Deep
Natural Language Understanding

22-Jan-2020

Omprakash Sonie
Flipkart

Course
Deep Learning for Natural Language Processing

Agenda:

- Advanced approaches
  - **Transformer**
  - BERT
  - Transformer-XL
  - XLNet
  - MT-DNN

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

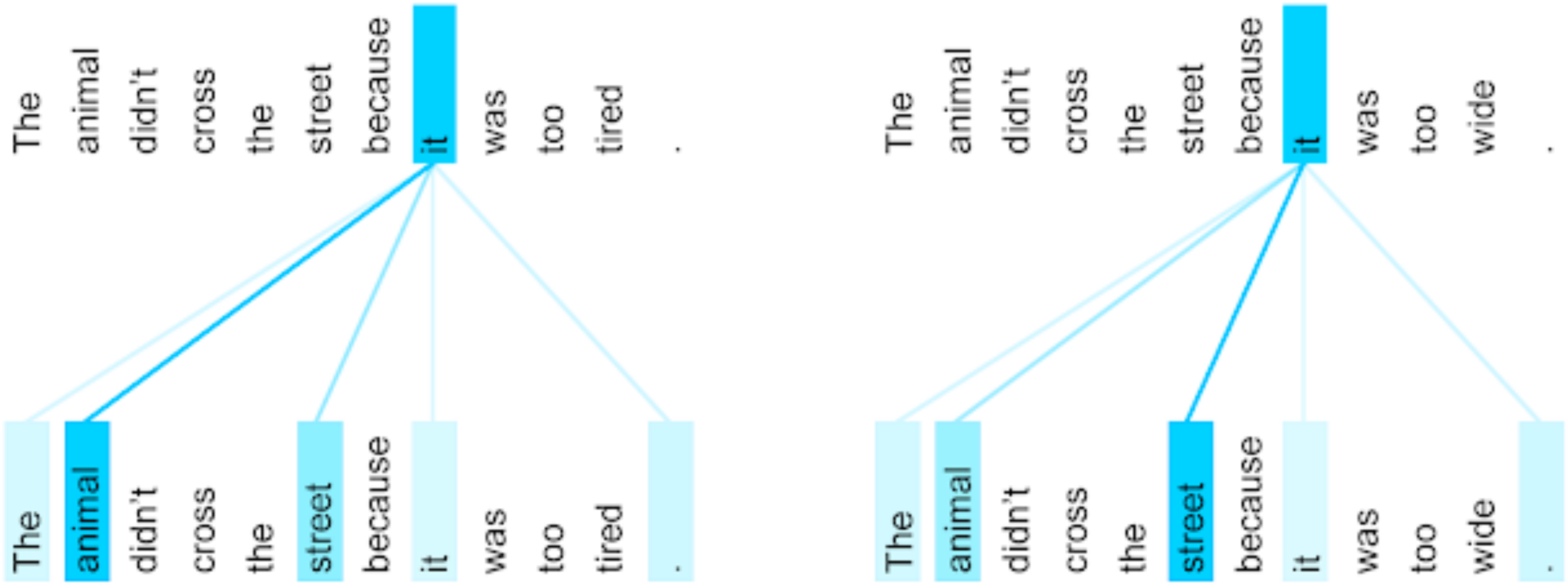
Llion Jones* Google Research llion@google.com

Aidan N. Gomez*† University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* illia.polosukhin@gmail.com

Attention



The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigué.

The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.

Agenda

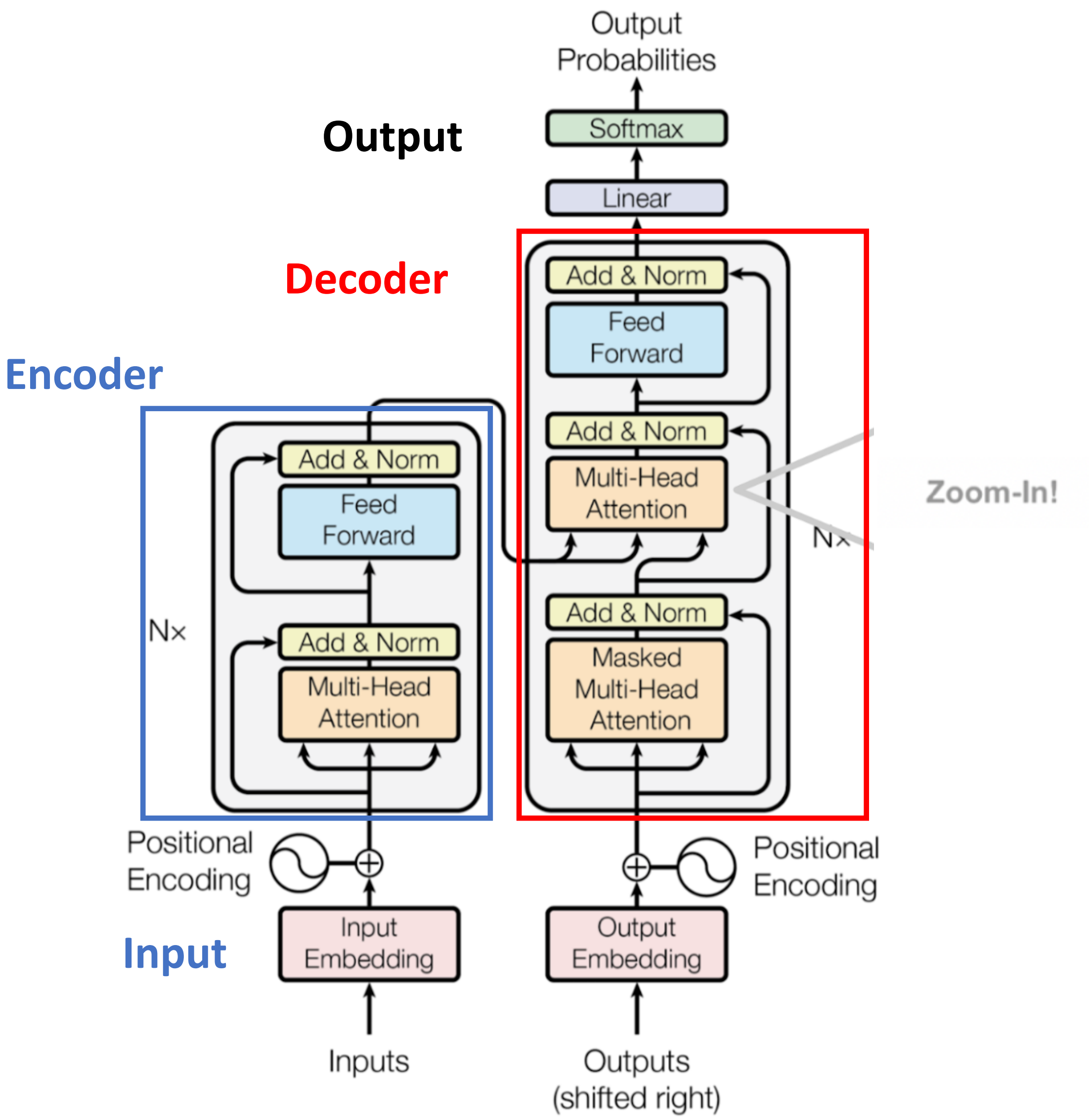
- Provide foundation for
  - BERT (Bidirectional Encoder Representation from Transformers)
- Assumption: Familiarity with
  - RNN/LSTM, Encoder-Decoder Architecture and Attention Mechanism

Why Transformer Network

- **Addresses drawbacks of RNN based architecture**
  - Hard to parallelize
  - Difficulty in learning long range dependencies
  - Complex
- **It uses only attention – No RNN or CNN**

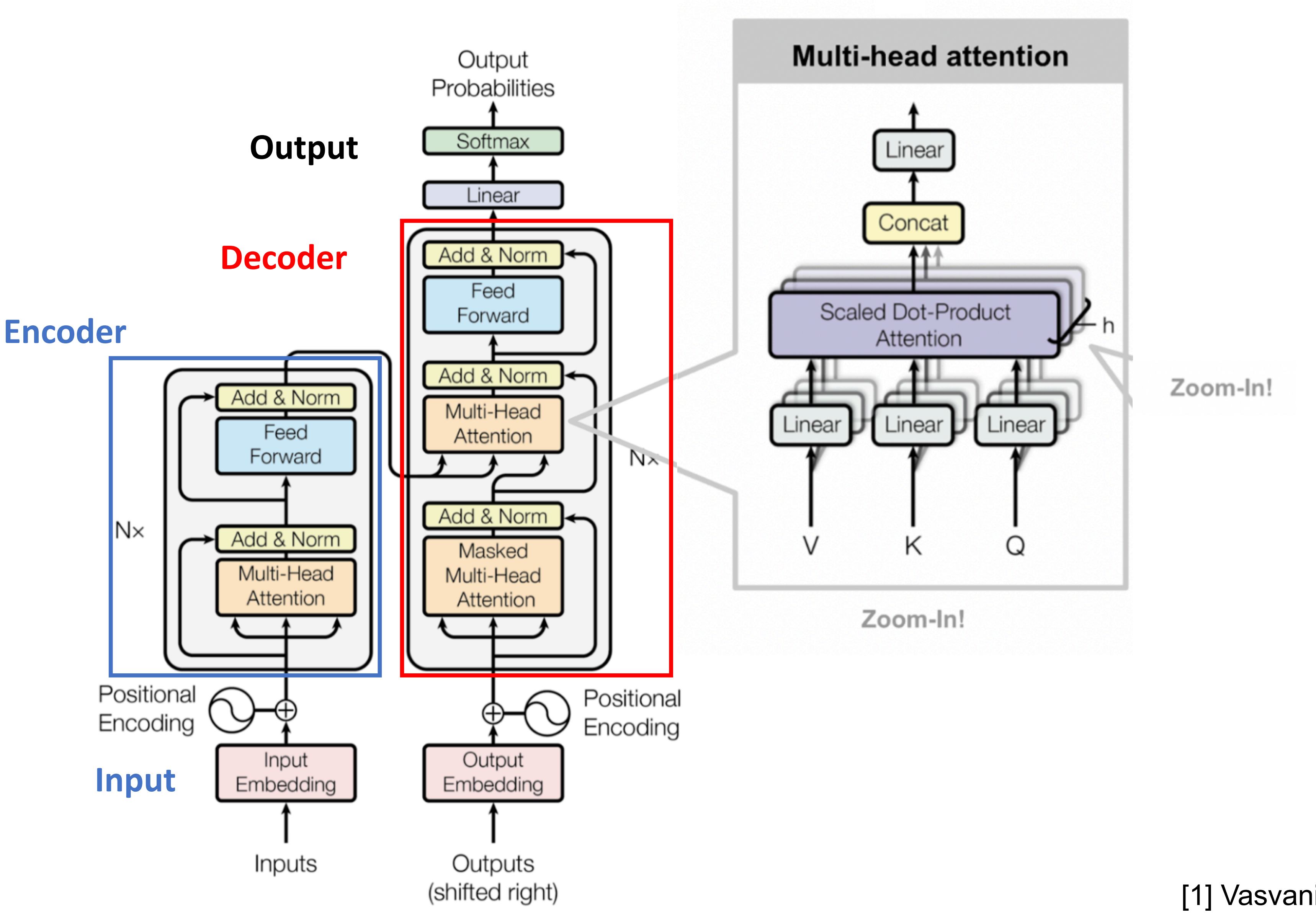
[1] Vasvani A., et. al. Attention Is All You Need

Full Model Architecture



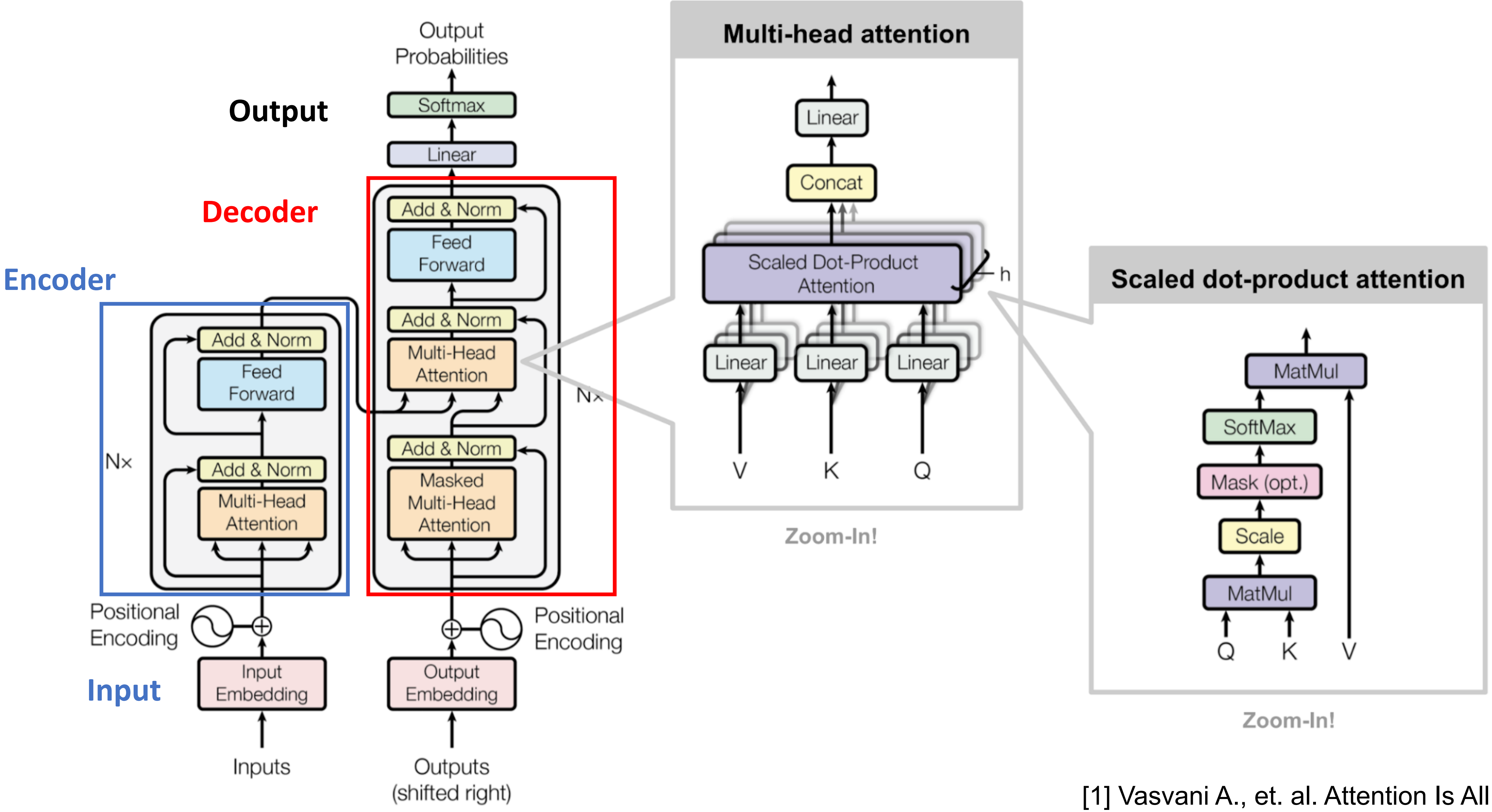
[1] Vasvani A., et. al. Attention Is All You Need

Full Model Architecture



[1] Vasvani A., et. al. Attention Is All You Need

Full Model Architecture

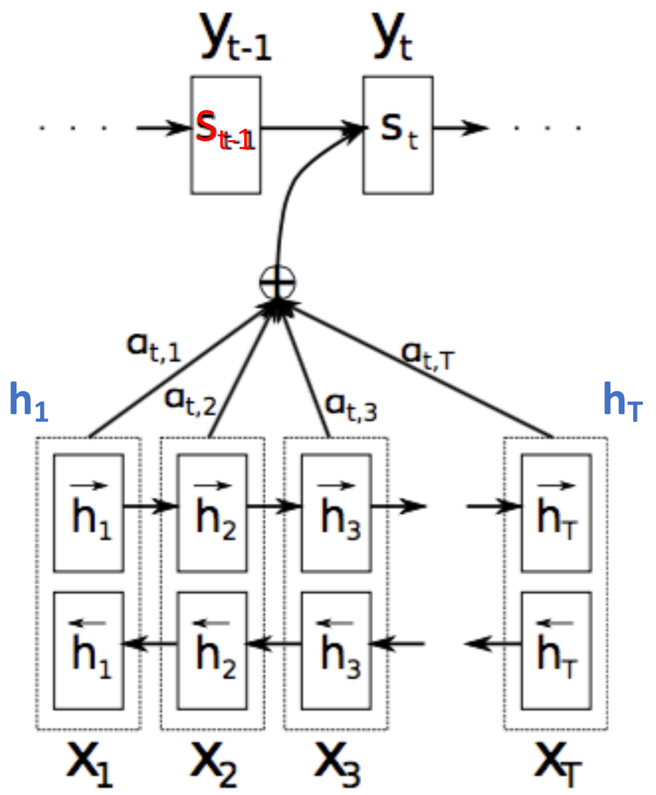


[1] Vasvani A., et. al. Attention Is All You Need

Scaled Dot Product Attention

- **Dot-Product Attention**
  - Query(Q): S_{t-1}
  - Keys(K): $[h_1, h_2, \ldots, h_T]$
  - Values(V): $[h_1, h_2, \ldots, h_T]$

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(QK^T)V$$



[1] Vasvani A., et. al. Attention Is All You Need

Scaled Dot Product Attention

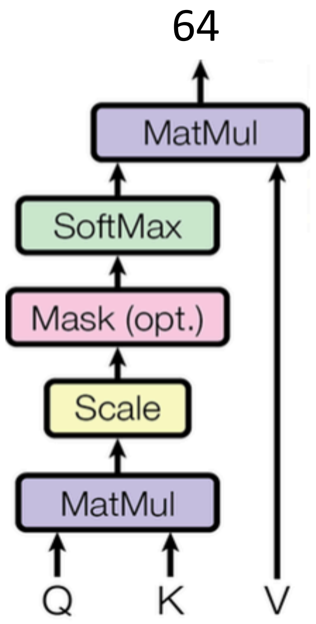
- **Dot-Product Attention**
  - Query(Q): S_{t-1}
  - Keys(K): $[h_1, h_2, \ldots, h_T]$
  - Values(V): $[h_1, h_2, \ldots, h_T]$

