SHAPE: Shifted Absolute Position Embedding for Transformers

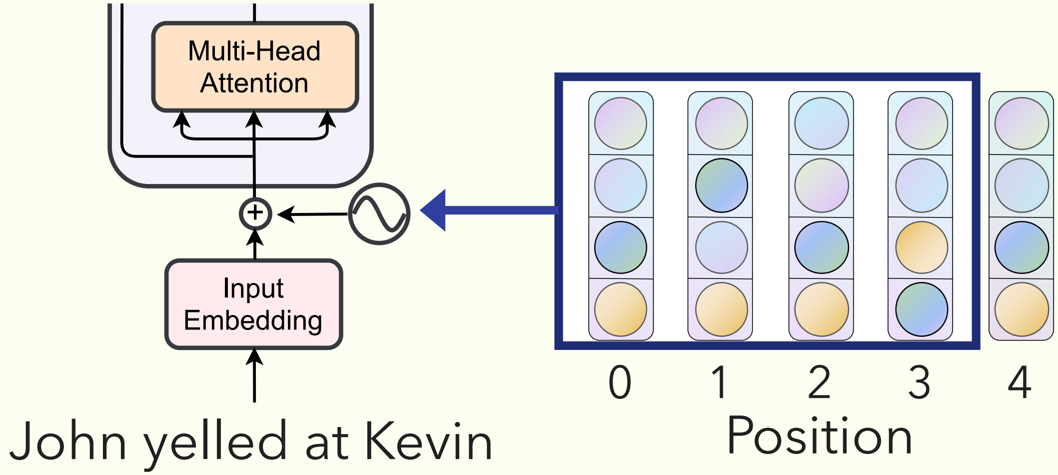
Shun Kiyono[1,2], Sosuke Kobayashi[2,3], Jun Suzuki[2,1], Kentaro Inui[2,1]

[1]RIKEN [2]Tohoku University [3]Preferred Networks, Inc.

Absolute Position Embedding (APE)

