Data-Informed Global Sparseness in Attention Mechanisms for Deep Neural Networks

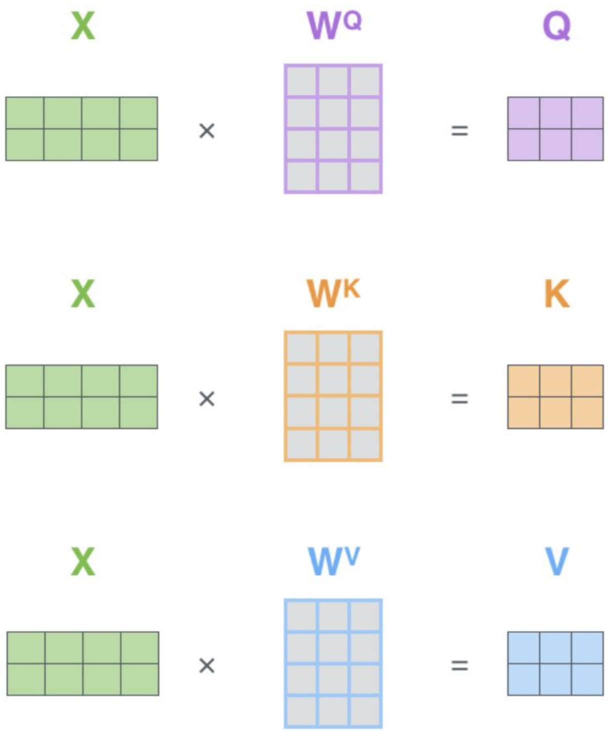
https://arxiv.org/abs/2012.02030

Ileana Rugina*, Rumen Dangovski*, Li Jing, Preslav Nakov, Marin Soljačić

Self-Attention

Self-Attention Mechanisms

word embeddings



$$X \in \mathbb{R}^{N \times h}$$

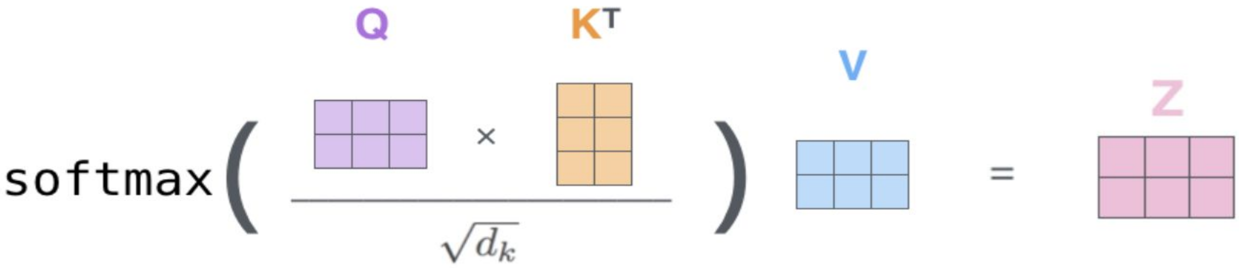
$$Q = XW^Q, Q \in \mathbb{R}^{N \times d}$$

$$K = XW^K, K \in \mathbb{R}^{N \times d}$$

$$V = XW^V, V \in \mathbb{R}^{N \times d}$$

http://jalammar.github.io/illustrated-transformer/

Self-Attention Mechanisms



$$A = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right), A \in \mathbb{R}^{N \times N}$$

