Adaptively Sparse Transformers

Gonçalo Correia Instituto de Telecomunicações, Lisbon

Vlad Niculae IT

André Martins IT & Unbabel

github.com/deep-spin/entmax goncalomcorreia.github.io

TL;DL (Too Long; Didn't Listen)

We replace the softmax function in Transformers with α-entmax, creating a model that is able to learn its own sparsity in each attention head.

(Bahdanau et al., 2015)

A bit of context... On Seq2Seq

United Nations elections end today

Slide Credit: Vlad Niculae

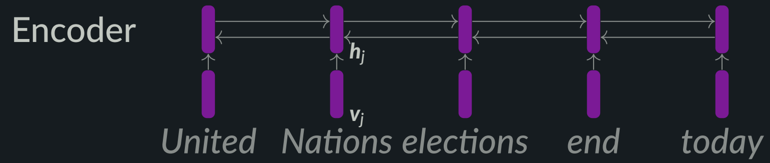
(Bahdanau et al., 2015)

A bit of context… On Seq2Seq

Encoder

(Bahdanau et al., 2015)

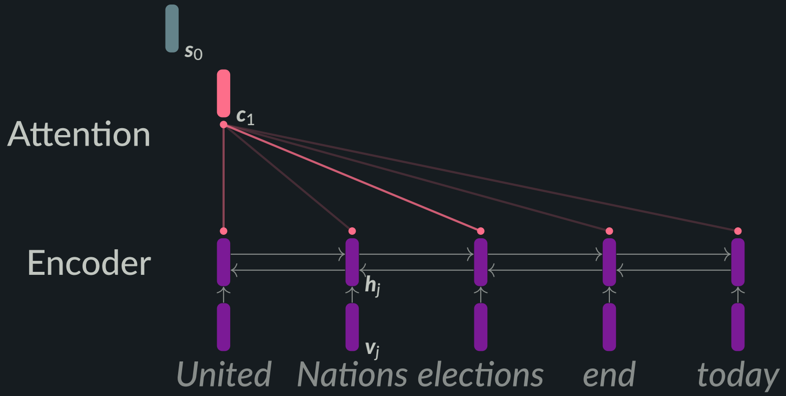
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

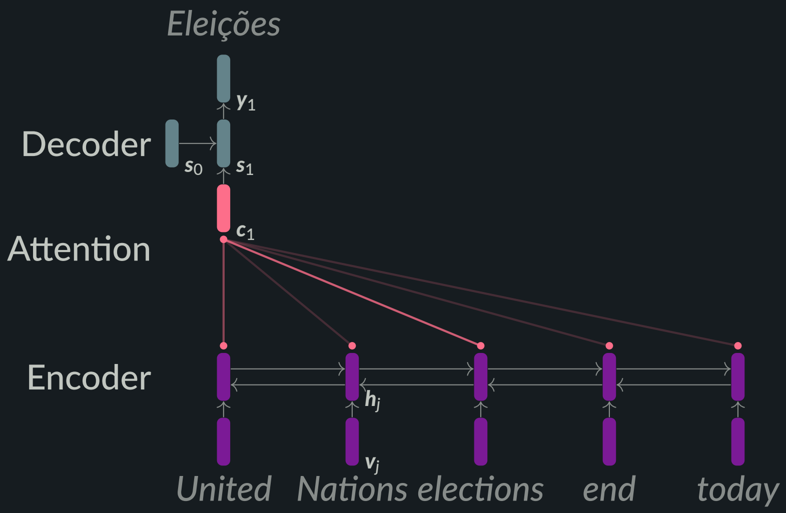
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

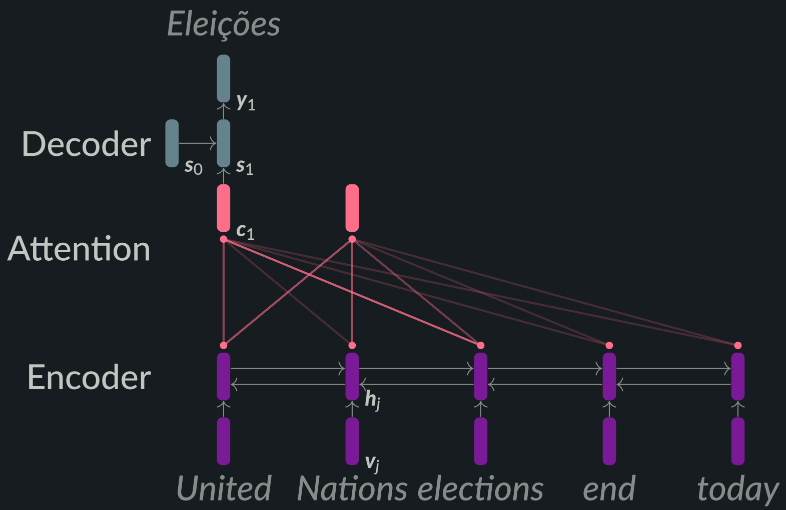
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

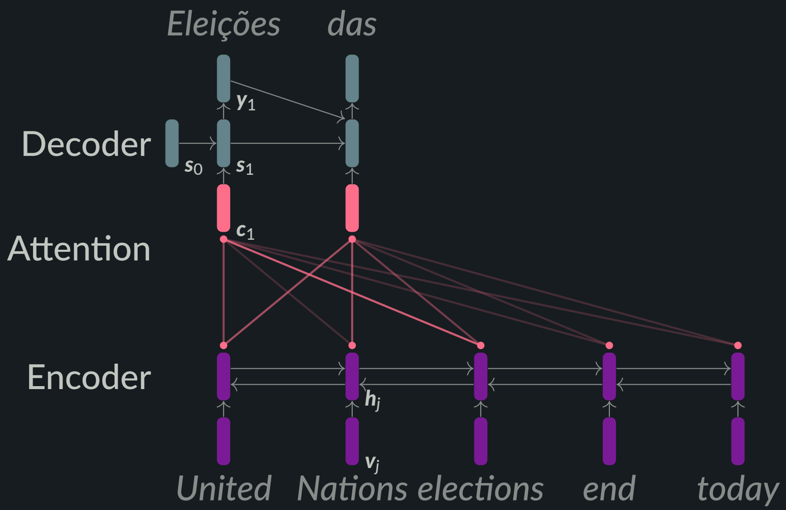
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