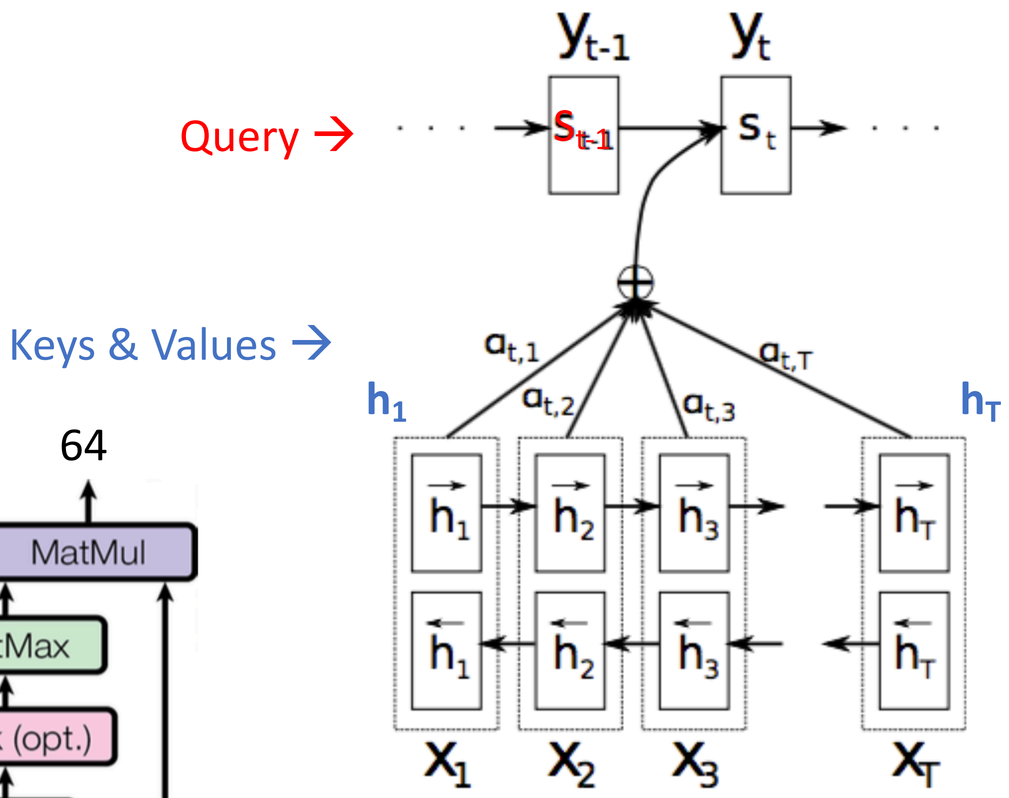
$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V$$

- **Scaling**
  - For large dimension, QK^T will explode
  - Softmax --> extremely small

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

d_k is dimension of key (e.g. 64)





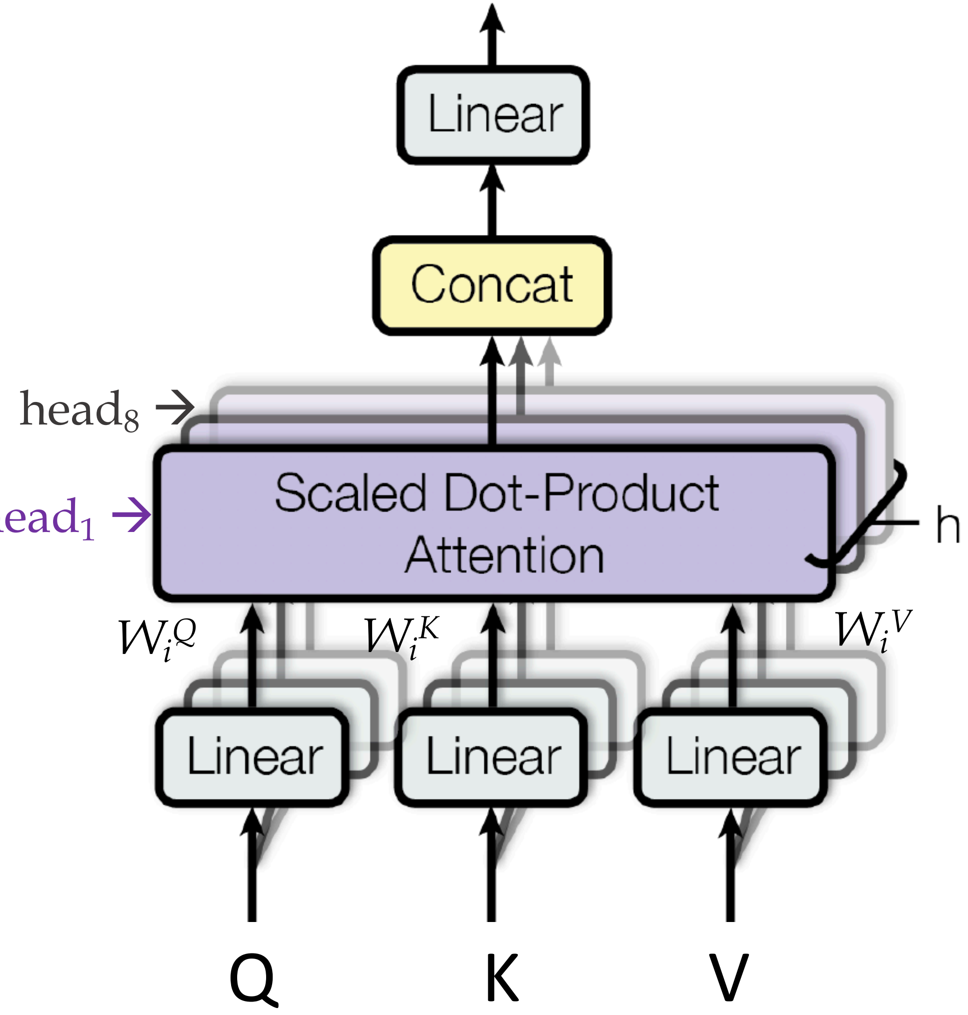
[1] Vasvani A., et. al. Attention Is All You Need

Multi Head Attention

- Apply different attention at different positions
  - Split Q, K, V of size 512 into 8 parts of size 64
  - Calculate attention in 8 different heads

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)$$

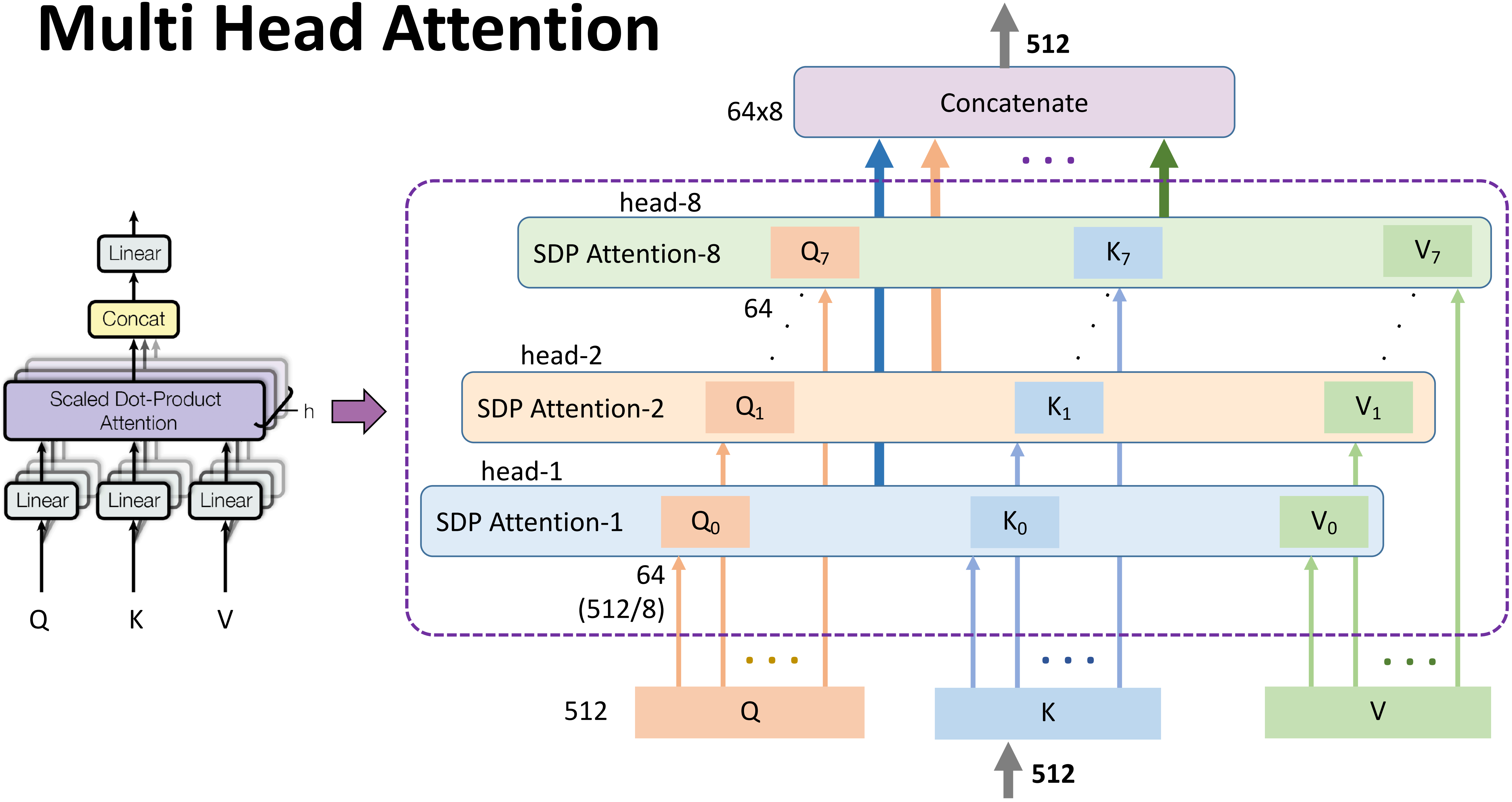
$$\mathrm{head}_1 = \mathrm{Attention}(QW_1^Q, KW_1^K, VW_1^V)$$



Attention on
- Projection of Query with Weight W_1^Q
- Projection of Key with Weight W_1^K
- Projection of Value with Weight W_1^V

[1] Vasvani A., et. al. Attention Is All You Need

Multi Head Attention



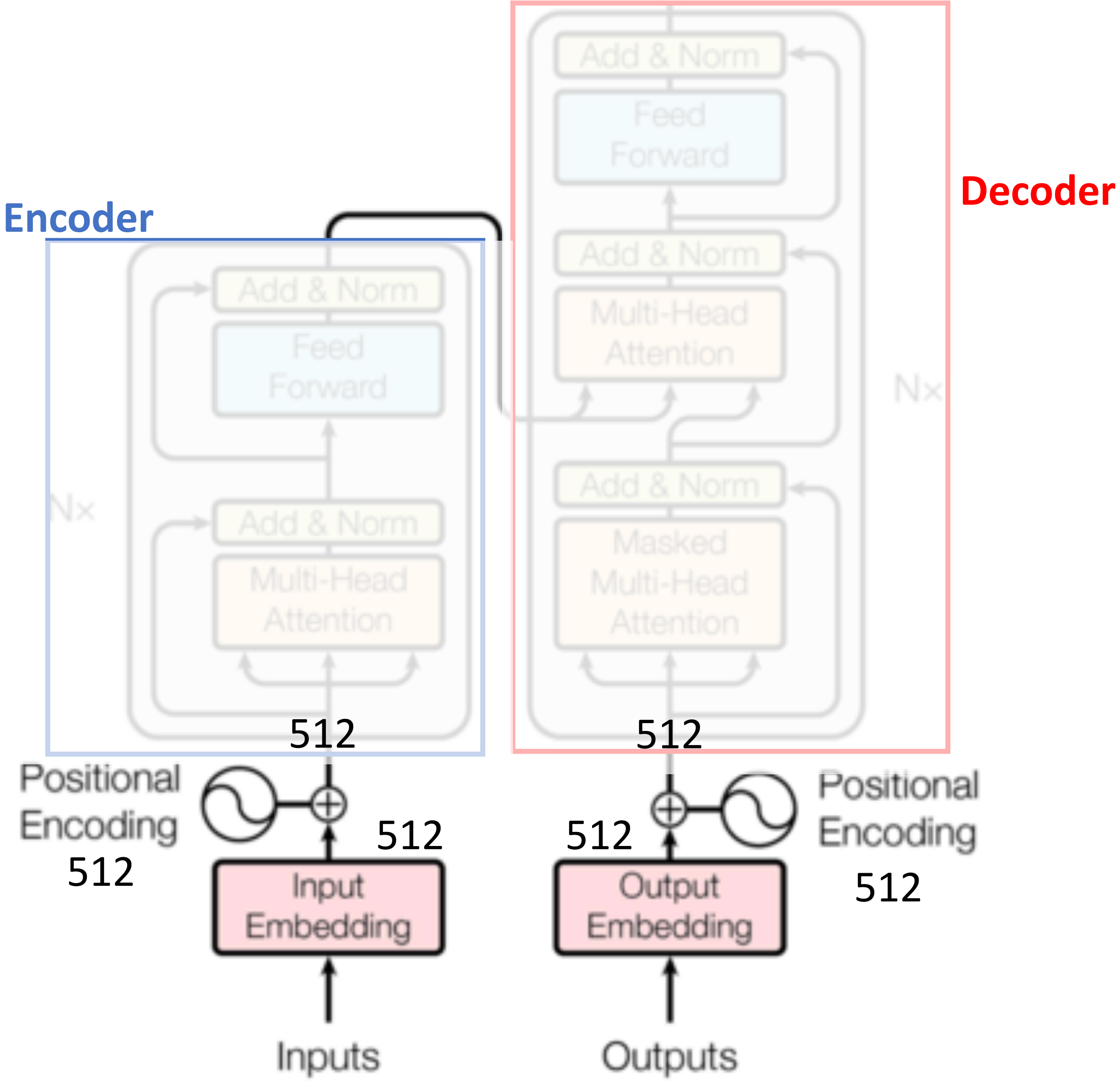
[1] Vasvani A., et. al. Attention Is All You Need

Embedding and Positional Encoding

- **Embedding:**
  - Source and target sequences (tokens)
  - d_{model} = 512

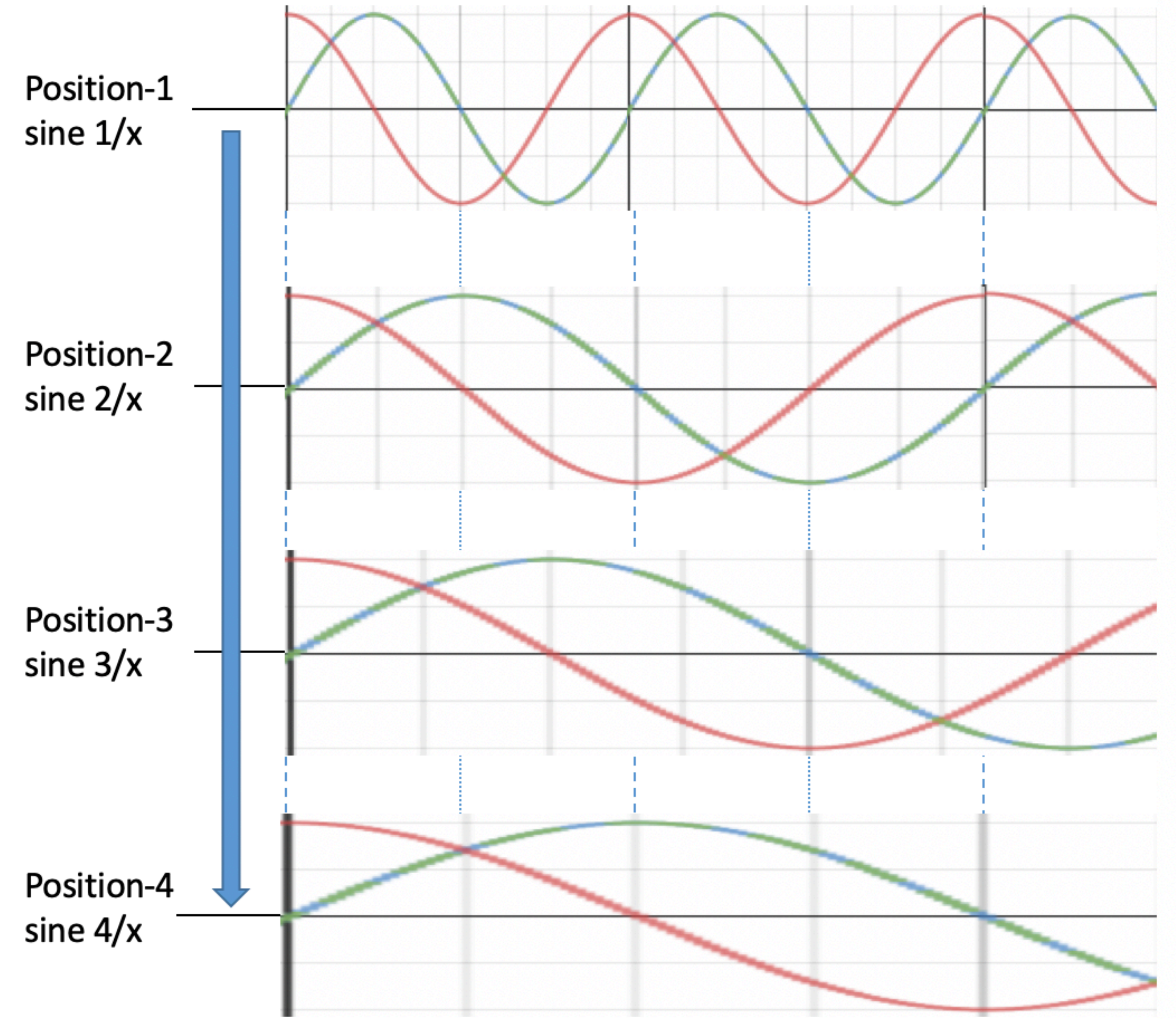
- **Positional Encoding (fixed):**
  - Need to include position information
  - Use sine and cosine functions of different frequencies
  - Dimension = 512

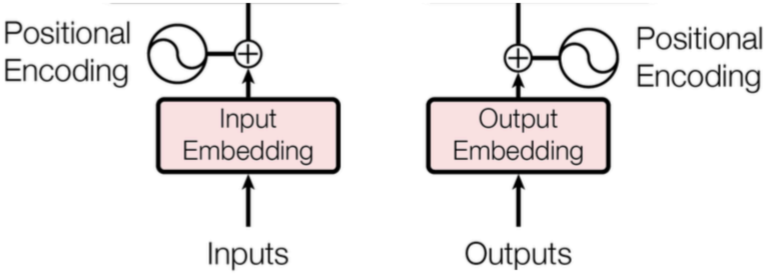
- **Embedding and Positional Encoding**
  - Are added and fed to Encoder and Decoder