$$Z = AV, Z \in \mathbb{R}^{N \times d}$$

http://jalammar.github.io/illustrated-transformer/

Related Work

Kernel Trick Ideas

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) = \phi(Q)\phi(K)^T$$

$$Z = \left(\phi(Q)\phi(K)^T\right)V = \phi(Q)\left(\phi(K)^T V\right)$$

Random Features for Large-Scale Kernel Machines - NeurIPS 2007

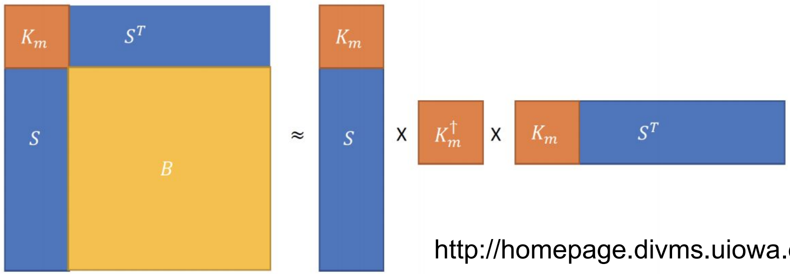
- https://people.eecs.berkeley.edu/~brecht/papers/07.rah.rec.nips.pdf

Random Feature Attention - ICLR 2021

- https://arxiv.org/abs/2103.02143
- https://arxiv.org/abs/2009.14794

Low-Rank Approximations

- Nyström approximation (- also a kernel method) https://arxiv.org/abs/2102.03902



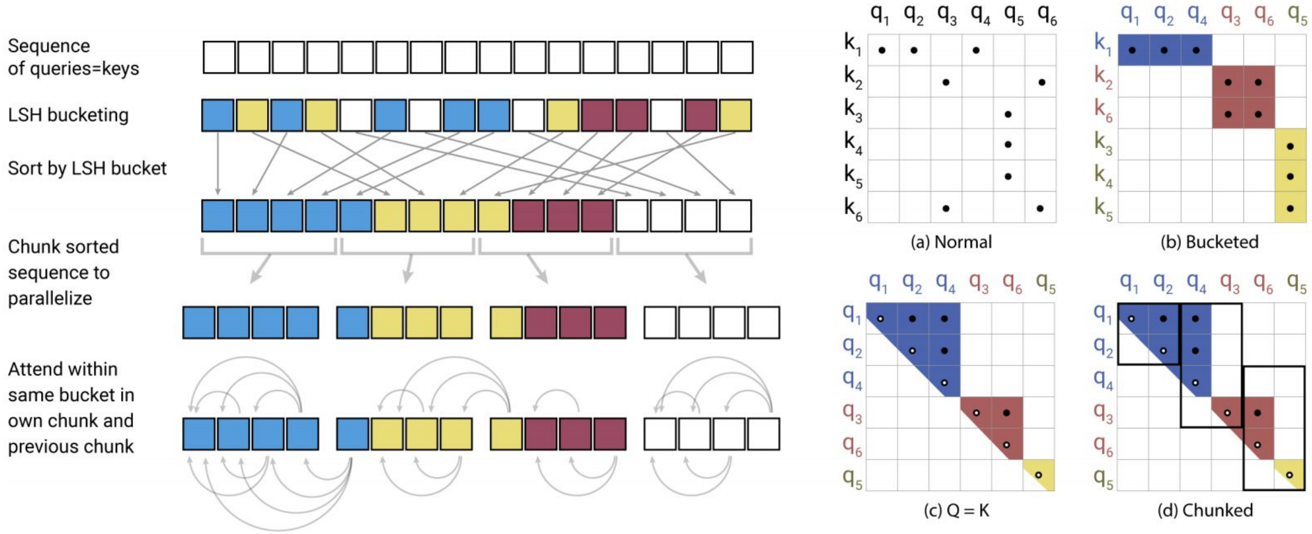
http://homepage.divms.uiowa.edu/~zli79/talks/dual_perspective_nystrom_method.pdf

