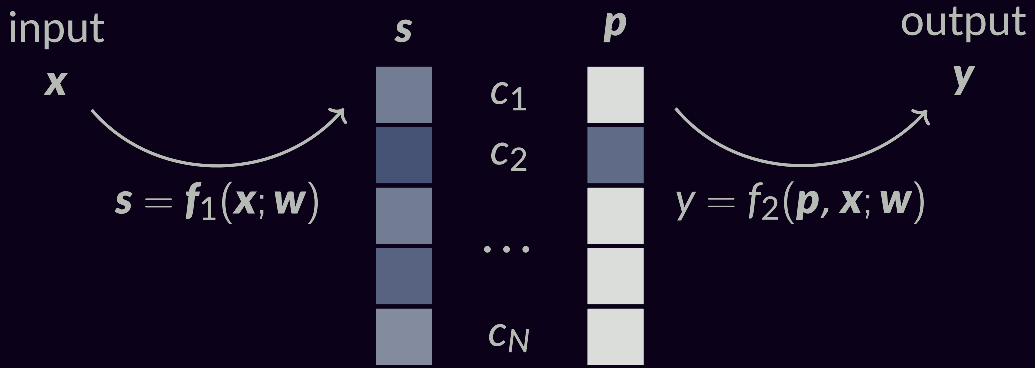
How to select an item from a set?



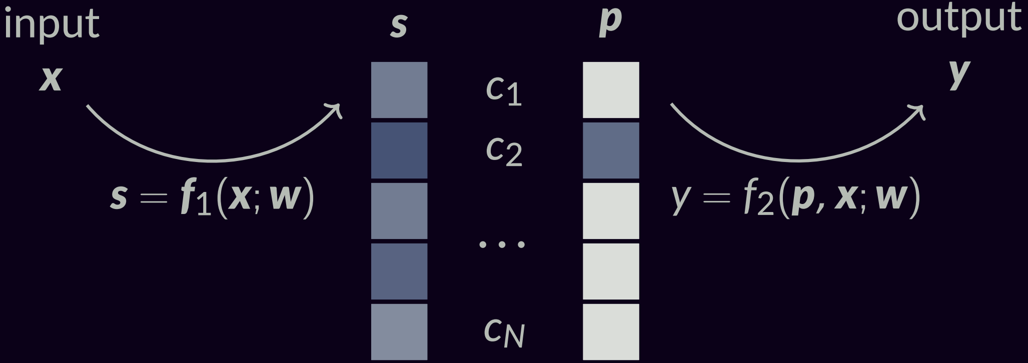
How to select an item from a set?



How to select an item from a set?

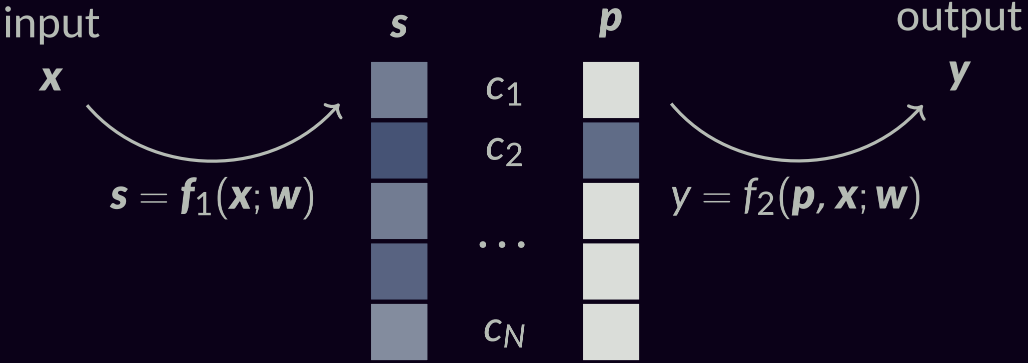


How to select an item from a set?



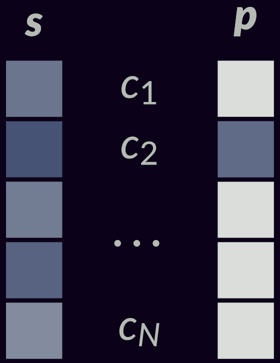
$$\frac{\partial y}{\partial \boldsymbol{w}} = ?$$

How to select an item from a set?