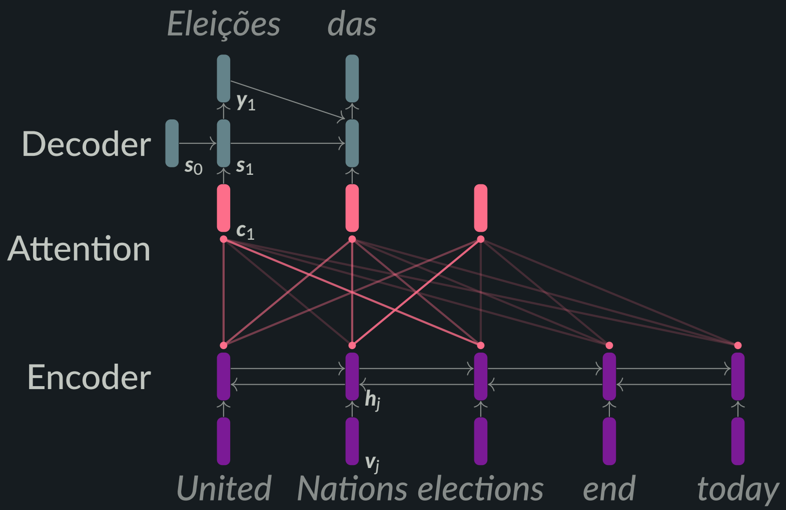
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

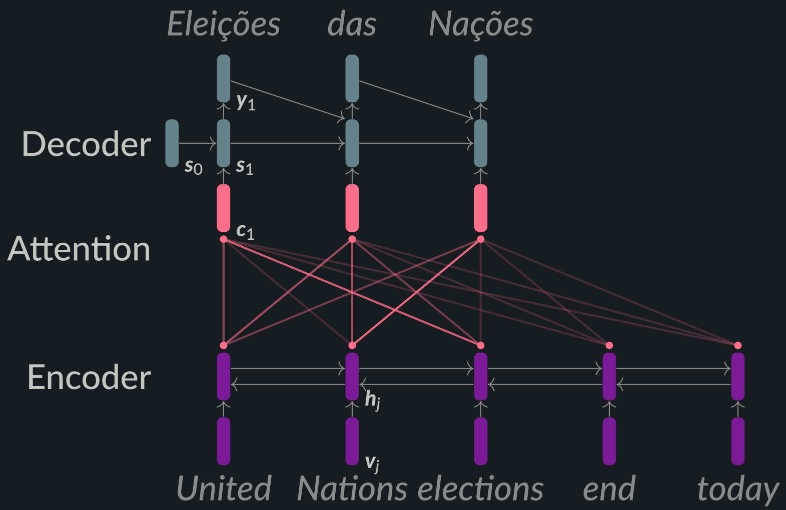
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

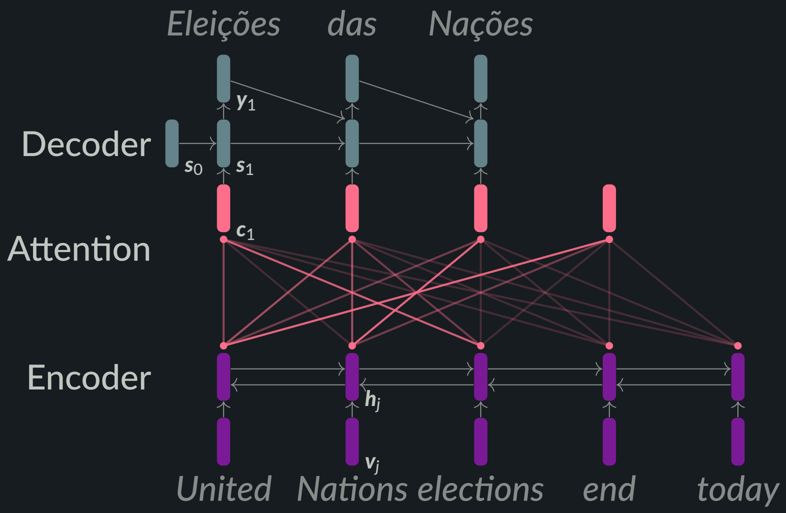
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

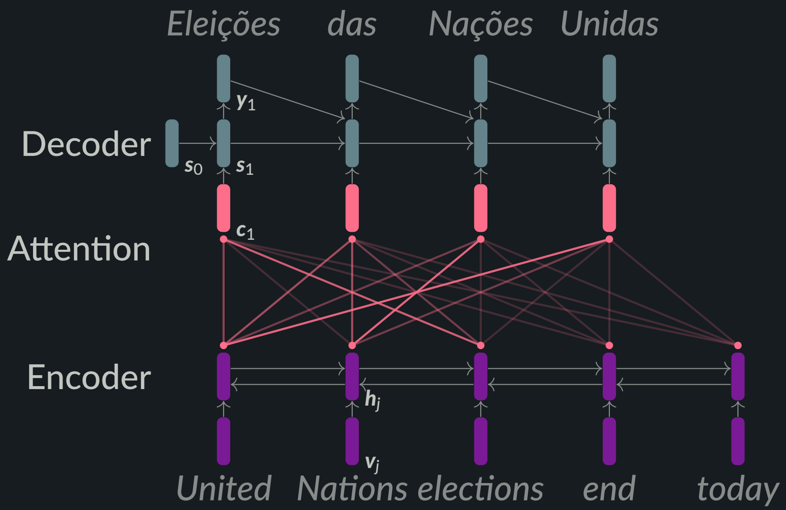
A bit of context... On Seq2Seq



Slide Credit: Vlad Niculae

(Bahdanau et al., 2015)

A bit of context... On Seq2Seq



attention weights computed with *softmax*:

for some decoder state s_t, compute contextually weighted average of input c_t:

$$z_j = \boldsymbol{s}_t^\top \boldsymbol{W}^{(a)} \boldsymbol{h}_j$$

$$\pi_j = \operatorname{softmax}_j(\boldsymbol{z})$$

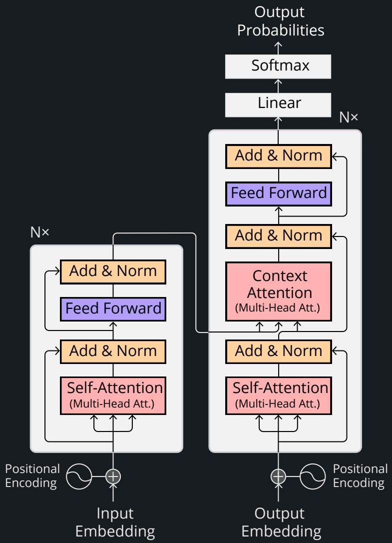
$$\boldsymbol{c}_t = \sum_j \pi_j \boldsymbol{h}_j$$

Slide Credit: Vlad Niculae

(Vaswani et al., 2017)

A bit of context… on Transformers

What if… Attention is all you need?