- JL Lemma: https://arxiv.org/abs/2006.04768
  - dimensionality reduction technique
  - random orthogonal projections of a set on points on smaller subspaces preserves L2 norms
  - project valued and keys

$$V \in \mathbb{R}^{N \times d} \rightarrow V' \in \mathbb{R}^{k \times d}$$

$$K \in \mathbb{R}^{k \times d} \rightarrow K' \in \mathbb{R}^{k \times d}$$

LSH https://arxiv.org/abs/2001.04451



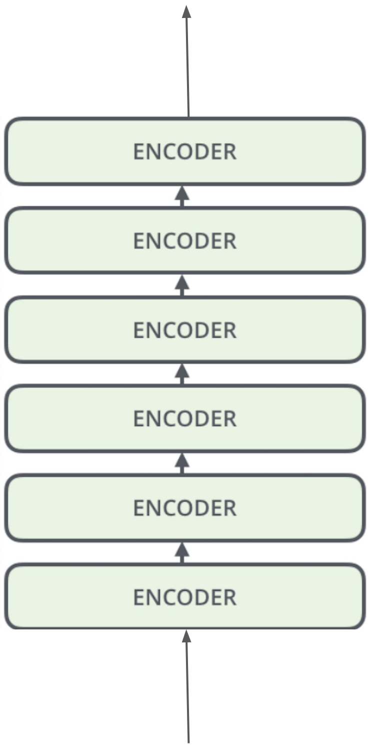
Idea:

1. use locality-based hashing as a clustering mechanism to group related tokens together
2. dot product similarity https://arxiv.org/pdf/1509.02897.pdf