Code

```
pos_embed(pos_idx)
```

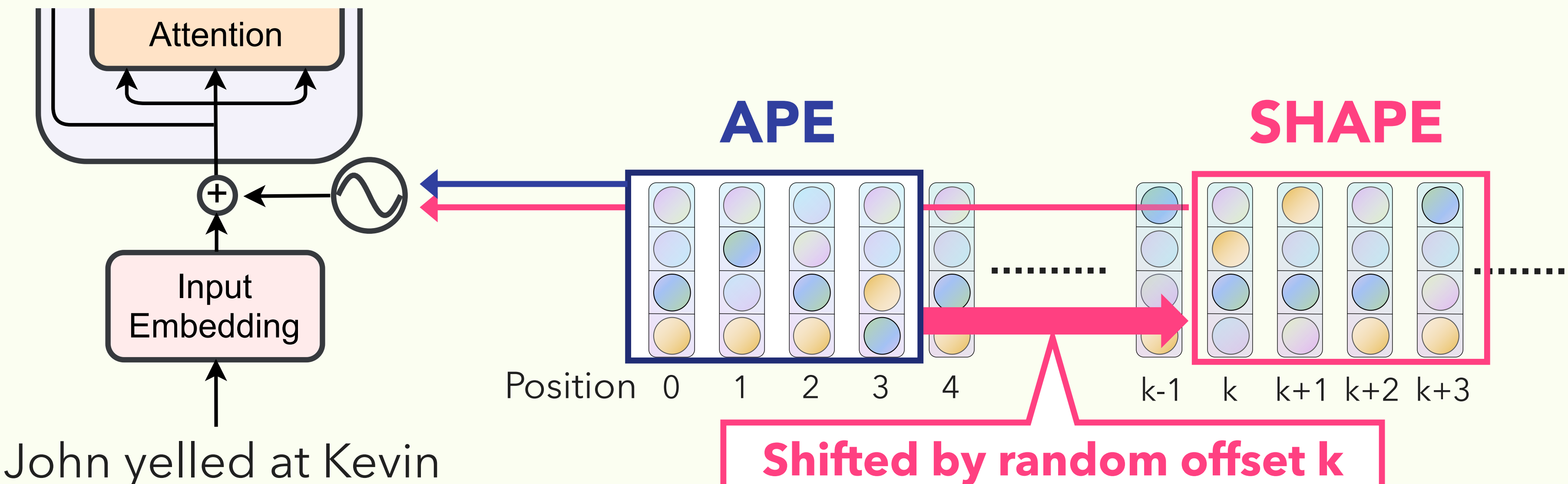


SHAPE is "Shifted" APE

Adding a single line of code is all you need

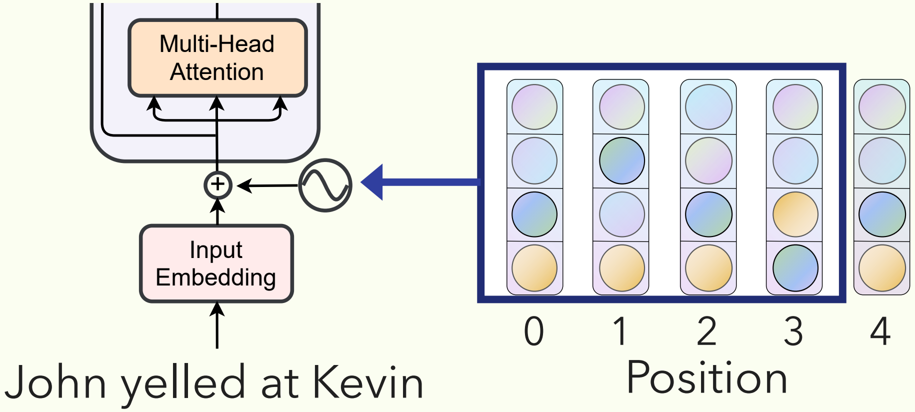
Code

```
pos_idx += self.training * torch.randint(0, K)
pos_embed(pos_idx)
```



Absolute Position Embedding (APE)

- Represent each position with unique embedding
  - e.g., sinusoidal wave [Vaswani+2017]
- 😀 Simple, fast, and easy to implement
- 😰 Poor performance on **unseen lengths**
  - i.e., APE is bad at extrapolattion



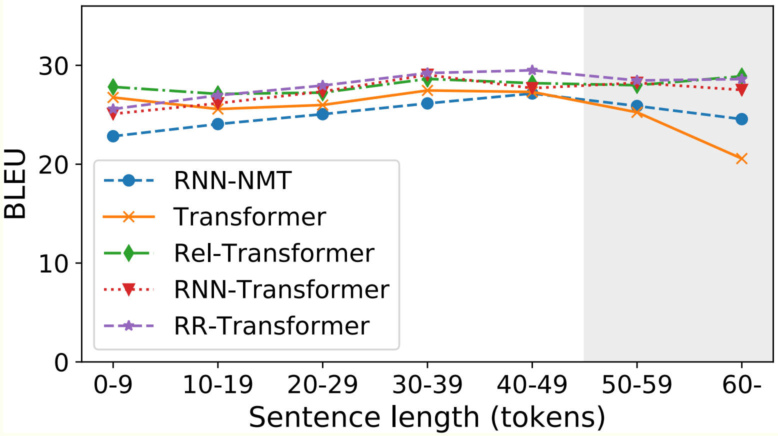
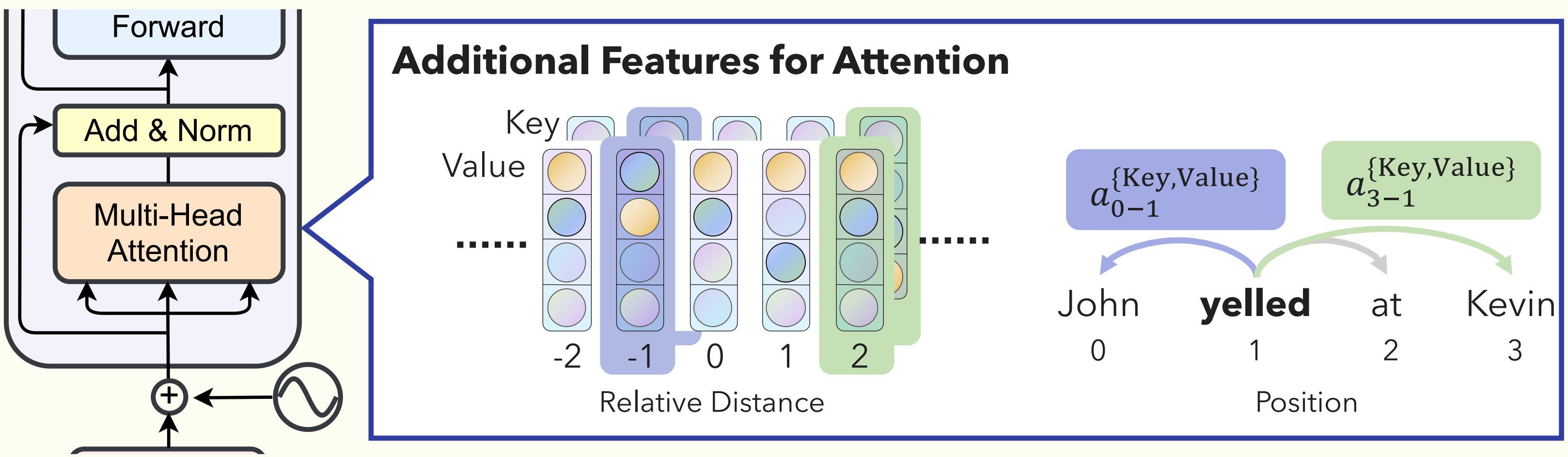


