Tencent AI Lab

Towards Better Transformer For Long-range Sequence Modeling

Haofei Yu 03/31/2022

Outline

Tencent AI Lab

1. Background: Logic behind Efficient Transformer

2. RoadMap: Research Lines in Efficient Transformer

3. Example Works: Methods for Building Long-term Memory

4. Challenges: Need for Benchmark and New Evaluation Metric

5. Future Works: Ensemble Methods and Sparse Modeling

Background

Tencent AI Lab

Why We Need to Research Long-range Sequence Modeling?

Long Sequence Modeling Scenario is common:

1. Wikitext-103 / PG-19 / Enwik8 / Arxiv / Github are common datasets for Long-range Language Modeling
2. Besides Pure Text Language Modeling, Music / Speech / Video / Image / Document-level Machine Translation can be considered as Long-range Sequence Modeling Task

Why Vanilla Transformer cannot do Long-range Sequence Modeling?

1. Memory Cost is high when input sequence length is too long
2. Computation Cost is high when input sequence length is too long
3. Conflict between **Long-term Dependency** and **Memory/Computation Cost**

Natural Ideas Based on the Conflict:

1. Use external independent memory to augment Language Modeling to gain long-term dependency and focus on relatively short span
2. Utilize internal model sparsity to reduce memory cost and focus on long span
3. Replace original attention module with linear-time attention mechanisms and focus on long span

RoadMap

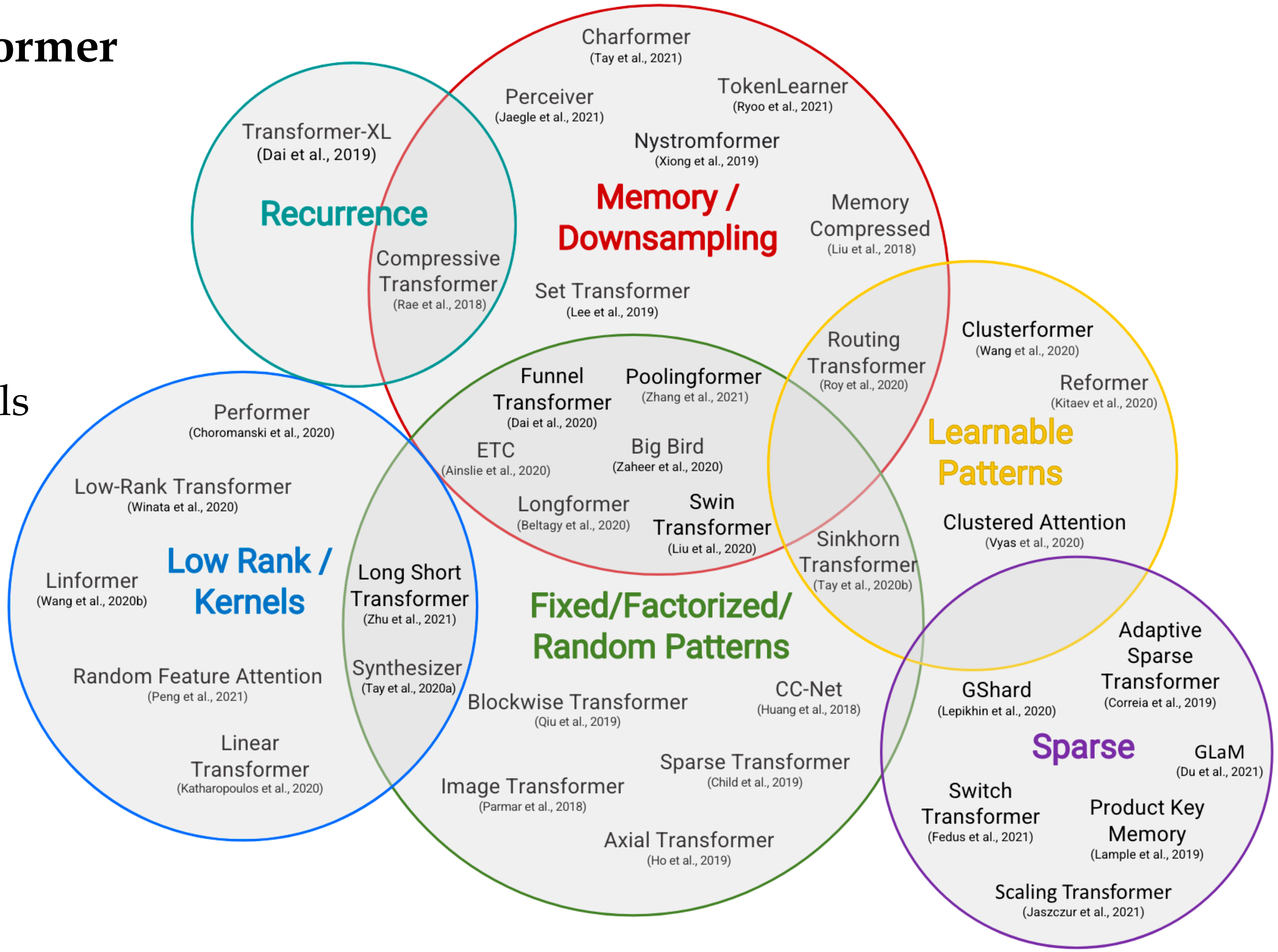
Tencent AI Lab

Methods RoadMap in Efficient Transformer

Today's Topic focus on Long-Term Memory Modeling. It can be considered as a combination of Recurrence, Memory and Learnable Patterns

1.Adaptive Semiparametric Language Models (from DeepMind) lies in the crossing point between **recurrence and memory**

2. Not All Memories are Created Equal: Learning to Forget by Expiring (from FAIR) lies in the crossing point between **memory and learnable patterns**