[1] Vasvani A., et. al. Attention Is All You Need

Positional Encoding



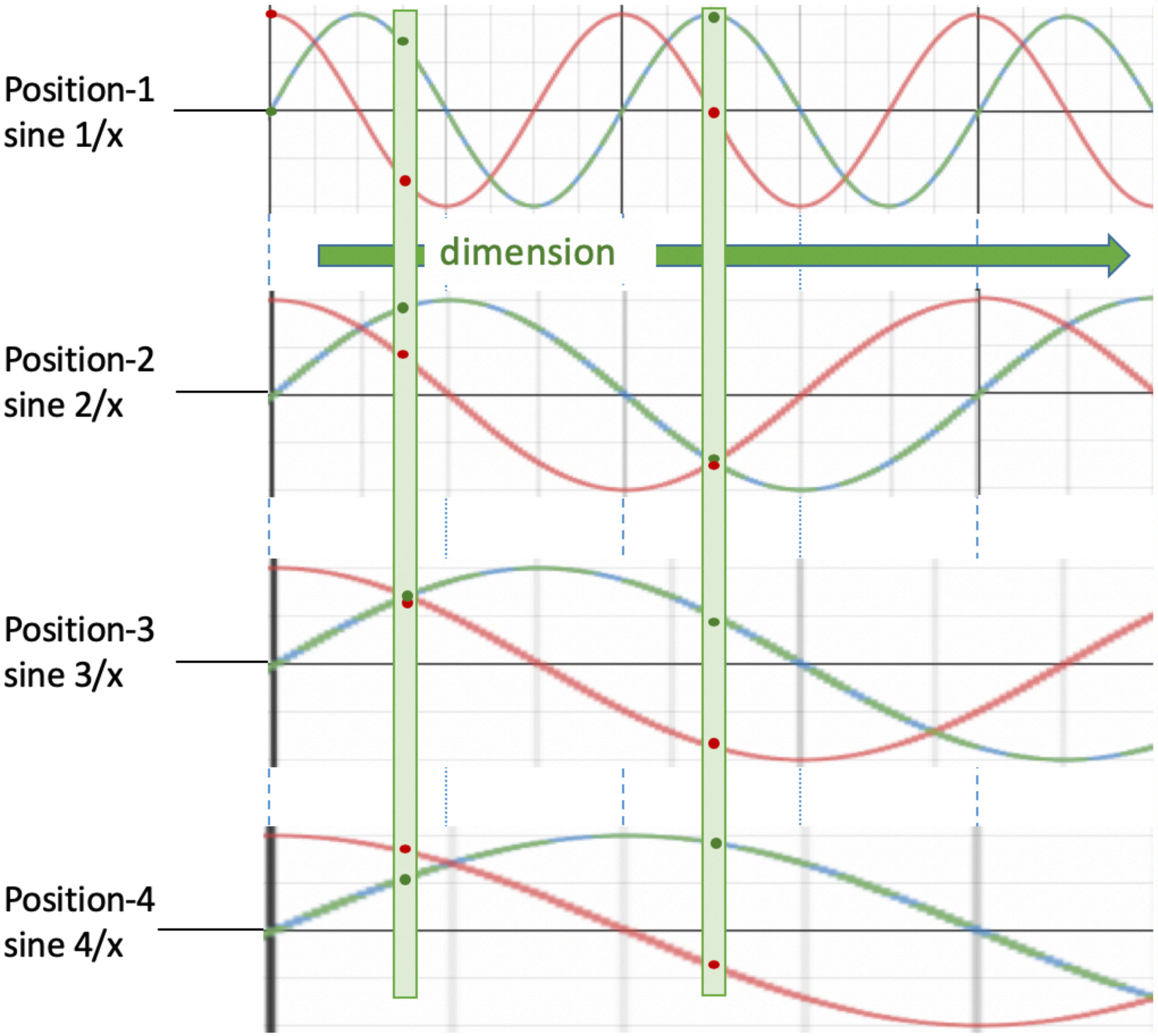


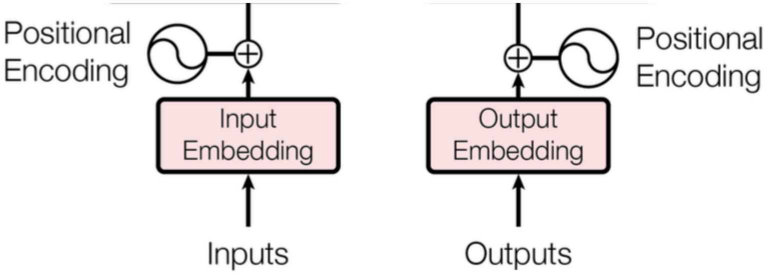
- Use sine and cosine functions of different frequencies

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

[1] Vasvani A., et. al. Attention Is All You Need

Positional Encoding

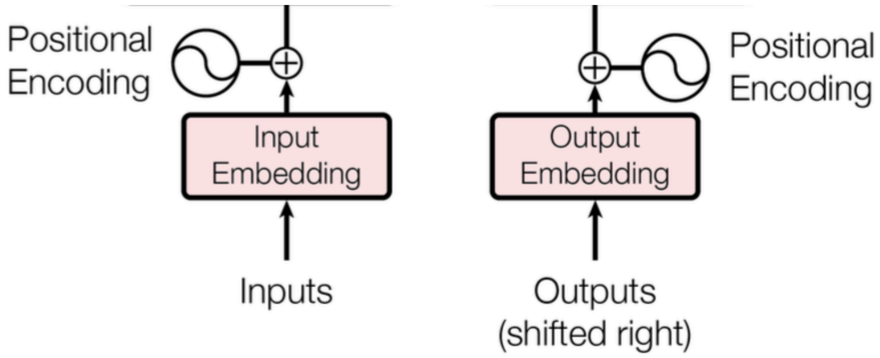




$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

[1] Vasvani A., et. al. Attention Is All You Need

Positional Encoding



- Easily learn to attend by relative positions
- Any fixed offset k, PE(pos+k) can be represented as a linear function of PE(pos)

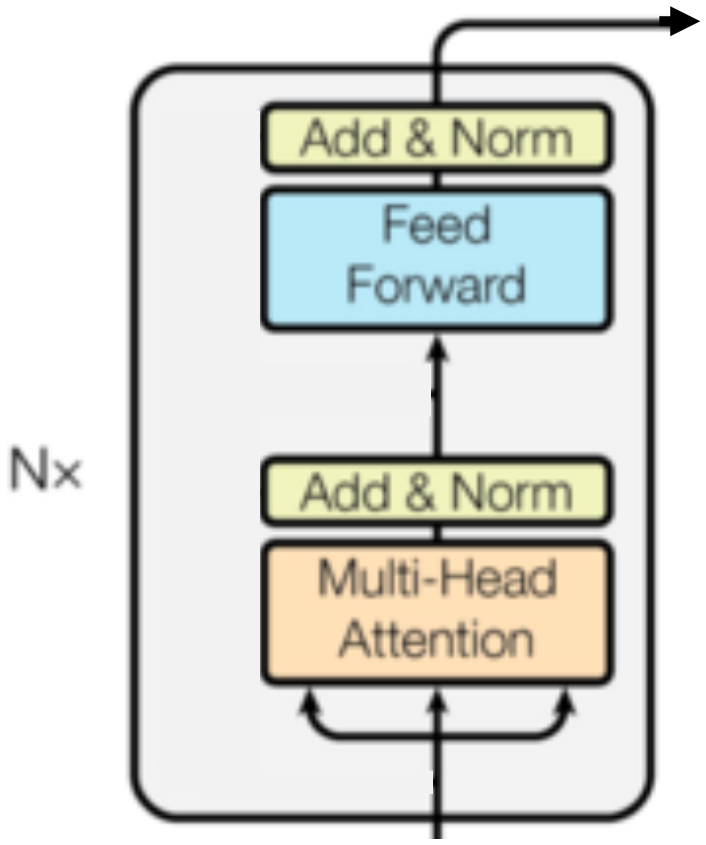
- $$\begin{bmatrix} \sin(pos+k) \\ \cos(pos+k) \\ \dots \end{bmatrix} = \begin{bmatrix} \sin(pos)\cos(k) + \cos(pos)\sin(k) \\ \cos(pos)\cos(k) - \sin(pos)\sin(k) \\ \dots \end{bmatrix}$$

- Model will extrapolate to sequence lengths longer than the ones seen during training

[1] Vasvani A., et. al. Attention Is All You Need

Encoder

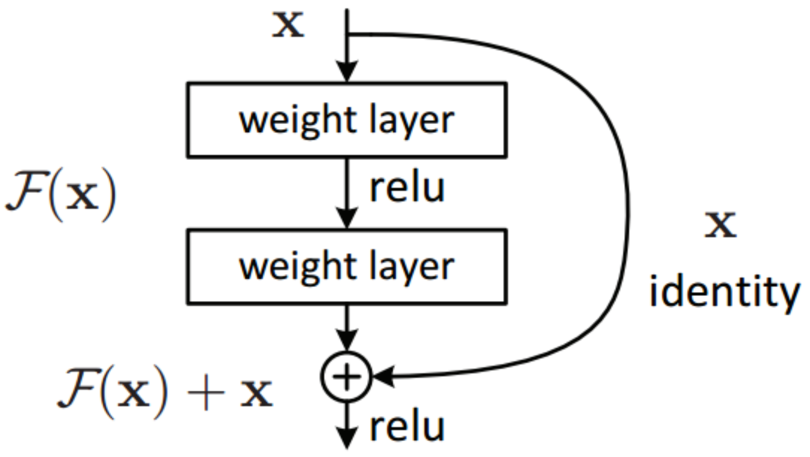
- **Six Layers (stacked)**
- **Each layer has two sub-layers**
  - Multi-head attention (self-attention)
  - Feed-Forward
    - hidden layer d_{ff} = 2048, input and output = 512
    - $FFN(x) = ReLU(xW1 + b1)\,W2 + b2$

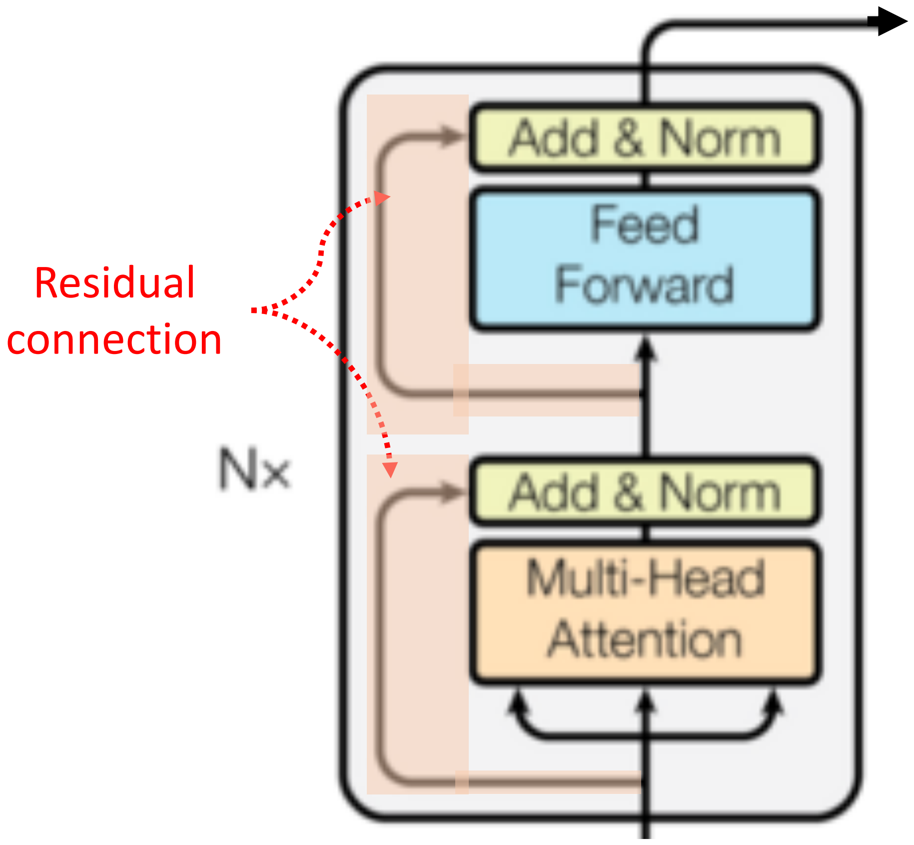


[1] Vasvani A., et. al. Attention Is All You Need

Encoder

- **Six Layers (stacked)**
- **Each layer has two sub-layers**
  - Multi-head attention (self-attention)
  - Feed-Forward
    - hidden layer d_{ff} = 2048, input and output = 512
    - $FFN(x) = ReLU(xW1 + b1)\,W2 + b2$
- **Residual connection**
  - Copy of data is fed to upper layer
  - So that we can train deeper networks





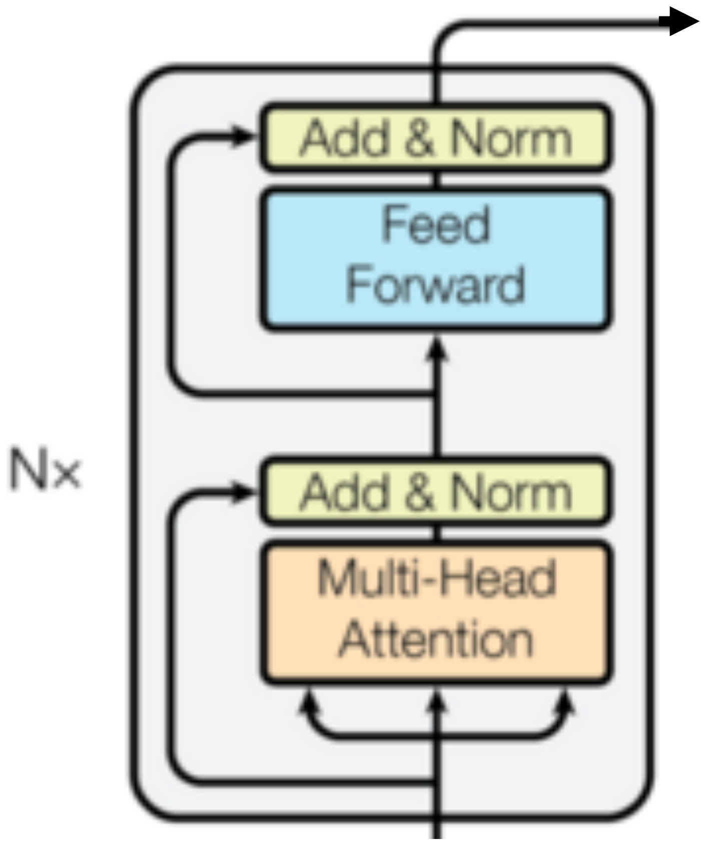
[1] Vasvani A., et. al. Attention Is All You Need

Encoder

Encoder output =

`LayerNorm [x + FFN [LayerNorm [x + MultiHead (Q, K, V)]]]`

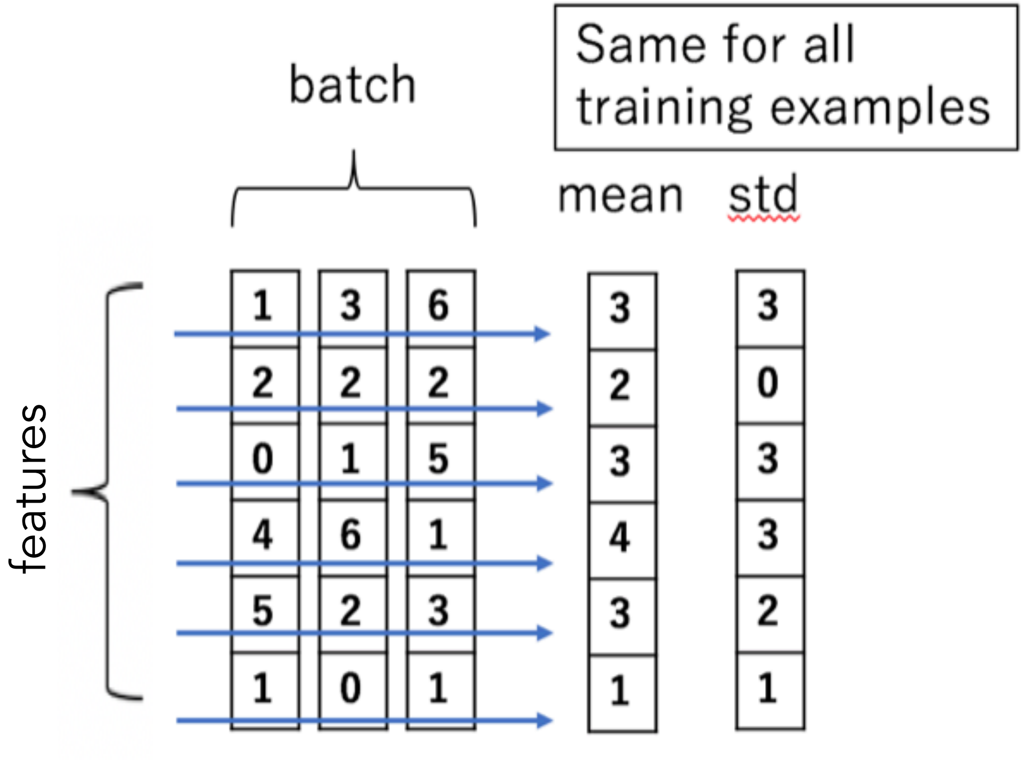
- **Layer normalization**
  - In batch normalization, the statistics are computed across the batch and are the same for each example in the batch
  - In layer normalization, the statistics are computed across each feature and are independent of other examples
  - Faster convergence



[1] Vasvani A., et. al. Attention Is All You Need

Batch Normalization

Batch normalization normalizes the input features **across the batch dimension**

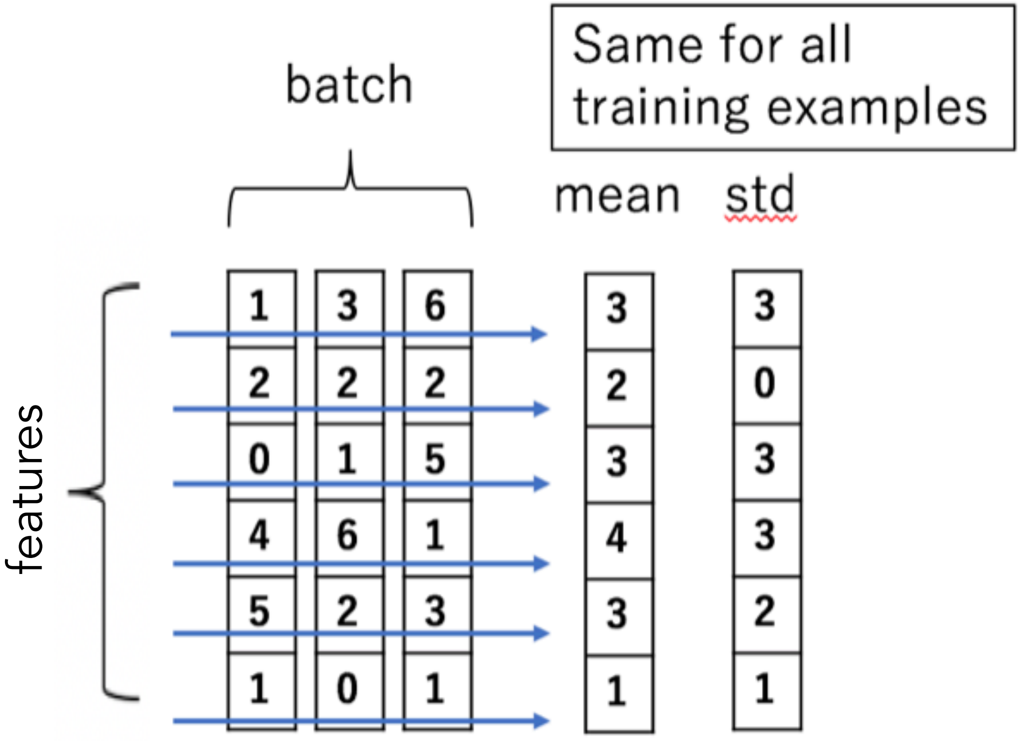


the statistics are computed across the **batch** and are the same for each example in the batch