Figure from [Neishi+2019]

Relative Position Embedding (RPE)

[Shaw+2018]

- Consider distance between token pair in self-attention
- 😀 Robust to unseen length by **shift invariance**
- 😰 Computationally more expensive
- 😰 Incompatible with lightweight self-attention variants
  - Performer, Linformer, etc…

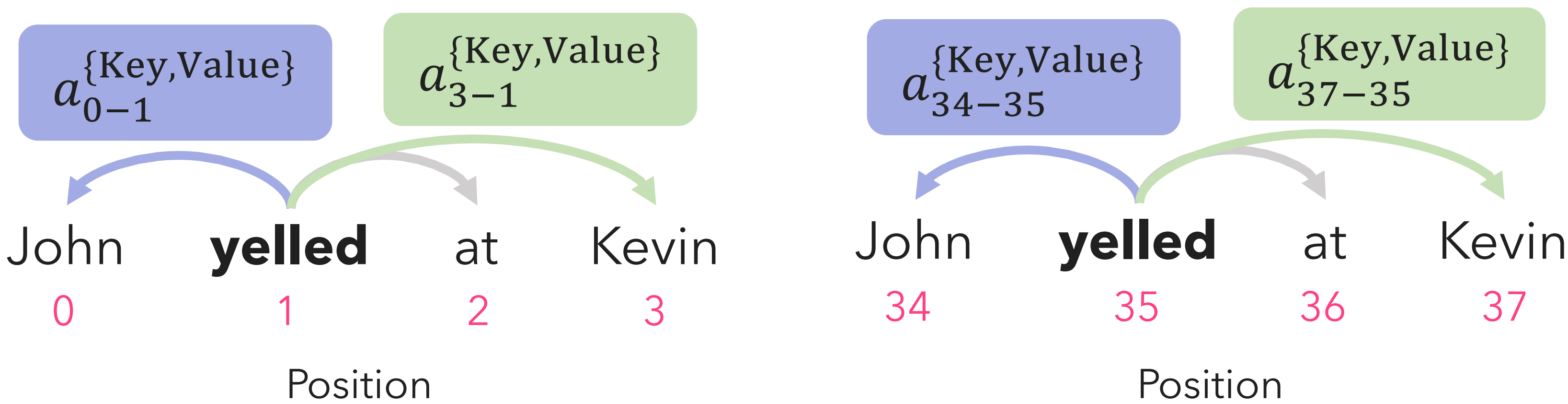


Research Question: APE with Shift Invariance?