Example Works

Tencent AI Lab

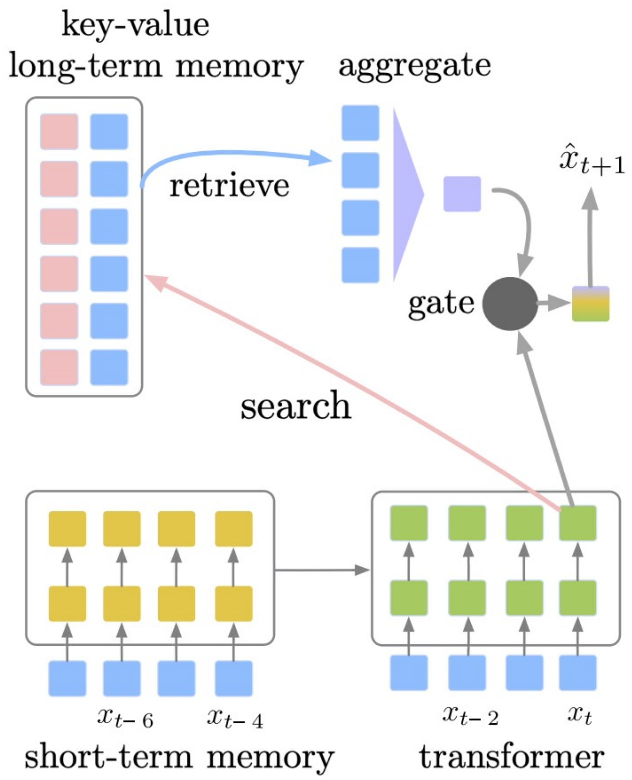
1. **Adaptive Semiparametric Language Models (DeepMind)**

What is the motivation behind this paper?
To jointly use Long-term Memory and Short-term Memory with the gate mechanism
(Approximately the combination of kNN-LM and Transformer-XL)

How to model long-term memory in this paper?
Equip Models with external independent K-V offline database

What is the memory mechanism in this paper?
Combine Short-term and Long-term Memory with Gate Mechanism

Short-term Memory: have the same setting as Transformer-XL
Long-term Memory: Key-Value Database
(Key is a vector representation for previous condition context,
Value is a vector representation for predicted target token,
Build based on pretrained encoder like BERT)
Gate Mechanism: Allow model to use Long-term Memory for Strong Prediction
Allow model to use Short-term Memory for Easy Prediction

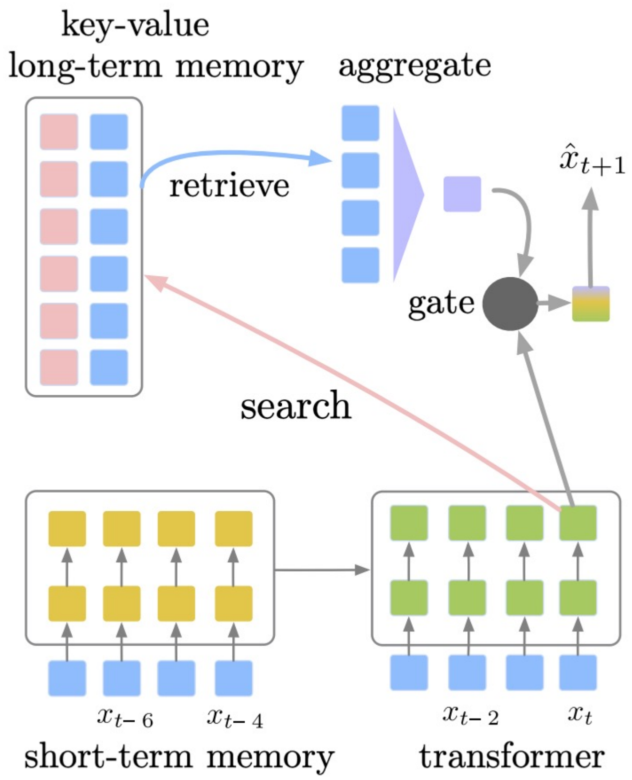


Example Works

1. **Adaptive Semiparametric Language Models (DeepMind)**

How to use long-term memory in training and inference stage?
1.Pre-compute K-V long-term memory offline
2.Do kNN search in K-V Database
3.Use Context-related Gate Mechanism to token-level adaptively decide use short-term or long-term memory

What is its advantage compared with kNN-LM?
1.kNN-LM's hyper-parameter is tuned on dev set only, SPALM is tuned by training
2.kNN-LM's hyper-parameter is fixed for each token, SPALM is adaptive



Example Works

1. **Adaptive Semiparametric Language Models (DeepMind)**

What Experiment can confirm its performance?

Test on WikiText-103 (word-level).

	Model	# Params	Dev	Test
	Transformer-XL[a]	257M	–	18.3
	Adaptive Input[b]	247M	18.0	18.7
	Compressive[c]	257M	16.0	17.1
	kNN-LM[d]	247M	16.1	16.1
$M = 512$	Transformer	142M	20.8	21.8
	Transformer-XL	142M	18.7	19.6
	kNN-LM	142M	18.1	18.5
	SpaLM	142M	17.9	18.8
	$\hookrightarrow$ + kNN		17.6	18.0
$M = 3072$	Transformer-XL	142M	18.3	19.1
	kNN-LM	142M	17.7	18.0
	SpaLM	142M	17.4	18.3
	$\hookrightarrow$ + kNN		17.2	**17.6**

Test on WMT Dataset(word-level).