- Difficult to apply to recurrent connections

Batch Normalization

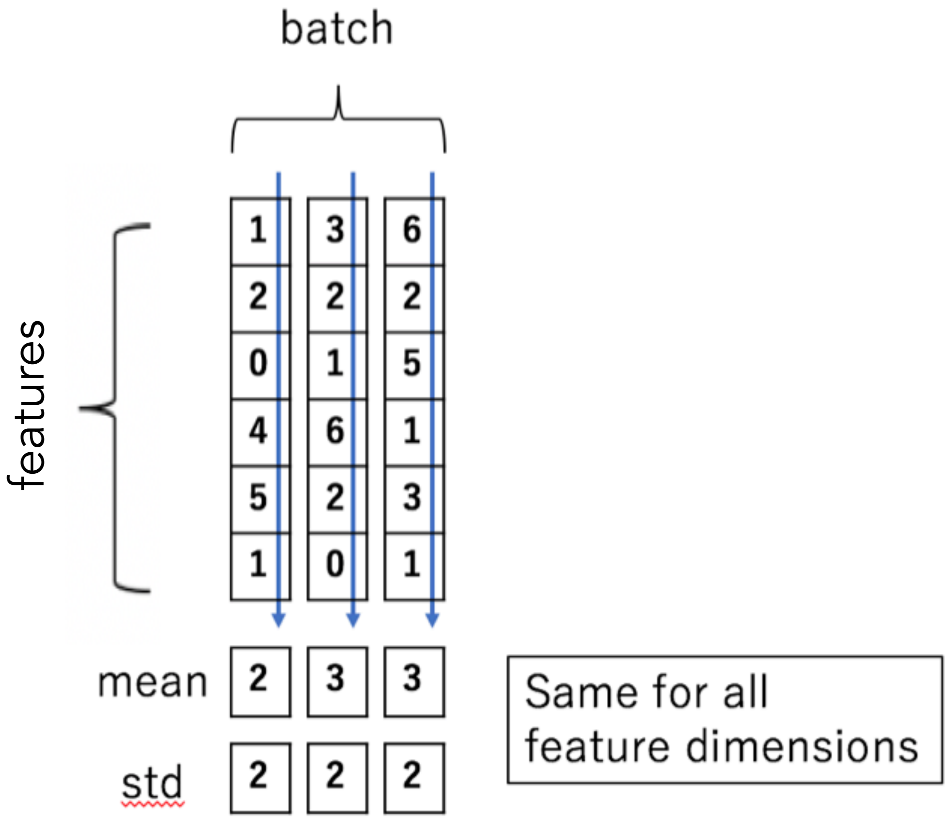
Batch normalization normalizes the input features **across the batch dimension**



the statistics are computed across the **batch** and are the same for each example in the batch

Layer Normalization

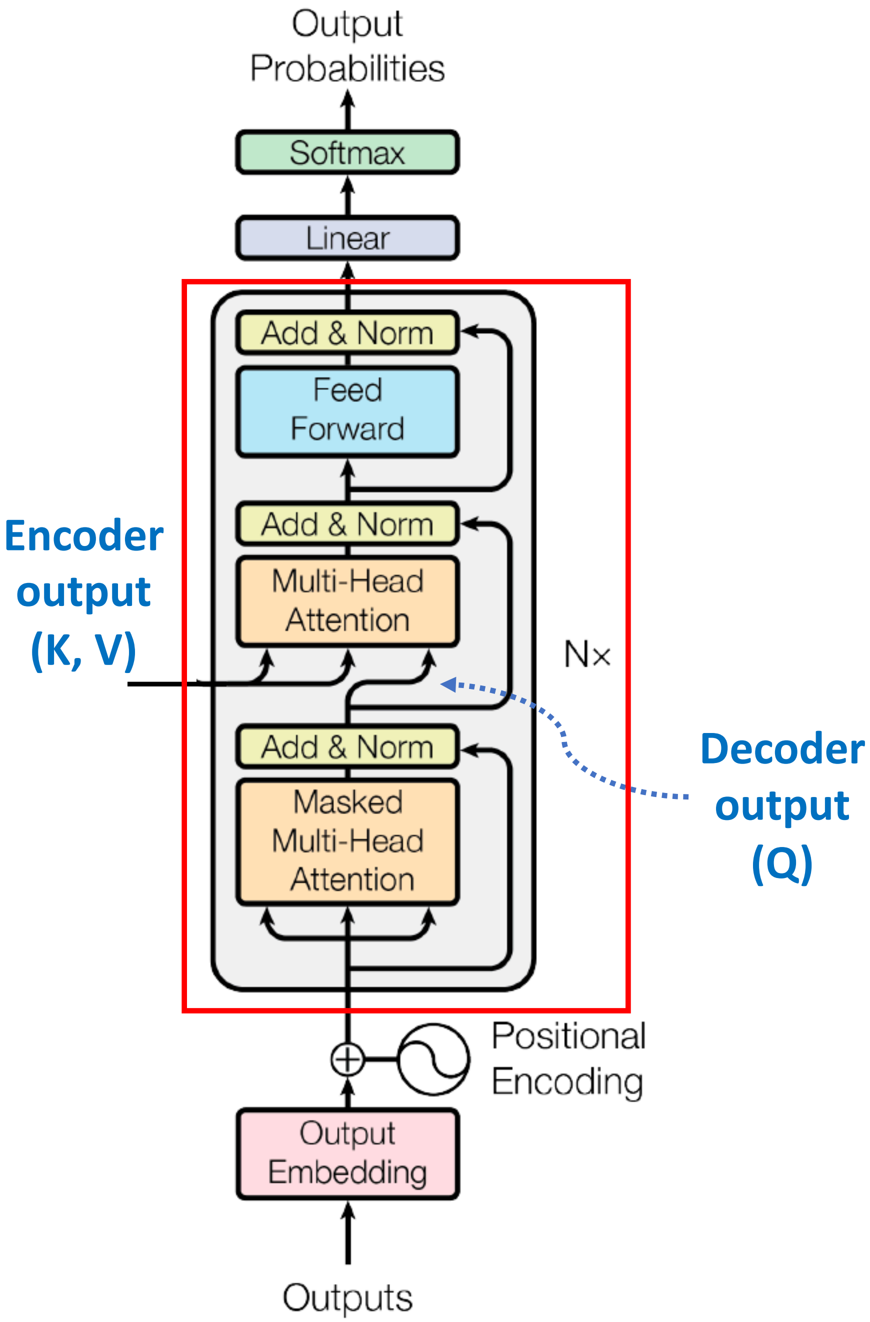
Layer normalization normalizes the **inputs across the features**



statistics are computed across **each feature** and are independent of other examples

Decoder

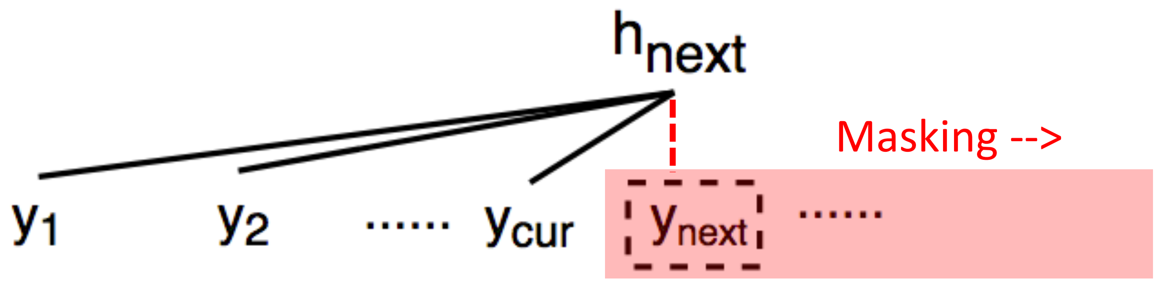
- **Six Layers (stacked)**
- **Each layer has three sub-layers**
  - Masked Multi-head attention
  - Multi-head attention
  - Feed-Forward
- **Multi-head attention**
  - Output of Encoder is fed as K and V
  - Output of Masked Multi Head is fed as Q

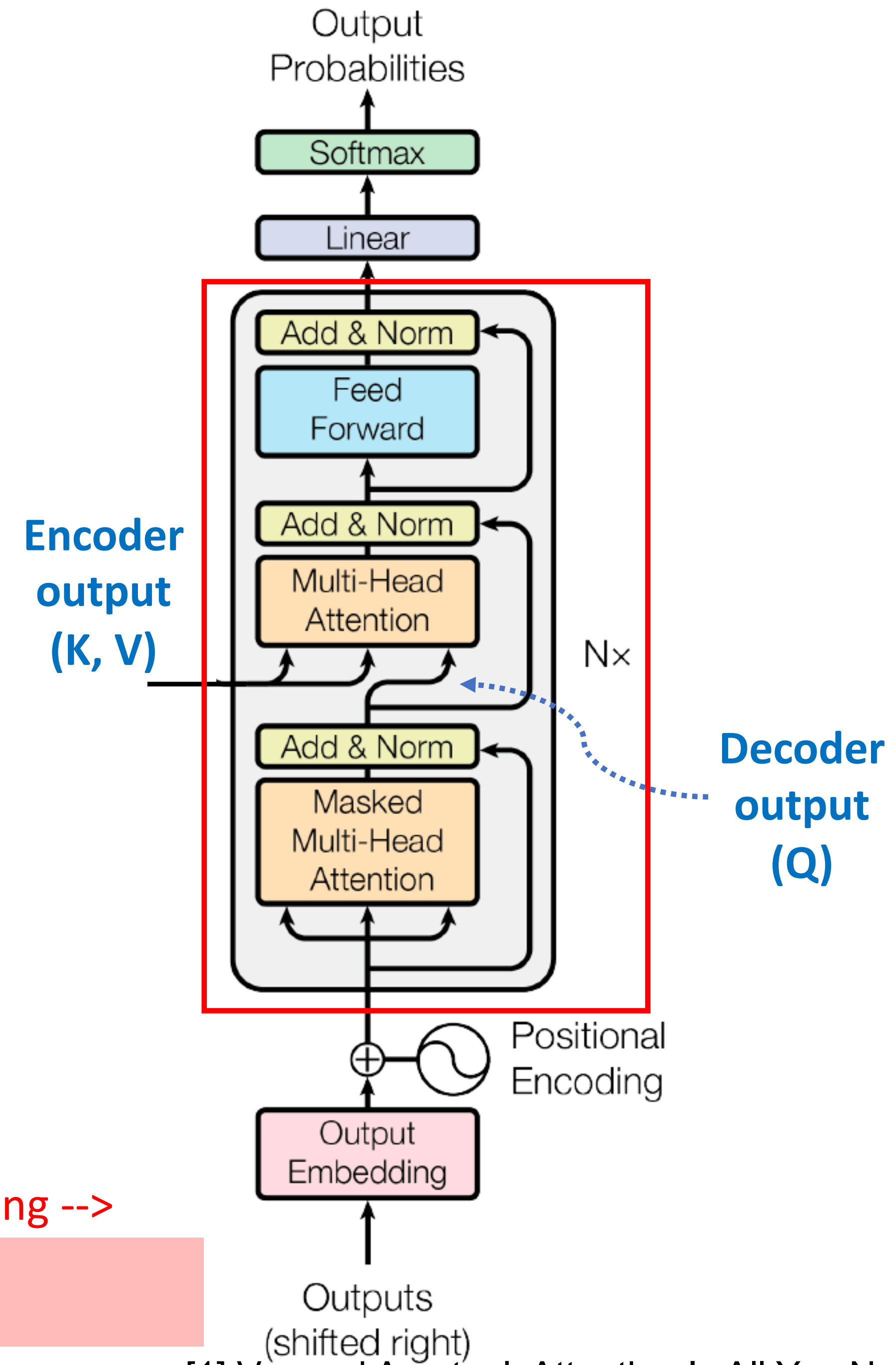


[1] Vasvani A., et. al. Attention Is All You Need

Decoder

- **Masked Multi-head attention**
  - Don’t want to look into future target sequence when predicting current position
  - Mask subsequent positions (shifted right)
  - Output is fed to Multi-head attention as Q



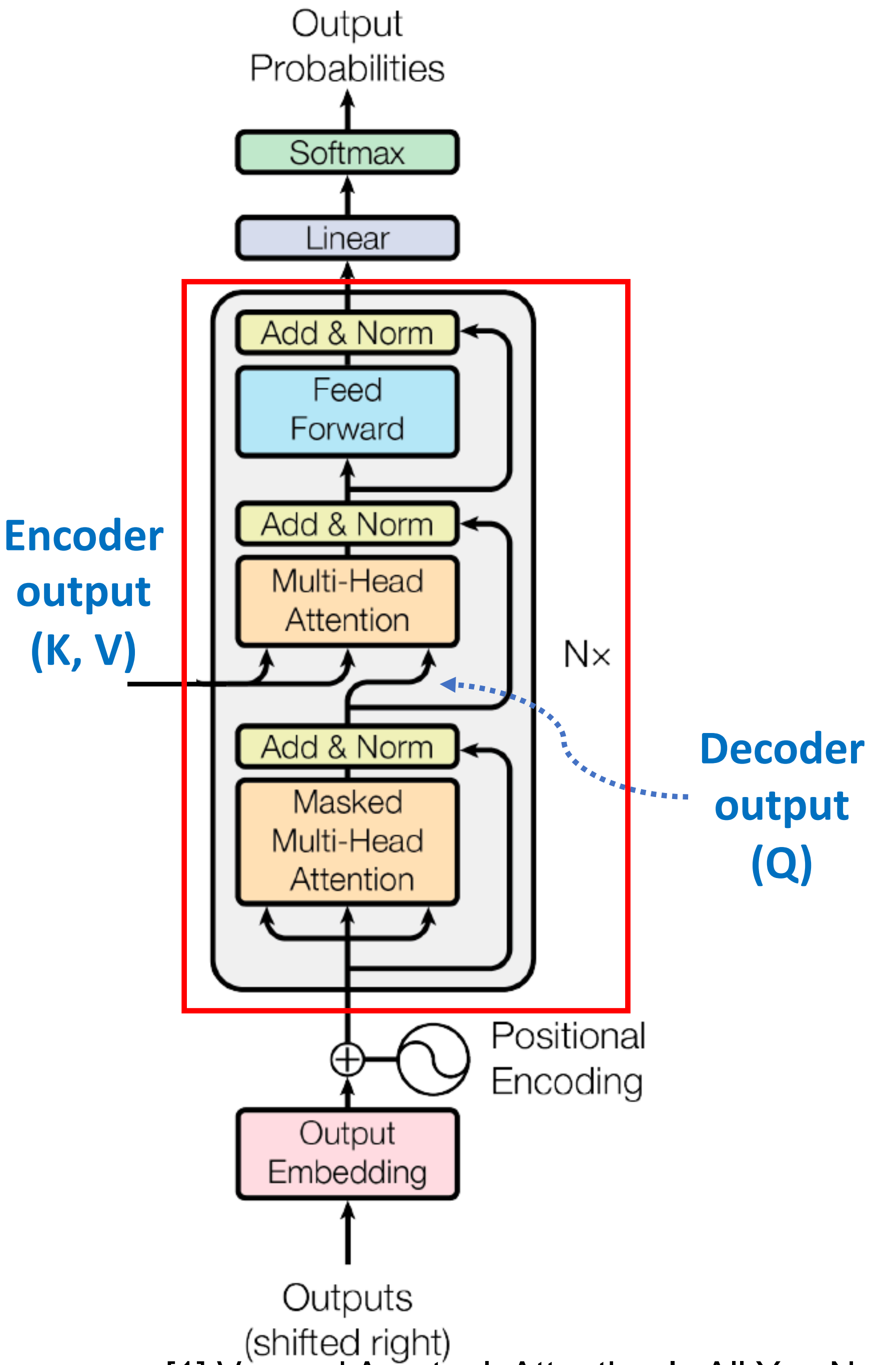


[1] Vasvani A., et. al. Attention Is All You Need

Decoder & Output

- **Masked Multi-head attention**
  - Don’t want to look into future target sequence when predicting current position
  - Mask subsequent positions (shifted right)
  - Output is fed to Multi-head attention as Q

- **Output**
  - **Fully connected layer**
  - **Softmax**



[1] Vasvani A., et. al. Attention Is All You Need

Agenda:

- Advanced approaches
    - Transformer
    - **BERT**
    - Transformer-XL
    - XLNet
    - MT-DNN

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language

Masked Language Model (MLM) - Task-1

- 15% of tokens in random will be chosen
- 80% masked
- 10% random
- 10% unchanged and predicted
- Sum of cross-entropy losses over all the masked tokens

- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`

- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`

- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Next Sentence Prediction: Task-2

- To understand relationship between sentences
- 50% of the time next sentence is chosen, 50% some other random sentence
- Binary classification (0 - next sentence, 1 - random)

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`

Label = `IsNext`

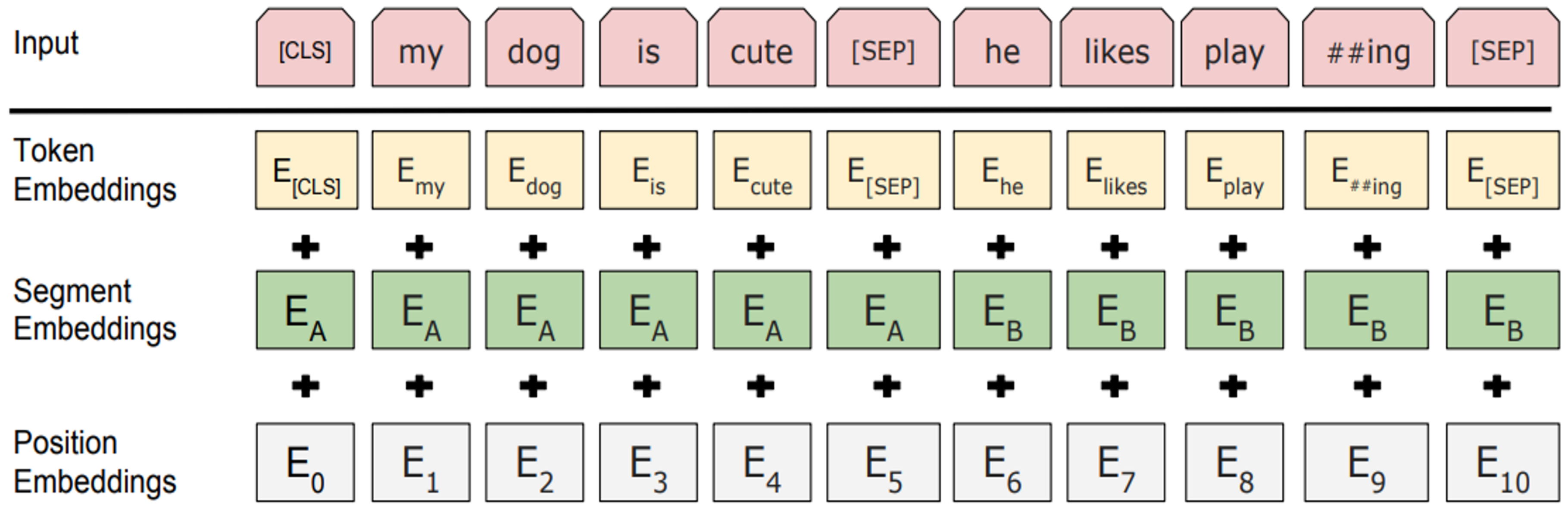
Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT input representation:

- The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.