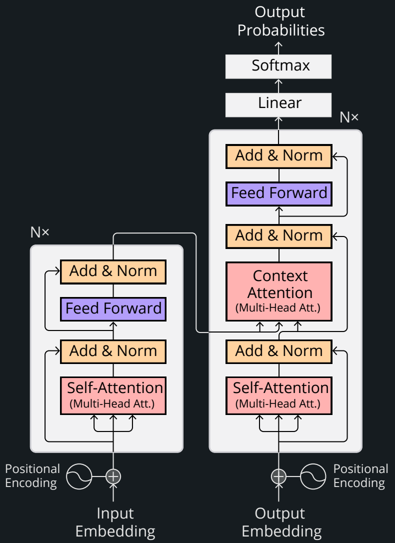


(Vaswani et al., 2017)

A bit of context... on Transformers

What if... Attention is all you need?

Key idea: Instead of Recurrent Neural Networks (RNNs), let's use attention mechanisms!



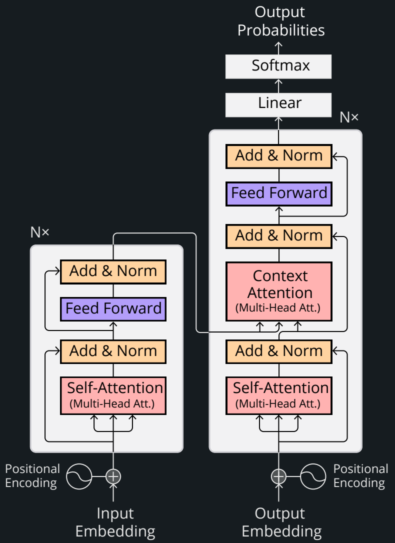
(Vaswani et al., 2017)

A bit of context… on Transformers

What if… Attention is all you need?

Key idea: Instead of Recurrent Neural Networks (RNNs), let's use attention mechanisms!

- In place of the RNNs, use self-attention



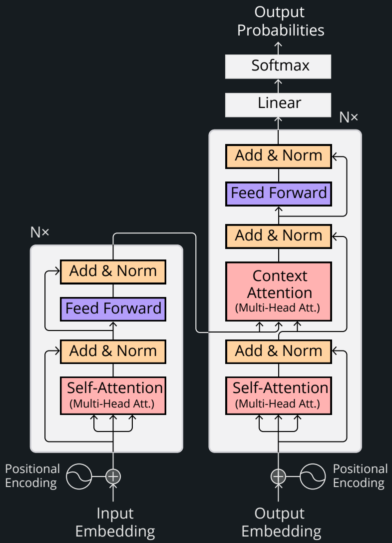
(Vaswani et al., 2017)

A bit of context... on Transformers

What if... Attention is all you need?

Key idea: Instead of Recurrent Neural Networks (RNNs), let's use attention mechanisms!

- In place of the RNNs, use self-attention
- Do this with multiple heads (i.e. attention mechanisms in parallel)



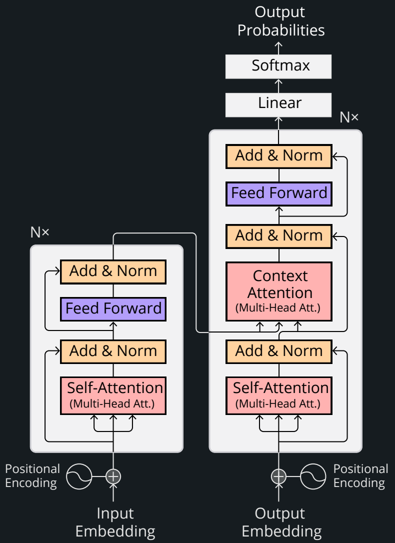
(Vaswani et al., 2017)

A bit of context... on Transformers

What if... Attention is all you need?

Key idea: Instead of Recurrent Neural Networks (RNNs), let's use attention mechanisms!

- In place of the RNNs, use self-attention
- Do this with multiple heads (i.e. attention mechanisms in parallel)
- ... and do it through several layers



Introduction

Transformers have been achieving incredible SOTA results in the last couple of years!

Introduction

Transformers have been achieving incredible SOTA results in the last couple of years!

But they seem overparameterized...

Introduction

Transformers have been achieving incredible SOTA results in the last couple of years!

But they seem overparameterized...

Attention heads aid visualization but they are completely dense.

Introduction

Transformers have been achieving incredible SOTA results in the last couple of years!

But they seem overparameterized...

Attention heads aid visualization but they are completely dense.

Our solution is to bet on sparsity:

- for interpretability
- for discovering linguistic structure
- for efficiency

Introduction

Transformers have been achieving incredible SOTA results in the last couple of years!

But they seem overparameterized...

Attention heads aid visualization but they are completely dense.

Our solution is to bet on sparsity:

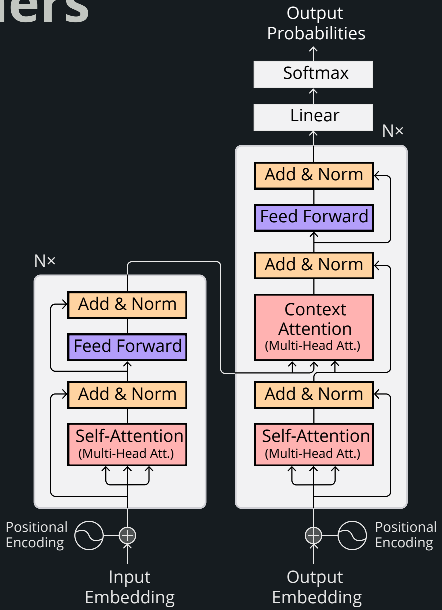
- for interpretability
- for discovering linguistic structure
- for efficiency

(Vaswani et al., 2017)

Transformers

In each attention head:

$$\bar{V} = \textbf{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V.$$



(Vaswani et al., 2017)

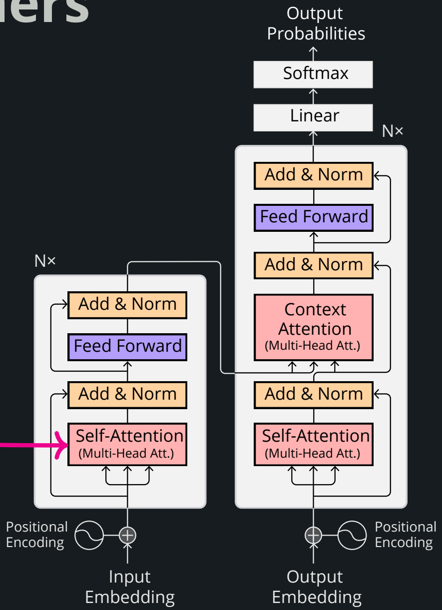
Transformers

In each attention head:

$$\bar{V} = \mathbf{softmax}\left(\frac{\mathbf{QK}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}.$$

Attention in three places:

- Self-attention in the encoder



(Vaswani et al., 2017)