Model	# Params	Dev	Test
Transformer	148M	16.0	16.3
Transformer-XL	148M	15.6	15.5
kNN-LM	148M	13.1	15.2
SpaLM	148M	13.0	**14.0**

Test on EnWiki8(char-level)

Model	# Params	Dev	Test
18L Transformer-XL[a]	88M	–	1.03
24L Transformer-XL[a]	277M	–	0.99
Longformer[c]	102M	–	0.99
Compressive[d]	277M	–	0.97
Transformer	104M	1.07	1.05
Transformer-XL	104M	1.03	1.01
kNN-LM	104M	1.04	1.02
SpaLM	104M	1.02	**1.00**

Example Works

Tencent AI Lab

1. **Adaptive Semiparametric Language Models (DeepMind)**

Any Extra Findings?

Test on WikiText-103 (word-level).

	Model	# Params	Dev	Test
	Transformer-XL[a]	257M	–	18.3
	Adaptive Input[b]	247M	18.0	18.7
	Compressive[c]	257M	16.0	17.1
	kNN-LM[d]	247M	16.1	16.1
$M = 512$	Transformer	142M	20.8	21.8
	Transformer-XL	142M	18.7	19.6
	kNN-LM	142M	18.1	18.5
	SPALM	142M	17.9	18.8
	$\hookrightarrow$ + kNN		17.6	18.0
$M = 3072$	Transformer-XL	142M	18.3	19.1
	kNN-LM	142M	17.7	18.0
	SPALM	142M	17.4	18.3
	$\hookrightarrow$ + kNN		17.2	**17.6**

kNN-LM and SPALM model have complementary function: incorporating long-term memory during training and incorporating probabilities during testing have additional effects

Example Works

Tencent AI Lab

1. **Adaptive Semiparametric Language Models (DeepMind)**

What is the function of Long-Term Memory?

Long-Term Memory helps model to generate common phrases and named entities (that exist in the training set), especially when they are encountered for the first time and have not appeared in the extended context

... Several companies have pulled their advertising from the TV show following the revelations ...
... Liberal Democrat leader Jo Swinson has said she would work with Donald Trump in government as ...
... Additionally , the airline has purchased six Boeing 787 - 9 Dream liner aircraft that are scheduled ...

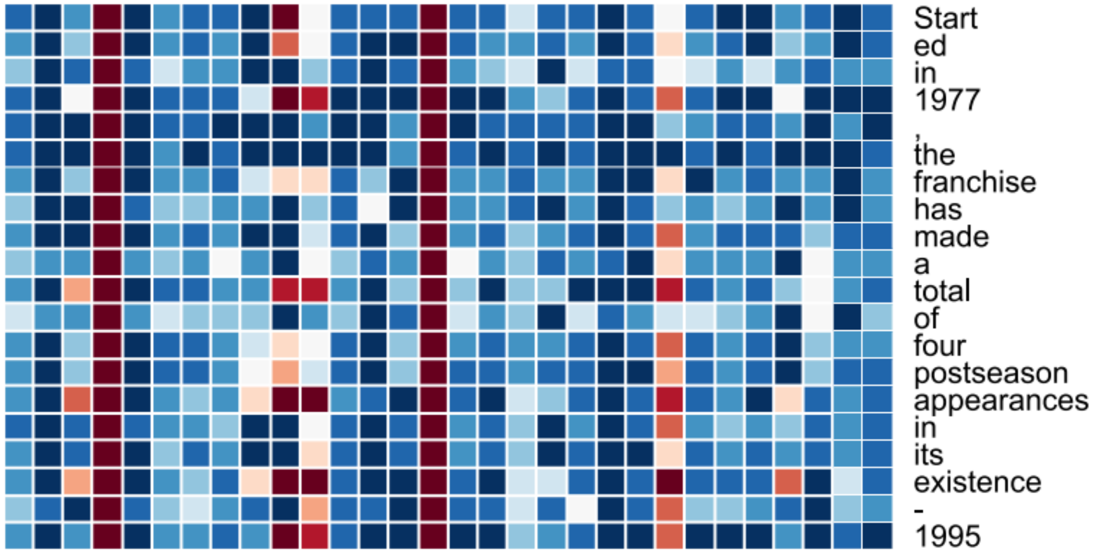
Figure 4: Three example sequences from the WMT test set. We highlight words where both p_{TXL} and p_{SPALM} are larger than $p_{\text{transformer}} + 0.1$ in green and $p_{\text{SPALM}} > p_{\text{TXL}} + 0.1$ in blue. See §5.2 for details.

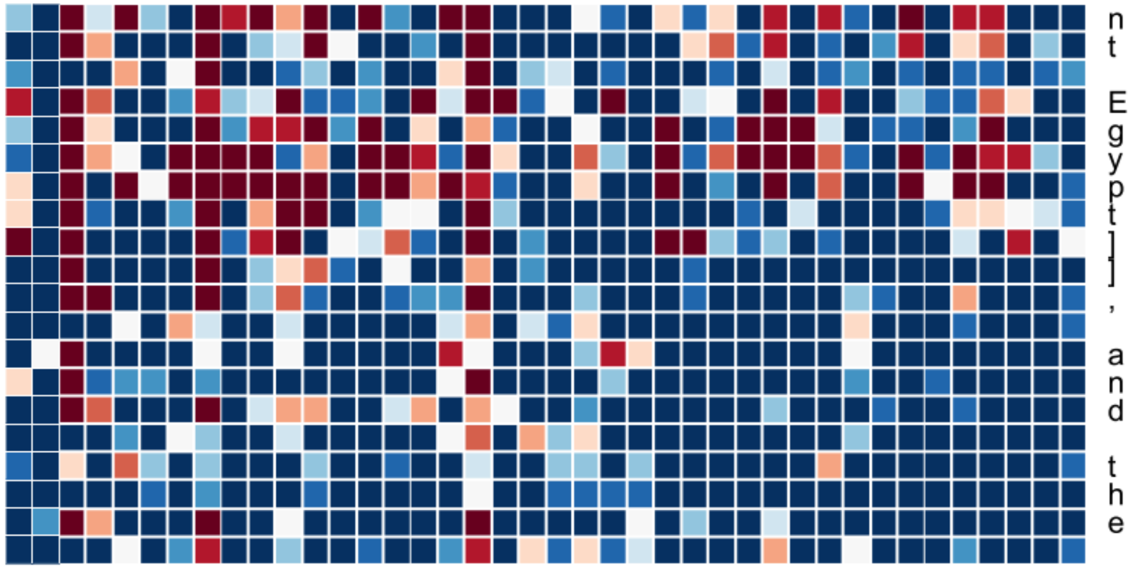
Example Works

1. **Adaptive Semiparametric Language Models (DeepMind)**

What is exactly the function of the Gate Mechanism? Is it really working to use long-term memory?