[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT Model Details

- Pre-training corpus:
  - BooksCorpus (800M words) and English Wikipedia (2,500M words)
- The first sentence receives the A embedding and the second receives the B embedding
  - 50% of the time B is the actual next sentence that follows A and
  - 50% of the time it is a random sentence, which is done for the “next sentence prediction” task.
- Sampled such that the combined length is 512 tokens
- Wordpiece embedding with 30k tokens
- Base Model:
  - Number of layers = 12, hidden size = 768, number of heads = 12

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT – Fine Tuning - sequence-level classification

- Special [CLS] word embedding
  - The final hidden state (i.e., the output of the Transformer) for the first token (Vector C)
- New parameters added during fine-tuning are for a classification layer (K classifier labels) $W \in \mathbb{R}^{K \times H}$
- Label probabilities are computed with a standard softmax

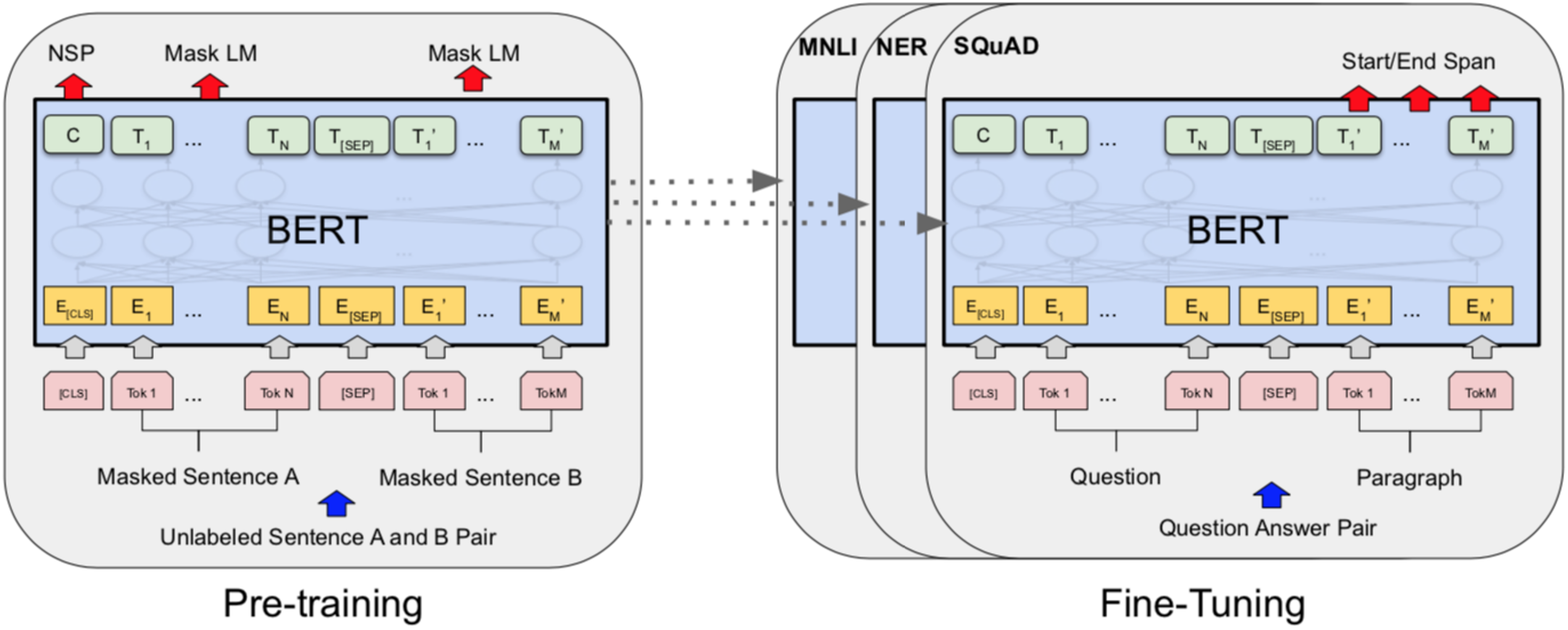
$$P = \mathrm{softmax}(CW^T)$$

- All of the parameters of BERT and W are fine-tuned jointly to maximize the log-probability of the correct label

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

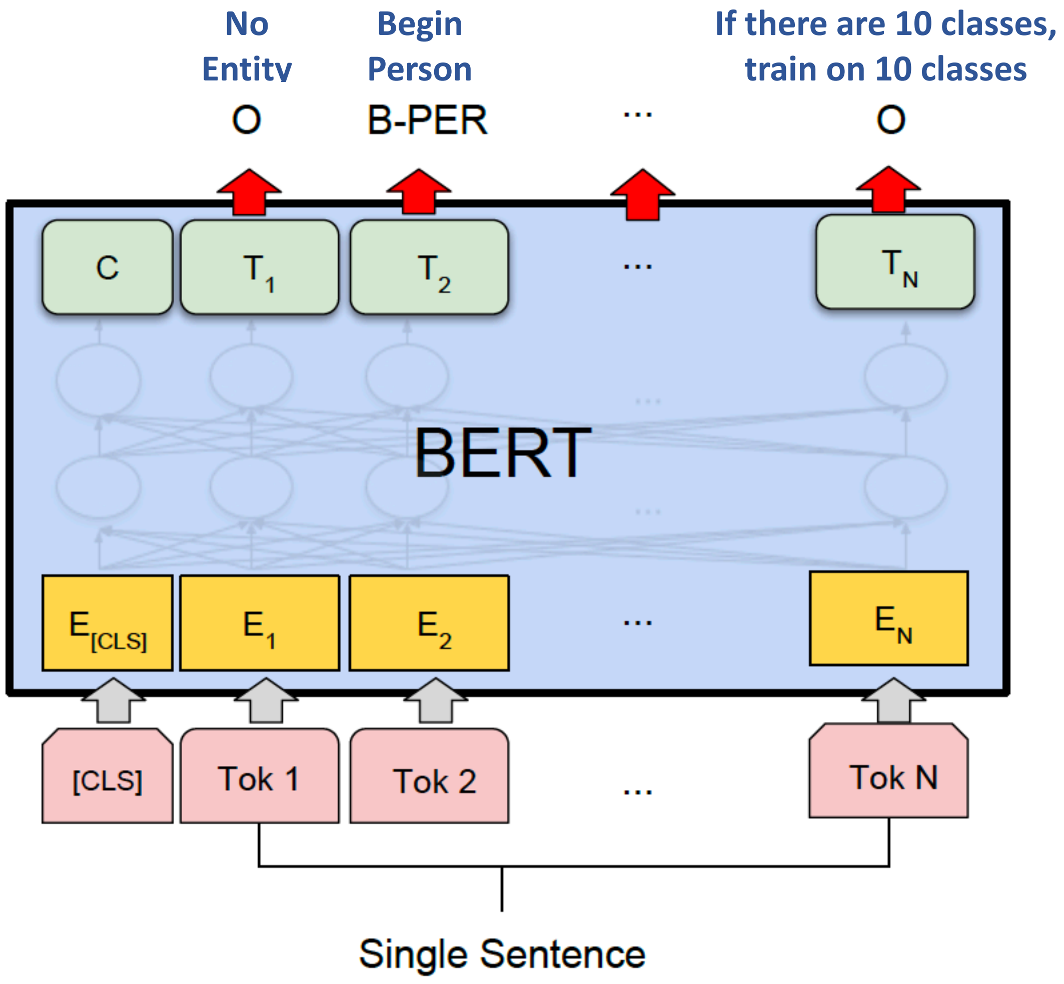
BERT for All Tasks

- Tagging (NER)
- Question Answer
- Classification



[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Single Sentence Tagging Tasks (e.g NER)



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER (Named Entity Recognition)

This dataset consists of 200k training words which have been annotated as

- Person
- Organization
- Location,
- Miscellaneous
- Other (non-named entity)

For fine-tuning, feed the final hidden representation Ti for to each token i into a classification layer over the NER label set

Jim	Hen	##son	was	a	puppet	##eer
I-PER	I-PER	X	O	O	O	X

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

SQUAD – Question Answer

- Given a passage from Wikipedia (containing all the required information) and a question, identify the relevant portion in the passage
- Identify start and end word constructing the answer
- Jointly learning the start and end position

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

The Black Death is thought to have originated in the arid plains of Central Asia, where it then travelled along the Silk Road, reaching Crimea by 1343. From there, it was most likely carried by Oriental rat fleas living on the black rats that were regular passengers on merchant ships. Spreading throughout the Mediterranean and Europe, the Black Death is estimated to have killed 30–60% of Europe's total population. In total, the plague reduced the world population from an estimated 450 million down to 350–375 million in the 14th century. The world population as a whole did not recover to pre-plague levels until the 17th century. The plague recurred occasionally in Europe until the 19th century.

Where did the black death originate?
Ground Truth Answers: the arid plains of Central Asia | Central Asia | Central Asia

How did the black death make it to the Mediterranean and Europe?
Ground Truth Answers: merchant ships. | merchant ships | Silk Road

How much of the European population did the black death kill?
Ground Truth Answers: 30–60% of Europe's total population | 30–60% of Europe's total population | 30–60%

SQuAD – Stanford Question Answering Dataset

SQUAD – Question Answer

Given a question and a paragraph from Wikipedia containing the answer, the task is to predict the answer text span in the paragraph. For example:

- Input Question:

  `Where do water droplets collide with ice crystals to form precipitation?`

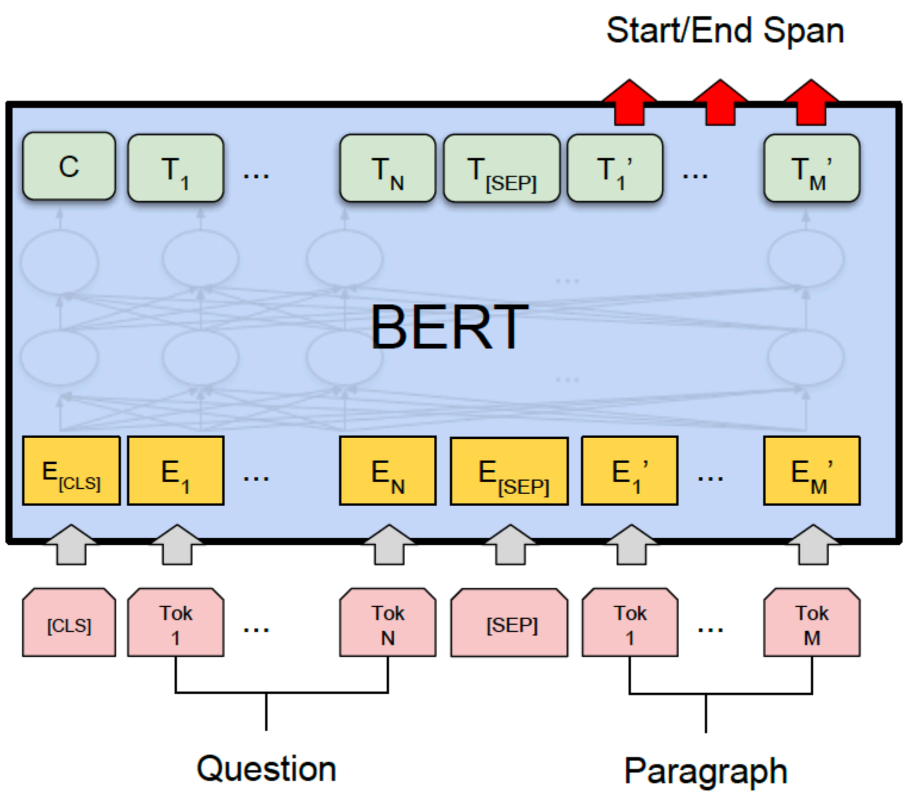
- Input Paragraph:

  `... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. ...`

- Output Answer:

  `within a cloud`

Represent the input question and paragraph as a single packed sequence.
Question using the A embedding and the paragraph using the B embedding.



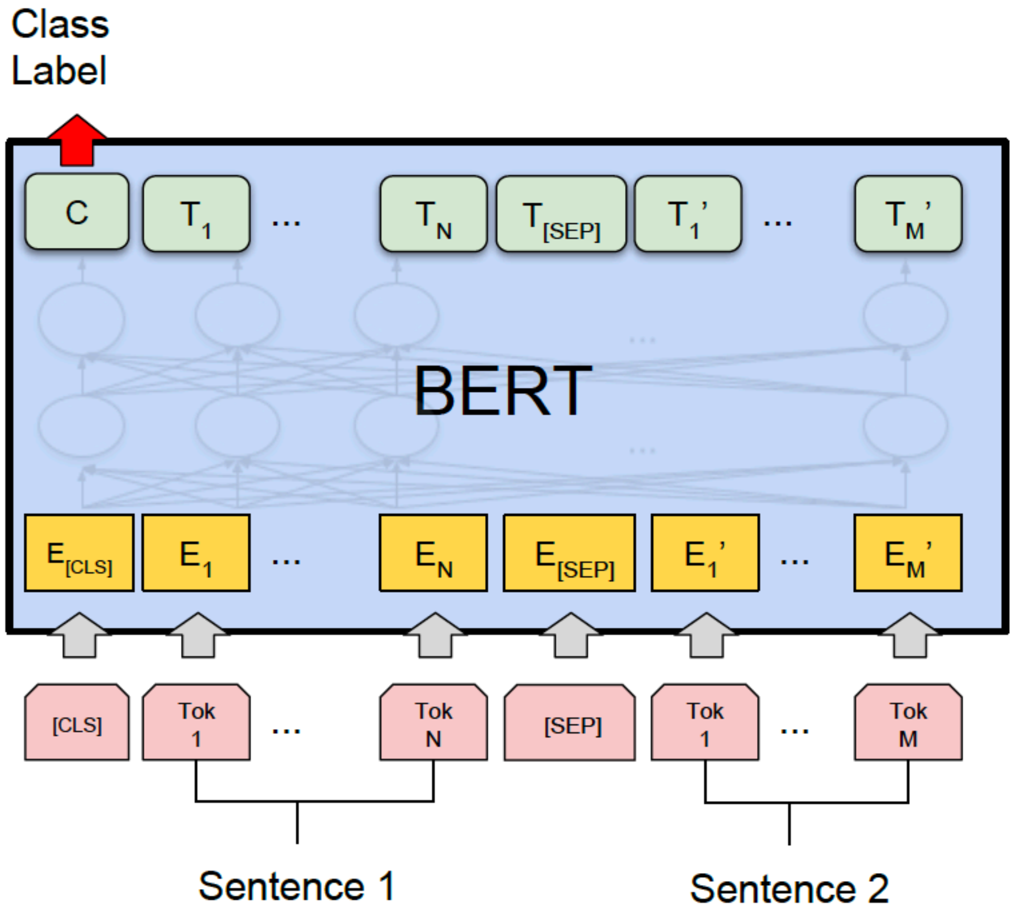
The only new parameters learned during fine-tuning are a start vector and an end vector.

The probability of word i being the start of the answer span is computed as a dot product between Ti and S followed by a softmax over all of the words in the paragraph. Same way learn end vector

The training objective is the loglikelihood of the correct start and end positions.

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Sentence Pair Classification



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

QQP: Quora Question Pairs is a *binary classification* task where the goal is to determine if two questions asked on Quora are semantically equivalent.

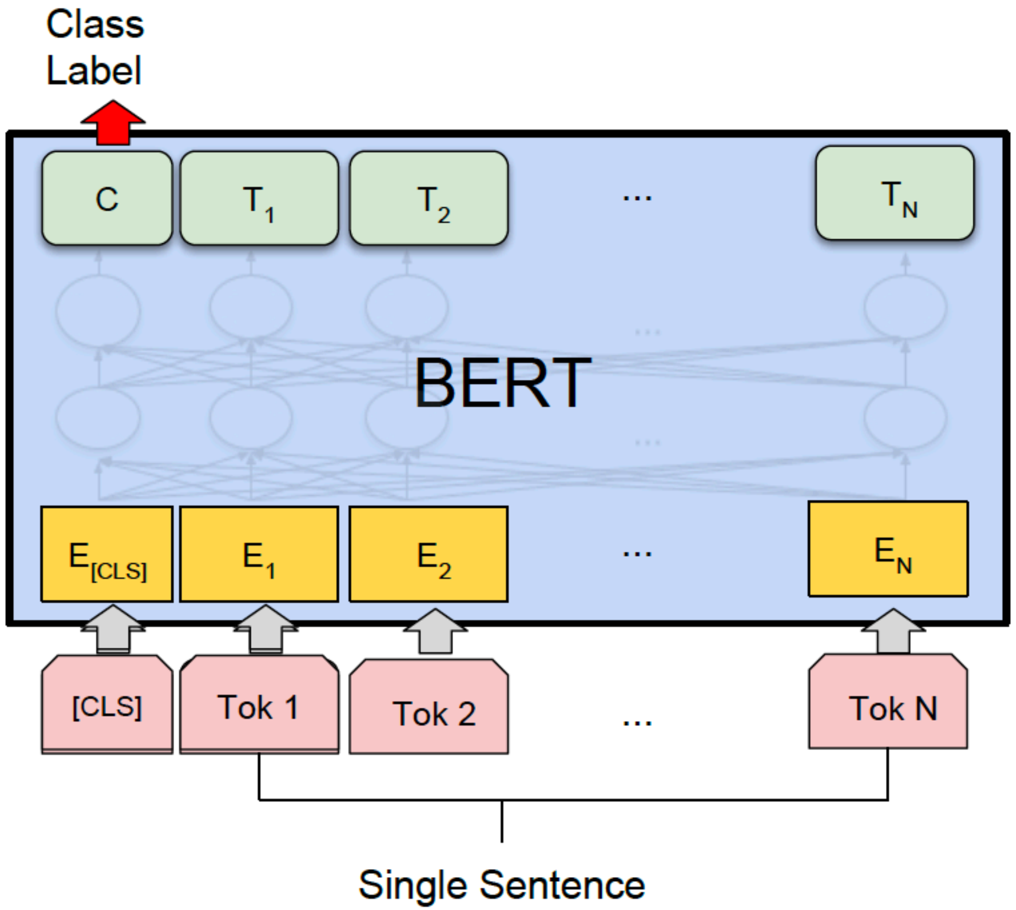
MNLI: Multi-Genre Natural Language Inference: Given a pair of sentences, the goal is to predict whether the second sentence is an

- *entailment*
- *contradiction, or*
- *neutral*

with respect to the first one.