- **Shift invariance** in RPE seems the key

→ Spatial shift does not change function's output

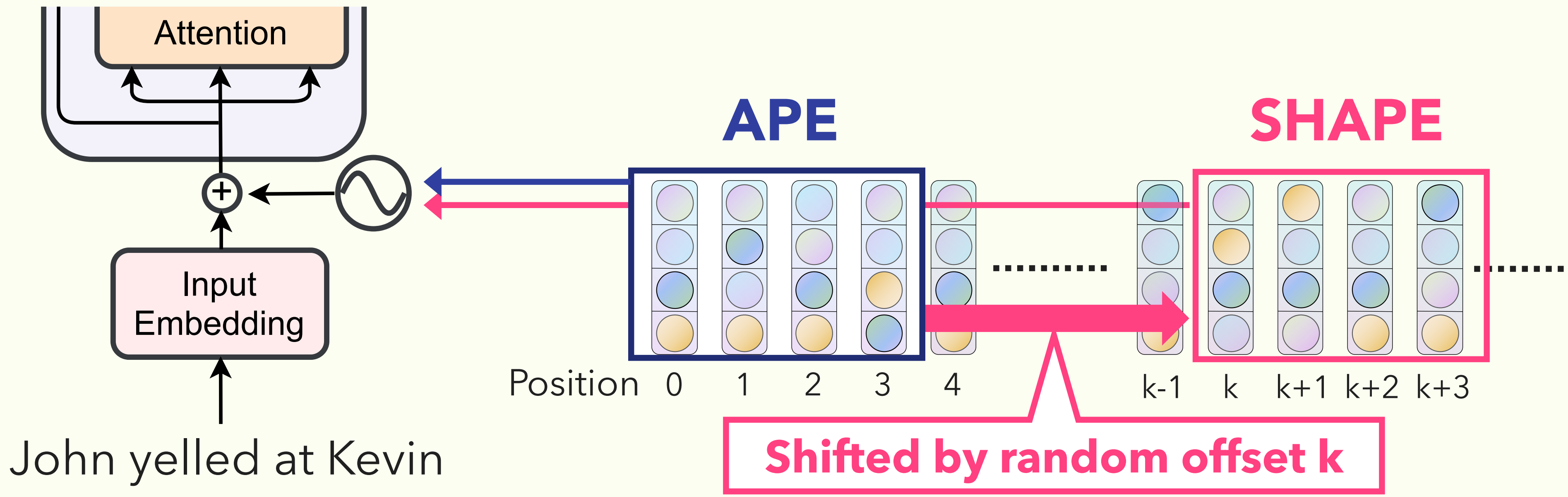
e.g. RPE is shift-invariant: $a_{0-1}^{\{\text{Key,Value}\}}$ is same as $a_{34-35}^{\{\text{Key,Value}\}}$



Can we achieve shift inariance while using APE?

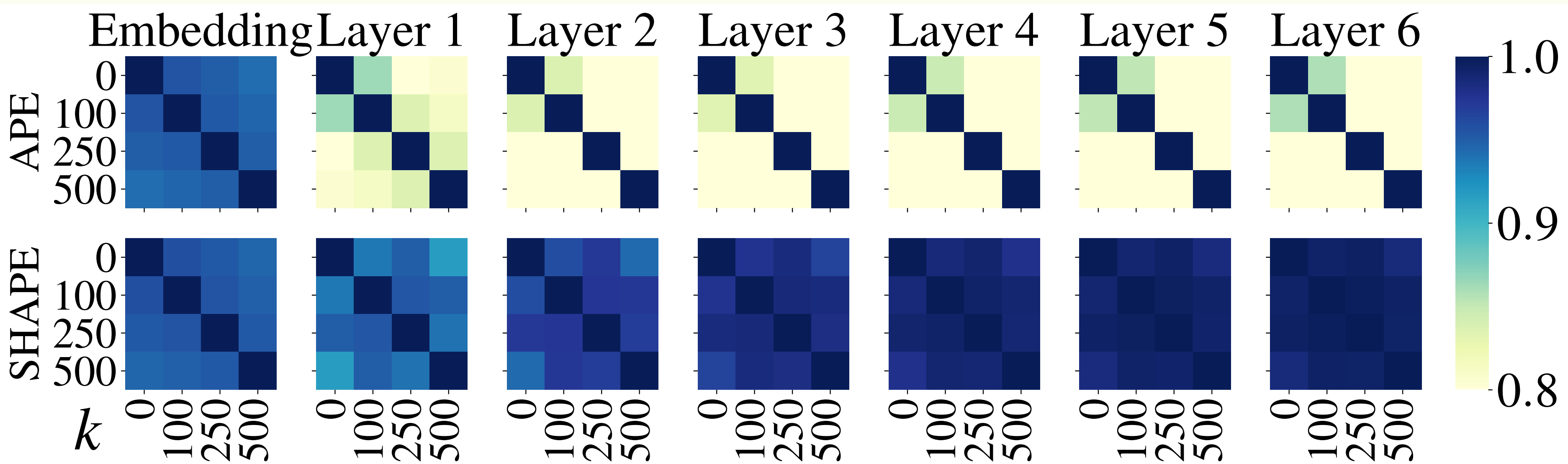
APE+Random Shift for Shift Invariance

- **Shifted Absolute Position Embedding (SHAPE)**
  - APE is randomly shifted by offset $k \sim \mathcal{U}(0, K)$
- Model cannot use absolute position to learn task
  - Instead learns to use relative position?



