Neural Machine Translation with Joint Representation

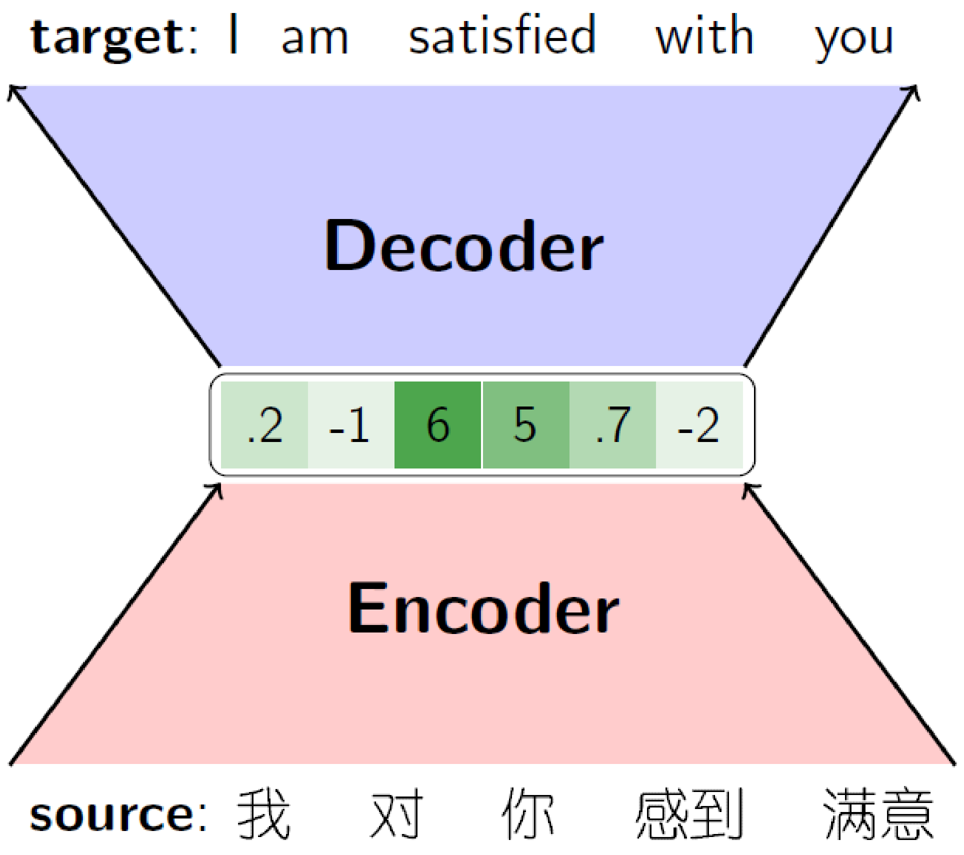
李炎洋[1]，王强[1]，肖桐[1]，刘彤冉[2]，朱靖波[1]

东北大学
自然语言处理实验室[1]

中国科学院
心理研究所[2]

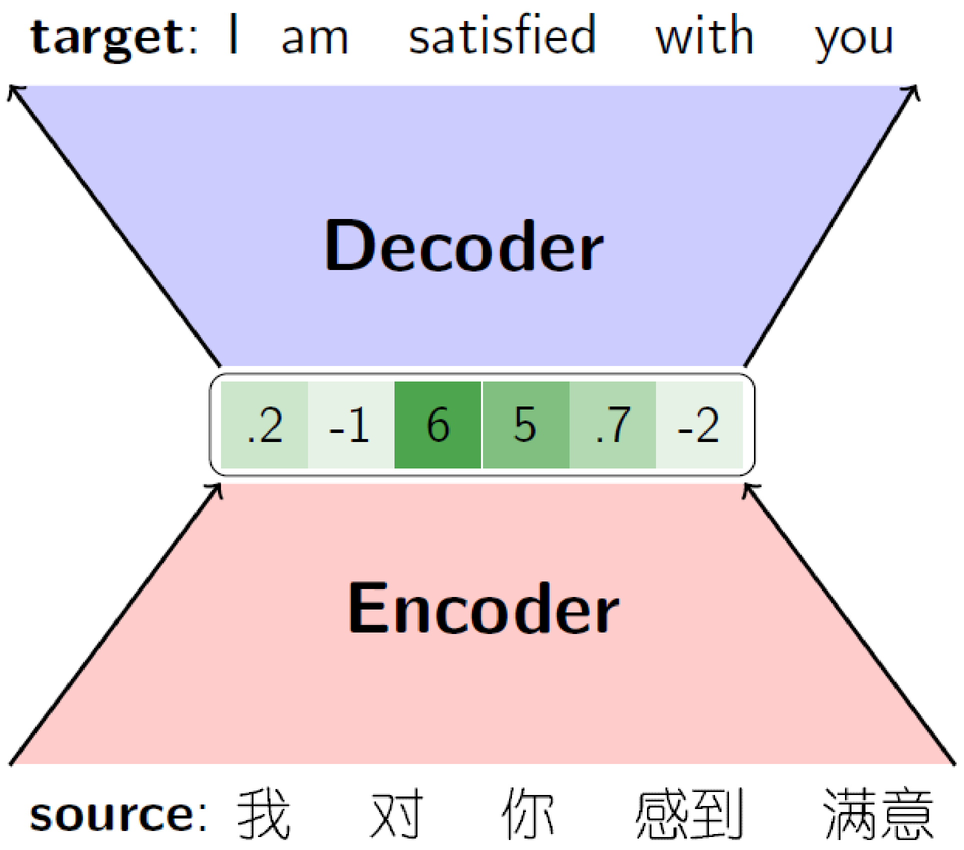
Neural Machine Translation (NMT)

- Existing NMT models are based on the Encoder-Decoder framework
  - Encoder converts the source sentence to a continuous representation
  - Decoder converts the continuous representation to the target sentence



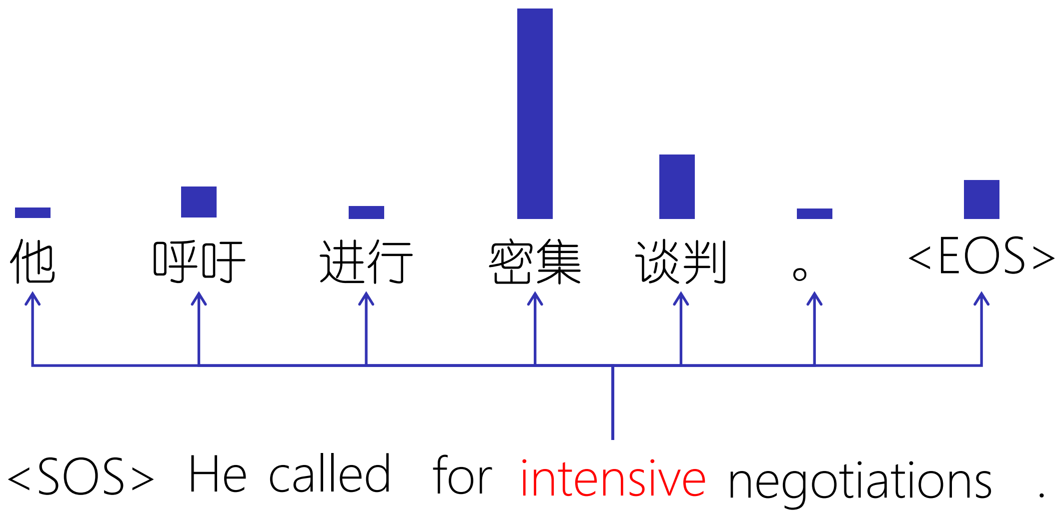
Limitations

- Existing NMT models are based on the Encoder-Decoder framework
  - Encoder converts the source sentence to a continuous representation
  - Decoder converts the continuous representation to the target sentence
- But it processes the source and target sentences separately



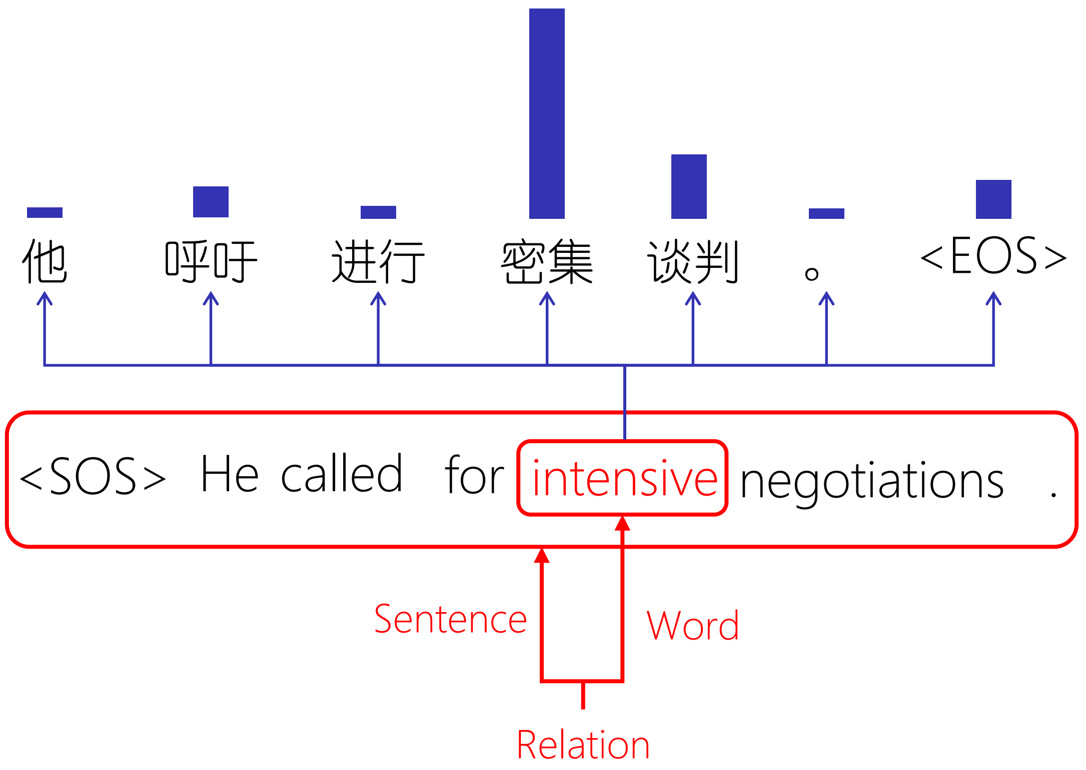
Limitations

- Existing NMT models are based on the Encoder-Decoder framework
  - Encoder converts the source sentence to a continuous representation
  - Decoder converts the continuous representation to the target sentence
- But it processes the source and target sentences separately
- Attention that bridges the encoder and decoder has alleviated this issue

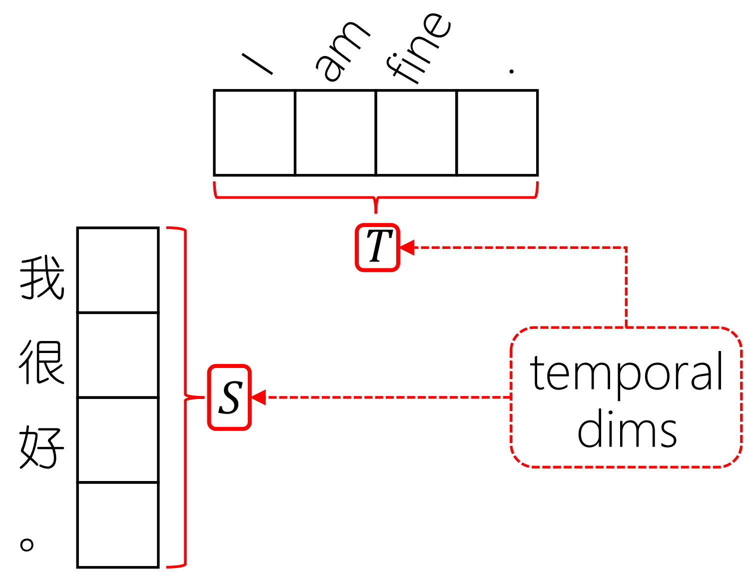


Limitations

- Existing NMT models are based on the Encoder-Decoder framework
  - Encoder converts the source sentence to a continuous representation
  - Decoder converts the continuous representation to the target sentence
- But it processes the source and target sentences separately
- Attention that bridges the encoder and decoder has alleviated this issue
- But it only models the relations of one target word to the source sentence
  - Fail to model those of the target sentence to the source one