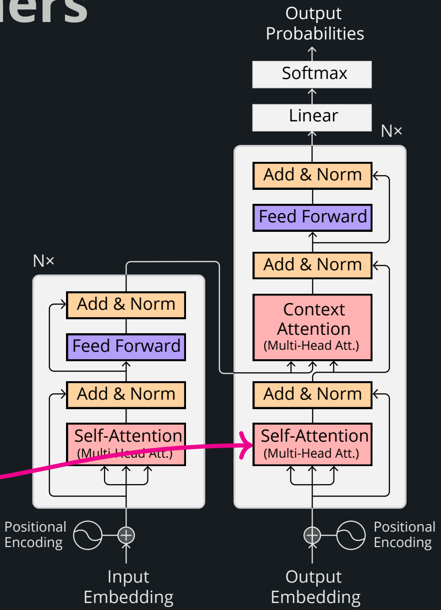
Transformers

In each attention head:

$$\bar{V} = \textbf{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V.$$

Attention in three places:

- Self-attention in the encoder
- Self-attention in the decoder



(Vaswani et al., 2017)

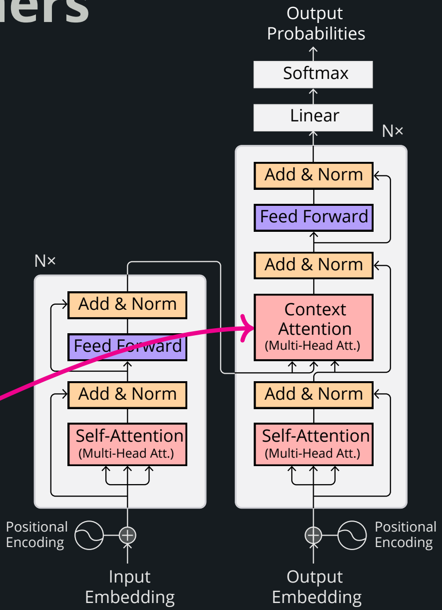
Transformers

In each attention head:

$$\bar{V} = \mathbf{softmax}\left(\frac{QK^{T}}{\sqrt{d_k}}\right)V.$$

Attention in three places:

- Self-attention in the encoder
- Self-attention in the decoder
- Contextual attention



Sparse Transformers

Sparse Transformers

Key idea: replace softmax in attention heads by a sparse normalizing function! 🙌

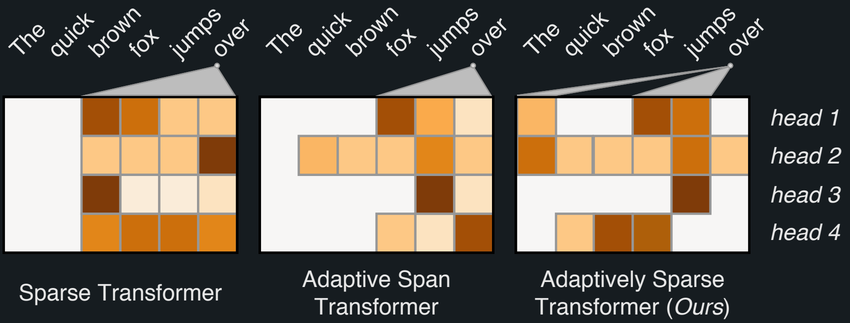
Adaptively Sparse Transformers

Key idea: replace softmax in attention heads by a sparse normalizing function! 🙌

Another key idea: use a normalizing function that is adaptively sparse via a learnable α! 🙌 🙌 🙌

(Child et al., 2019; Sukhbaatar et al., 2019)

Related Work: Other Sparse Transformers



Our model allows non-contiguous attention for each head.

What is softmax?

Softmax exponentiates and normalizes: $\textbf{softmax}(z_i) := \frac{\exp(z_i)}{\sum_j \exp(z_j)}$

What is softmax?

Softmax exponentiates and normalizes: $\textbf{softmax}(z_i) := \dfrac{\exp(z_i)}{\sum_j \exp(z_j)}$

It's fully dense: $\textbf{softmax}(\boldsymbol{z}) > \boldsymbol{0}$

What is softmax?

Softmax exponentiates and normalizes: $\textbf{softmax}(z_i) := \frac{\exp(z_i)}{\sum_j \exp(z_j)}$

It's fully dense: $\textbf{softmax}(\boldsymbol{z}) > \mathbf{0}$

Argmax can be written as:

$$\textbf{argmax}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p} \in \triangle} \boldsymbol{z}^\top \boldsymbol{p}$$

What is softmax?

Softmax exponentiates and normalizes: $\textbf{softmax}(z_i) := \frac{\exp(z_i)}{\sum_j \exp(z_j)}$

It's fully dense: $\textbf{softmax}(\boldsymbol{z}) > \mathbf{0}$

Argmax can be written as:

$$\textbf{argmax}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{z}^\top \boldsymbol{p}$$

- Retrieves a **one-hot vector** for the highest scored index.

What is softmax?

Softmax exponentiates and normalizes: $\textbf{softmax}(z_i) := \frac{\exp(z_i)}{\sum_j \exp(z_j)}$

It's fully dense: $\textbf{softmax}(\boldsymbol{z}) > \mathbf{0}$

Argmax can be written as:

$$\textbf{argmax}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p} \in \triangle} \boldsymbol{z}^\top \boldsymbol{p}$$

- Retrieves a **one-hot vector** for the highest scored index.
- Sometimes used as hard attention, but not differentiable!

(Niculae and Blondel, 2017)

Ω-Regularized Argmax

For convex Ω, define the **Ω-regularized argmax transformation**:

$$\mathbf{argmax}_{\Omega}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p} \in \triangle} \boldsymbol{z}^\top \boldsymbol{p} - \Omega(\boldsymbol{p})$$

(Niculae and Blondel, 2017)

Ω-Regularized Argmax

For convex Ω, define the **Ω-regularized argmax transformation:**

$$\mathbf{argmax}_{\Omega}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{z}^\top \boldsymbol{p} - \Omega(\boldsymbol{p})$$

- Argmax corresponds to **no regularization,** $\Omega \equiv 0$

(Niculae and Blondel, 2017)

Ω-Regularized Argmax

For convex Ω, define the **Ω-regularized argmax transformation:**

$$\mathbf{argmax}_{\Omega}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p}\in\Delta} \boldsymbol{z}^{\top}\boldsymbol{p} - \Omega(\boldsymbol{p})$$

- Argmax corresponds to **no regularization,** $\Omega \equiv 0$
- Softmax amounts to **entropic regularization,** $\Omega(\boldsymbol{p}) = \sum_{i=1}^{K} p_i \log p_i$

(Niculae and Blondel, 2017)

Ω-Regularized Argmax

For convex Ω, define the **Ω-regularized argmax transformation:**

$$\mathbf{argmax}_{\Omega}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p}\in\Delta} \boldsymbol{z}^{\top}\boldsymbol{p} - \Omega(\boldsymbol{p})$$

- Argmax corresponds to **no regularization**, $\Omega \equiv 0$
- Softmax amounts to **entropic regularization**, $\Omega(\boldsymbol{p}) = \sum_{i=1}^{K} p_i \log p_i$
- Sparsemax amounts to **ℓ_2-regularization**, $\Omega(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|^2$.