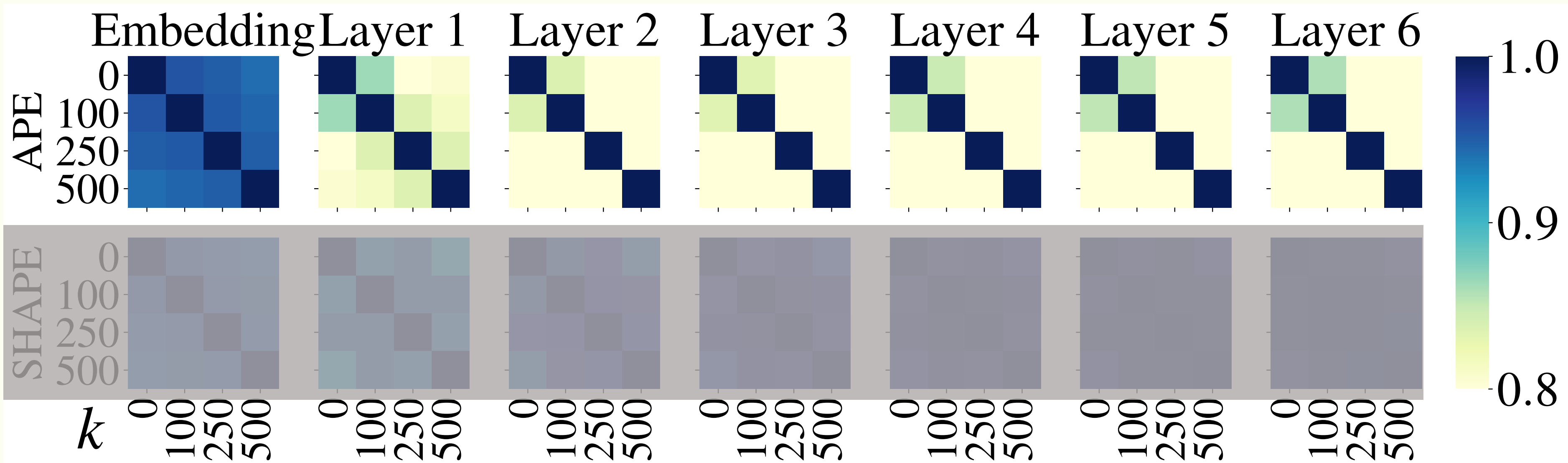
SHAPE Learns Shift Invariance

- Compare cosine similarities of hidden states
- APE: each k produces different hidden states
- SHAPE: hidden states are invariant to k