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Single Sentence Classification (e.g Sentiment)



(b) Single Sentence Classification Tasks: SST-2, CoLA

SST-2 The Stanford Sentiment Treebank:
Binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Agenda:

- Advanced approaches
  - Transformer
  - BERT
  - **Transformer-XL**
  - XLNet
  - MT-DNN

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Zihang Dai[*12], **Zhilin Yang**[*12], **Yiming Yang**[1], **Jaime Carbonell**[1],
Quoc V. Le[2], **Ruslan Salakhutdinov**[1]
[1]Carnegie Mellon University, [2]Google Brain
{dzihang,zhiliny,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

Agenda

- Foundation for XLNet
  - Transformer and BERT (already covered)
  - Transformer XL

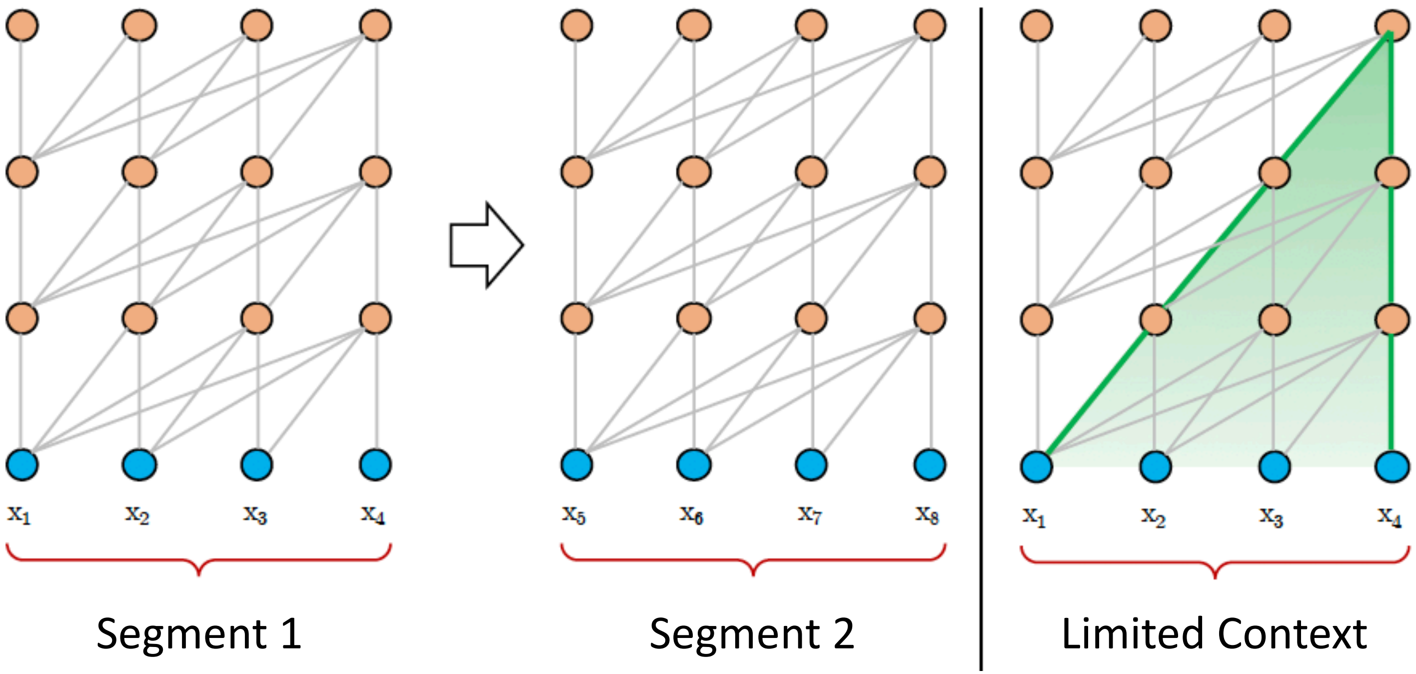
Problem Statement

- Attention is not recurrent, it can only deal with fixed-length context

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Problem Statement

- Attention is not recurrent, it can only deal with fixed-length context
- Context fragmentation: if context is long, it should be split up to segments



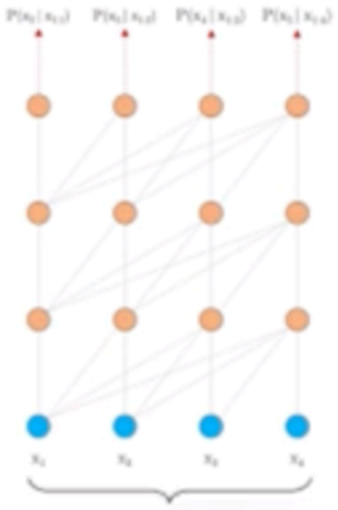
[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Solve for

- Capture longer-term dependency
- Resolve context fragmentation

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

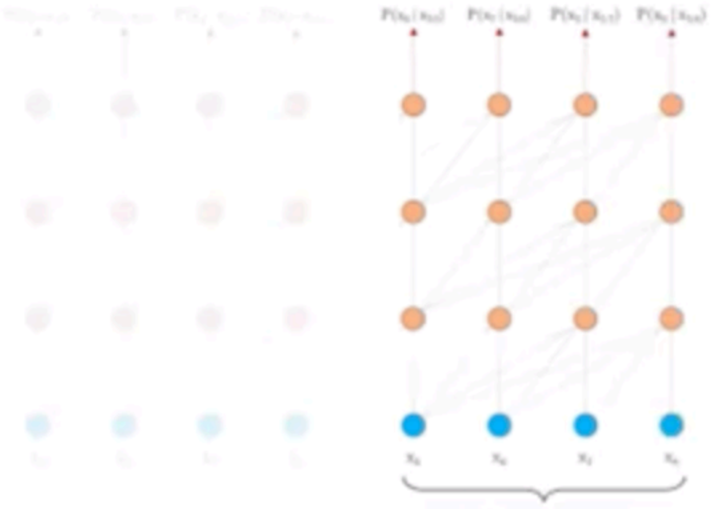
Vanilla Training



Current Segment

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

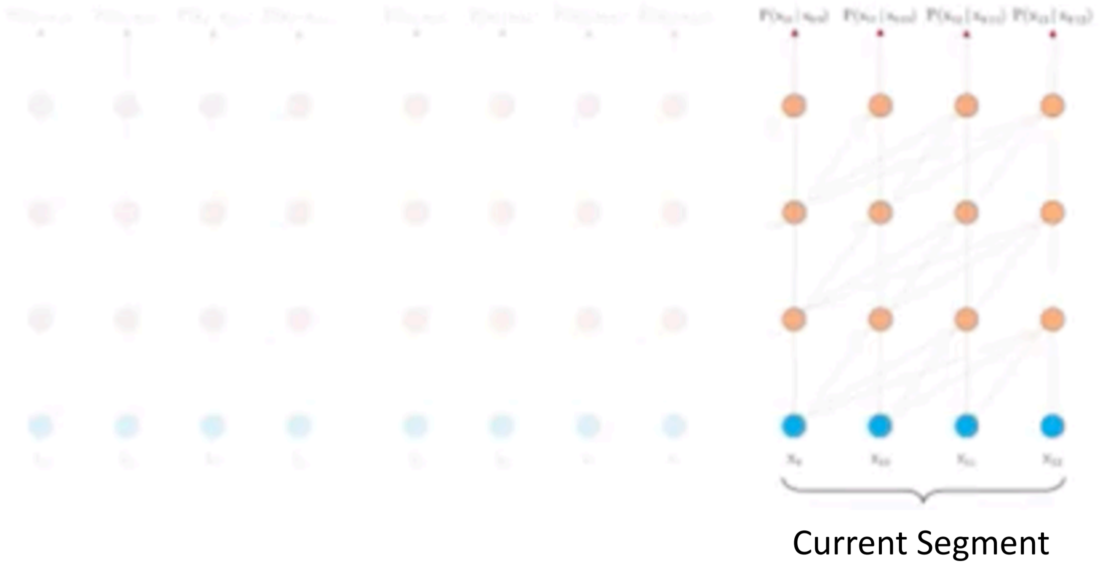
Vanilla Training



Current Segment

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

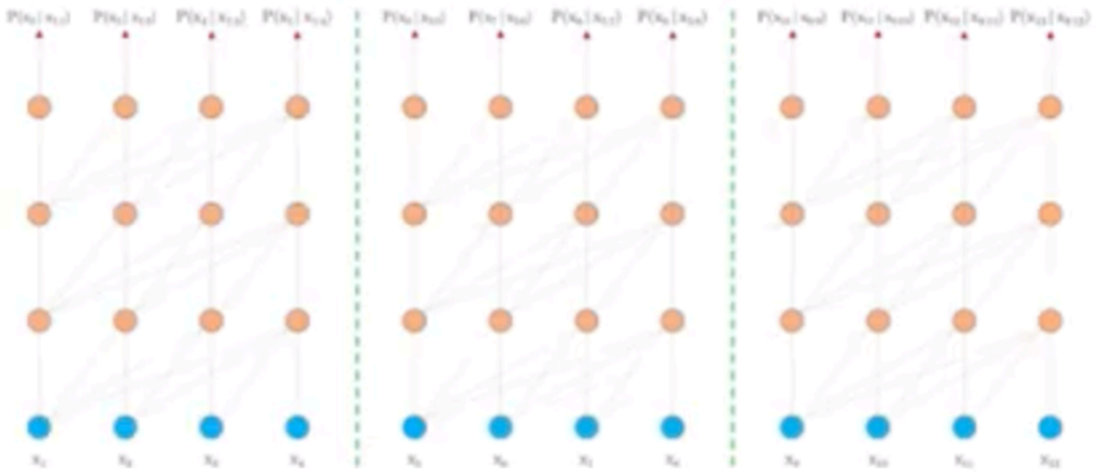
Vanilla Training



[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Vanilla Training

- Information never flows across segments in either forward or backward pass

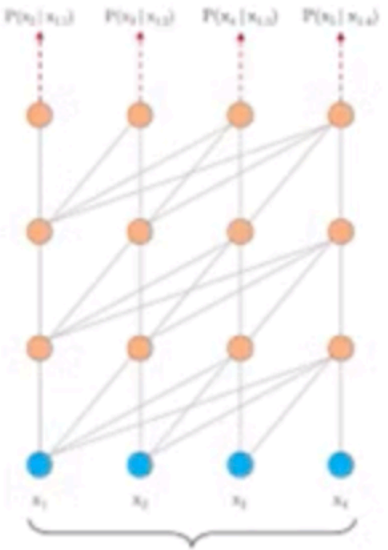


No information Flow
(either forward or backward)

No information Flow
(either forward or backward)

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Training

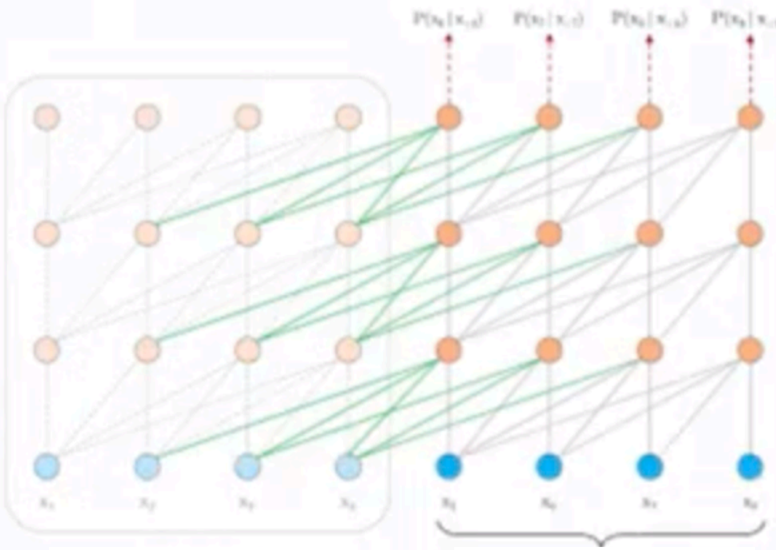


Current Segment

Segment-1

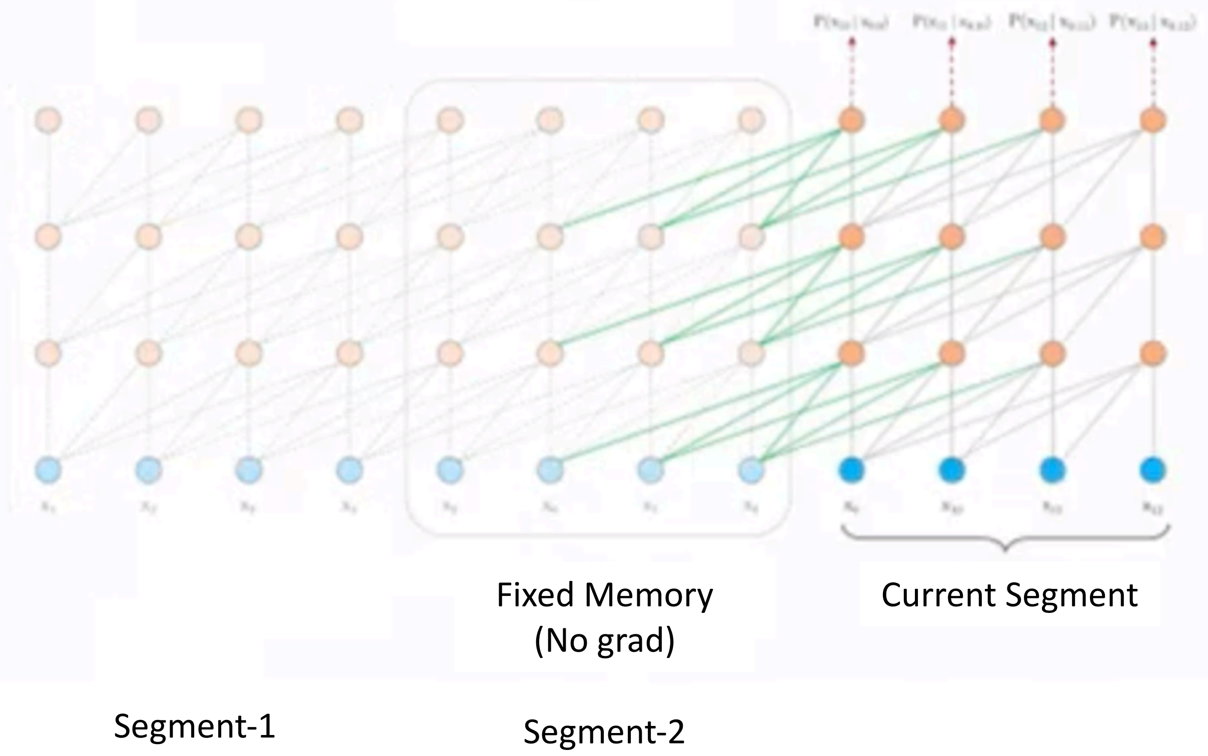
[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Training



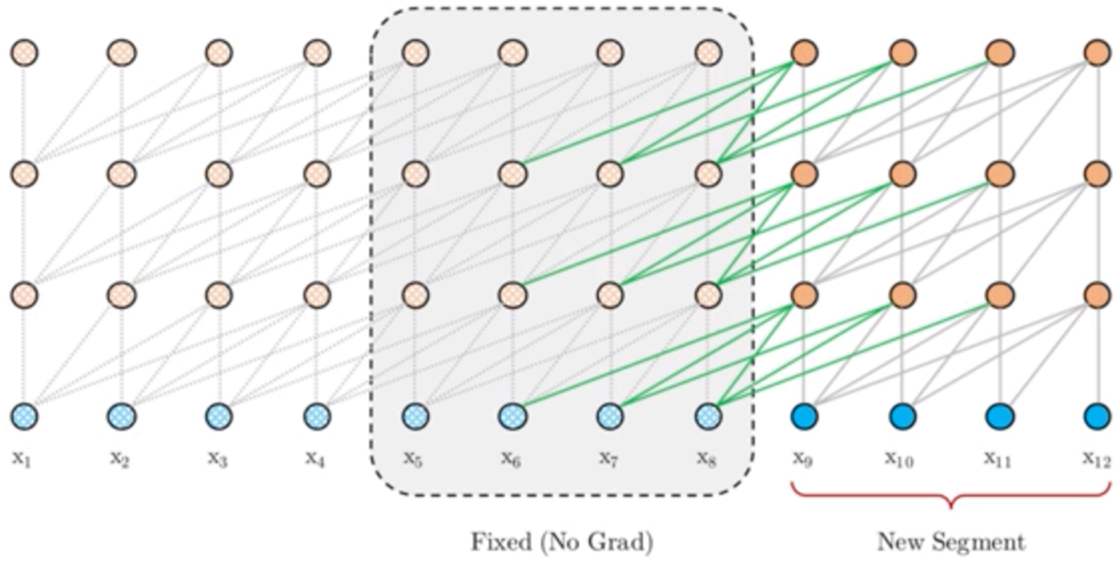
[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Training



[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Training



the key $\mathbf{k}_{\tau+1}^{n}$ and value $\mathbf{v}_{\tau+1}^{n}$ are conditioned on the extended context $\widetilde{\mathbf{h}}_{\tau+1}^{n-1}$ and hence $\mathbf{h}_{\tau}^{n-1}$ cached from the previous segment.

$$\widetilde{\mathbf{h}}_{\tau+1}^{n-1} = \left[\mathrm{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}\right],$$

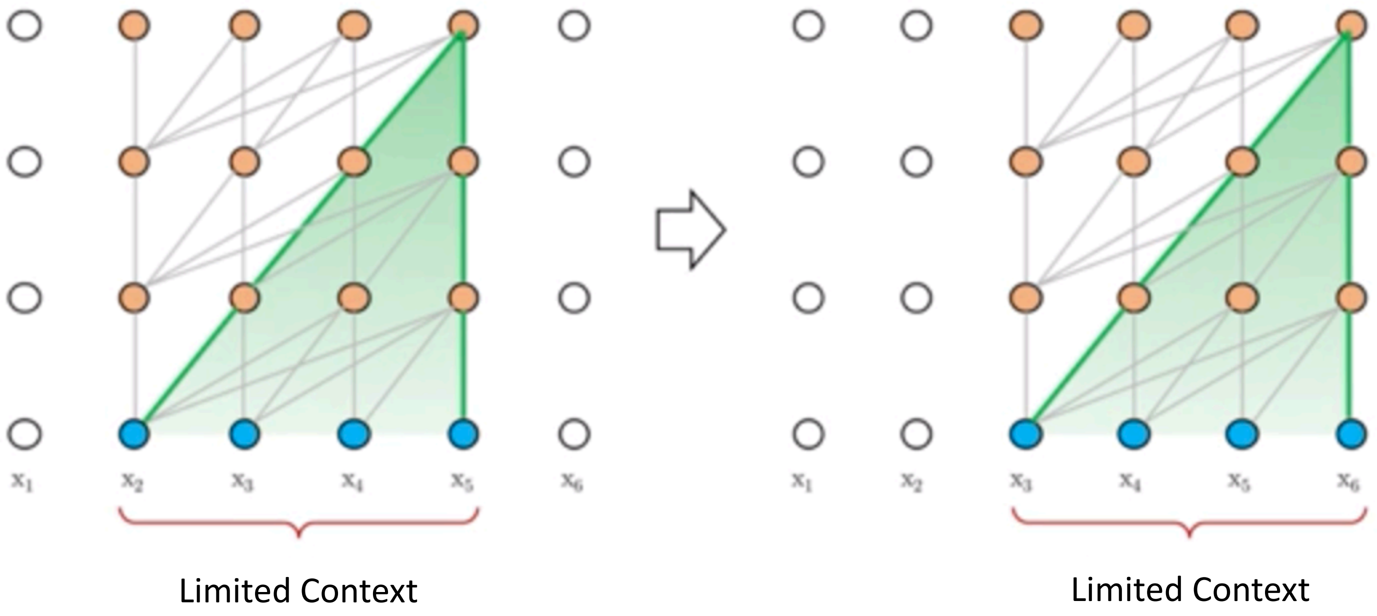
$$\mathbf{q}_{\tau+1}^{n}, \mathbf{k}_{\tau+1}^{n}, \mathbf{v}_{\tau+1}^{n} = \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_{q}^{\top}, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_{k}^{\top}, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_{v}^{\top},$$

$$\mathbf{h}_{\tau+1}^{n} = \text{Transformer-Layer}\left(\mathbf{q}_{\tau+1}^{n}, \mathbf{k}_{\tau+1}^{n}, \mathbf{v}_{\tau+1}^{n}\right).$$