(Martins and Astudillo, 2016)

(Niculae and Blondel, 2017)

Ω-Regularized Argmax

For convex Ω, define the **Ω-regularized argmax transformation:**

$$\mathbf{argmax}_{\Omega}(\boldsymbol{z}) := \arg\max_{\boldsymbol{p}\in\Delta} \boldsymbol{z}^\top \boldsymbol{p} - \Omega(\boldsymbol{p})$$

- Argmax corresponds to **no regularization**, $\Omega \equiv 0$
- Softmax amounts to **entropic regularization**, $\Omega(\boldsymbol{p}) = \sum_{i=1}^{K} p_i \log p_i$
- Sparsemax amounts to **ℓ_2-regularization**, $\Omega(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|^2$.

Is there something in-between?

α-Entmax

Parametrized by $\alpha \geq 0$:

$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_{i=1}^{K} p_i^\alpha\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^{K} p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$

α-Entmax

Parametrized by $\alpha \geq 0$:

$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_{i=1}^{K} p_i^\alpha\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^{K} p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$

- **Argmax** corresponds to $\alpha \to \infty$

α-Entmax

Parametrized by $\alpha \geq 0$:

$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_{i=1}^{K} p_i^\alpha\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^{K} p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$

- **Argmax** corresponds to $\alpha \to \infty$
- **Softmax** amounts to $\alpha \to 1$

α-Entmax

Parametrized by $\alpha \geq 0$:

$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_{i=1}^{K} p_i^\alpha\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^{K} p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$

- **Argmax** corresponds to $\alpha \to \infty$
- **Softmax** amounts to $\alpha \to 1$
- **Sparsemax** amounts to $\alpha = 2$.

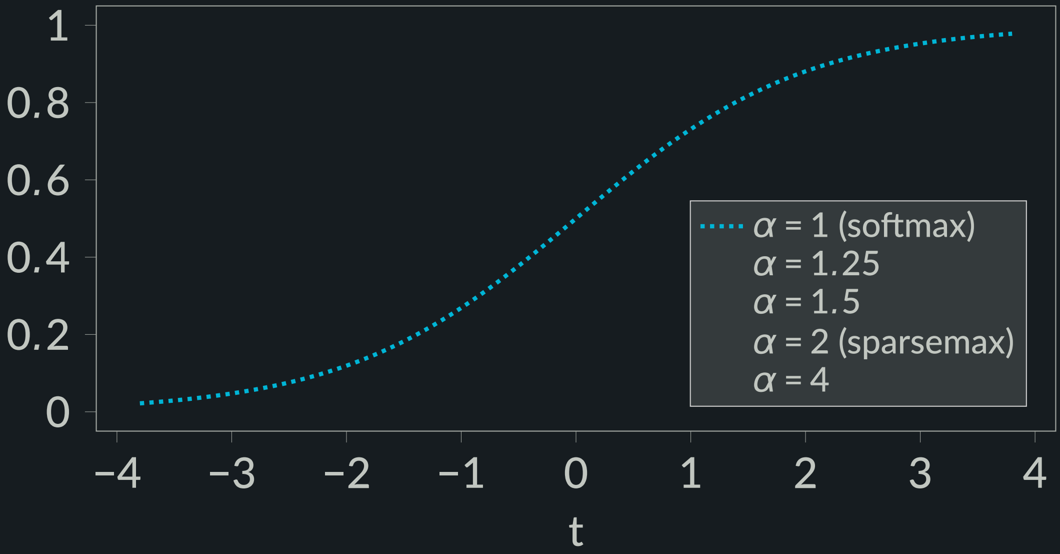
α-Entmax

Parametrized by $\alpha \geq 0$:

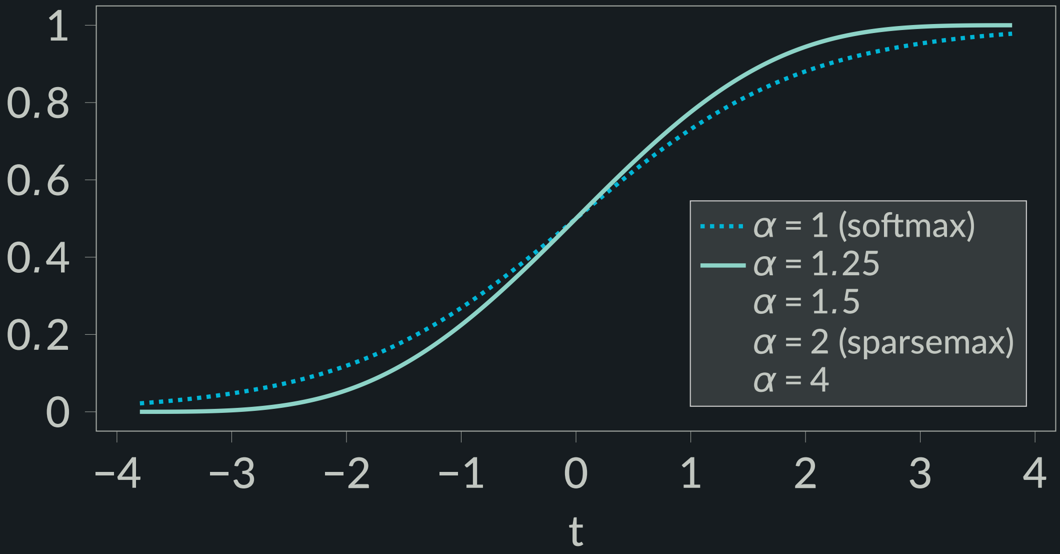
$$\Omega_\alpha(\boldsymbol{p}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_{i=1}^{K} p_i^\alpha\right) & \text{if } \alpha \neq 1 \\ \sum_{i=1}^{K} p_i \log p_i & \text{if } \alpha = 1. \end{cases}$$