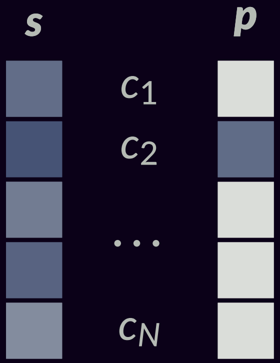
$\frac{\partial y}{\partial \boldsymbol{w}} = ?$ or, essentially, $\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$

Winner Takes It All



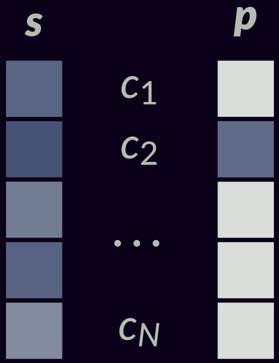
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



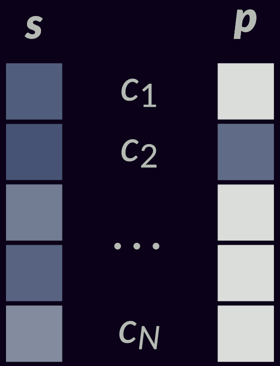
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



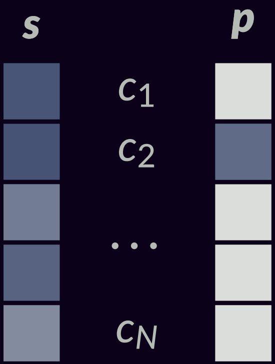
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



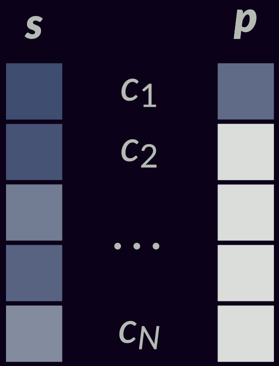
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



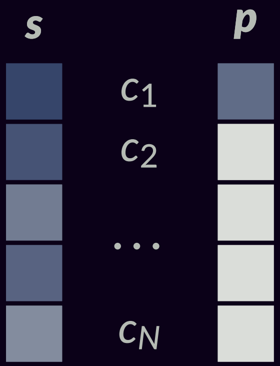
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



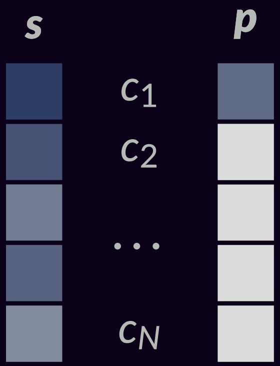
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



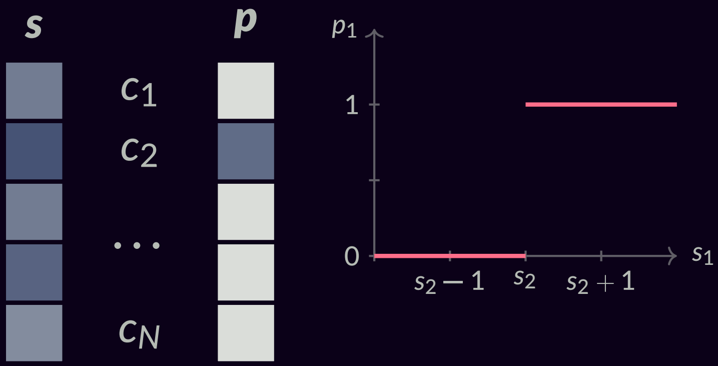
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

Winner Takes It All



$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = ?$$

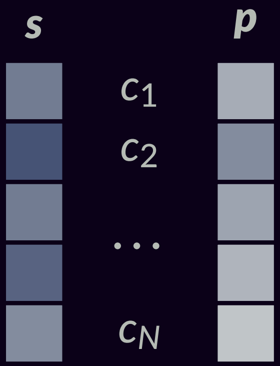
Argmax

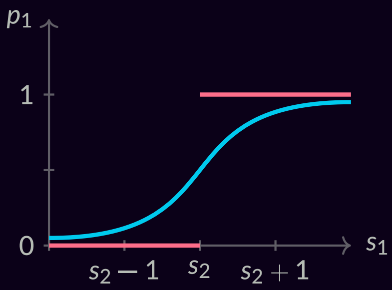


$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = \boldsymbol{0}$$

Argmax vs. Softmax

$$p_j = \exp(s_j)/Z$$





$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}} = \operatorname{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top$$

An Inconvenient Truth

We need gradients.
So you can't just take the highest-scoring item.