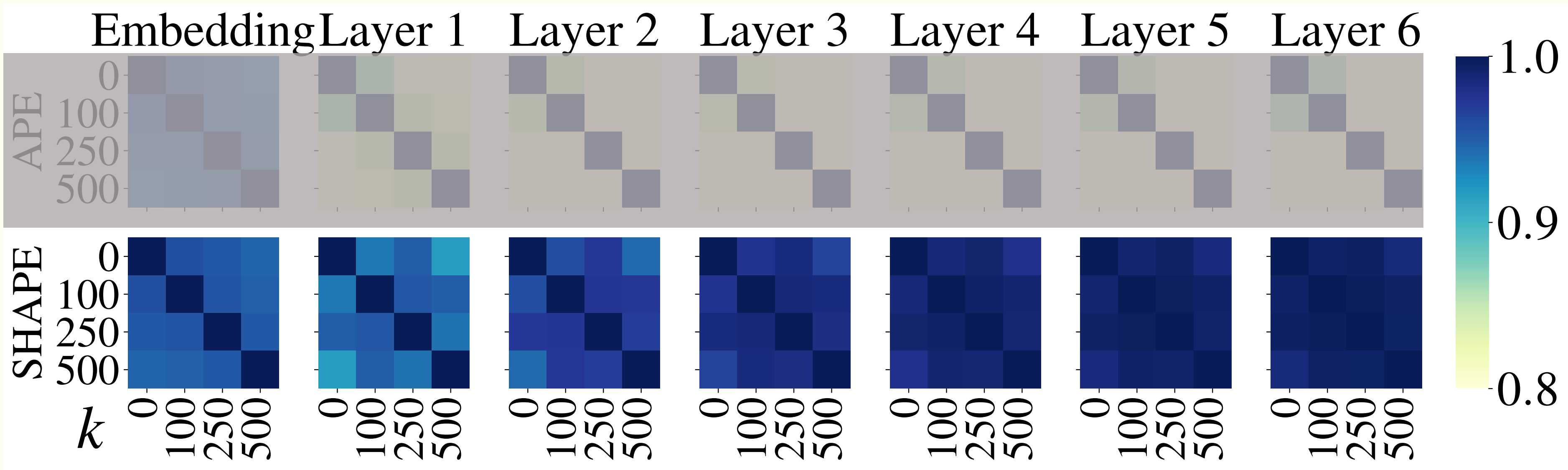
SHAPE Learns Shift Invariance

- Compare cosine similarities of hidden states
- **APE: each k produces different hidden states**
- SHAPE: hidden states are invariant to k



SHAPE Learns Shift Invariance

- Compare cosine similarities of hidden states
- APE: each k produces different hidden states
- **SHAPE: hidden states are invariant to k**



Experimental Configuration

- Model: Transformer with APE, RPE, or SHAPE
- Task: Machine translation (MT)
- Training data
  1. **Vanilla**
     - WMT 2016 EnDe [Ott+2018]
  2. **Extrapolate**
     - Remove sequences longer than 50 subwords from Vanilla
  3. Interpolate
     - Concatenate neighboring sequences

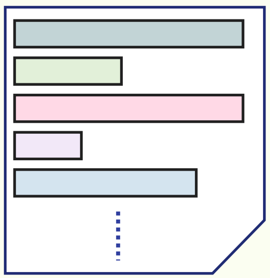
Details in paper or poster session 😀

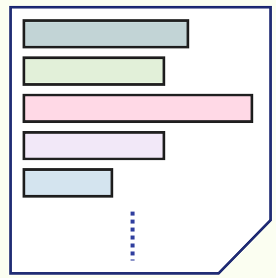
- Validation data: newstest2010-2013
- Test data: newstest2014-2016
- Evaluation: sacreBLEU

Machine Translation Experiment: Three Distinct Datasets

Why three? – To evaluate model performance on seen/unseen lengths

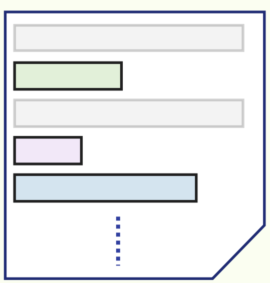
① **Vanilla: WMT EnDe 2016 Dataset [Ott+2018]**

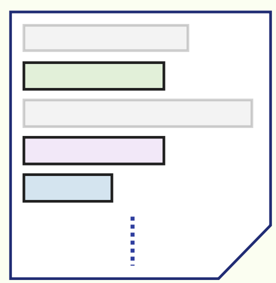
English 

German 

- Standard setting for MT
- Sanity check of baseline performance

② **Extrapolate: remove sequences longer than 50 subwords from Vanilla**

English 

German 

- Evaluate if model can extrapolate
- i.e. is model robust to unseen lengths?