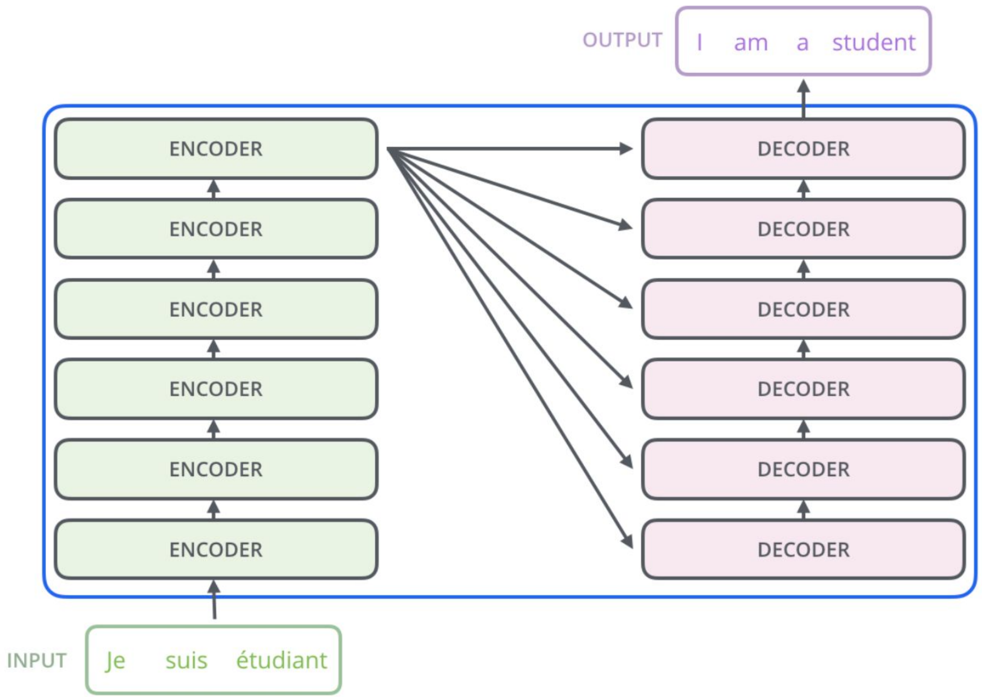
Transformer Models

Two Architectures

Encoder Models

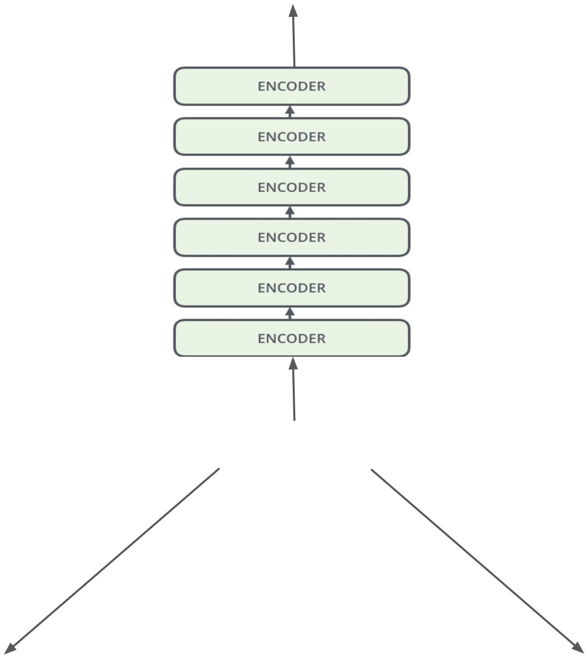


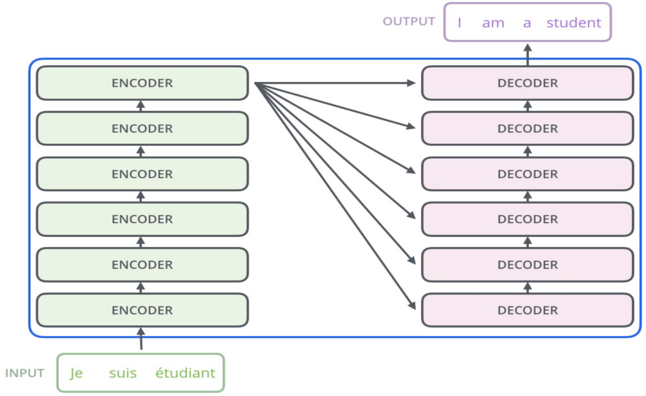
Encoder-Decoder Models



http://jalammar.github.io/illustrated-transformer/

Three Model Types





autoregressive

next token prediction

autoencoder

masked language modeling

seq-2-seq

autoregressive decoder

NLP applications

- Language modeling (next token prediction): autoregressive model
- Language understanding: autoencoder model
- Translation: seq2seq

Method

Attention Pruning Method

- Train a transformer model
- Perform a forward pass through all the train set
- Gather average attention pattern for each attention mechanism
- Learn which tokens correlate on this problem domain

Results

Language Modelling: prune 90% with good performance

p (%)	**Perplexity**
0	24.157
20	24.157
40	24.214
60	24.566
80	25.115
90	26.011

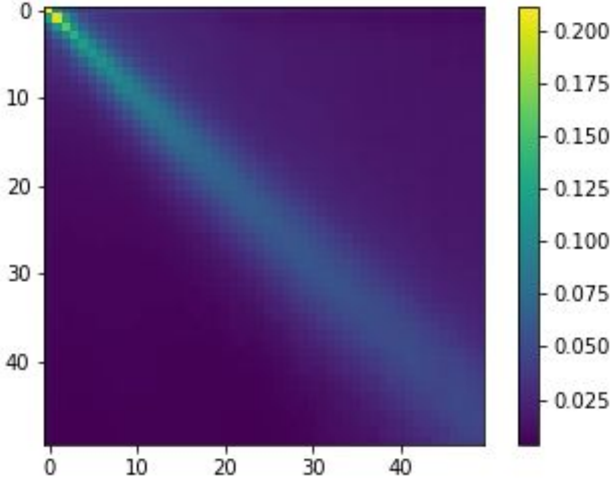
Transformer-XL-base trained on WikiText-103