But...

That does not mean that softmax is the only choice.

But...

That does not mean that softmax is the only choice.

- Softmax attention is dense

But...

That does not mean that softmax is the only choice.

- Softmax attention is dense
- Argmax is completely sparse

But...

That does not mean that softmax is the only choice.

- Softmax attention is dense
- Argmax is completely sparse
- Maybe there's some kind of middle ground?

Sparsemax

$$\mathrm{sparsemax}(\boldsymbol{s}) = \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} - 1/2\|\boldsymbol{p}\|_2^2$$
$$= \arg\min_{\boldsymbol{p}\in\triangle} \|\boldsymbol{p} - \boldsymbol{s}\|_2^2$$

Computation:

$$\boldsymbol{p}^\star = [\boldsymbol{s} - \tau\mathbf{1}]_+$$
$$s_i > s_j \Rightarrow p_i \geq p_j$$
$\mathcal{O}(d)$ via partial sort

(Held et al., 1974; Brucker, 1984; Condat, 2016)

Backward pass:

$$\boldsymbol{J}_{\mathrm{sparsemax}} = \mathrm{diag}(\boldsymbol{s}) - \frac{1}{|\mathcal{S}|}\boldsymbol{s}\boldsymbol{s}^\top$$
where $\mathcal{S} = \{j : p_j^\star > 0\}$,
$$s_j = [\![j \in \mathcal{S}]\!]$$

(Martins and Astudillo, 2016)

(Peters et al., 2019)

α-entmax

$$\pi_{\mathsf{H}^{\mathsf{t}}_{\alpha}}(\boldsymbol{s}) := \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathsf{H}^{\mathsf{t}}_{\alpha}(\boldsymbol{p})$$

Solution has the form:

$$\pi_{\mathsf{H}^{\mathsf{t}}_{\alpha}}(\boldsymbol{s}) = [(\alpha - 1)\boldsymbol{s} - \tau\mathbf{1}]_{+}^{1/\alpha-1}$$

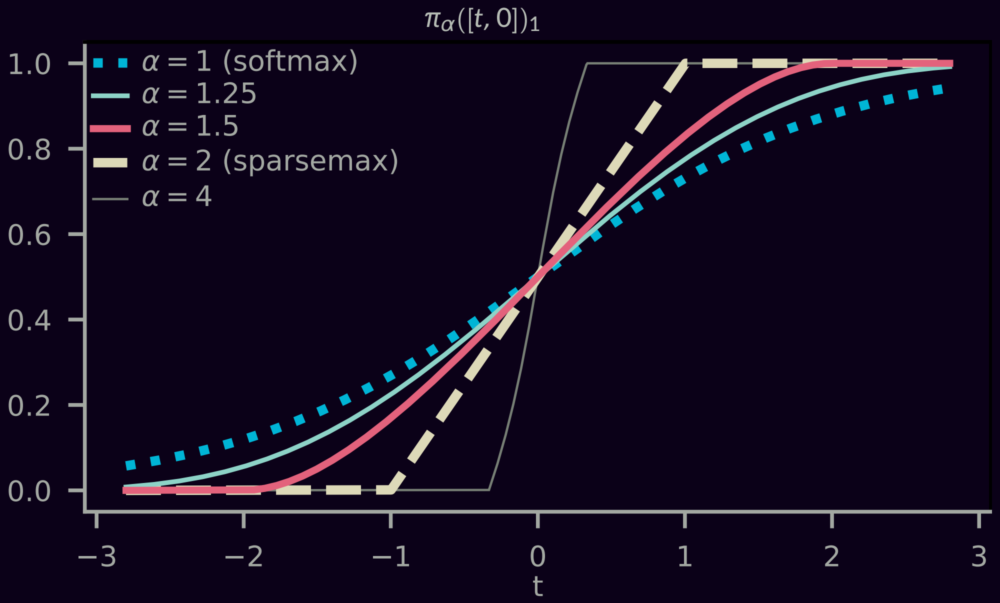
Algorithms:

bisection

- approximate; bracket $\tau \in [\tau_{\mathrm{lo}}, \tau_{\mathrm{hi}}]$
- gain 1 bit per $\mathcal{O}(d)$ iteration
- `float32` has 23 mantissa bits

sort-based

- exact algorithm, $\mathcal{O}(d \log d)$
- available only for $\alpha \in \{1.5, 2\}$
- huge speed-up from partial sorting when expecting sparse solutions



$\pi_\alpha([t, 0])_1$
$\alpha = 1$ (softmax)
$\alpha = 1.25$
$\alpha = 1.5$
$\alpha = 2$ (sparsemax)
$\alpha = 4$
1.0
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3
t

Structural Biases?