Gate Mechanism helps the model to achieve adaptive token-level long-term/short-term switch
Some Dimensions and some tokens are proved to be strongly effected by long-term memory





Example Works

Tencent AI Lab

1. **Adaptive Semiparametric Language Models (DeepMind)**

What is the drawbacks to use external independent memory to build long-term memory?

Needs a lot of extra offline computation!
Takes 6–8 hours to obtain neighbors for WikiText-103 and enwik8 with 1,000 CPUs and 18 hours for WMT with 9,000 CPUs

Is the kNN neighbor the more the better?

Nope. Too many neighbors can bring possible noise and will harm the performance.

# NNs	Perplexity
1	18.0
2	18.0
4	17.9
8	18.2
16	18.4

Table 5: SPALM perplexity on the WikiText-103 development set with different numbers of neighbors.

Example Works

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

What is the motivation of this paper?

Scale Transformer-XL to VERY LONG sequence

Treat the activation time of hidden states as learnable pattern

The whole mechanism is designed for **scalability**

What is the definition of Expire Span?

The Expire Span is an integer for each hidden states in each layer.

It ranges from 0 to sequence total span.

Its function is to decide the living period for each hidden states in the attention mechanism.

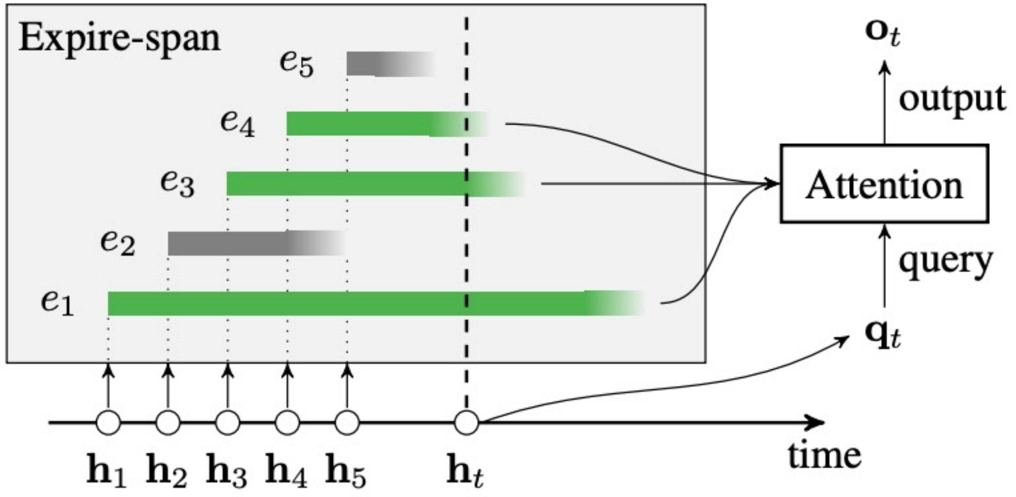


Figure 1. **Expire-Span.** For every memory $\mathbf{h}_i$, we compute an EXPIRE-SPAN e_i that determines how long it should stay in memory. Here, memories $\mathbf{h}_2$, $\mathbf{h}_5$ are already expired at time t, so the query $\mathbf{q}_t$ can only access $\{\mathbf{h}_1, \mathbf{h}_3, \mathbf{h}_4\}$ in self-attention.

Example Works

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

How to Train the Expire Span? (Key Design)

Overall, it is just an additional loss for attention mechanism.
In detail, the hardest point is to design gradient for attention mask as a function of expire span. And it is done using monotonically decreasing function related to time for soft masking.

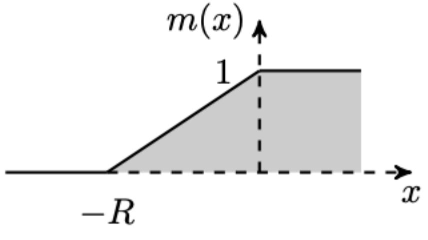


Figure 2. Soft Mask

$$e_i = L\sigma(\mathbf{w}^\top \mathbf{h}_i + b)$$

$$r_{ti} = e_i - (t - i)$$

$$m_{ti} = \max(0, \min(1, 1 + r_{ti}/R))$$

$$a'_{ti} = \frac{m_{ti} a_{ti}}{\sum_j m_{tj} a_{tj}}, \quad \mathbf{o}_t = \sum_i a'_{ti} \mathbf{v}_i.$$

$$\begin{aligned}
\frac{1}{T}\sum_t |C_t| &= \frac{1}{T}\sum_t \sum_{i<t} \mathbf{1}_{m_{ti}>0} \\
&= \frac{1}{T}\sum_i \left(R + \sum_{t>i} \mathbf{1}_{r_{ti}>0}\right) \\
&= \frac{1}{T}\sum_i \left(R + \sum_{t>i} \mathbf{1}_{e_i>t-i}\right) \\
&= R - 1 + \frac{1}{T}\sum_i \lfloor e_i \rfloor
\end{aligned}$$

$$L_{\text{total}} = L_{\text{task}} + \alpha \sum_i e_i / T$$

Example Works

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

What feature are Expire-Span want to design experiment to prove?