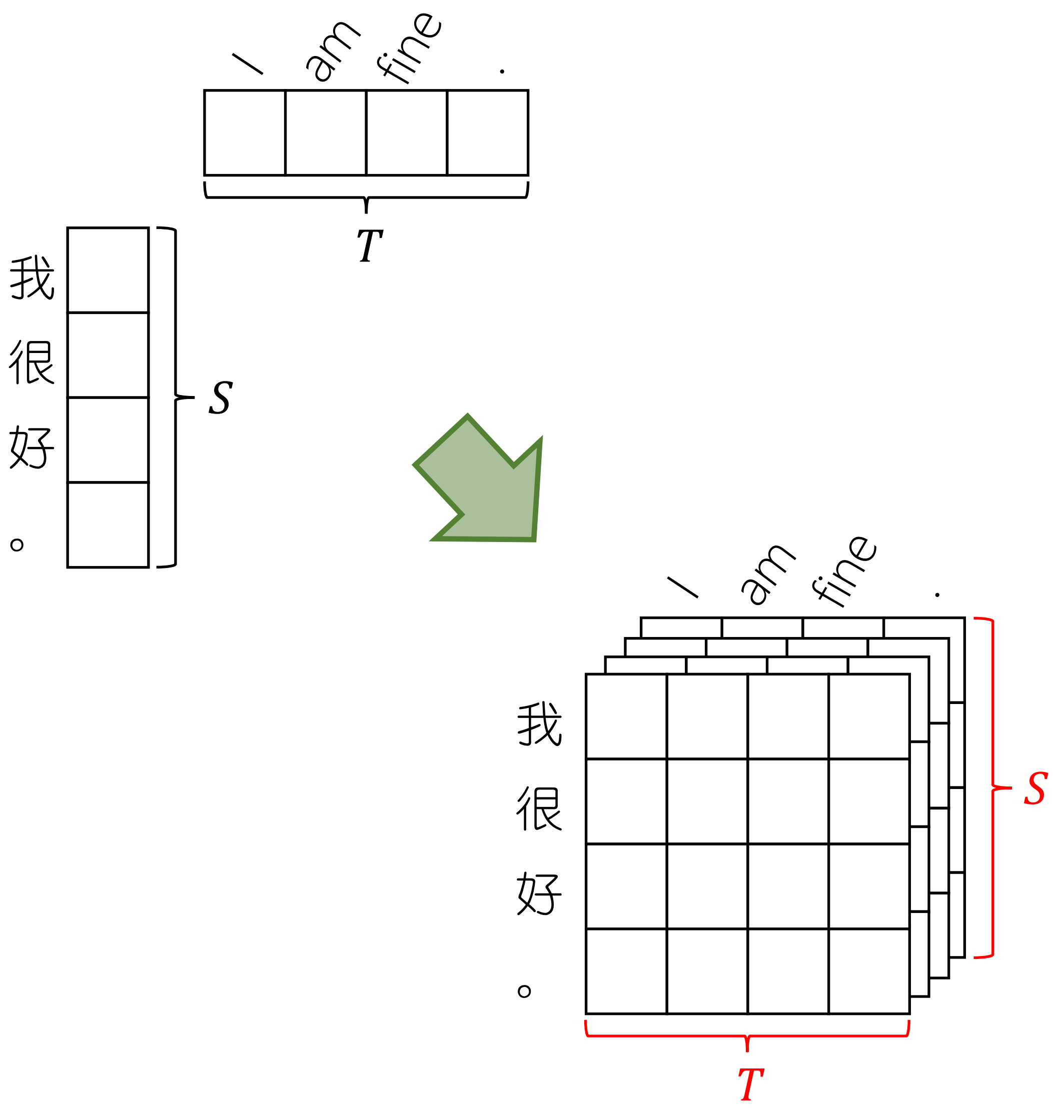


Joint Representation



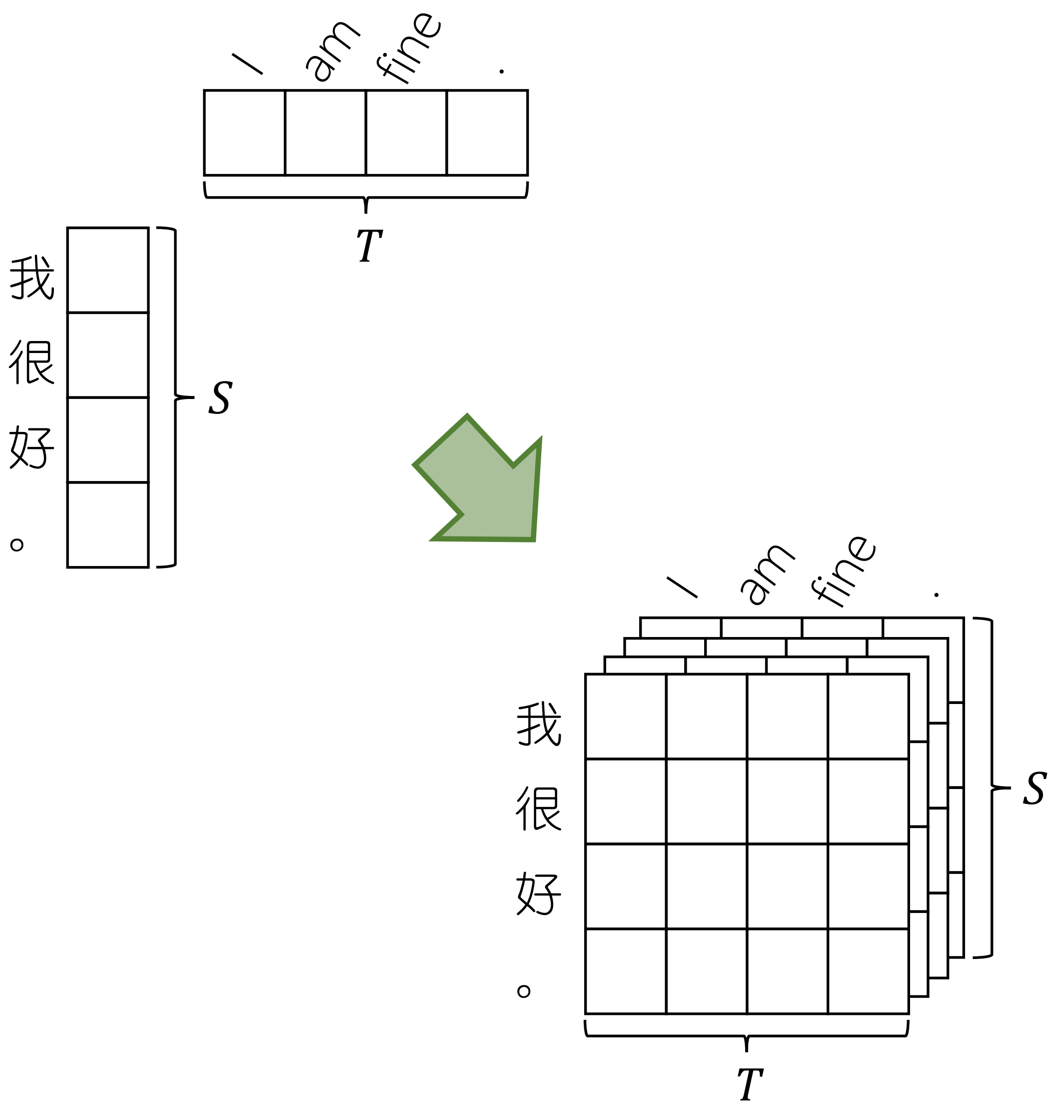
- The natural idea is to extend the raw representation from one source (S) or target (T) temporal dim

Joint Representation



- The natural idea is to extend the raw representation from one source (S) or target (T) temporal dim
- To two temporal dims $S \times T$ (Joint Representation)
- Which assigns one representation for each source-target token combination

Joint Representation

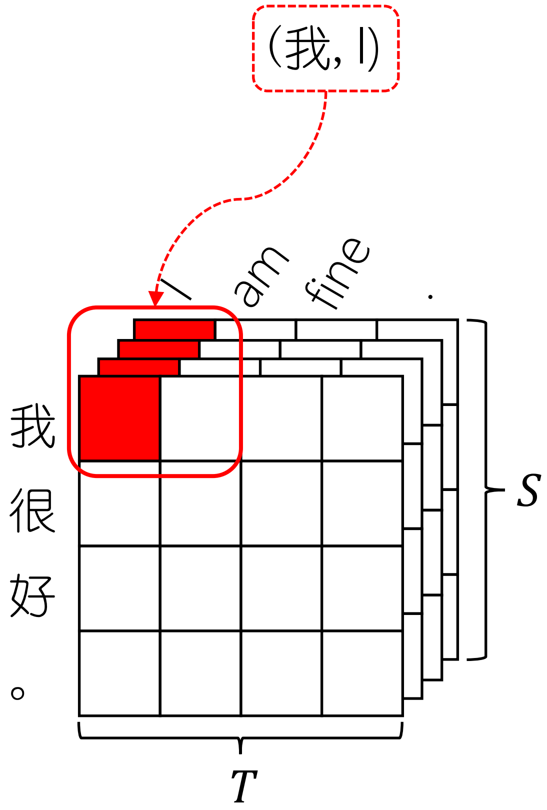


- The natural idea is to extend the raw representation from one source (S) or target (T) temporal dim
- To two temporal dims $S \times T$ (Joint Representation)
- Which assigns one representation for each source-target token combination

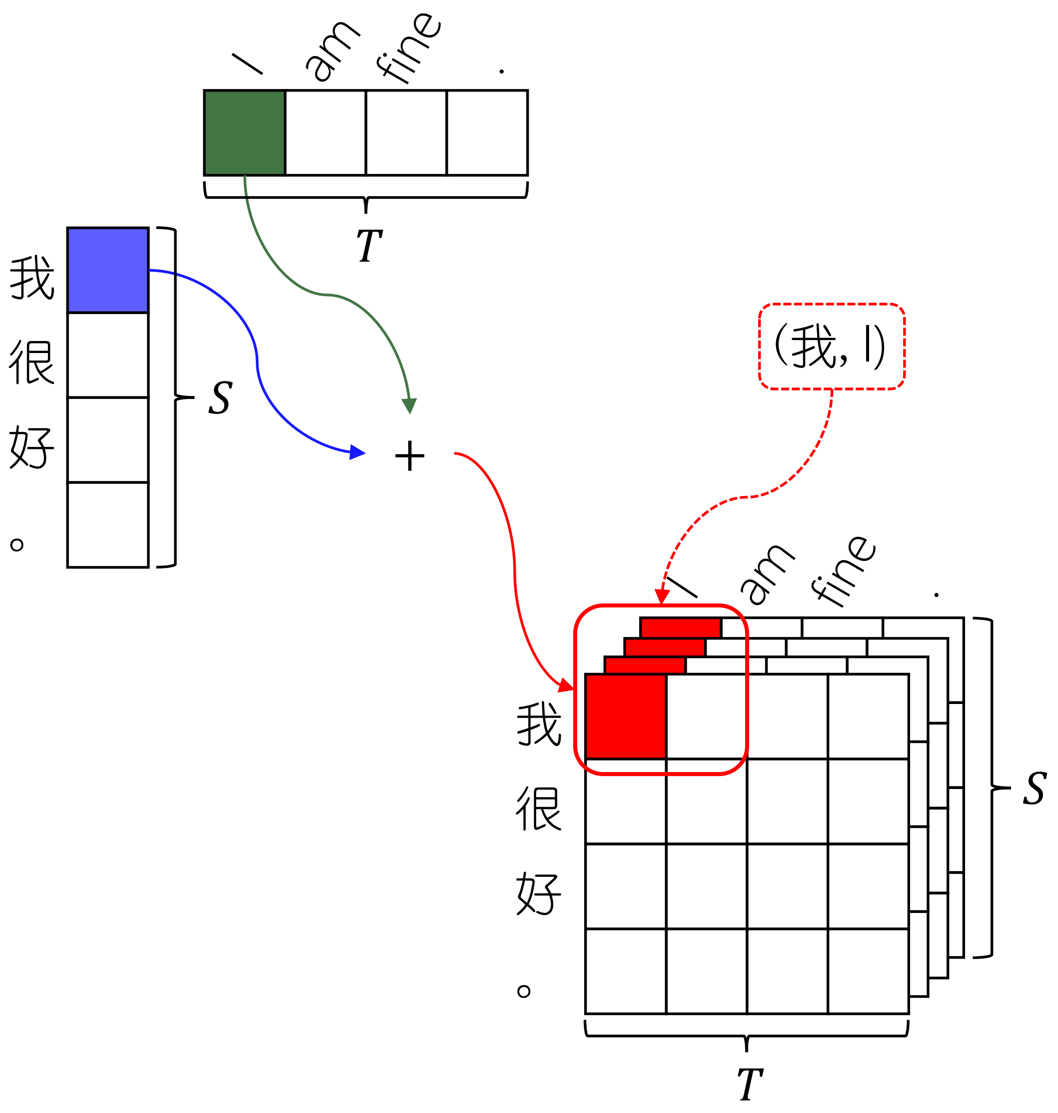
How to construct input embeddings for the joint representation?