Imagine you are an en→pt model.

Structural Biases?

Imagine you are an en→pt model.

I am going to the store →

Structural Biases?

Imagine you are an en→pt model.

I am going to the store → Vou à loja

Structural Biases?

Imagine you are an en→pt model.

I am going to the store → Vou à loja

When you generate “Vou”, where do you attend?

Structural Biases?

Imagine you are an en→pt model.

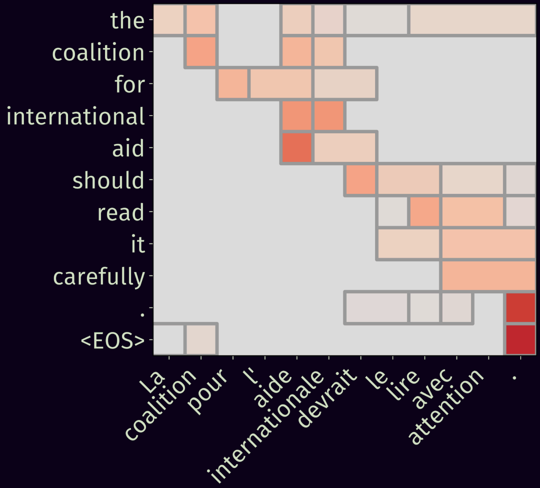
I am going to the store → Vou à loja

When you generate "Vou", where do you attend?

I am going to the store → Vou à loja

Enter fusedmax

Basic idea: penalize weight differences between adjacent positions.


the
coalition
for
international
aid
should
read
it
carefully
.
<EOS>
La
coalition
pour
l'
aide
internationale
devrait
le
lire
avec
attention
.

fusedmax

Fusedmax

$$\mathrm{fusedmax}(\boldsymbol{s}) = \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} - 1/2\|\boldsymbol{p}\|_2^2 - \sum_{2\le j\le d} |p_j - p_{j-1}|$$

$$= \arg\min_{\boldsymbol{p}\in\triangle} \|\boldsymbol{p} - \boldsymbol{s}\|_2^2 + \sum_{2\le j\le d} |p_j - p_{j-1}|$$

$$\mathrm{prox}_{\mathrm{fused}}(\boldsymbol{s}) = \arg\min_{\boldsymbol{p}\in\mathbb{R}^d} \|\boldsymbol{p} - \boldsymbol{s}\|_2^2 + \sum_{2\le j\le d} |p_j - p_{j-1}|$$

Proposition: $\mathrm{fusedmax}(\boldsymbol{s}) = \mathrm{sparsemax}\left(\mathrm{prox}_{\mathrm{fused}}(\boldsymbol{s})\right)$

(Niculae and Blondel, 2017)

Constrained Attention

Another structural idea: limit how much attention a word gets.

(Martins and Kreutzer, 2017; Malaviya et al., 2018)

Constrained Attention

$$\mathrm{softmax}(\boldsymbol{z}) = \arg\min_{\boldsymbol{p} \in \triangle^D} \mathrm{KL}(\boldsymbol{p} \| \mathrm{softmax}(\boldsymbol{z}))$$

(Martins and Kreutzer, 2017; Malaviya et al., 2018)