③ **Interpolate: concatenate neighboring sequences (omitted)**

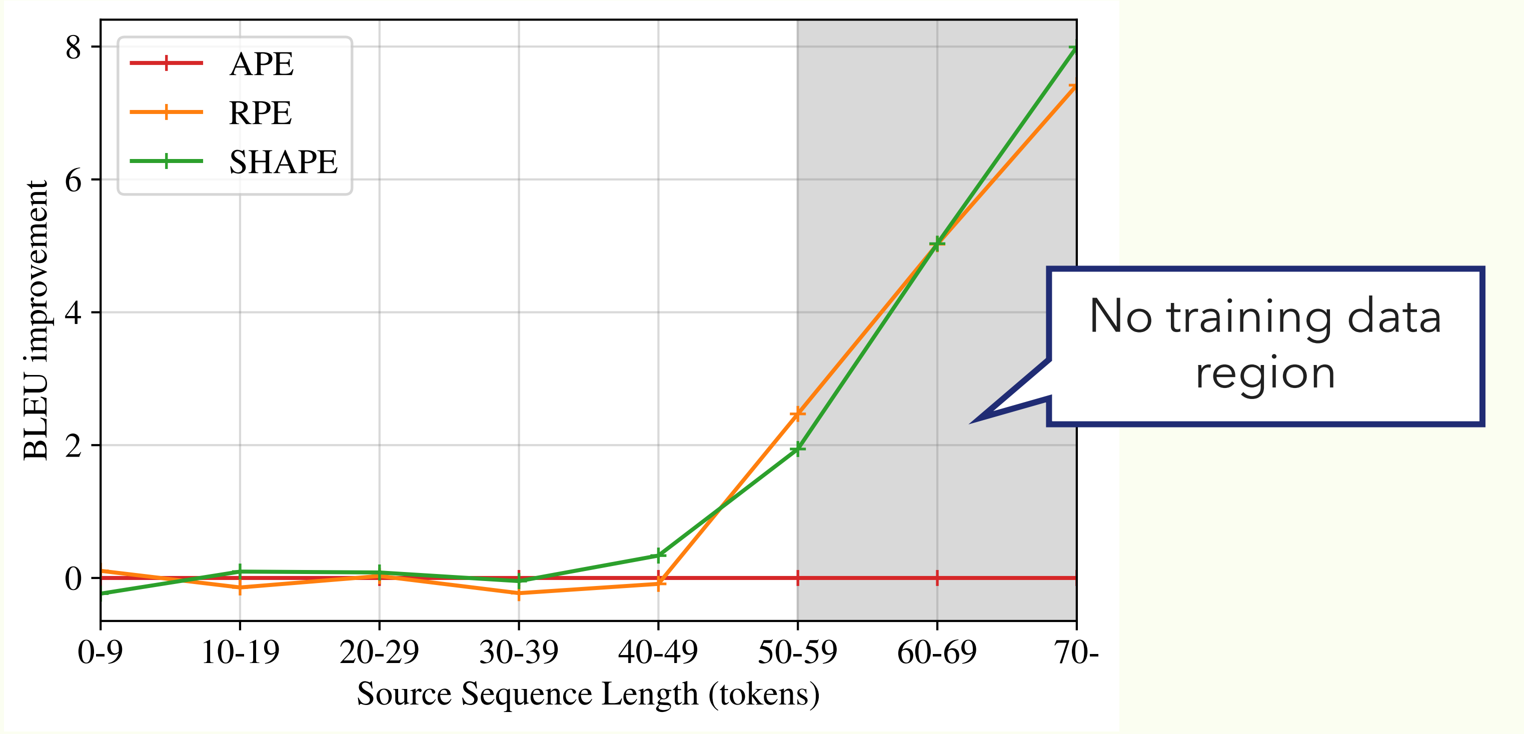
Result: RPE and SHAPE are Comparable

- On Extrapolate
  - Both RPE and SHAPE outperform APE
  - SHAPE is comparable to RPE
  - SHAPE is as fast as APE while RPE is not
- On Vanilla
  - All models achieve comparable performance
  - No risk of performance drop

Dataset	Model	Valid	Test	Speed
VANILLA	APE†	23.61	30.46	x1.00
	RPE†	23.67	30.54	x0.91
	SHAPE†	23.63	30.49	x1.01
EXTRAPOLATE	APE	22.18	29.22	x1.00
	RPE	22.97	29.86	x0.91
	SHAPE	22.96	29.80	x0.99

Length Analysis: Better Extrapolation

Relative BLEU improvement from baseline (APE)



- RPE and SHAPE can better extrapolate than APE
- SHAPE and RPE have comparable extrapolation ability

Conclusion

- SHAPE : shifted absolute position embedding
  - APE with shift invariance
  - As fast as APE & comparable performance to RPE
  - Easy implementation
  - No risk of performance drop from APE

Take Home PyTorch Code

```
pos_idx += self.training * torch.randint(0, K)
pos_embed(pos_idx)
```