Construct Initial Representation

- In principle, we need V^2 embeddings
- Each represents one possible source-target token combination



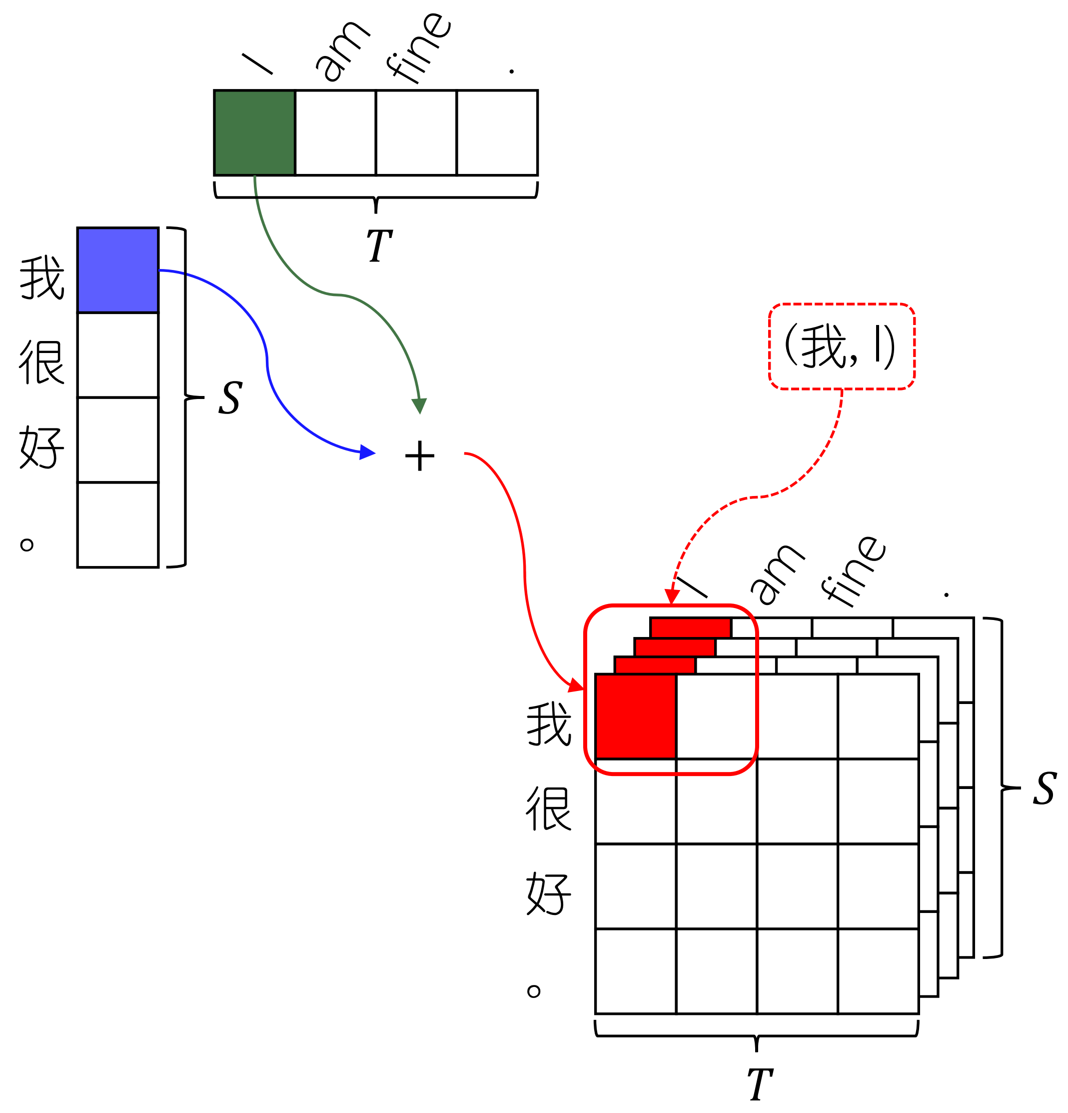
Construct Initial Representation



- In principle, we need V^2 embeddings
- Each represents one possible source-target token combination
- Without the context, words are almost independent, allowing us to decompose such embeddings to $2V$

$$\mathbf{embed}_{ij} = \mathbf{embed}_i + \mathbf{embed}_j$$

Construct Initial Representation

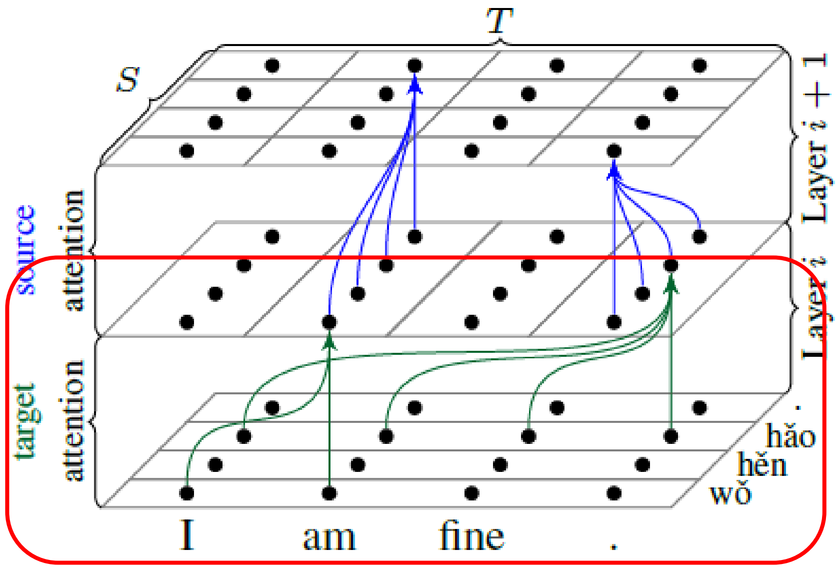


- In principle, we need V^2 embeddings
- Each represents one possible source-target token combination
- Without the context, words are almost independent, allowing us to decompose such embeddings to $2V$

$$\text{embed}_{ij} = \text{embed}_i + \text{embed}_j$$

How to perform the attention on the joint representation?

Separable Attention

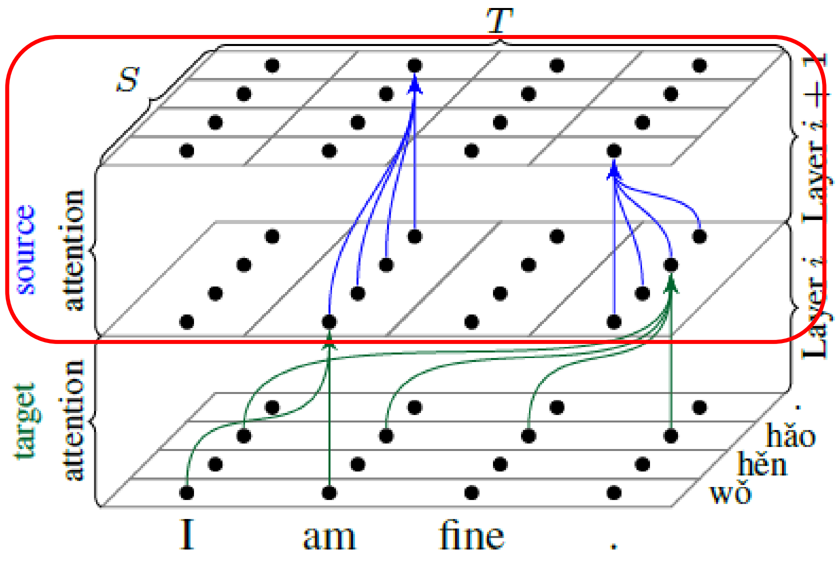


- Training
  1. Target attention: performs the attention along the target dim

$$\mathrm{SepAttn}(Q, K, V) = [\mathrm{split}_1, \ldots, \mathrm{split}_S]$$

$$\text{where } \mathrm{split}_i = \mathrm{Attention}(Q_i, K_i, V_i)$$

Separable Attention

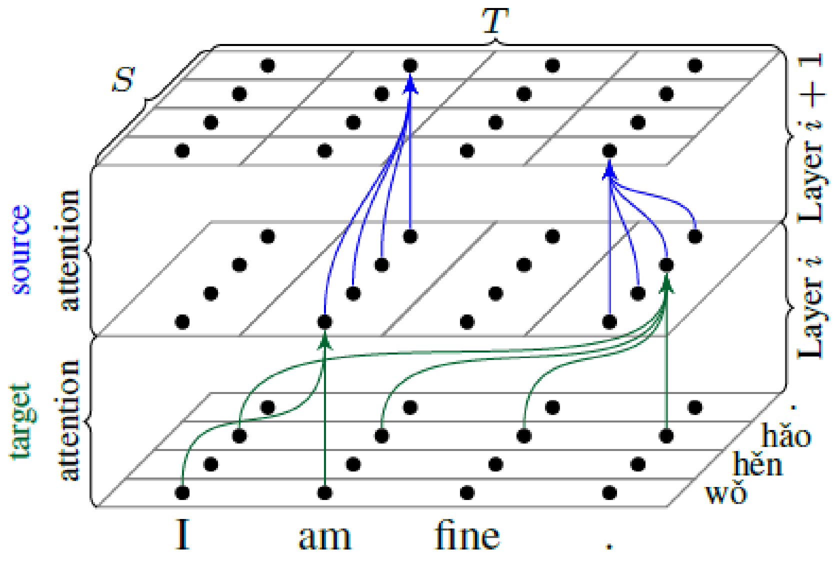


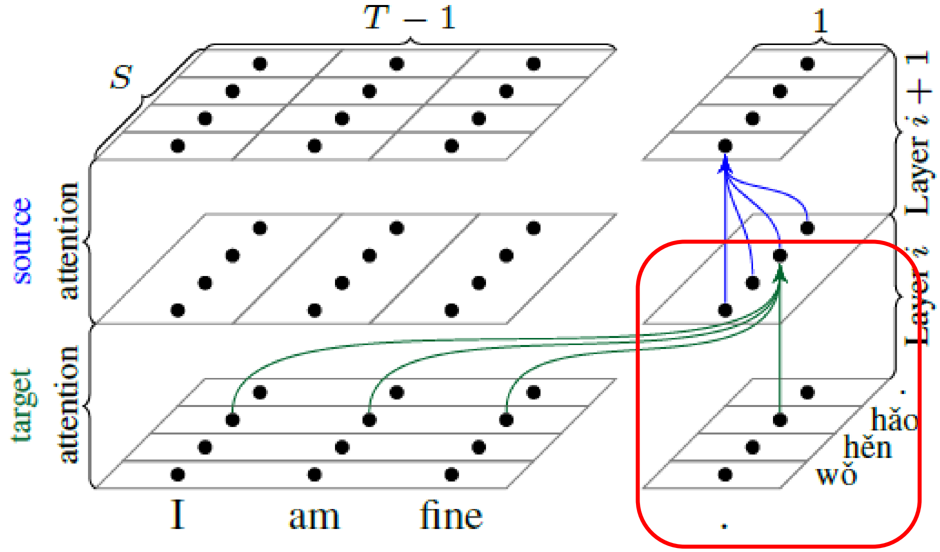
- Training
  1. Target attention: performs the attention along the target dim
  2. Source attention: performs the attention along the source dim