1. This method can gain long-term memory far away
2. This method can scale to VERG LONG sequence with efficient memory cost

Example Works

2. Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)

How to prove this design really capture long-term dependency?

1) Memorize One Piece of Key Information (Corridor Task)
2) Memorize Sequence Information (Portal Task)
3) Memorize Sequence Information with Distractor (Instruction Task)

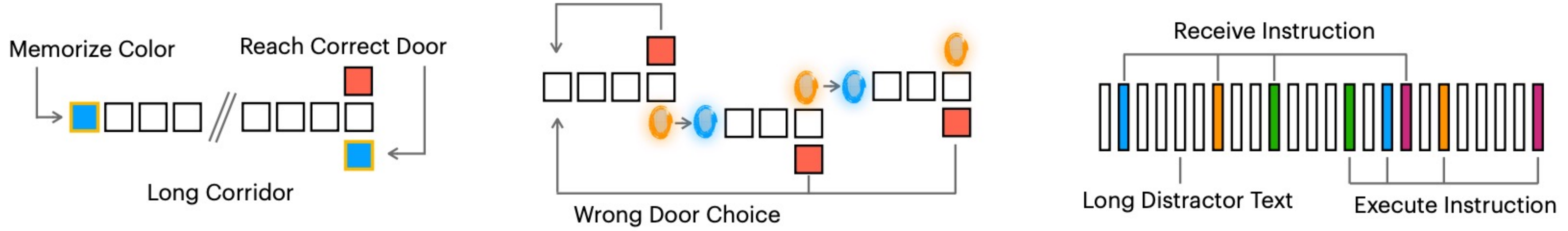


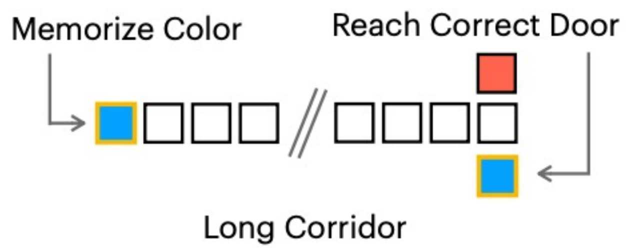
Figure 3. **Corridor Task (left)**- Agents must memorize the color of an object and walk through the door of the corresponding color at the end of a long corridor. **Portal Task (middle)**- An agent must trial-and-error to memorize the sequence of doors. **Instruction Task (right)**- A model must recognize instructions, memorize them, and execute when at the correct location.

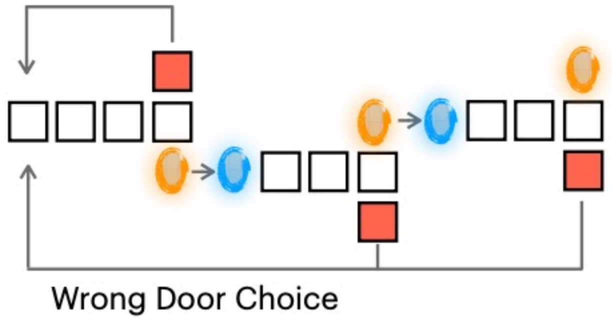
Example Works

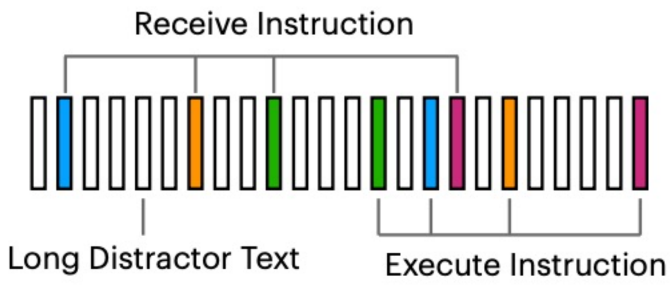
Tencent AI Lab

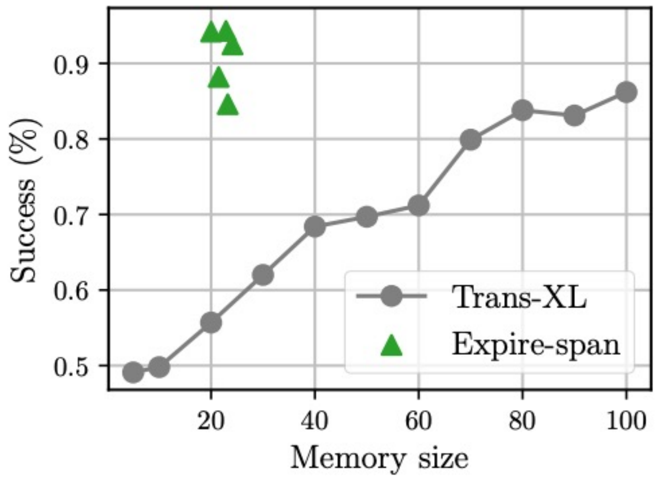
2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

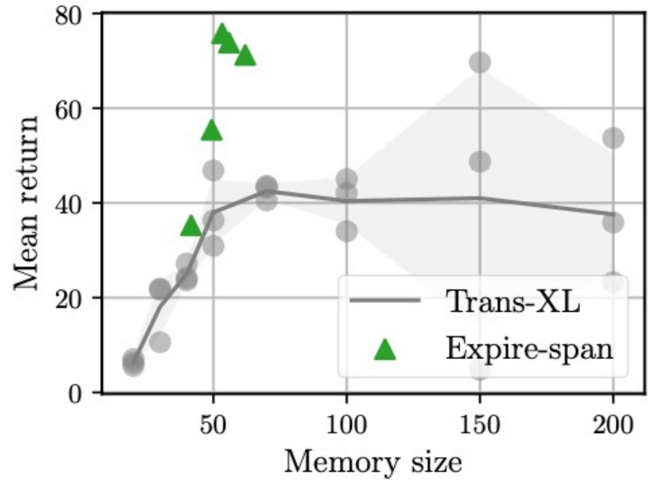
How to prove this design really capture long-term dependency with efficient memory?