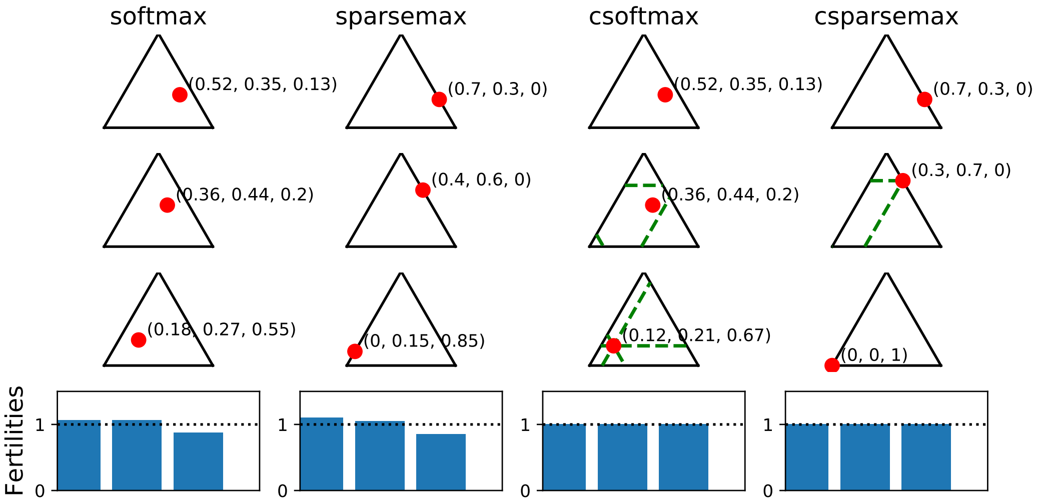
Constrained Attention

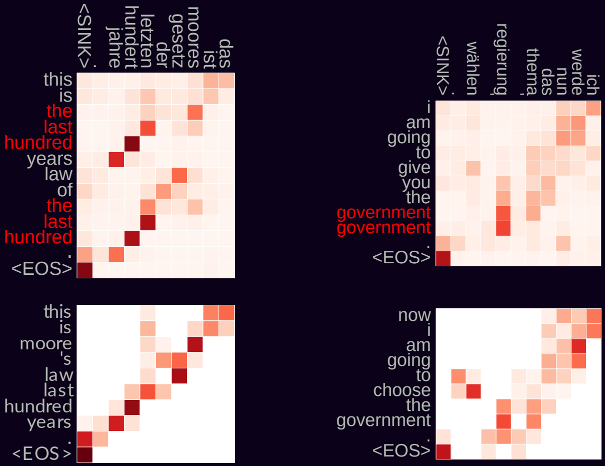
$$\operatorname{softmax}(\boldsymbol{z}) = \underset{\boldsymbol{p} \in \triangle^D}{\arg\min} \operatorname{KL}(\boldsymbol{p} \| \operatorname{softmax}(\boldsymbol{z}))$$

$$\operatorname{csoftmax}(\boldsymbol{z}, \boldsymbol{u}) = \underset{\boldsymbol{p} \in \triangle^D s.t. \boldsymbol{p} \leq \boldsymbol{u}}{\arg\min} \operatorname{KL}(\boldsymbol{p} \| \operatorname{softmax}(\boldsymbol{z}))$$

$\boldsymbol{u}$: max probability for each index

Example: Source Sentence with Three Words




<SINK>
.
jahre
hundert
letzten
der
gesetz
moores
ist
das
this
is
the
last
hundred
years
law
of
the
last
hundred
.
<EOS>
this
is
moore
's
law
last
hundred
years
.
<EOS>
<SINK>
.
wählen
regierung
,
.
thema
das
nun
werde
ich
i
am
going
to
give
you
the
government
government
.
<EOS>
now
i
am
going
to
choose
the
government
.
<EOS>

2. Transformers

Beyond seq2seq

The spirit of *attention mechanisms* reaches far:

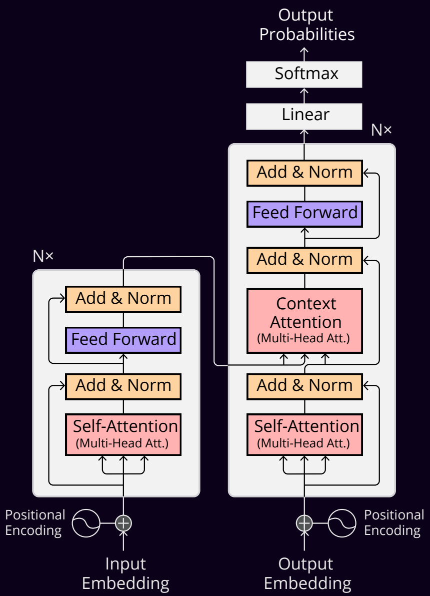
- Key-Value Attention
- Multi-head Attention
- Self-Attention

What do you get when you put these together?