$$\mathrm{SepAttn}(Q,K,V) = [\mathrm{split}_1, \ldots, \mathrm{split}_T]$$

$$\text{where } \mathrm{split}_i = \mathrm{Attention}(Q_i, K_i, V_i)$$

Separable Attention

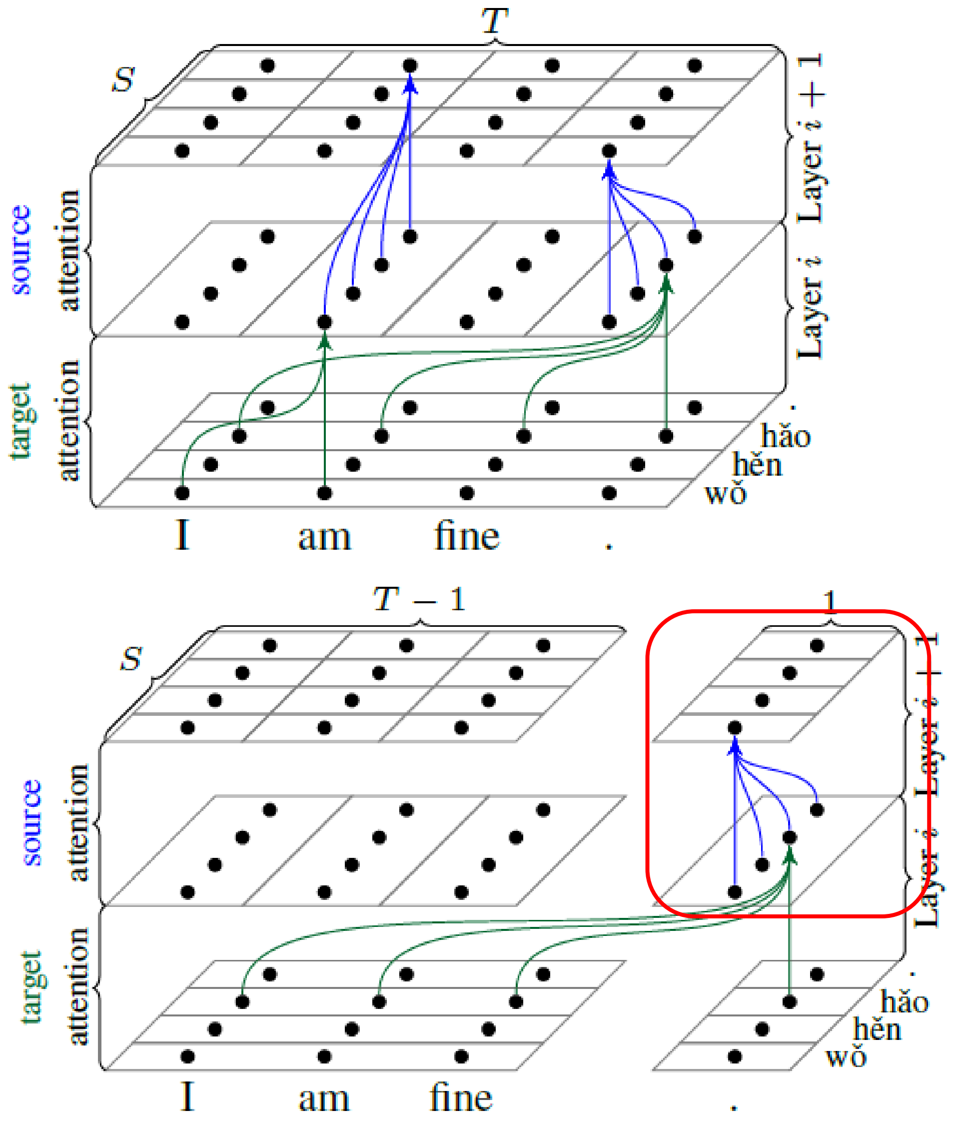




- Training
  1. Target attention: performs the attention along the target dim
  2. Source attention: performs the attention along the source dim

- Decoding ($\boldsymbol{T}$-th step)
  1. Target attention: only attends the previous $T-1$ target tokens for the T-th input

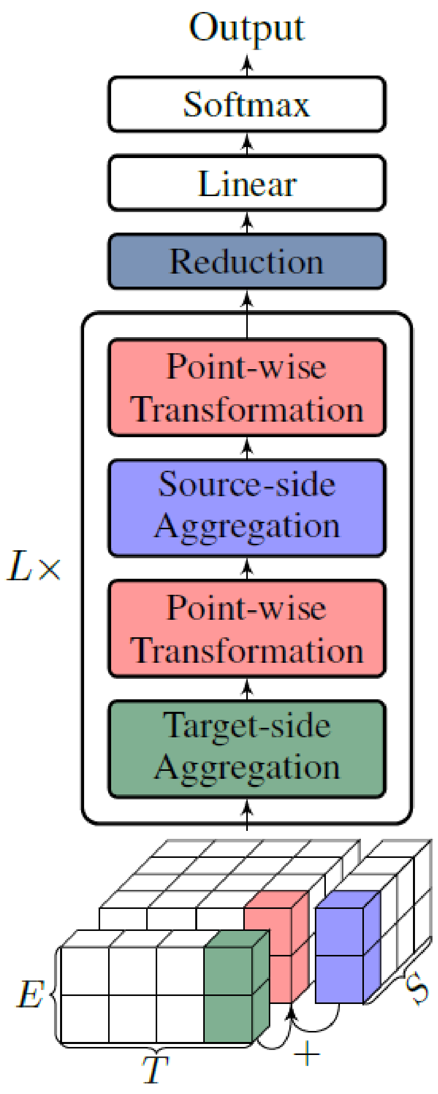
Separable Attention



- Training
  1. Target attention: performs the attention along the target dim
  2. Source attention: performs the attention along the source dim

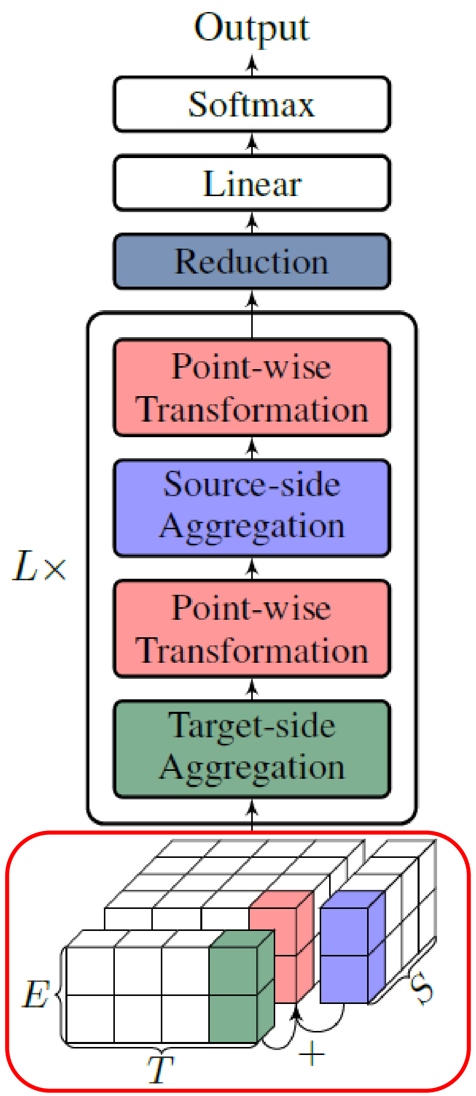
- Decoding ($\boldsymbol{T}$-th step)
  1. Target attention: only attends the previous $\boldsymbol{T-1}$ target tokens for the $\boldsymbol{T}$-th input
  2. Source attention: only attends all source tokens in the $\boldsymbol{T}$-th input

Putting Together



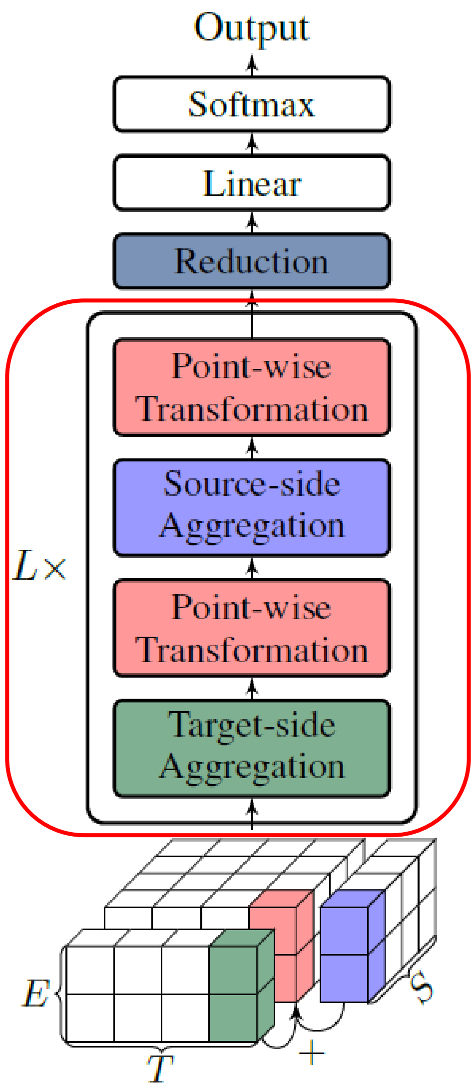
- General Structure

Putting Together



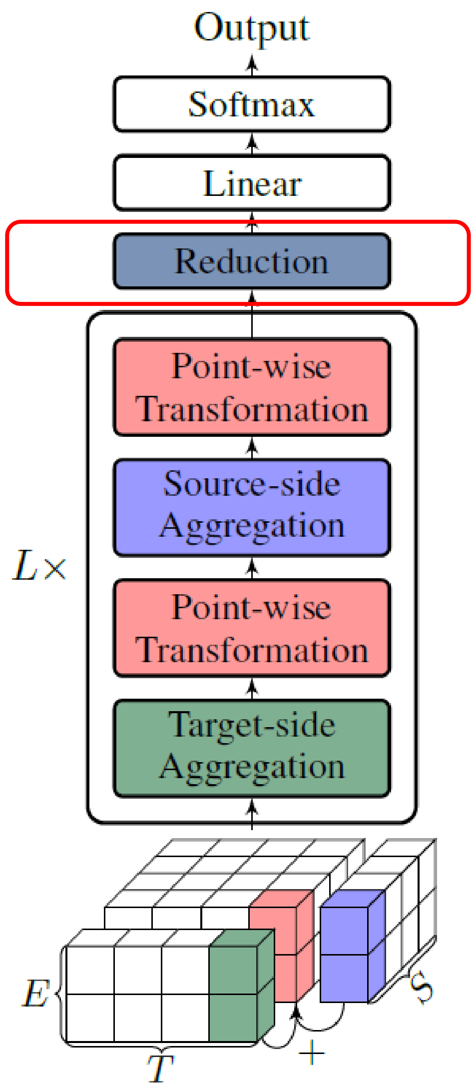
- General Structure
  - Construct the joint representation embeddings

Putting Together



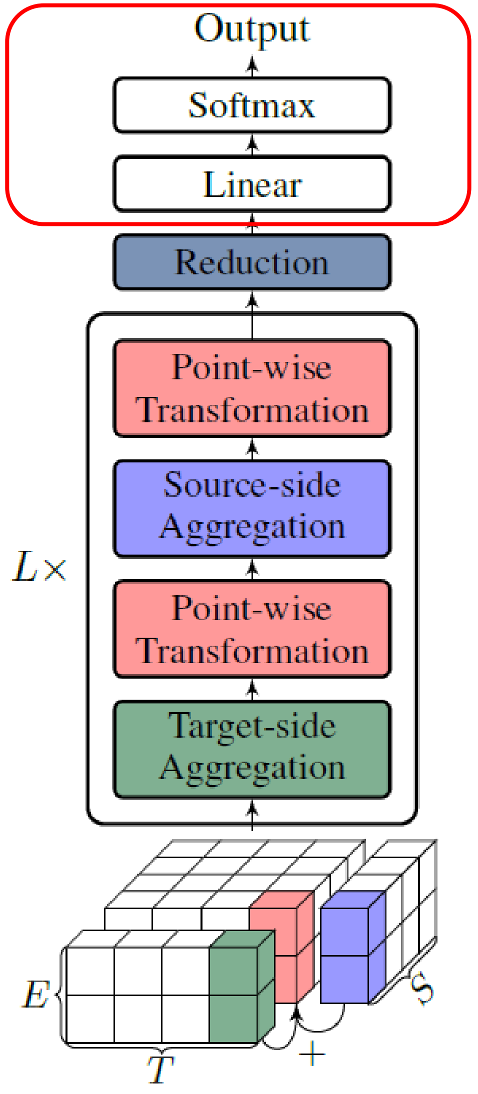
- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers

Putting Together



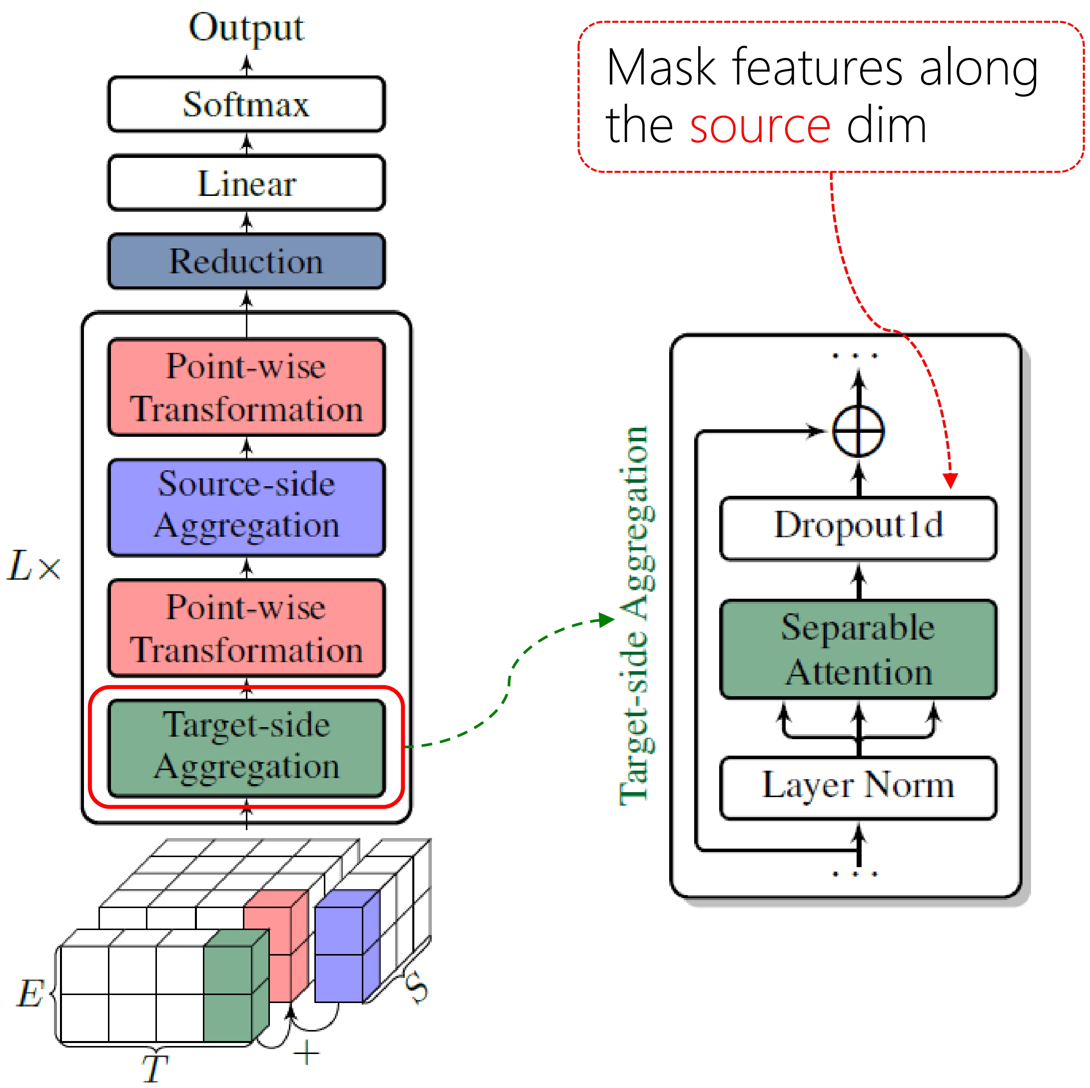
- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers
  - Reduce the source dim of joint representation

Putting Together



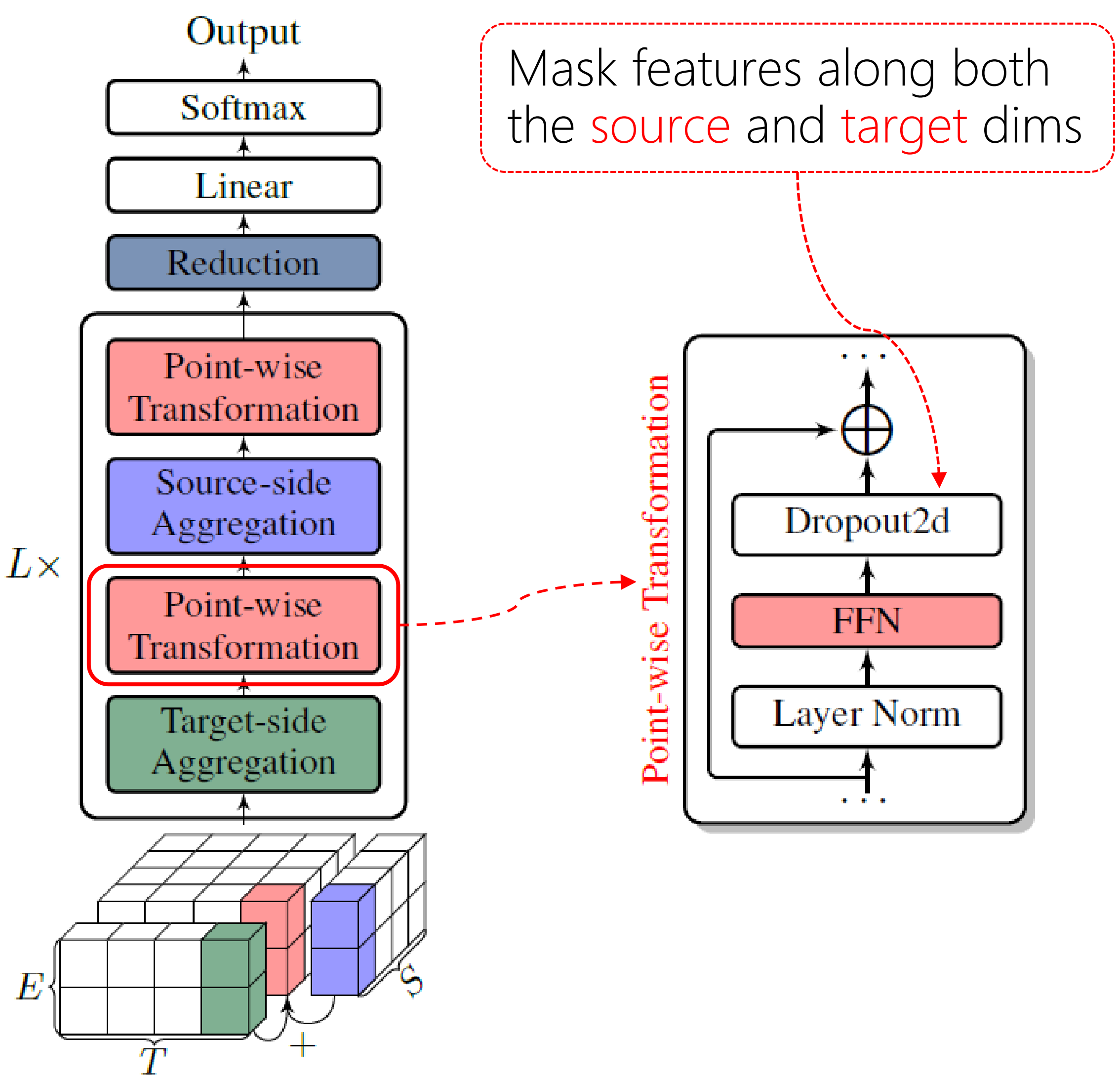
- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers
  - Reduce the source dim of joint representation
  - Predict the target sentence

Putting Together



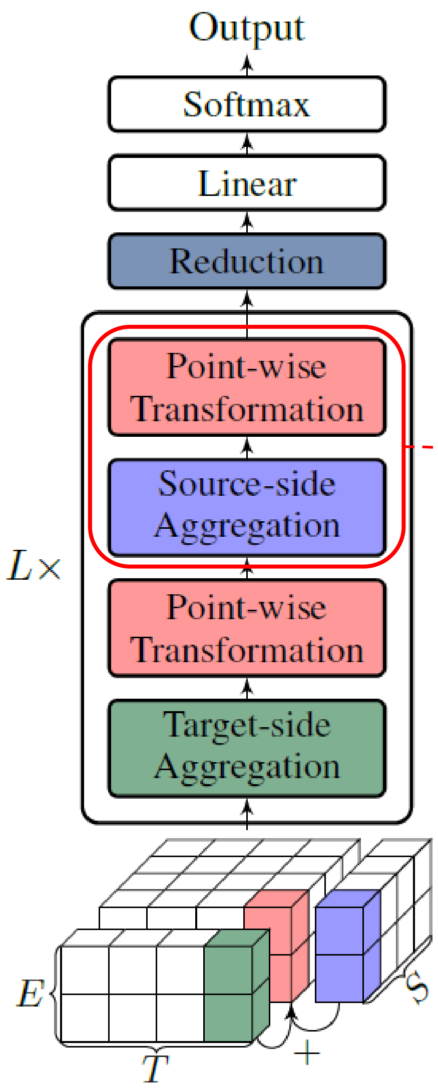
- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers
  - Reduce the source dim of joint representation
  - Predict the target sentence
- Layer
  - Perform target attention

Putting Together



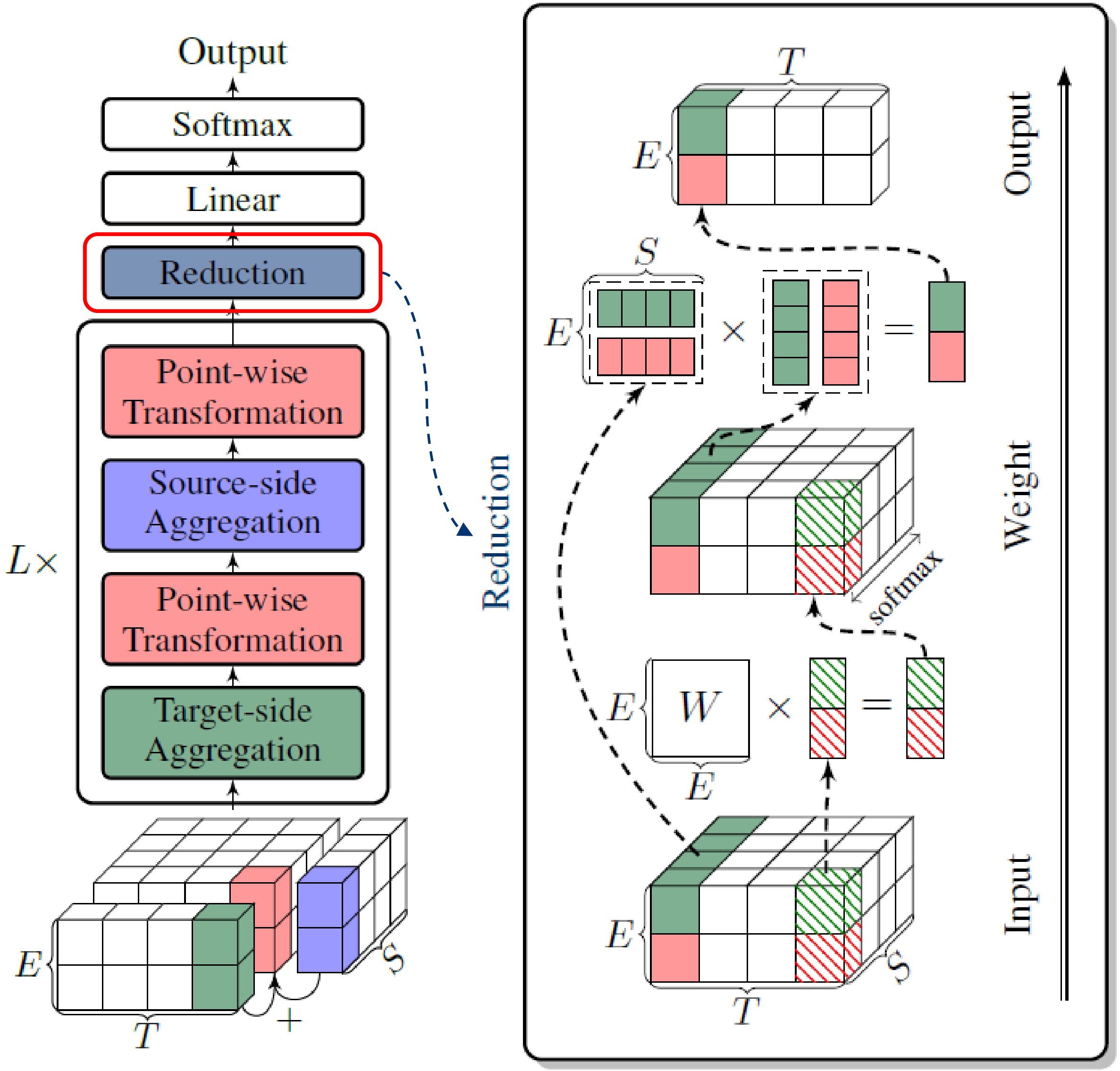
- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers
  - Reduce the source dim of joint representation
  - Predict the target sentence
- Layer
  - Perform target attention
  - Apply the non-linear transformation