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Vanilla Prediction

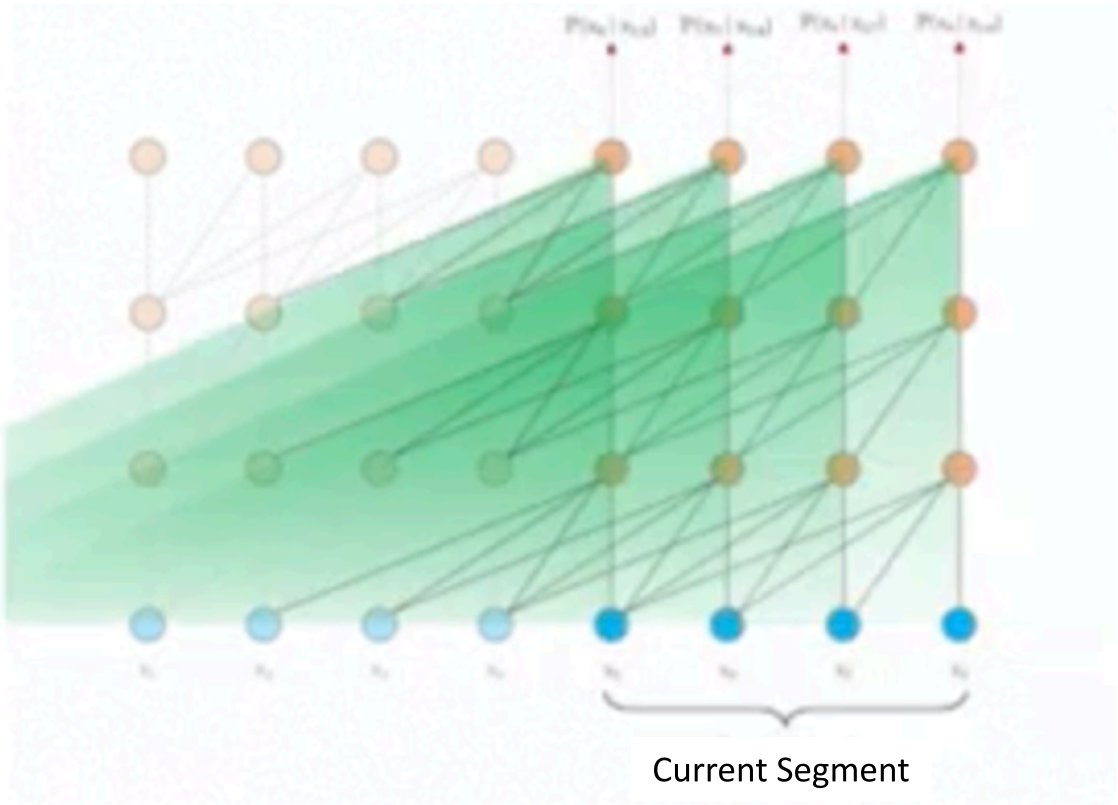
- During evaluation, vanilla model consumes a segment to make only one prediction at the last position, which is extremely expensive



[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Prediction

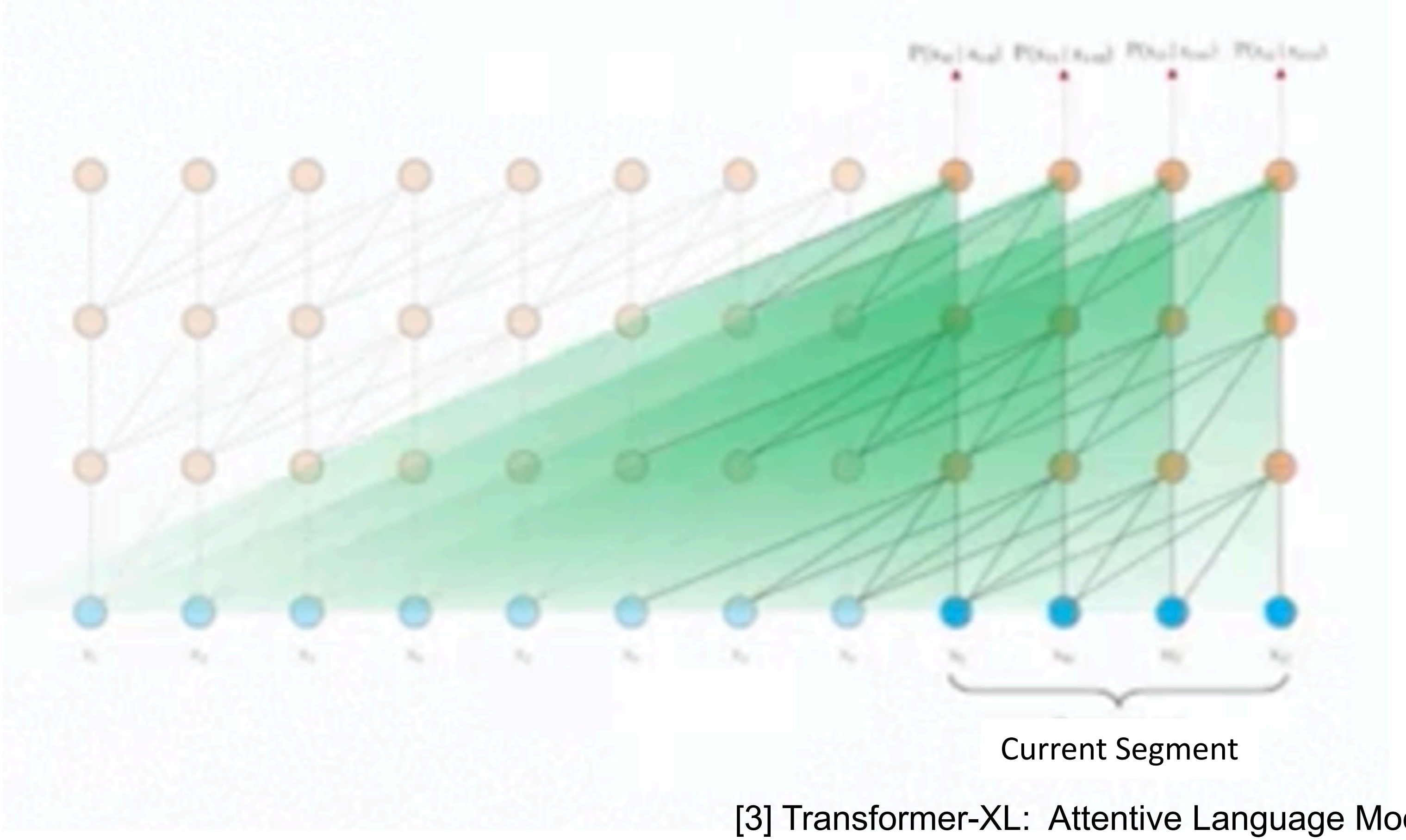
- Transformer-XL uses representations (memory) from previous segments instead of computing from scratch



[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Prediction

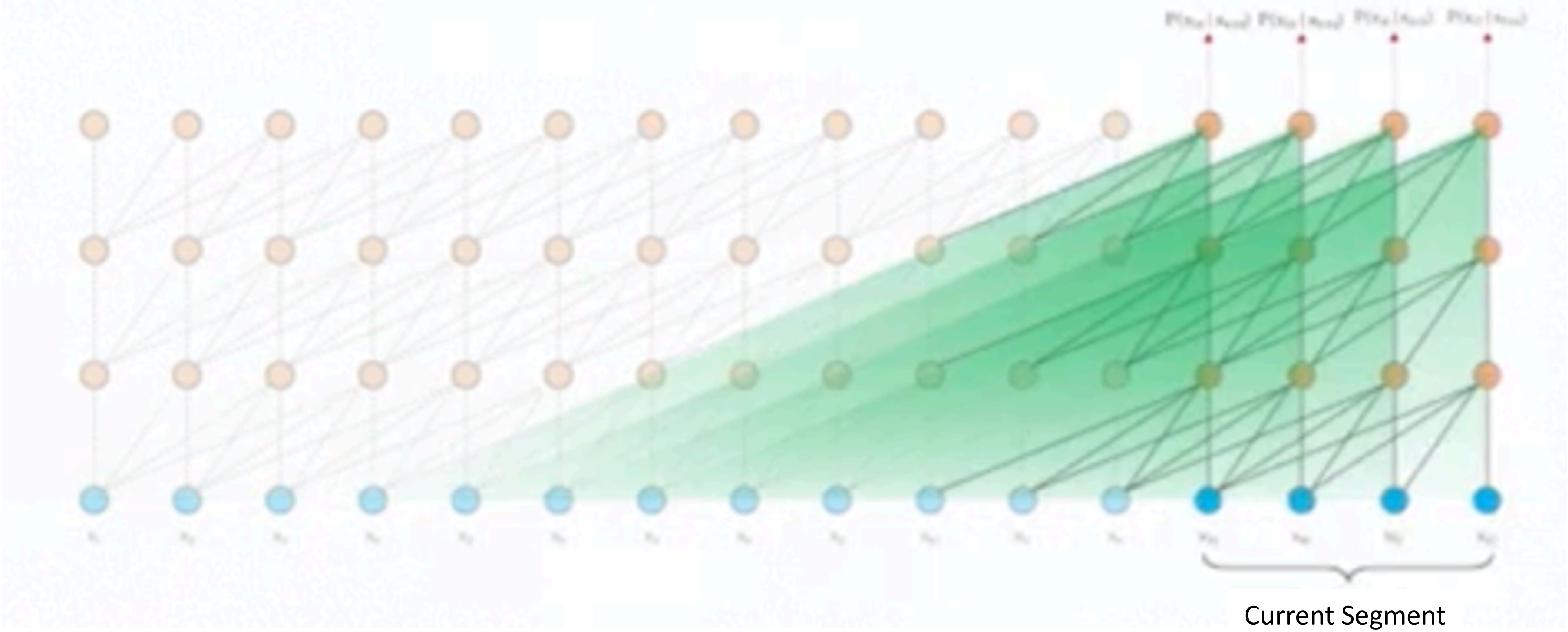
- Transformer-XL uses representations (memory) from previous segments instead of computing from scratch



[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Transformer-XL Prediction

- Transformer-XL uses representations (memory) from previous segments instead of computing from scratch



[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Summary

- Transformer model has weakness:
  - Capturing long-term dependency and
  - Resolving context fragmentation

- Transformer-XL suggests following to solve above problems
  - Segment-level recurrence and
  - Relative positional embedding so that recurrence works

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Agenda:

- Advanced approaches
  - Transformer
  - BERT
  - Transformer-XL
  - **XLNet**
  - MT-DNN

XLNet:
Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang[*1], **Zihang Dai**[*12], **Yiming Yang**[1], **Jaime Carbonell**[1],
Ruslan Salakhutdinov[1], **Quoc V. Le**[2]
[1]Carnegie Mellon University, [2]Google Brain
{zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

Autoregressive (AR) vs Autoencoding (AE)

AR language model (GPT)

$$\max_{\theta} \log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t})$$

backward

[x1, x2, x3, x4, x5, x6, x7, x8]

forward

Use observations from previous time steps
to predict the value at the next time stamp
Good at text generation

AE language model (BERT)

$$\max_{\theta} \log p_\theta(\bar{x} \mid \hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{x})$$

(masked token)

[x1, x2, x3, x4, x5, x6, x7, x8]

bidirectional

Based on Transformer
Cath dependencies from both sides
Good at language understanding

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

BERT's Limitations

- Model assumes that all masked tokens are independent
    - e.g. New York is not
- Generalized model **should not rely on data corruption** (masking)
- **<mask>** token doesn't appear in real world
- It lacks long-term dependency

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

XLNet

- Could we use the best of both worlds?
  - Autoregressive and Autoencoding

- Could it be useful to mainstream NLP tasks?

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

XLNet

1. Autoregressive model not just going forward or backward but with permutation on both sides
2. Uses two-Stream Self-Attention
3. Integrate recurrence mechanism - Transformer-XL

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

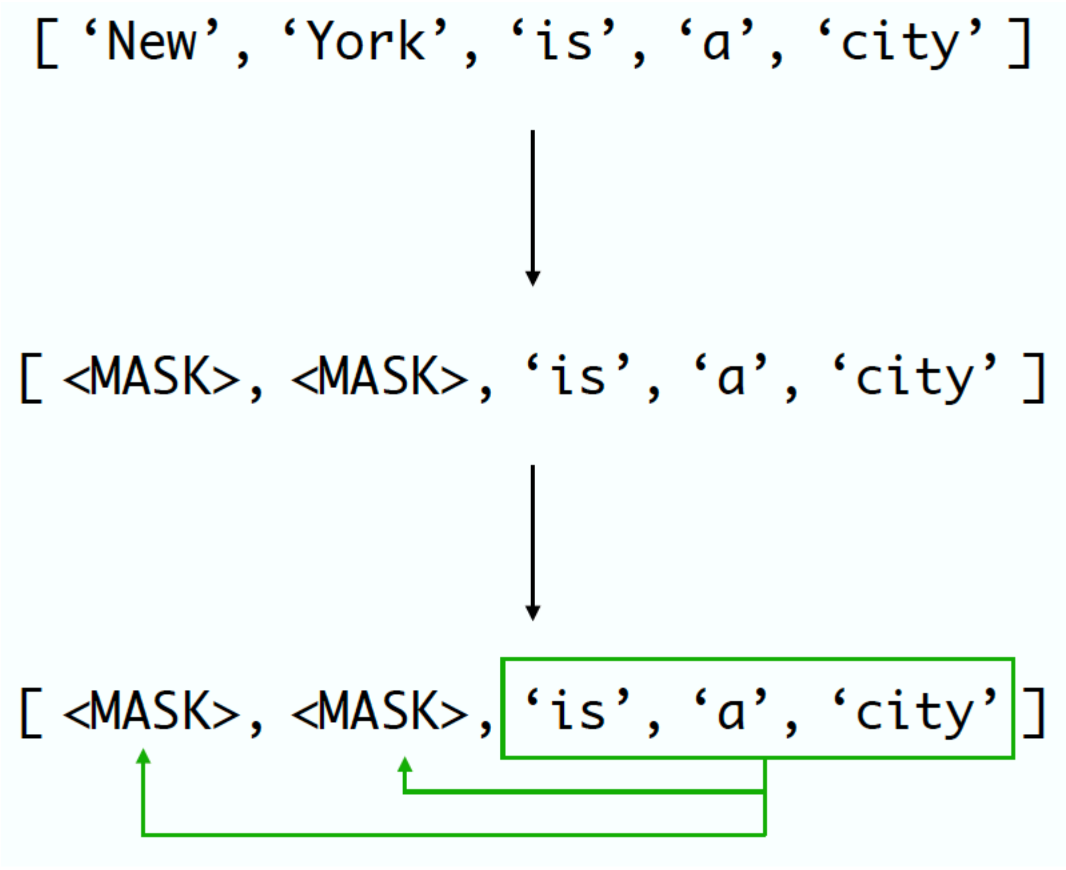
Permutation Language Modeling

origin sequence	[1, 2, 3, 4, 5, 6, 7, 8]
Forward AR	[1, 2, 3, 4, 5, 6, 7, 8]
Backward AR	[8, 7, 6, 5, 4, 3, 2, 1]
Permutation AR	[3, 2, 5, 6, 8, 1, 7, 4]

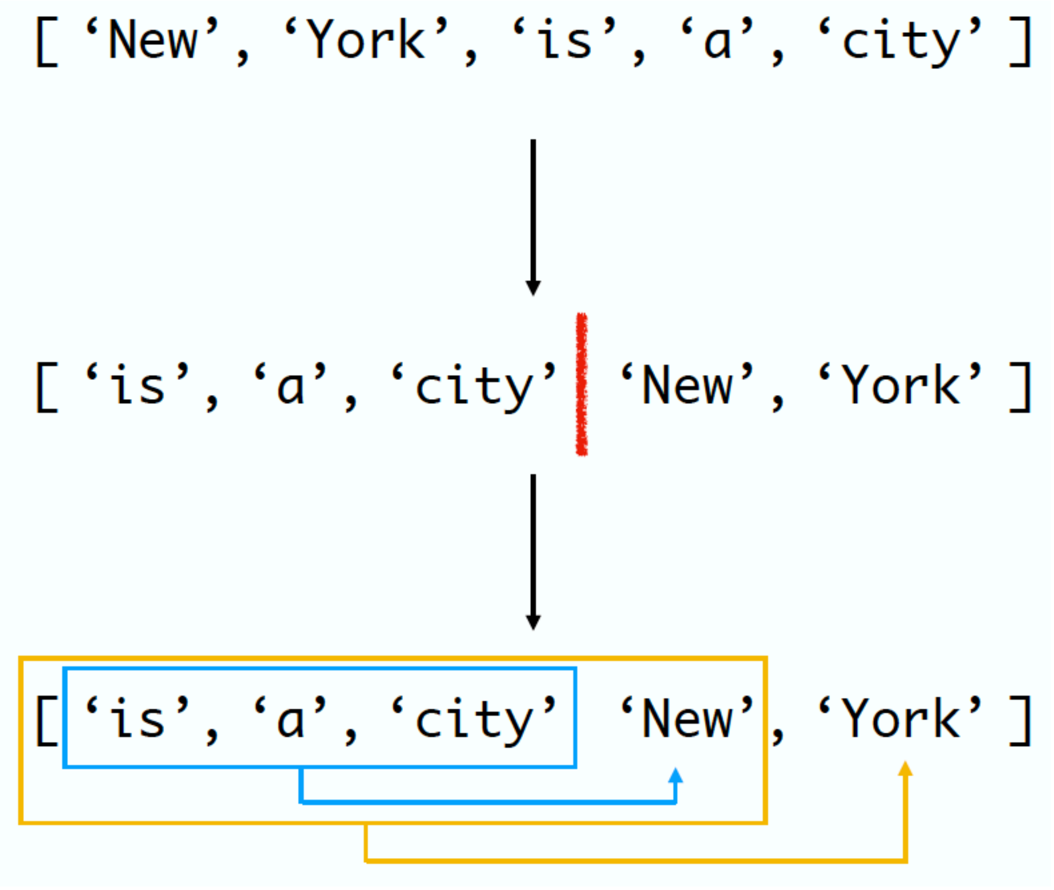
[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Permutation Language Modeling