A line of toys?

A line of toys?
A film franchise starring Shia LaBeouf?

A line of toys?
A film franchise starring Shia LaBeouf?
Wrong.


Output
Probabilities
Softmax
Linear
N×
Add & Norm
Feed Forward
Add & Norm
Context
Attention
(Multi-Head Att.)
Add & Norm
Self-Attention
(Multi-Head Att.)
N×
Add & Norm
Feed Forward
Add & Norm
Self-Attention
(Multi-Head Att.)
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding

(Graves et al., 2014; Sukhbaatar et al., 2015)

Key-Value Attention

idea: the objects we average *(values)*
and the objects used to compute scores *(keys)*
don't need to be identical!

$$s_j = \boldsymbol{h}_j^\top \boldsymbol{q}$$

$$\boldsymbol{u} = \mathrm{softmax}(\boldsymbol{s})^\top \boldsymbol{H}$$

$$s_j = \boldsymbol{k}_j^\top \boldsymbol{q}$$

$$\boldsymbol{u} = \mathrm{softmax}(\boldsymbol{s})^\top \boldsymbol{V}$$

(Vaswani et al., 2017)

Multi-head Attention

idea: compute k different attention averages,
& concatenate the outputs.

$$s_j = \boldsymbol{k}_j^\top \boldsymbol{q}$$

$$\boldsymbol{u} = \mathrm{softmax}(\boldsymbol{s})^\top \boldsymbol{V}$$

```
u = softmax(K @ q) @ V
```

$$s_j^{(i)} = (\boldsymbol{W}_k^{(i)} \boldsymbol{k}_j)^\top (\boldsymbol{W}_q^{(i)} \boldsymbol{q})$$

$$\boldsymbol{u}^{(i)} = \mathrm{softmax}(\boldsymbol{s}^{(i)})^\top (\boldsymbol{V}\boldsymbol{W}_v^{(i)})$$

$$\boldsymbol{u} = [\boldsymbol{u}^{(1)}; \cdots; \boldsymbol{u}^{(k)}]$$

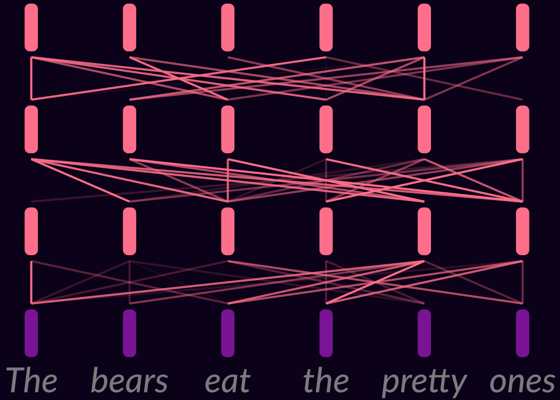
```
for i in range(num_heads):
  Ki = K @ Wk[i].t()
  Vi = V @ Wv[i].t()
  qi = q @ Wq[i].t()
  ui = softmax(Ki @ qi) @ Vi
u = concat(ui)
```

(Cheng et al., 2016)

Self-attention

Attention as an *encoder layer*

...

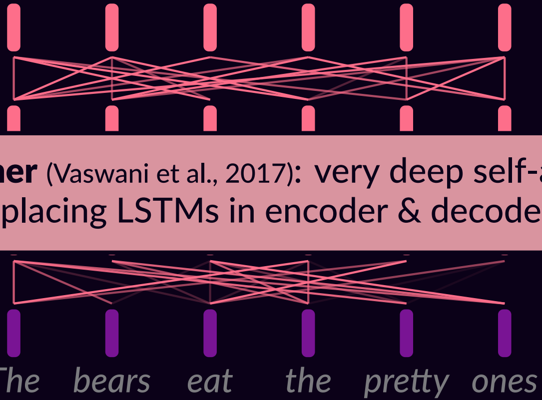


(Cheng et al., 2016)

Self-attention

Attention as an *encoder layer*

. . .



Transformer (Vaswani et al., 2017): very deep self-attention replacing LSTMs in encoder & decoder

Could you say that again?