Putting Together



- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers
  - Reduce the source dim of joint representation
  - Predict the target sentence
- Layer
  - Perform target attention
  - Apply the non-linear transformation
  - Perform source attention
  - Apply the non-linear transformation again

Putting Together



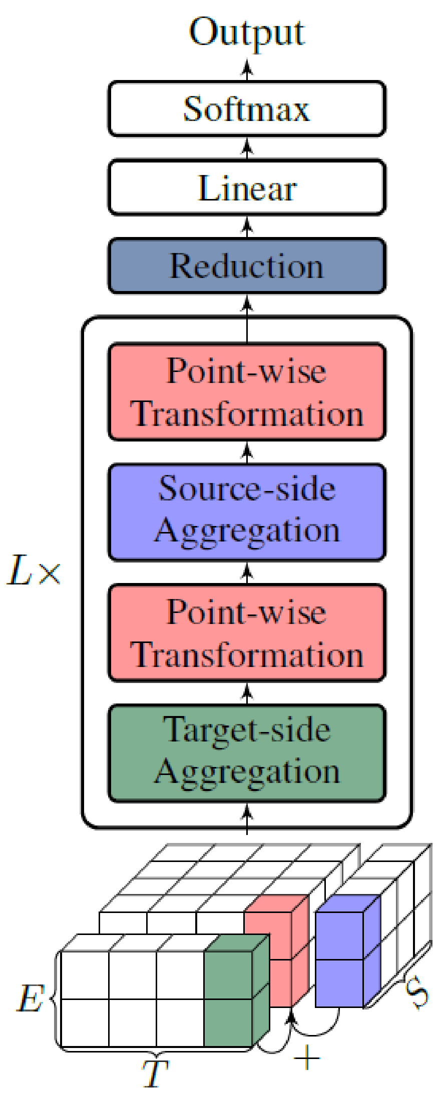
- General Structure
  - Construct the joint representation embeddings
  - Pass through a stack of identical layers
  - Reduce the source dim of joint representation
  - Predict the target sentence
- Reduction
  - A feature-wise attention, similar to the source attention but with a learnable query W

$$\mathrm{Reduction}(x) = [\mathrm{head}_1, \dots, \mathrm{head}_E]$$

$$\text{where } \mathrm{head}_i = \mathrm{softmax}(W_i x^T) x_i$$

*E: the embedding size

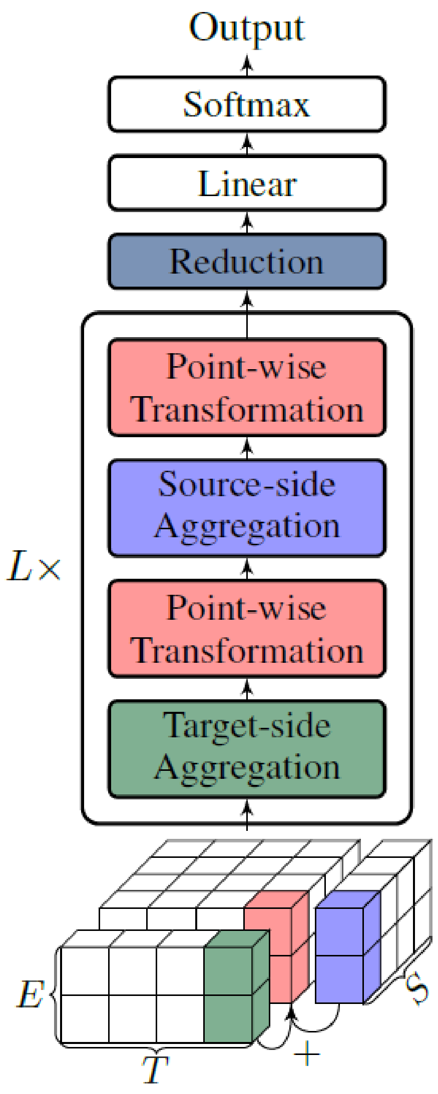
Effectiveness & Efficiency



- This gives us Reformer-base
  - It enjoys the best theoretical soundness
  - Accesses any token with $O(1)$ path length
  - But not done ...
- Two efficiency downsides
  - Duplicate computation
  - Computation allocation

Layer Type	Complexity	Path Length
Separable Attention	$O(n^3d)$	$\boxed{O(1)}$
Self-Attention	$O(n^2d)$	$\boxed{O(l)}$
Recurrent	$O(nd^2)$	$O(l+n)$
Convolution	$O(knd^2)$	$O(l+n\log_k(n))$

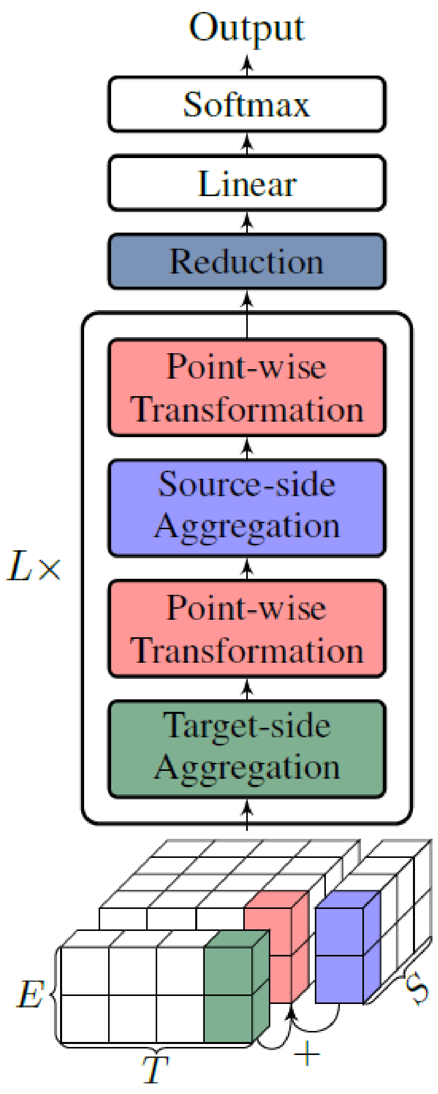
Effectiveness & Efficiency



- Duplicate Computation
  - Start from the embeddings at each step
  - Recompute the abstract (source) information

Layer Type	Complexity	Path Length
Separable Attention	$O(n^3d)$	$O(1)$
Self-Attention	$O(n^2d)$	$O(l)$
Recurrent	$O(nd^2)$	$O(l+n)$
Convolution	$O(knd^2)$	$O(l+n\log_k(n))$

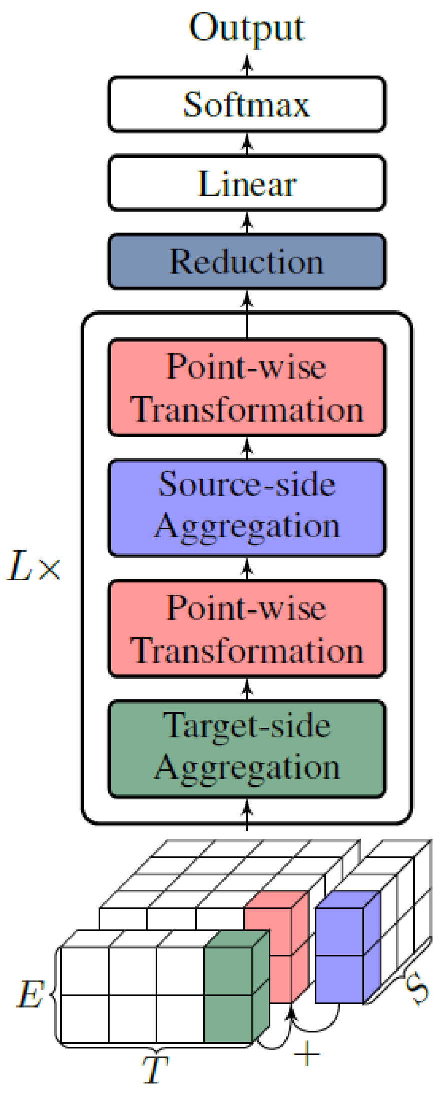
Effectiveness & Efficiency



- Duplicate Computation
  - Start from the embeddings at each step
  - Recompute the abstract (source) information
- Computation Allocation
  - Each step: #Source tokens ≫ #Target tokens
  - Require more source-side operations

Layer Type	Complexity	Path Length
Separable Attention	$O(n^3 d)$	$O(1)$
Self-Attention	$O(n^2 d)$	$O(l)$
Recurrent	$O(nd^2)$	$O(l+n)$
Convolution	$O(knd^2)$	$O(l+n\log_k(n))$

Effectiveness & Efficiency

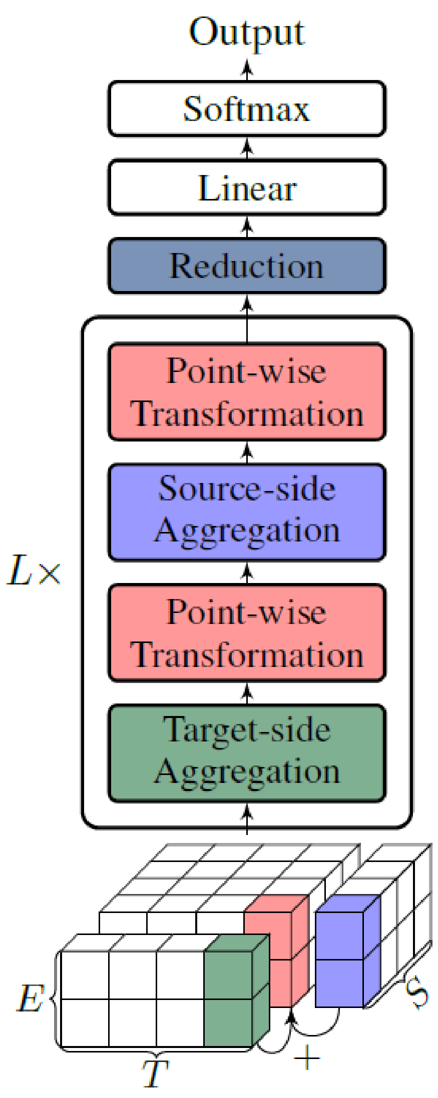


- Duplicate Computation
  - Start from the embeddings at each step
  - Recompute the abstract (source) information
- Computation Allocation
  - Each step: #Source tokens ≫ #Target tokens
  - Require more source-side operations