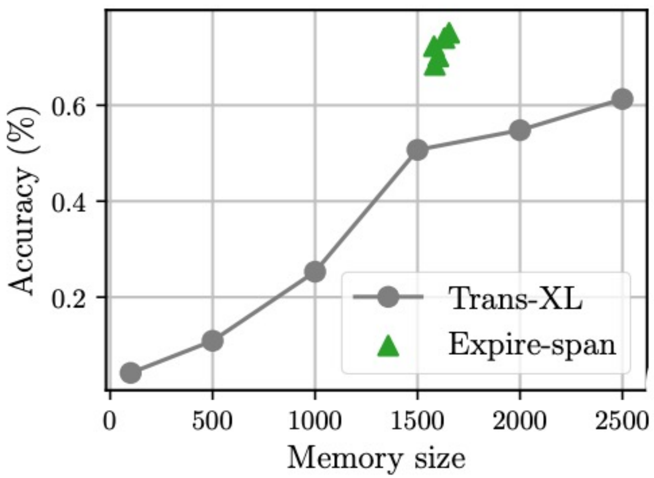












Example Works

Tencent AI Lab

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

How to prove this design really can scale to VERY LONG sequence with long-term dependency?

Copy task on enwik8

Model	Maximum span	Accuracy (%)
Transformer-XL	2k	26.7
EXPIRE-SPAN	16k	29.4
EXPIRE-SPAN	128k	**52.1**

Table 1. **Copy Task.** We report accuracy on the test set.

Performance on enwik8

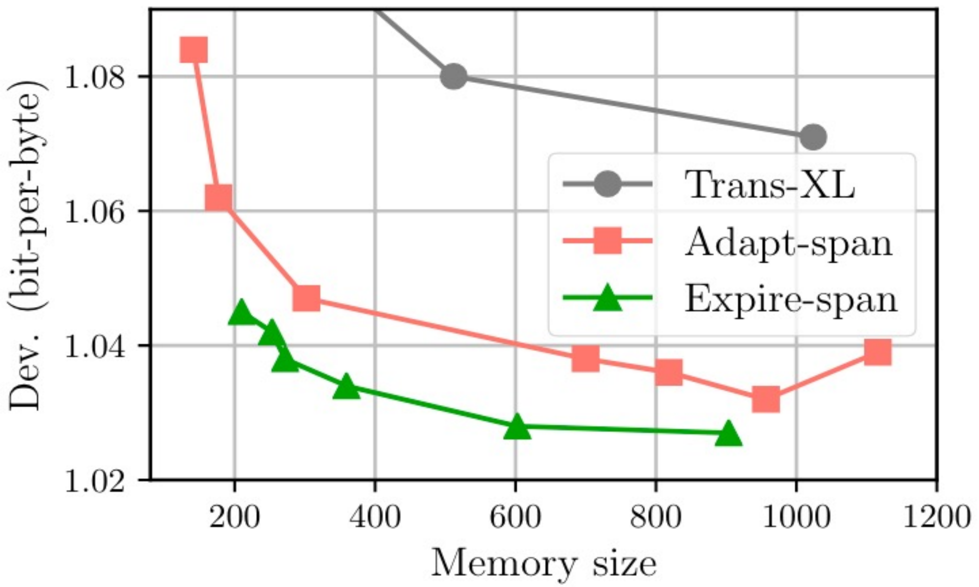
Model	Params	Test
Small models		
Trans-XL 12L (Dai et al., 2019)	41M	1.06
Adapt-Span 12L (Sukhbaatar et al., 2019a)	39M	1.02
Our Trans-XL 12L baseline	38M	1.06
EXPIRE-SPAN 12L	38M	**0.99**
Trans-XL 24L (Dai et al., 2019)	277M	0.99
Sparse Trans. (Child et al., 2019)	95M	0.99
Adapt-Span 24L (Sukhbaatar et al., 2019a)	209M	0.98
All-Attention (Sukhbaatar et al., 2019b)	114M	0.98
Compressive Trans. (Rae et al., 2020)	277M	0.97
Routing Trans. (Roy et al., 2020)	-	0.99
Feedback Trans. (Fan et al., 2020b)	77M	0.96
EXPIRE-SPAN 24L	208M	**0.95**