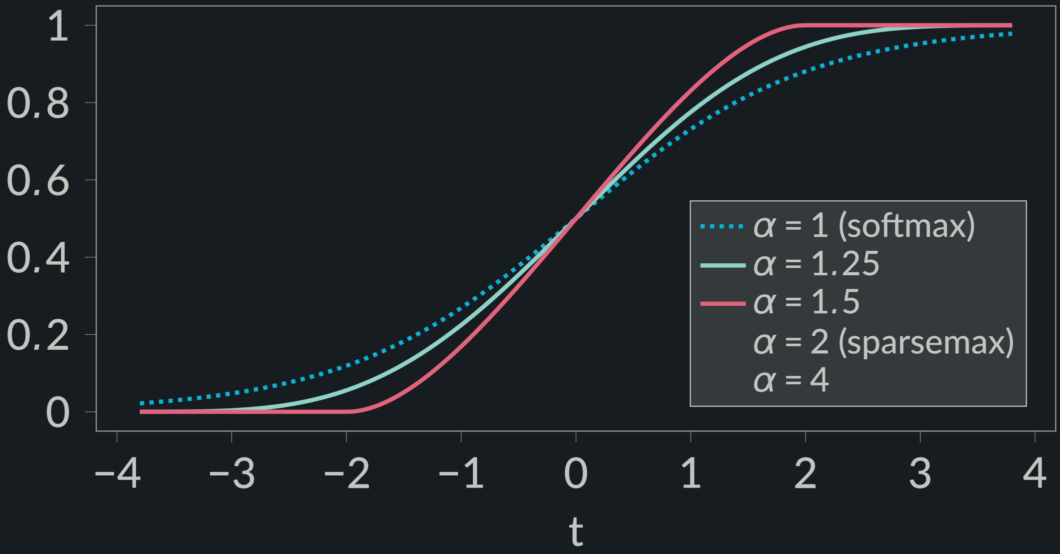
- **Argmax** corresponds to $\alpha \to \infty$
- **Softmax** amounts to $\alpha \to 1$
- **Sparsemax** amounts to $\alpha = 2$.

Key result: **can be sparse for** $\alpha > 1$, propensity for sparsity increases with α.

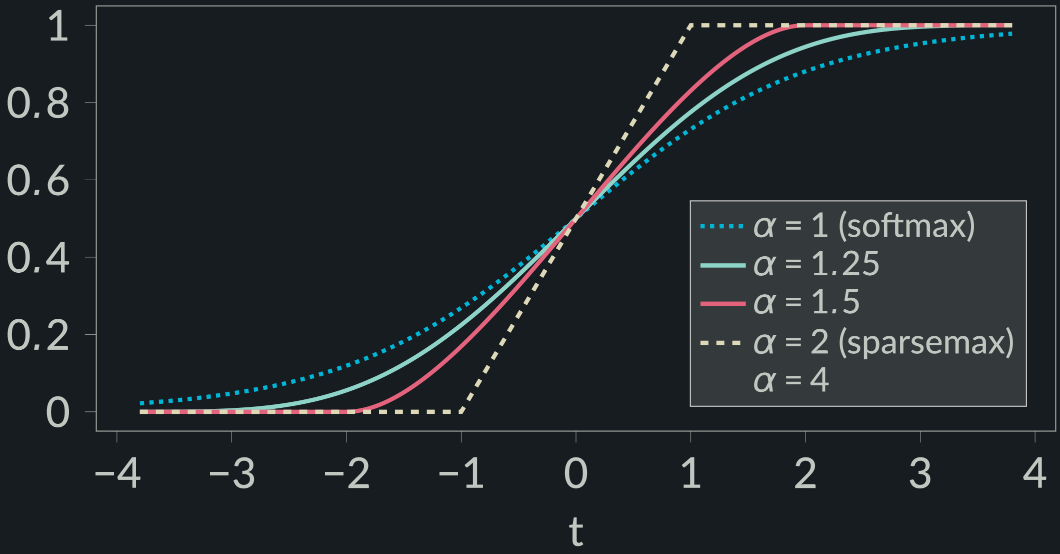


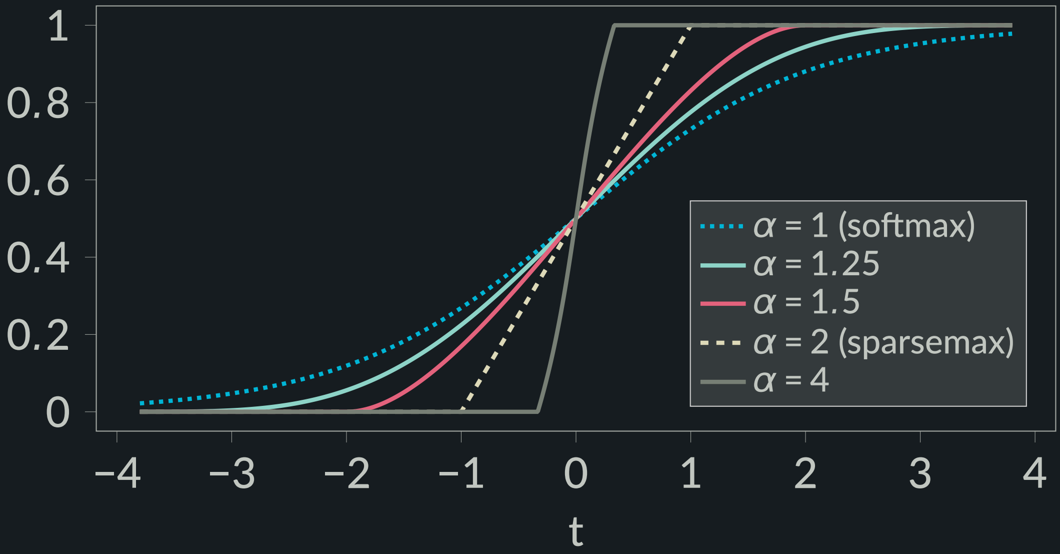
1
0.8
0.6
0.4
0.2
0
−4
−3
−2
−1
0
1
2
3
4
t
α = 1 (softmax)
α = 1.25
α = 1.5
α = 2 (sparsemax)
α = 4


1
0.8
0.6
0.4
0.2
0
−4
−3
−2
−1
0
1
2
3
4
t
α = 1 (softmax)
α = 1.25
α = 1.5
α = 2 (sparsemax)
α = 4



1
0.8
0.6
0.4
0.2
0
−4
−3
−2
−1
0
1
2
3
4
t
α = 1 (softmax)
α = 1.25
α = 1.5
α = 2 (sparsemax)
α = 4


1
0.8
0.6
0.4
0.2
0
−4
−3
−2
−1
0
1
2
3
4
t
α = 1 (softmax)
α = 1.25
α = 1.5
α = 2 (sparsemax)
α = 4


1
0.8
0.6
0.4
0.2
0
−4
−3
−2
−1
0
1
2
3
4
t
α = 1 (softmax)
α = 1.25
α = 1.5
α = 2 (sparsemax)
α = 4

Learning α

Learning α

Key contribution:

a closed-form expression for $\dfrac{\partial\, \alpha\text{-entmax}(\mathbf{z})}{\partial \alpha}$ 😱

Learning α

Key contribution:

a closed-form expression for $\dfrac{\partial\, \alpha\text{-entmax}(\mathbf{z})}{\partial \alpha}$ 😱

Requires argmin differentiation $\rightarrow$ see paper for details!

Learning α

Key co

a cl

:pip install entmax

Check `github.com/deep-spin/entmax`

Requires argmin differentiation $\longrightarrow$ see paper for details!