Each self-attention head uses key-value attention:

- Take a sequence of embeddings $X = \mathbf{x}_1 \ldots \mathbf{x}_n$, $\mathbf{x}_i \in \mathbb{R}^D$

Could you say that again?

Each self-attention head uses key-value attention:

- Take a sequence of embeddings $X = \mathbf{x}_1 \ldots \mathbf{x}_n$, $\mathbf{x}_i \in \mathbb{R}^D$
- queries: $Q = XW_q$, $\mathbf{q}_i \in \mathbb{R}^{D_q}$

Could you say that again?

Each self-attention head uses key-value attention:

- Take a sequence of embeddings $X = \mathbf{x}_1 \dots \mathbf{x}_n, \mathbf{x}_i \in \mathbb{R}^D$
- queries: $Q = XW_q, \mathbf{q}_i \in \mathbb{R}^{D_q}$
- keys: $K = XW_k, \mathbf{k}_i \in \mathbb{R}^{D_q}$

Could you say that again?

Each self-attention head uses key-value attention:

- Take a sequence of embeddings $X = \mathbf{x}_1 \ldots \mathbf{x}_n$, $\mathbf{x}_i \in \mathbb{R}^D$
- queries: $Q = XW_q$, $\mathbf{q}_i \in \mathbb{R}^{D_q}$
- keys: $K = XW_k$, $\mathbf{k}_i \in \mathbb{R}^{D_q}$
- values: $V = XW_v$, $\mathbf{v}_i \in \mathbb{R}^{D_v}$

Could you say that again?

Each self-attention head uses key-value attention:

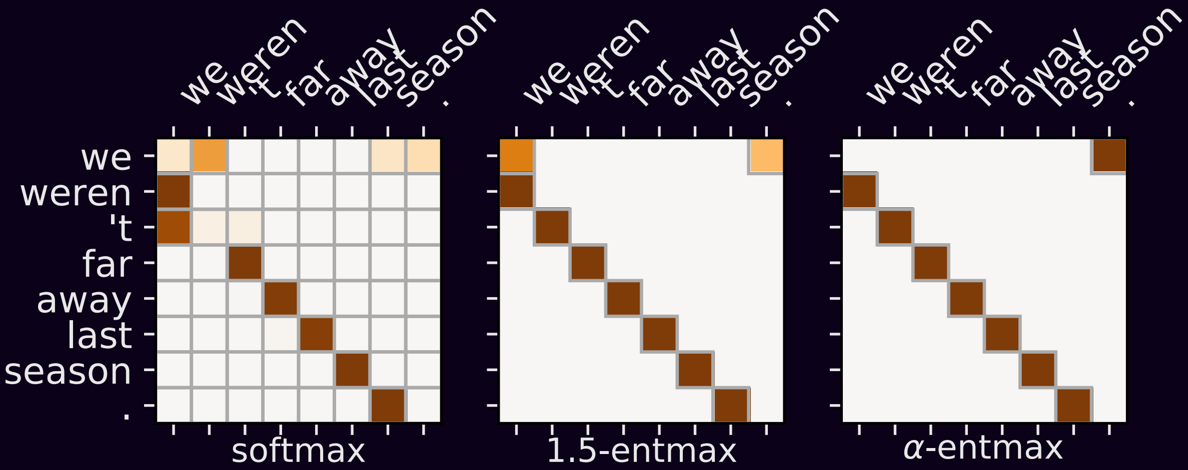
- Take a sequence of embeddings $X = \mathbf{x}_1 \dots \mathbf{x}_n$, $\mathbf{x}_i \in \mathbb{R}^D$
- queries: $Q = XW_q$, $\mathbf{q}_i \in \mathbb{R}^{D_q}$
- keys: $K = XW_k$, $\mathbf{k}_i \in \mathbb{R}^{D_q}$
- values: $V = XW_v$, $\mathbf{v}_i \in \mathbb{R}^{D_v}$