Both require to stack more high-complexity separable attention

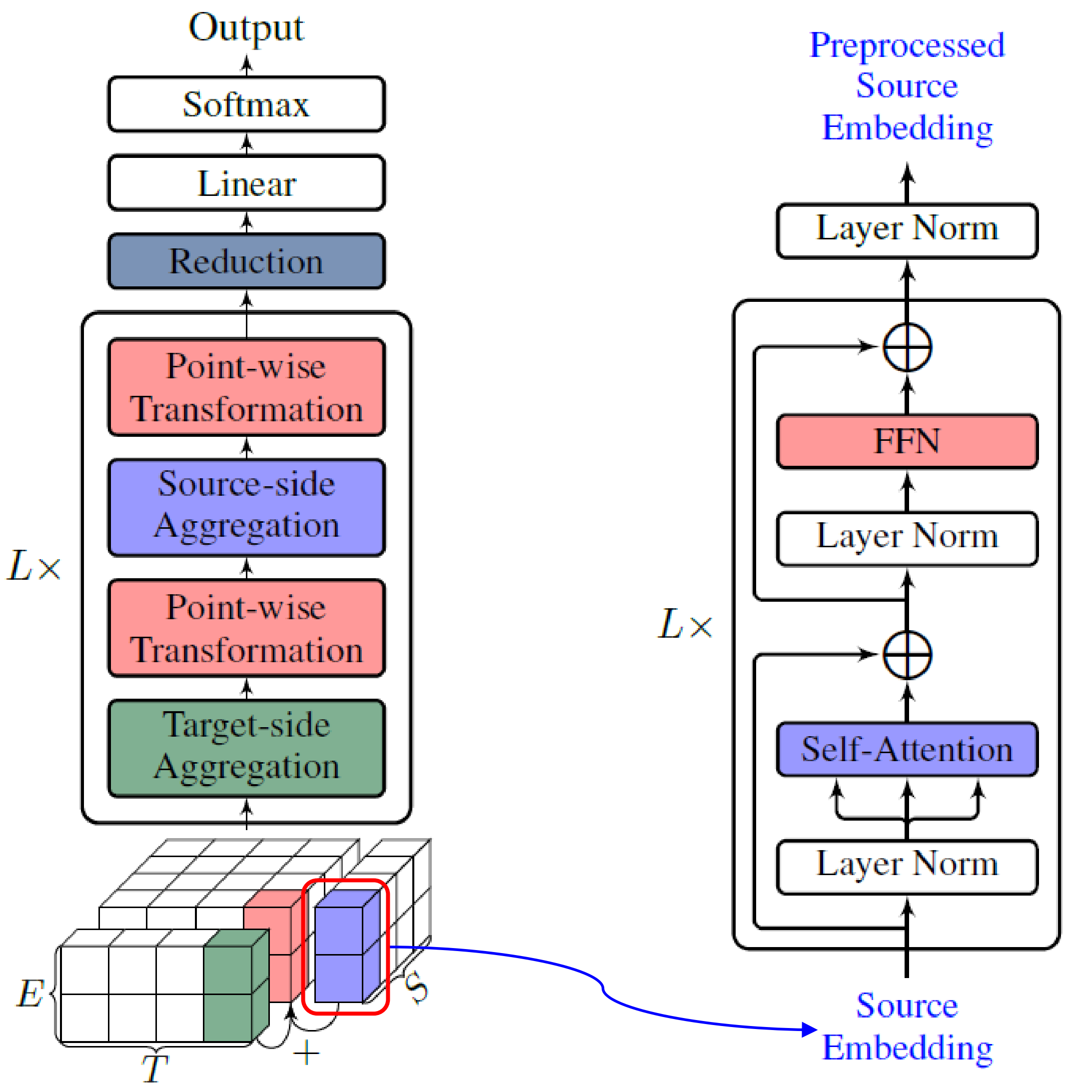
Layer Type	Complexity	Path Length
Separable Attention	$O(n^3d)$	$O(1)$
Self-Attention	$O(n^2d)$	$O(l)$
Recurrent	$O(nd^2)$	$O(l+n)$
Convolution	$O(knd^2)$	$O(l+n\log_k(n))$

A Faster Model



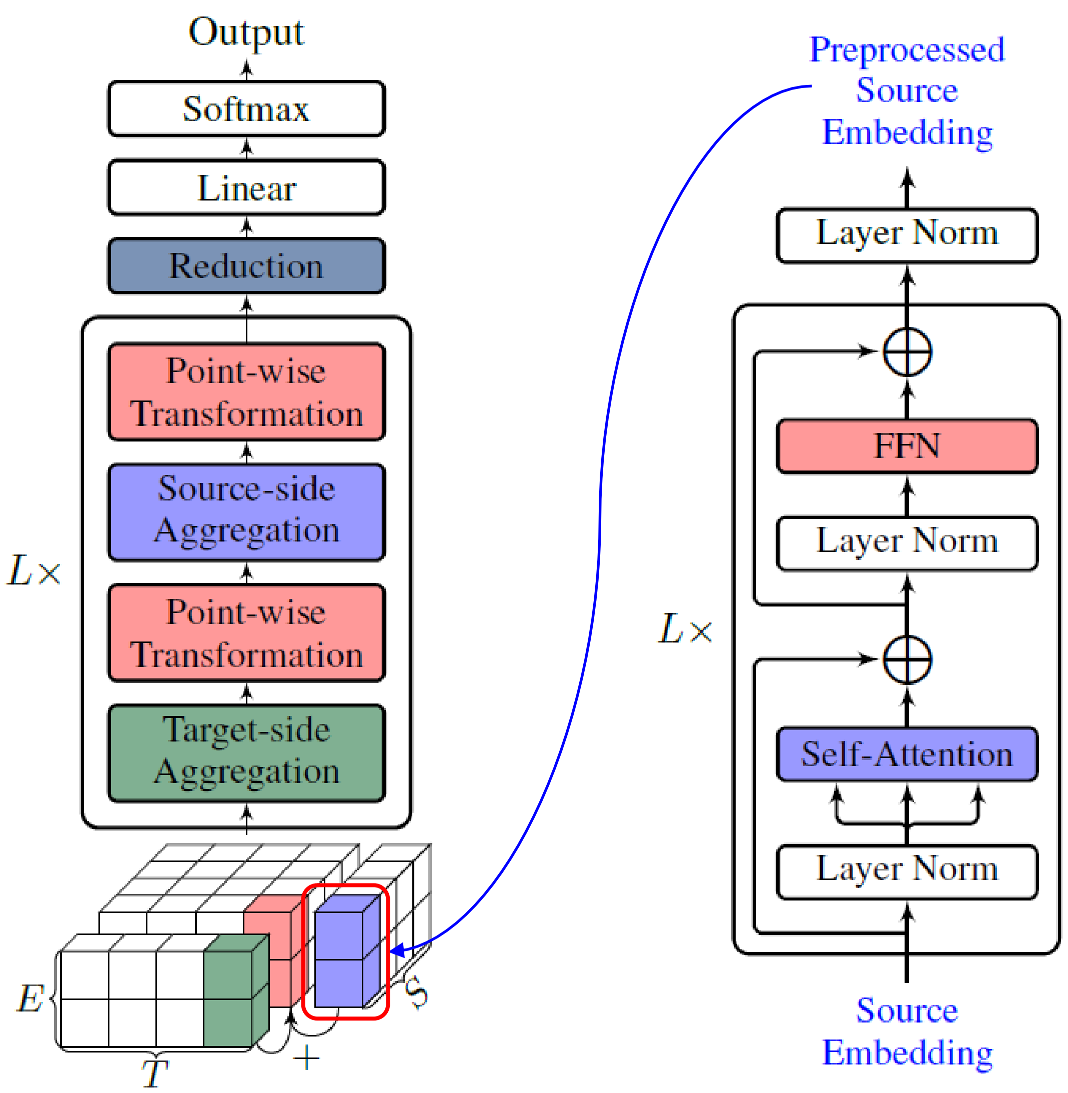
- Reformer-fast

A Faster Model



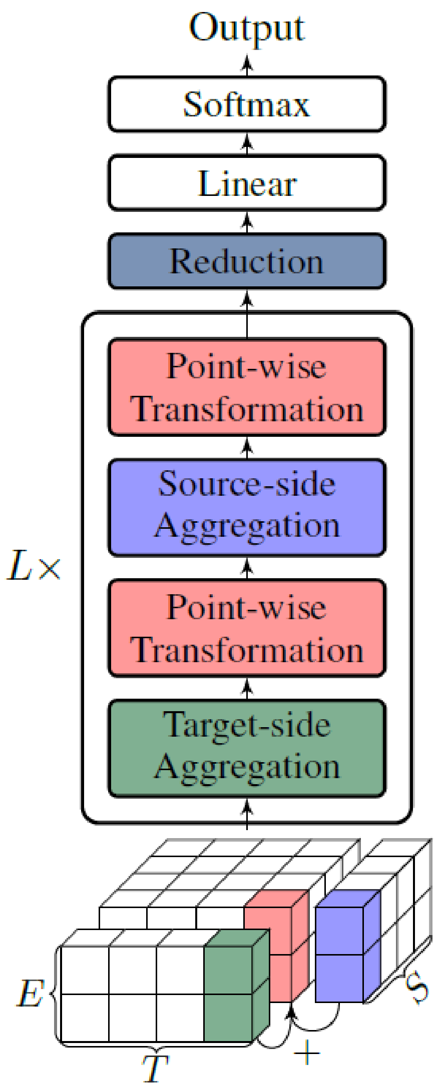
- Reformer-fast
  - PreNet processes the source embeddings first

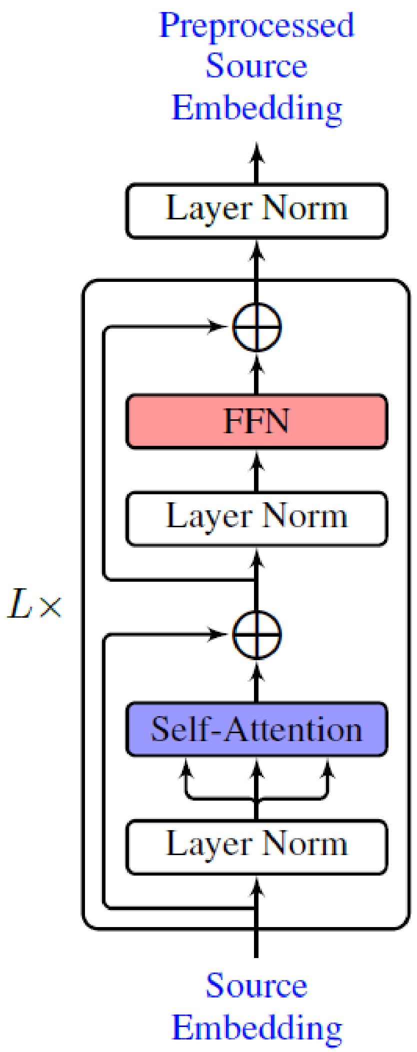
A Faster Model



- Reformer-fast
  - PreNet processes the source embeddings first
  - Then the output is used as the original source embeddings in Reformer-base

A Faster Model





- Reformer-fast
  - PreNet processes the source embeddings first
  - Then the output is used as the original source embeddings in Reformer-base
- Pros & Cons
  - PreNet reduces #Separable-Attention & has low complexity
  - But increase the path length from $O(1)$ to $O(L)$ for accessing any source token

A Larger Model

- How to increase the model capacity?
  - Enlarging both the embedding size and hidden dims as Transformer-big does not work in Reformer

A Larger Model

- How to increase the model capacity?
  - Enlarging both the embedding size and hidden dims as Transformer-big does not work in Reformer
  - Obtaining a larger model is equal to perform gradient descent with the step size α on both the network height l and width w to optimize the validation set performance $\mathcal{L}$ with the constraint β on the number of parameters P

$$\max_{\alpha} \mathcal{L}(l + \alpha\mathcal{L}'_l, w + \alpha\mathcal{L}'_w)$$

$$\text{s.t.} \ \frac{P(l + \alpha\mathcal{L}'_l, w + \alpha\mathcal{L}'_w)}{P(l, w)} \approx \beta$$

  - We estimate the gradient $\mathcal{L}'$ by its definition (take height l as the example)

$$\mathcal{L}'_l = \lim_{\delta \to 0} \frac{\mathcal{L}(l + \delta, w) - \mathcal{L}(l, w)}{\delta} \approx \frac{\mathcal{L}(l + \epsilon, w) - \mathcal{L}(l, w)}{\epsilon}$$

  - ϵ is a small number that is manually defined

Experiments

- Setup
  - Corpus: IWSLT15 (Vi-En), IWSLT14 (De-En, En-De) and NIST12 (Zh-En)
  - Baseline: Transformer-small/base, 256/512 embedding size, 1024/2048 hidden dim, 6 layers
  - Ours: similar to the baseline, except 7/5 layers for Reformer-base/fast