BLEU Scores

activation	de→en	ja→en	ro→en	en→de
softmax	29.79	21.57	32.70	26.02
1.5-entmax	29.83	22.13	33.10	25.89
α-entmax	29.90	21.74	32.89	26.93

BLEU Scores

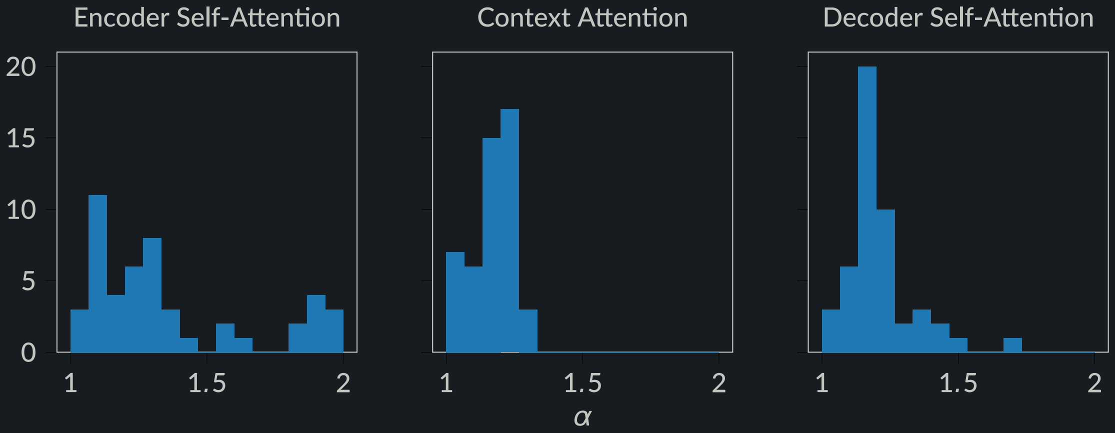
activation	de→en	ja→en	ro→en	en→de
softmax	29.79	21.57	32.70	26.02
1.5-entmax	29.83	22.13	33.10	25.89
α-entmax	29.90	21.74	32.89	26.93

BLEU Scores

activation	de→en	ja→en	ro→en	en→de
softmax	29.79	21.57	32.70	26.02
1.5-entmax	29.83	22.13	33.10	25.89
α-entmax	29.90	21.74	32.89	26.93

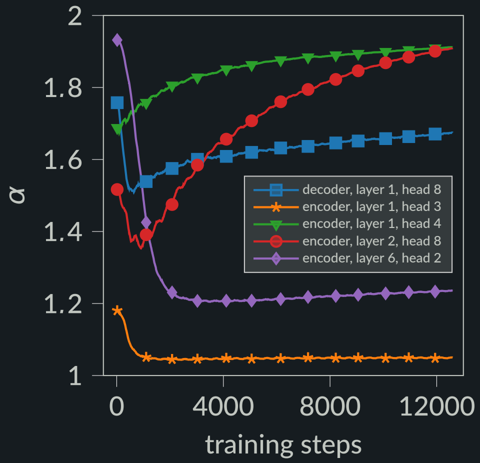
For analysis for other language pairs, see Appendix A.

Learned α

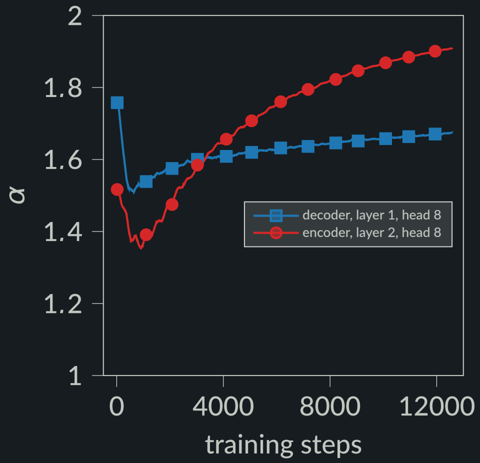


Bimodal for the encoder, mostly unimodal for the decoder.

Trajectories of α During Training

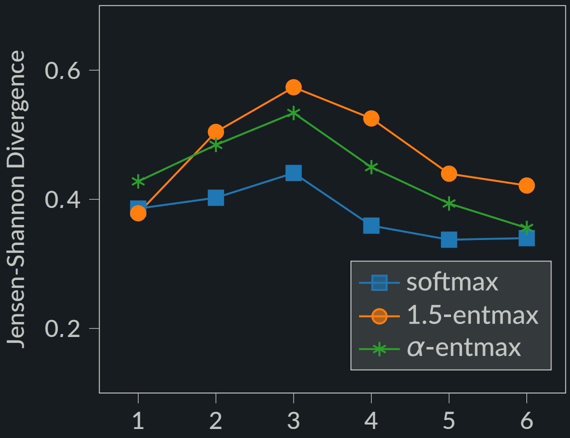


Trajectories of α During Training

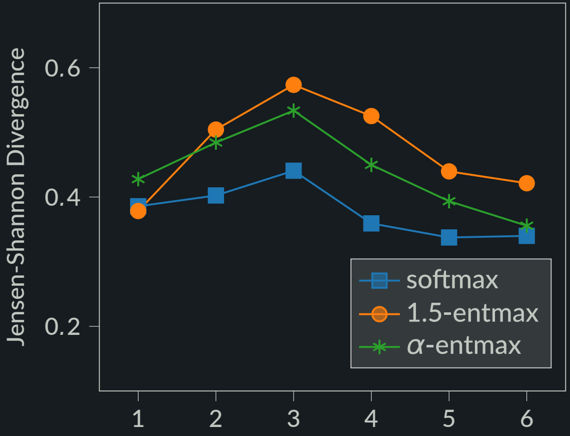


Some heads choose to start dense before becoming sparse.