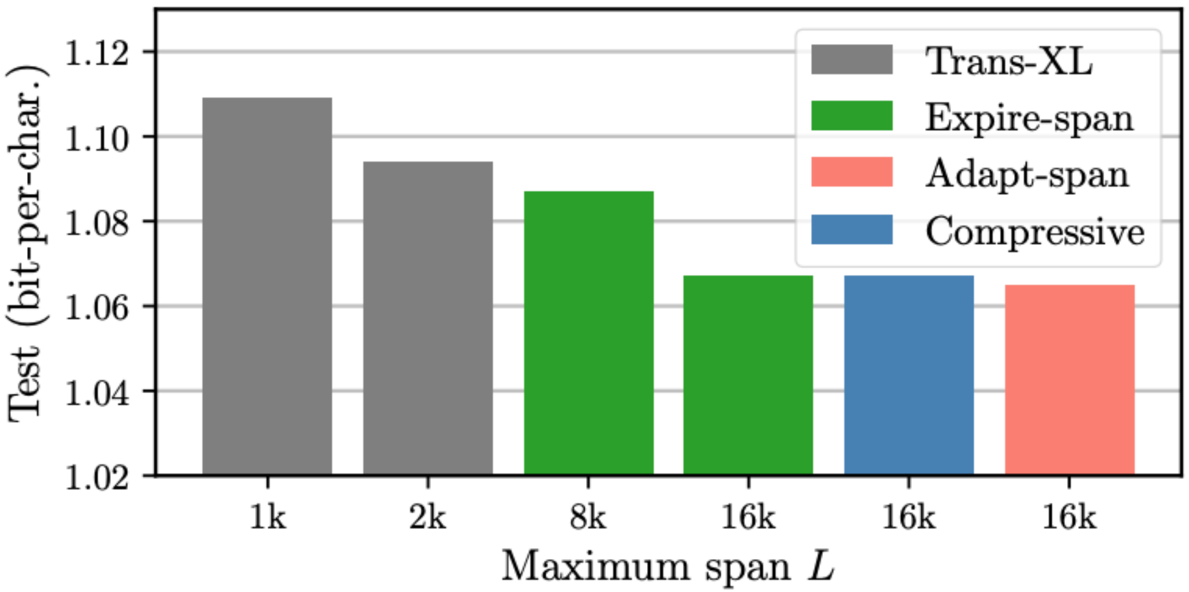
Example Works

Tencent AI Lab

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

How to prove this design have efficient memory cost?





Example Works

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

How to prove this design have efficient memory cost?

	Model	Performance	GPU Memory (GB)	Time/Batch (ms)
Enwik8	Transformer-XL	1.06 bpb	27	649
	Compressive Transformer	1.05 bpb	21	838
	Adaptive-Span	1.04 bpb	20	483
	EXPIRE-SPAN	**1.03** bpb	**15**	**408**
Char-level PG-19	Compressive Transformer	1.07 bpc	17	753
	Adaptive-Span	1.07 bpc	**13**	427
	EXPIRE-SPAN	1.07 bpc	15	**388**
Object Collision	Compressive Transformer	63.8% Error	**12**	327
	Adaptive-Span	59.8% Error	17	365
	EXPIRE-SPAN	**52.2%** Error	**12**	**130**

Example Works

Tencent AI Lab

2. **Not All Memories are Created Equal: Learning to Forget by Expiring (FAIR)**

How to prove expire-span performance improvements come from long-term information?

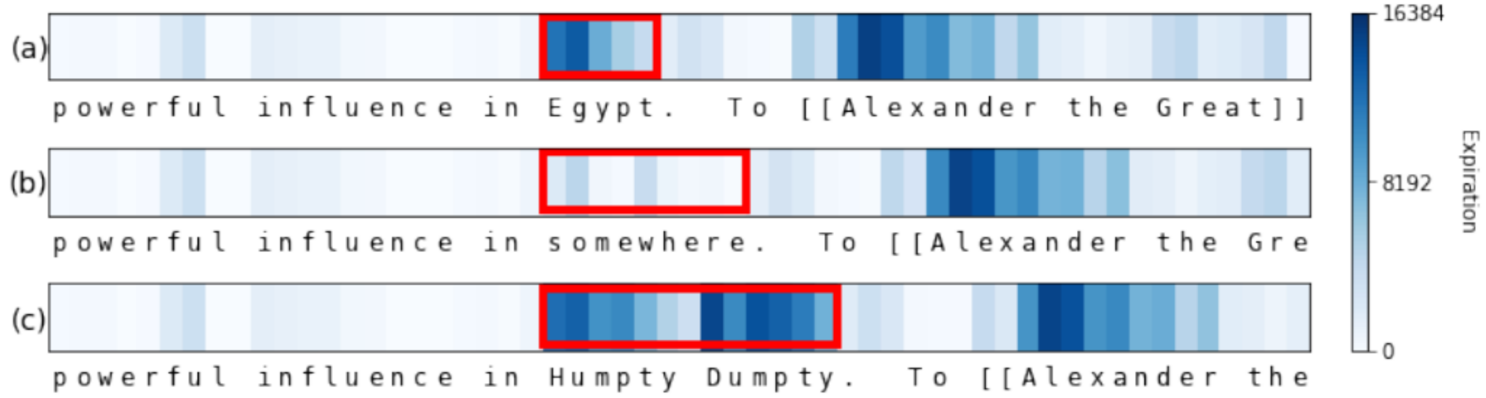


Figure 8. **Expiration in EXPIRE-SPAN on Enwik8**. In **(a)**, the model strongly memorizes two areas, "Egypt" and "Alexander". In **(b)**, if we replace "Egypt" with "somewhere", then it's forgotten fast. In **(c)**, we insert "Humpty Dumpty" and the model retains these rare words in memory.

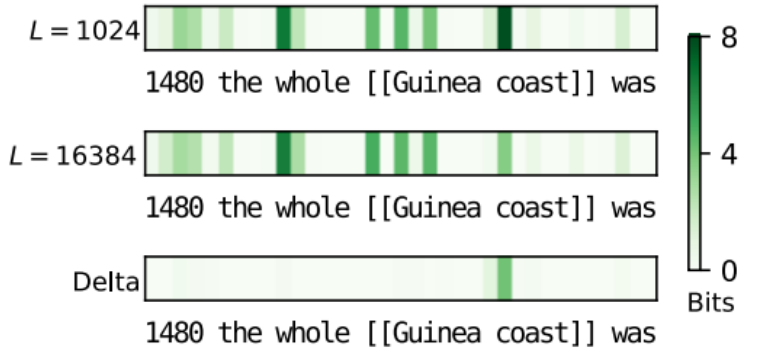


Figure 9. **Accuracy Needs Memory.** As the maximum span is artificially decreased at inference time from 16k to only 1k, the predic tion is less accurate.