Experiments

- Setup
  - Corpus: IWSLT15 (Vi-En), IWSLT14 (De-En, En-De) and NIST12 (Zh-En)
  - Baseline: Transformer-small/base, 256/512 embedding size, 1024/2048 hidden dim, 6 layers
  - Ours: similar to the baseline, except 7/5 layers for Reformer-base/fast
- Results
  - Both Reformer-base & fast outperform the baseline in all test sets
  - Reformer-base and Reformer-fast are of similar performances

System	Vi-En		De-En		En-De		Zh-En		
	tst2012	tst2013	valid	test	valid	test	MT06	MT05	MT08
baseline	24.70	27.53	34.44	33.63	28.19	27.54	49.63	48.23	43.10
Reformer-base	24.42	27.18	**35.87**	**34.92**	**29.42**	28.32	50.00	48.72	**45.04**
Reformer-fast	**24.98**	**28.26**	**35.87**	34.87	29.31	**28.36**	**50.82**	**49.29**	44.64

Experiments

- Ablation Study
  - Dropout 1/2d improves the generalization
  - PreNet improves efficiency

System	PPL	BLEU	Params	Speed
baseline	5.39	34.44	16M	1×
Reformer	5.00	35.16	16M	0.47×
+Dropout 1/2d	4.82	$\boxed{35.87}$	16M	0.52×
+PreNet	4.89	35.87	17M	$\boxed{0.73×}$

Experiments

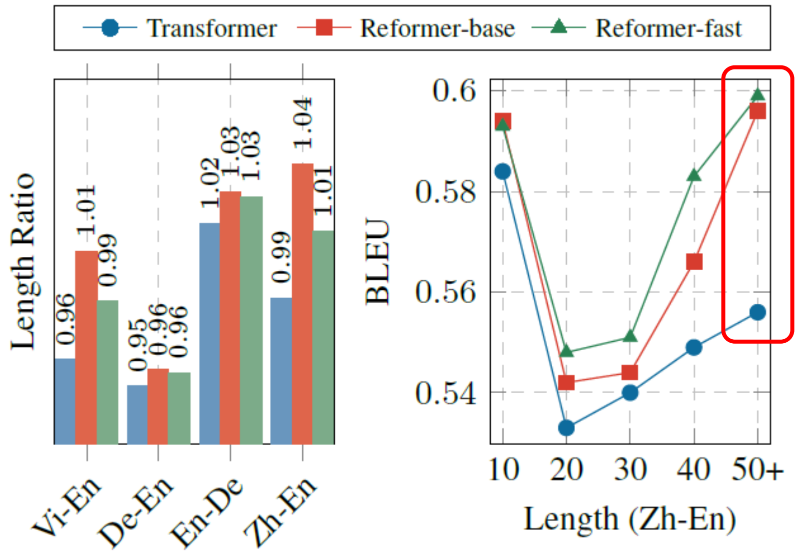
- Ablation Study
  - Dropout 1/2d improves the generalization
  - PreNet improves efficiency
- Larger Models
  - Reformer-fast always add 2 layers and 50% hidden dim with $\beta = 2$
  - Reformer-fast outperforms Transformer-big with ~50% fewer parameters

System	PPL	BLEU	Params	Speed
baseline	5.39	34.44	16M	1×
Reformer	5.00	35.16	16M	0.47×
+Dropout 1/2d	4.82	35.87	16M	0.52×
+PreNet	4.89	35.87	17M	0.73×

System	De-En		Zh-En	
	test	Params	MT08	Params
baseline	33.63	16M	43.10	101M
+scaling	34.41	42M	44.60	291M
Reformer-fast	34.87	17M	44.64	105M
+scaling	**35.11**	27M	**46.66**	146M

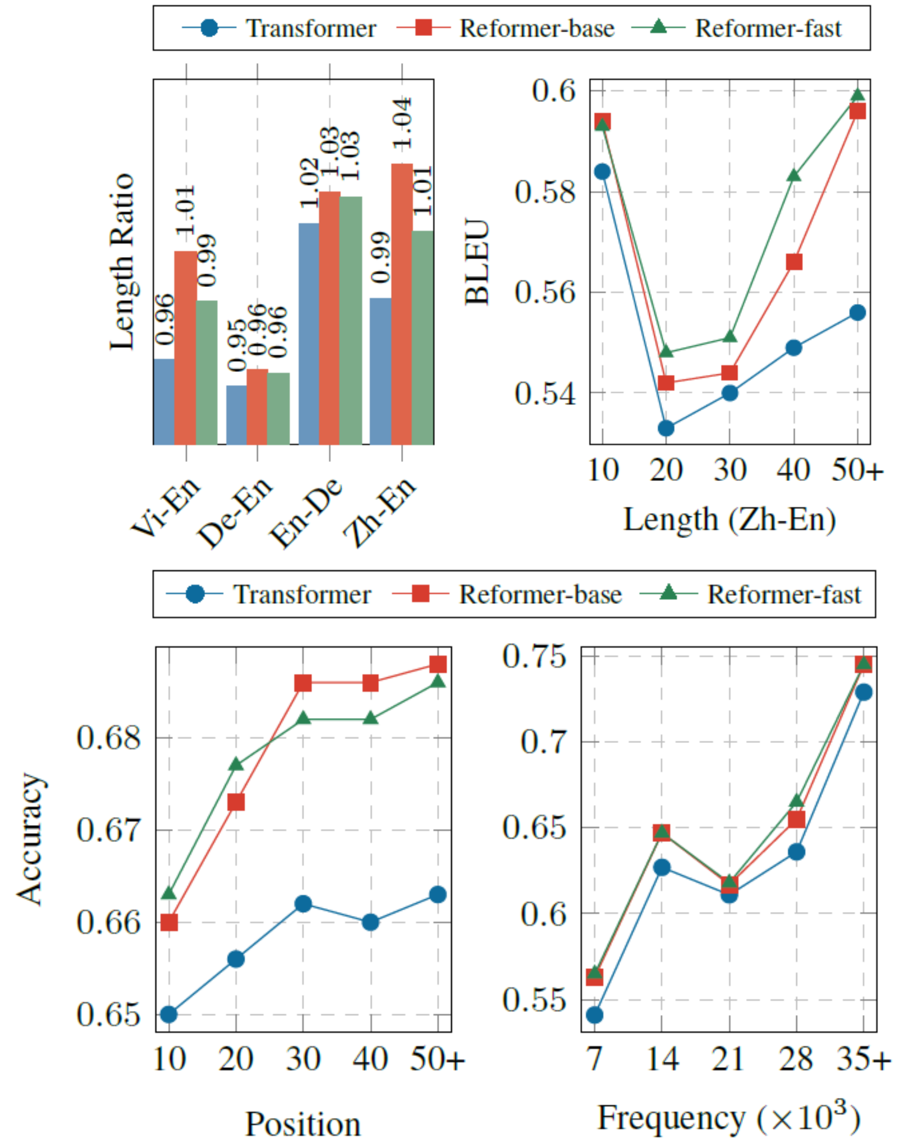
Experiments

- Analysis
  - Our models tend to produce long translations
  - Our models perform better for long sentences



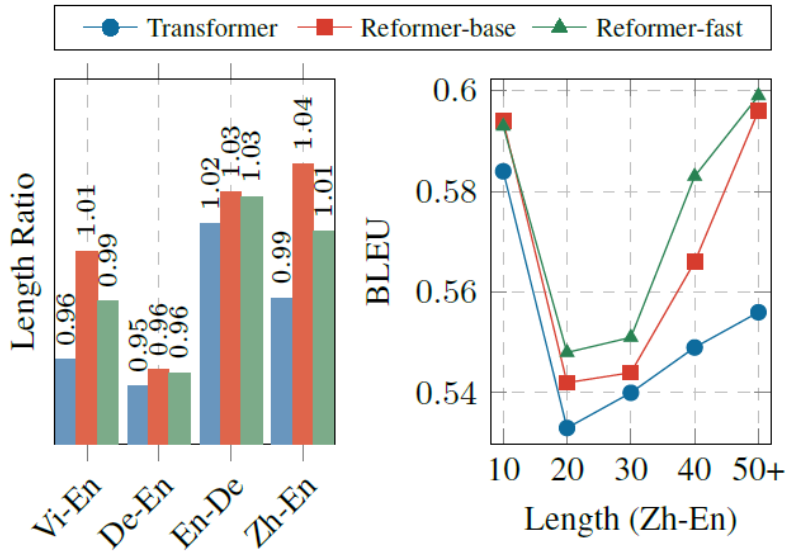
Experiments

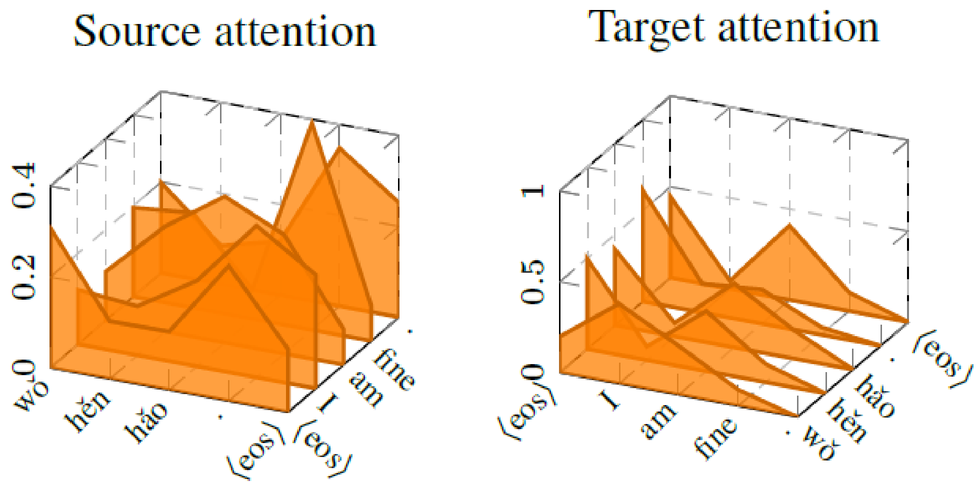
- Analysis
  - Our models tend to produce long translations
  - Our models perform better for long sentences
  - Our models have higher accuracies than the baseline

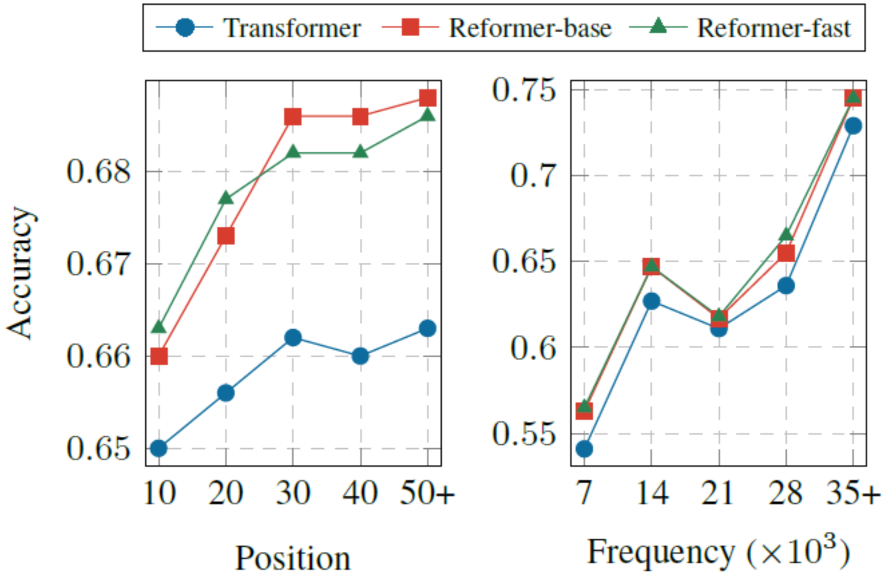


Experiments

- **Analysis**
  - Our models tend to produce long translations
  - Our models perform better for long sentences
  - Our models have higher accuracies than the baseline
  - The case study shows the attention distribution varies if it conditions on different source/target tokens







Conclusion

- Propose two attention-based models built on top of joint representation.
- They outperform the Transformer baseline in either the base or the big setup in various datasets.
- These models are still primitive and we expect more future work on them.
- The code is publicly available at https://github.com/lyy1994/reformer.

Thank you :)

东北大学自然语言处理实验室

Natural Language Processing Laboratory at Northeastern University