Important: each head has different W_q, W_k, W_v

$$\text{softmax}\left(\frac{QK^{\top}}{\sqrt{D}}\right)V$$

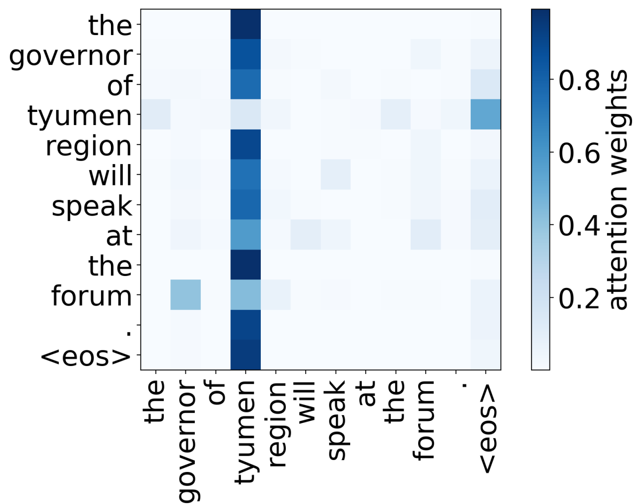
So what do heads learn?

Previous Position

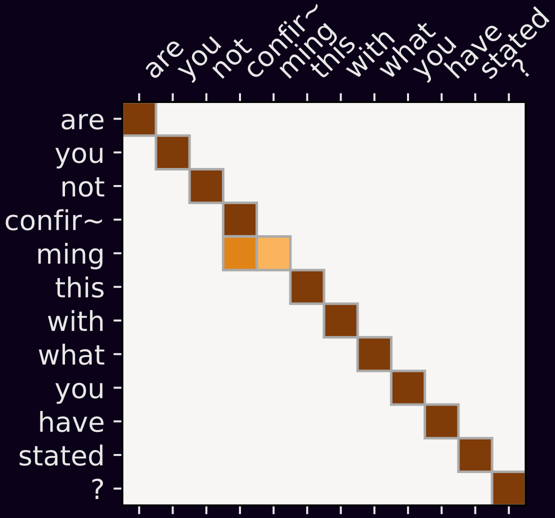


(Voita et al., 2019)

Rare Words



Merging Subwords



Replacing Recurrence

Self-attention is the *only* place where positions interact.
What do we gain over RNN-based models?
What do we lose?

References I

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate". In: *Proc. of ICLR*.

Brucker, Peter (1984). "An $O(n)$ algorithm for quadratic knapsack problems". In: *Operations Research Letters* 3.3, pp. 163–166.

Cheng, Jianpeng, Li Dong, and Mirella Lapata (2016). "Long Short-Term Memory-Networks for Machine Reading". In: *Proc. of EMNLP*.

Condat, Laurent (2016). "Fast projection onto the simplex and the ℓ_1 ball". In: *Mathematical Programming* 158.1-2, pp. 575–585.

Correia, Gonçalo M, Vlad Niculae, and André FT Martins (2019). "Adaptively Sparse Transformers". In: *Proc. EMNLP-IJCNLP (to appear)*.

Graves, Alex, Greg Wayne, and Ivo Danihelka (2014). "Neural Turing Machines". In: *arXiv preprint arXiv:1410.5401*.

Held, Michael, Philip Wolfe, and Harlan P Crowder (1974). "Validation of subgradient optimization". In: *Mathematical Programming* 6.1, pp. 62–88.

Luong, Minh-Thang, Hieu Pham, and Christopher D Manning (2015). "Effective approaches to attention-based neural machine translation". In: *Proc. of EMNLP*.

Malaviya, Chaitanya, Pedro Ferreira, and André F. T. Martins (2018). "Sparse and constrained attention for neural machine translation". In: *Proc. of ACL*.

Martins, André FT and Ramón Fernandez Astudillo (2016). "From softmax to sparsemax: A sparse model of attention and multi-label classification". In: *Proc. of ICML*.

References II

Martins, André FT and Julia Kreutzer (2017). “Learning What’s Easy: Fully Differentiable Neural Easy-First Taggers”. In: *Proc. of EMNLP*, pp. 349–362.

Niculae, Vlad and Mathieu Blondel (2017). “A regularized framework for sparse and structured neural attention”. In: *Proc. of NeurIPS*.

Peters, Ben, Vlad Niculae, and André FT Martins (2019). “Sparse Sequence-to-Sequence Models”. In: *Proc. ACL*.

Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus (2015). “End-to-end memory networks”. In: *Proc. of NeurIPS*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). “Attention Is All You Need”. In: *Proc. of NeurIPS*.

Voita, Elena, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov (July 2019). “Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5797–5808. doi: `10.18653/v1/P19-1580`. url: `https://www.aclweb.org/anthology/P19-1580`.