Challenges

Tencent AI Lab

C1. Build A Benchmark for Decoder-only Efficient Transformer with Standard Tasks
Existing Work: Long range arena: A benchmark for efficient transformers Build a Benchmark for Encoder-only Efficient Transformer, but without Decoder-Only Benchmark for Generation
It defines a suite of tasks consisting of sequences ranging from 1K to 16K tokens including multimedia

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	FAIL	54.39
Local Attention	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans.	17.07	63.58	**59.59**	**44.24**	71.71	FAIL	51.24
Longformer	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer	**37.27**	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird	36.05	64.02	59.29	40.83	74.87	FAIL	**55.01**
Linear Trans.	16.13	**65.90**	53.09	42.34	75.30	FAIL	50.55
Performer	18.01	65.40	53.82	42.77	**77.05**	FAIL	51.41
Task Avg (Std)	29 (9.7)	61 (4.6)	55 (2.6)	41 (1.8)	72 (3.7)	FAIL	52 (2.4)

Challenges

Tencent AI Lab

C2.Find a Different Evaluation Metric For Long-range Sequence Generation to evaluate

Existing Metrics only include Perplexity and Bit-Per-Character to evaluate Language Model
Long-range Sequence Generation may face structure problems except fluency.

In the paper *Recipes for building an open-domain chatbot* from FAIR, authors mentioned that:

> etc. While several recent works have extended neural architectures to possess longer contexts (Dai et al., 2019; Rae et al., 2020; Kitaev et al., 2020; Beltagy et al., 2020), we have neither implemented those, nor do we believe the current evaluation setup is the right one for measuring their success.

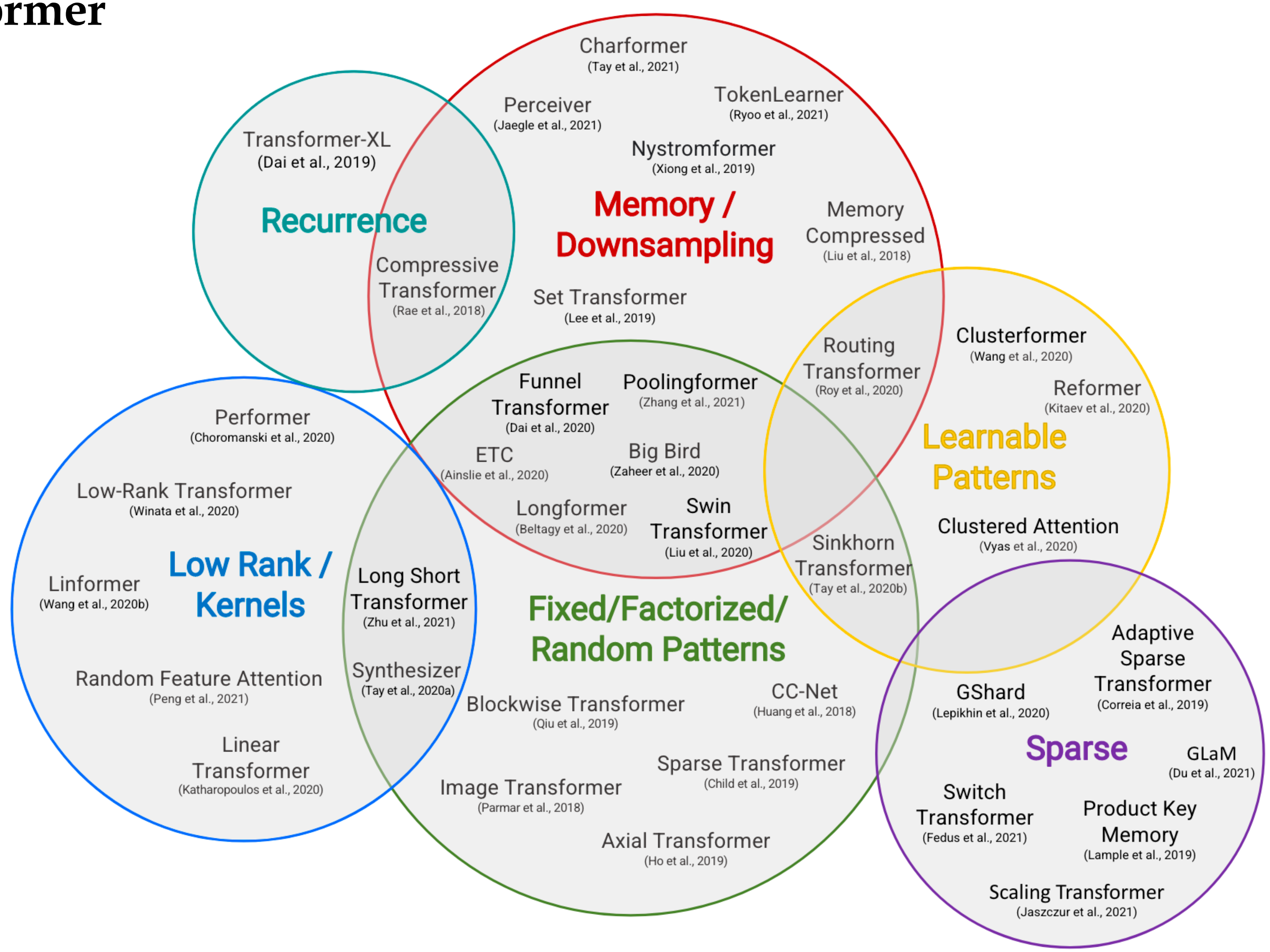
Future Works

Tencent AI Lab

Review the RoadMap for Efficient Transformer

1. To ensemble different forms of methods concerning with memory modeling to achive a more flexible architecture (Adaptive Semiparametric Language Models)

2. In the RoadMap, one separate but novel part is related to **Sparse** (as a new independent line of work)
Sparse models like MoE typically achieve a high parameter to FLOP ratio by sparsely activating a subset of parameters or activations.



Thanks for Listening