Head Diversity per Layer

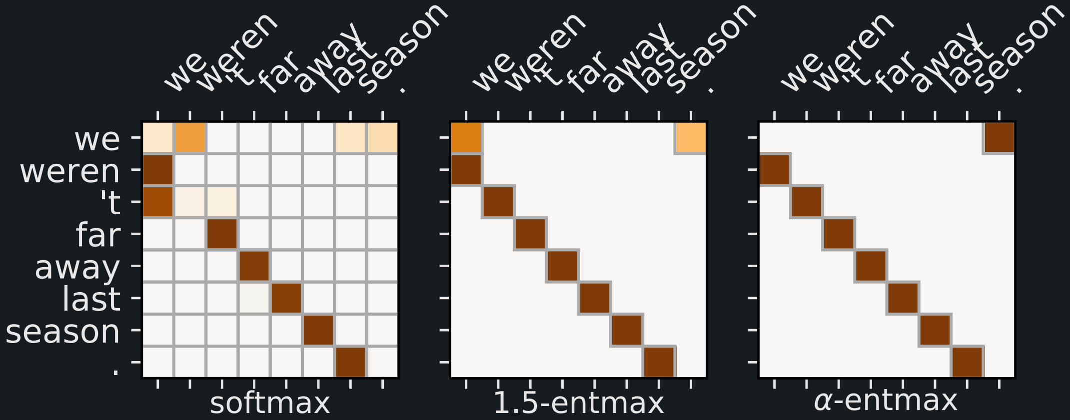


Head Diversity per Layer



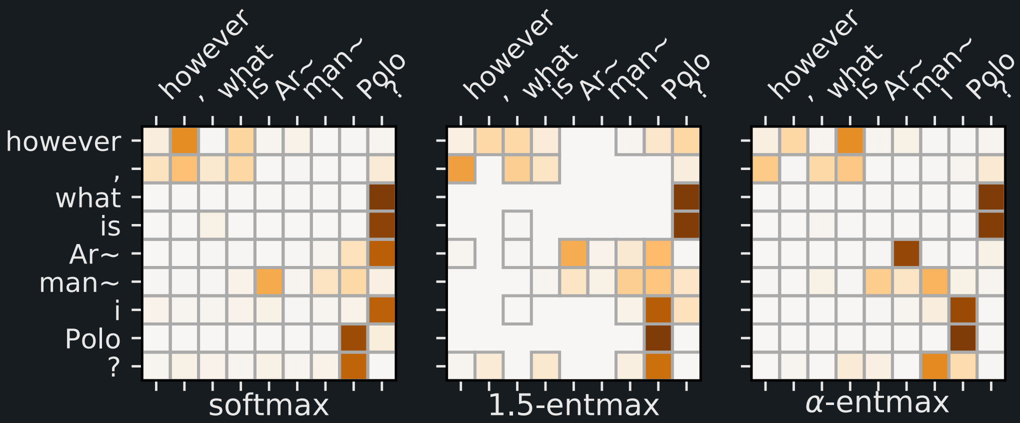
Specialized heads are important as seen in Voita et al. (2019)!

Previous Position Head



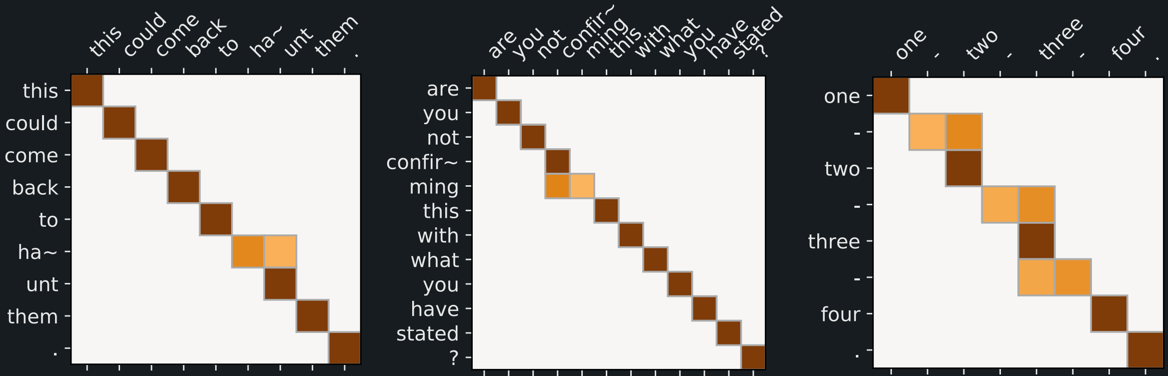
This head role was also found in Voita et al. (2019)! Learned $\alpha = 1.91$.

Interrogation-Detecting Head



Learned $\alpha = 1.05$.

Subword-Merging Head



Learned $\alpha = 1.91$.

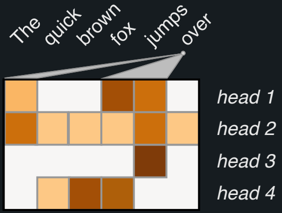
Key Takeaways

Introduce adaptive sparsity
for Transformers via α-entmax with a gradient learnable α.

Key Takeaways

Introduce adaptive sparsity
for Transformers via α-entmax with a gradient learnable α.

adaptive sparsity

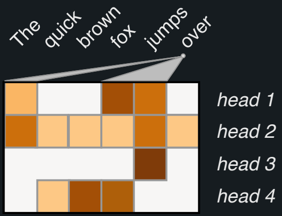


github.com/deep-spin/entmax goncalomcorreia.github.io

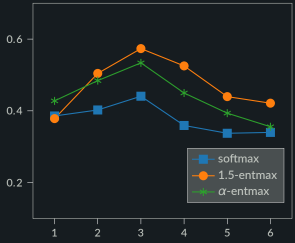
Key Takeaways

Introduce adaptive sparsity
for Transformers via α-entmax with a gradient learnable α.

adaptive sparsity



reduced head redundancy

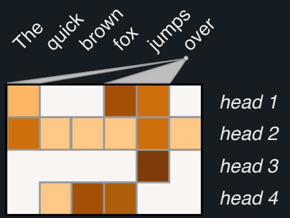


github.com/deep-spin/entmax goncalomcorreia.github.io

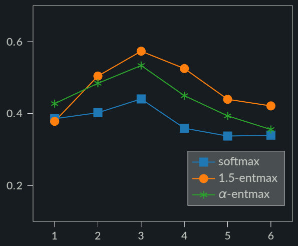
Key Takeaways

Introduce adaptive sparsity
for Transformers via α-entmax with a gradient learnable α.

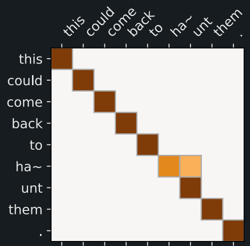
adaptive sparsity



reduced head redundancy



clearer head roles



github.com/deep-spin/entmax goncalomcorreia.github.io

Thank you!

Questions?

`:pip install entmax`

Check `github.com/deep-spin/entmax`

Acknowledgements

This work was supported by the European Research Council (ERC StG DeepSPIN 758969) and by the Fundação para a Ciência e Tecnologia through contract UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal).

References I

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

Child, Rewon, Scott Gray, Alec Radford, and Ilya Sutskever (2019). “Generating long sequences with sparse transformers”. In: *arXiv preprint arXiv:1904.10509*.

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*.

Niculae, Vlad and Mathieu Blondel (2017). “A Regularized Framework for Sparse and Structured Neural Attention”. In: *arXiv preprint arXiv:1705.07704*.

Peters, Ben, Vlad Niculae, and André F. T. Martins (2019). “Sparse Sequence-to-Sequence Models”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Sukhbaatar, Sainbayar, Edouard Grave, Piotr Bojanowski, and Armand Joulin (2019). “Adaptive Attention Span in Transformers”. In: *arXiv preprint arXiv:1905.07799*.

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 (2019). “Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned”. In: *Proc. ACL*.