NLU: prune 60% with good performance

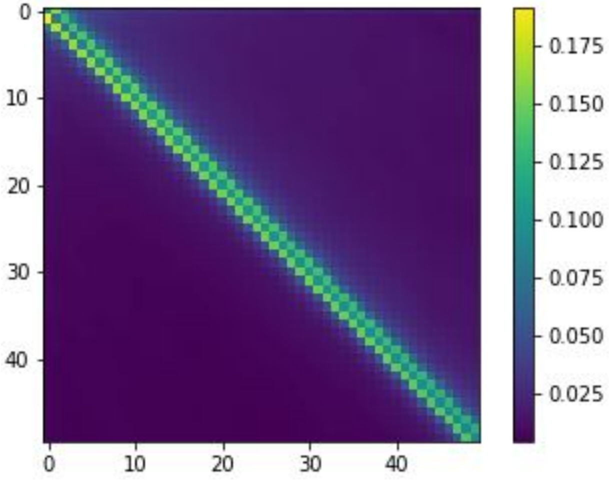
	Accuracy (%)									
p (%)	MNLI-m	MNLI-mm	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Average
0	84.07	83.44	90.91	87.51	65.7	91.97	88.77	57.78	88.39	82.06
20	84.00	83.42	89.72	86.37	64.44	91.11	87.12	55.99	87.34	81.06
40	83.42	83.70	88.76	84.73	62.82	89.91	84.45	52.33	86.48	79.62
50	83.32	82.87	87.81	83.84	62.09	89.05	83.08	48.56	85.28	78.43
60	82.54	81.98	87.19	83.10	61.37	88.82	82.04	45.05	81.70	77.09
80	79.29	78.64	82.37	81.32	57.22	84.52	78.57	34.80	65.89	71.40
90	75.40	75.23	77.23	77.45	49.46	80.56	79.41	20.28	51.39	65.16

BERT base fine tuned on GLUE

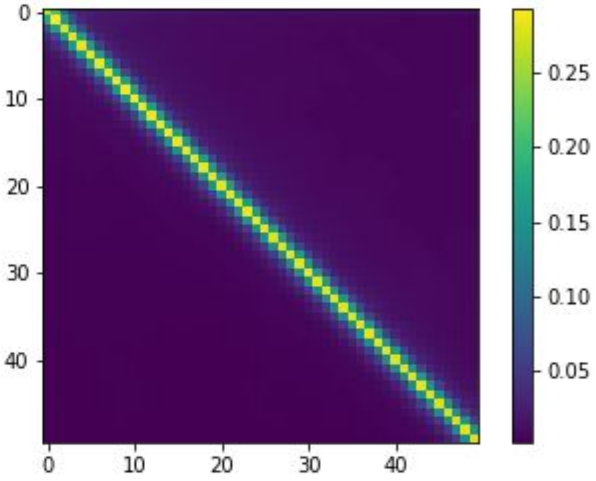
Translation Attention Patterns



encoder-decoder



encoder self-attention



encoder entmax self-attention

Translation Performance: Cross-Attention is more brittle

p (%)	**Self-Enc**	**Self-Dec**	**Cross**
0		34.94	
20	34.53	34.94	33.50
40	33.70	34.94	24.38
50	33.56	35.08	22.60
60	33.68	34.91	15.08
80	33.67	34.90	6.39

IWSLT14 de-en

Translation: prune* 60-80% with good performance

p (%)	**IWSLT14 de-en**	**WMT17 en-de**
0	34.94	26.73
20	34.92	26.21
40	33.70	26.60
50	33.68	26.19
60	33.64	26.44
80	33.81	21.88

* prune only self-attention mechanisms

SQuAD Results https://www.deepspeed.ai/

Kernel	Pruning	Efficiency	
	p (%)	Time (s)	Memory (GB)
CUDA	0	95.80	6.24
Triton	0	95.41	6.85
Triton	90	86.44 (↓9.4%)	5.00 (↓27%)

Performance:

- p = 0 yields 81.02 Exact and 88.63 F1
- p = 90 yields 79.62 Exact and 87.32 F1

Thank you!