Masked Language Model



Permutation Language Model

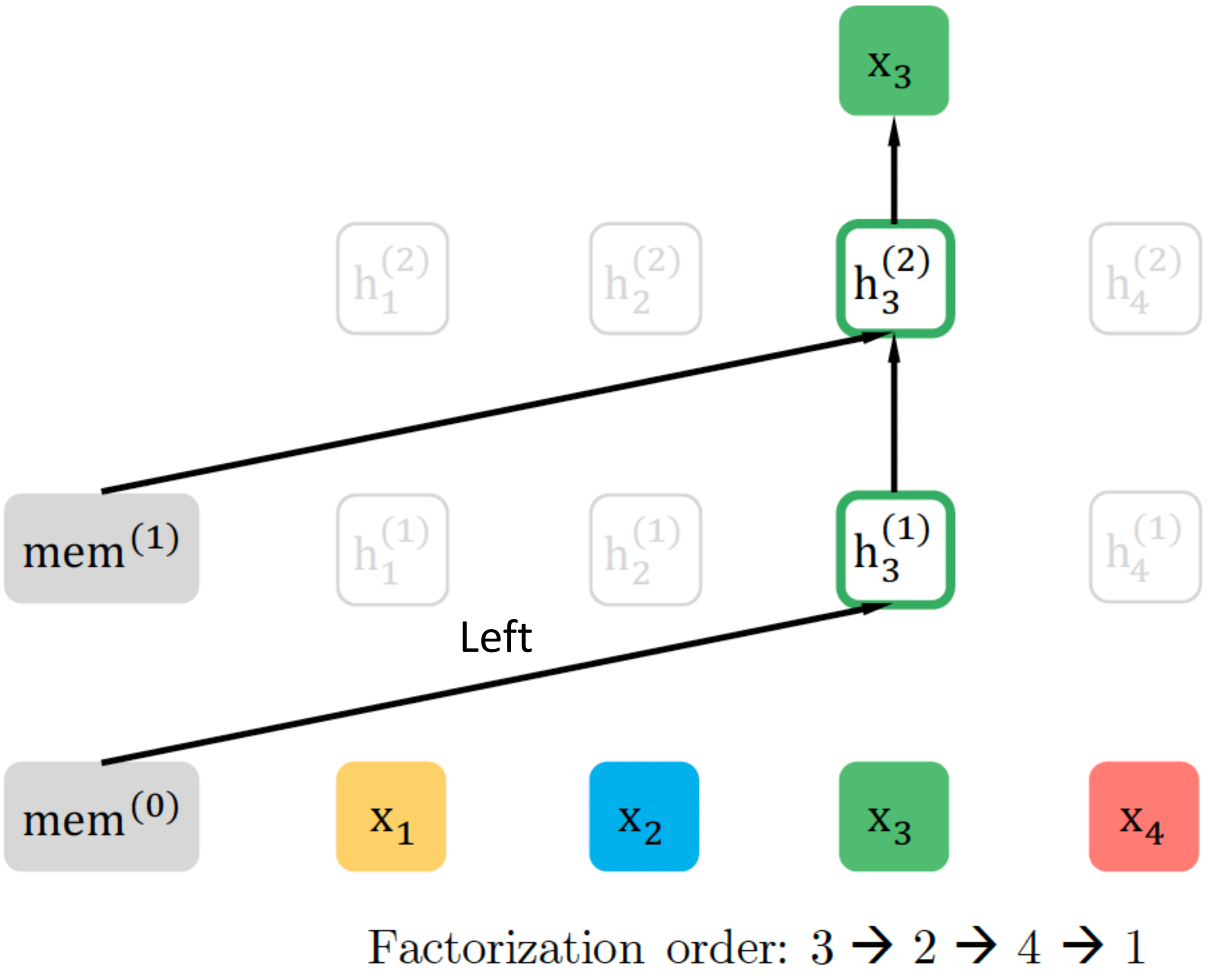


[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Permutation Language Modeling

Permutation

[3, 2, 4, 1]

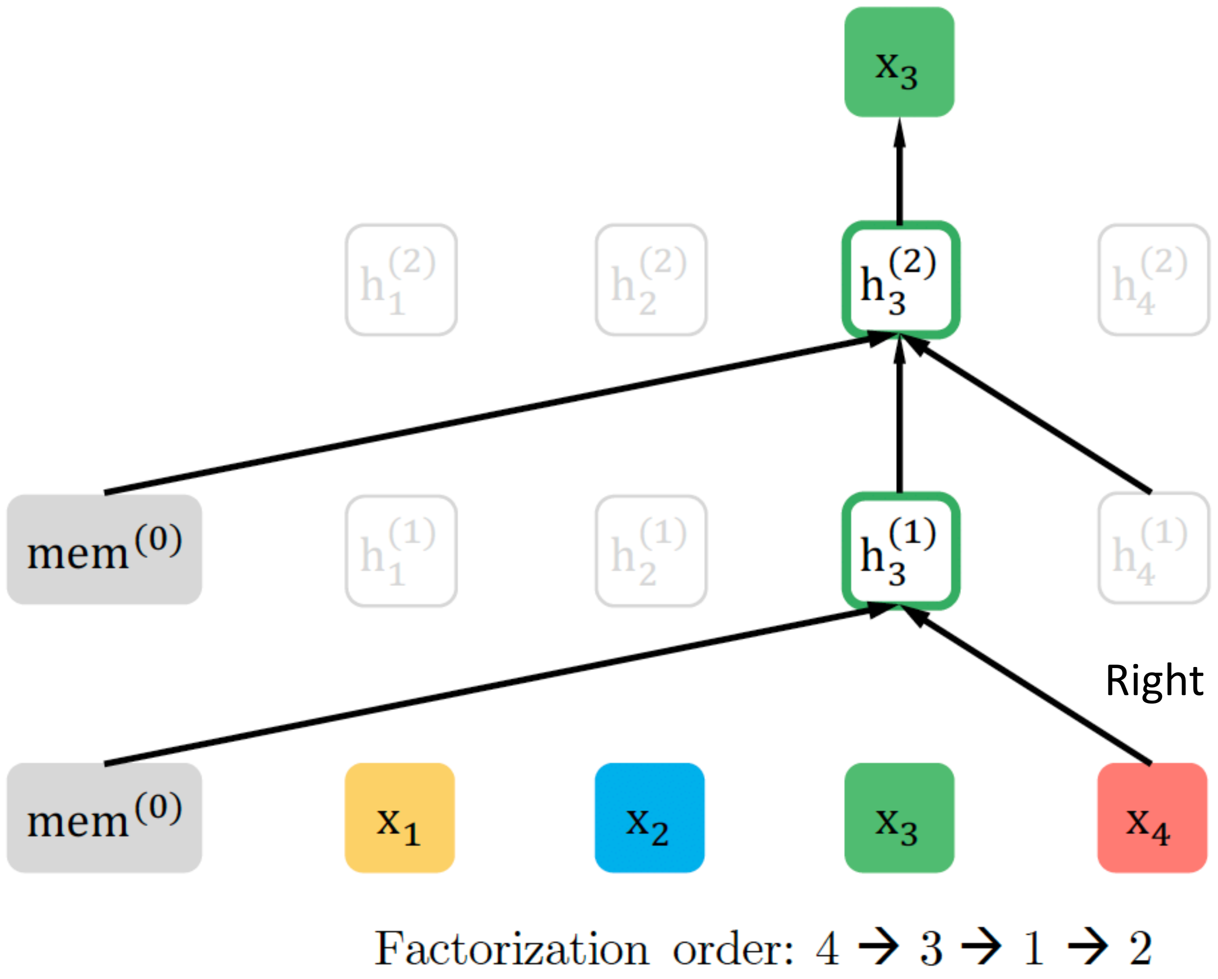


[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Permutation Language Modeling

Permutation

[4, 3, 1, 2]

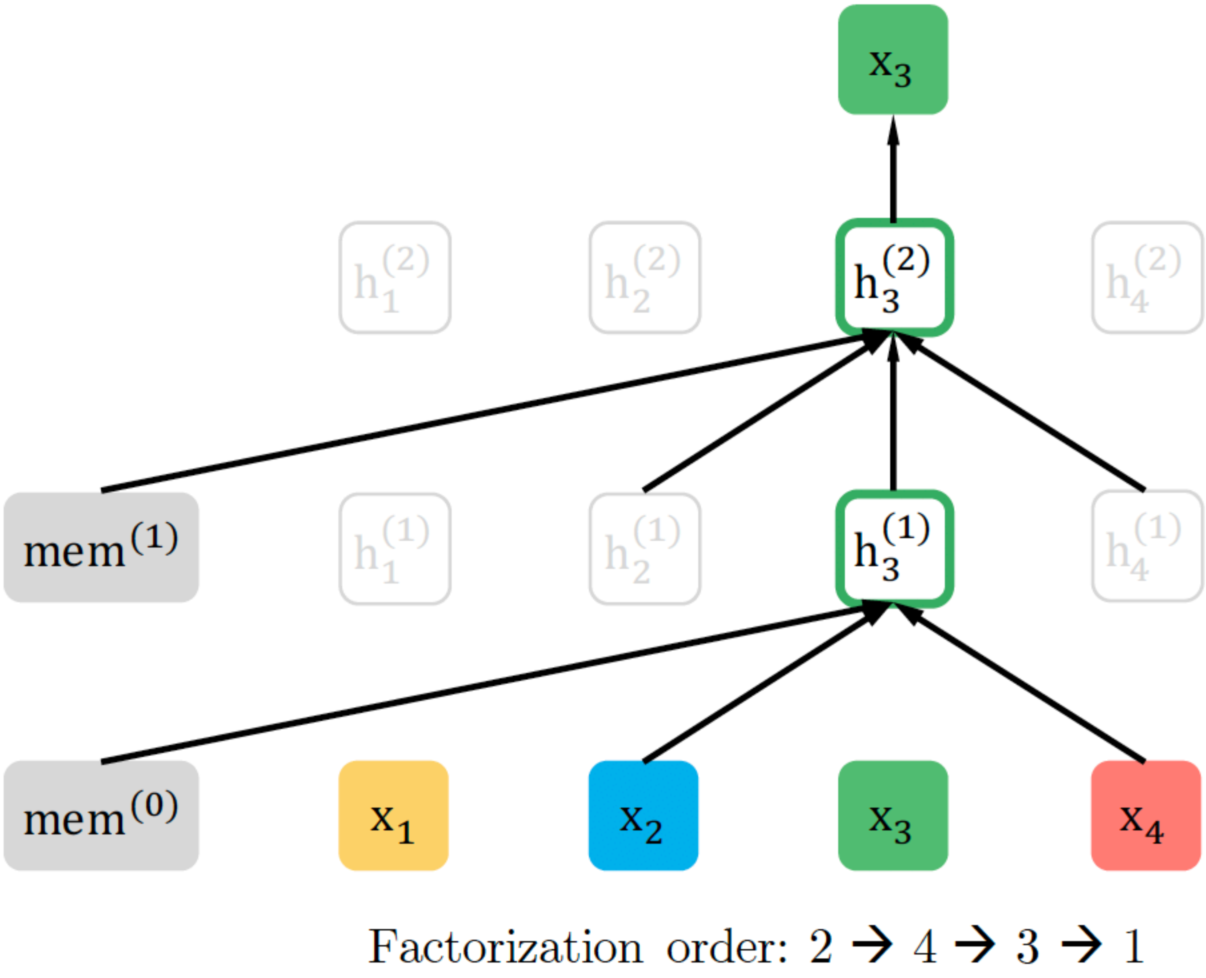


[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Permutation Language Modeling

Permutation

[2, 4, 3, 1]



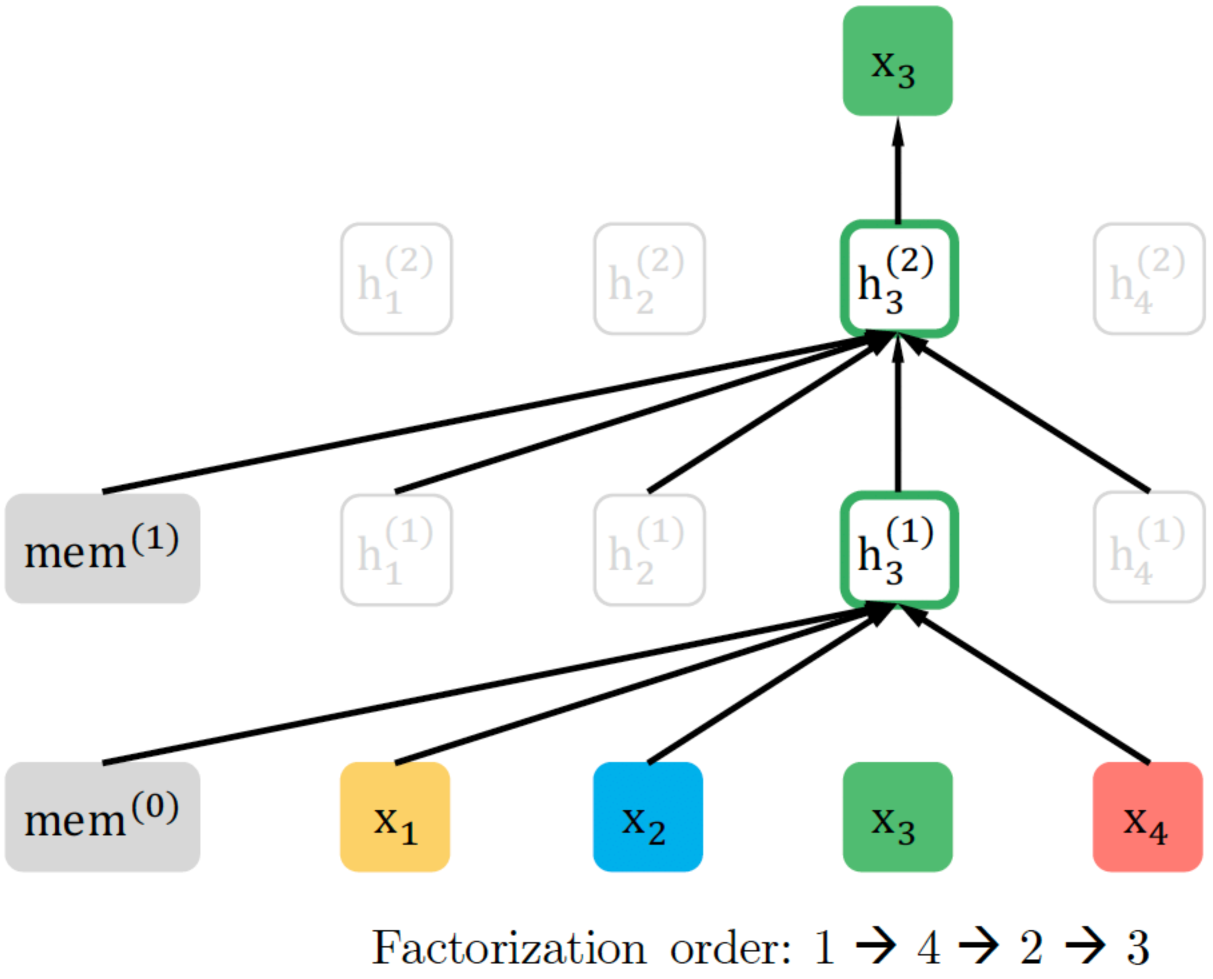
Left & Right

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Permutation Language Modeling

Permutation

[1, 4, 2, 3]



Left & Right

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

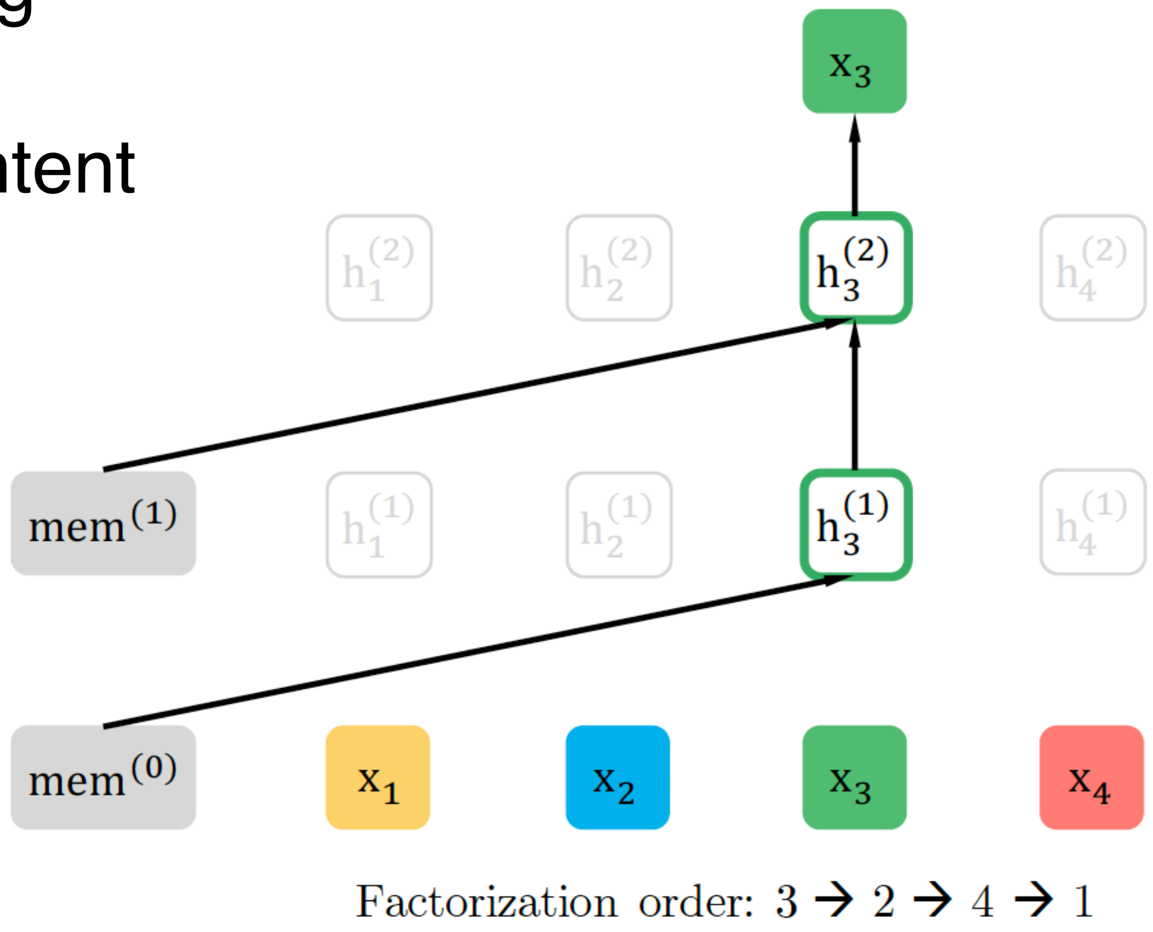
Two Stream Self-Attention

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Two-Stream Self-Attention

From the input token corresponding to the token to predict

- Keep positional encoding
- Remove embedding content



[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Two-Stream Self-Attention mechanism

From the input token corresponding to the token to predict

- Keep positional encoding
  - *Query can be used to pass the positional encoding*

- Remove embedding content
  - *Block the Values (embedding content)*

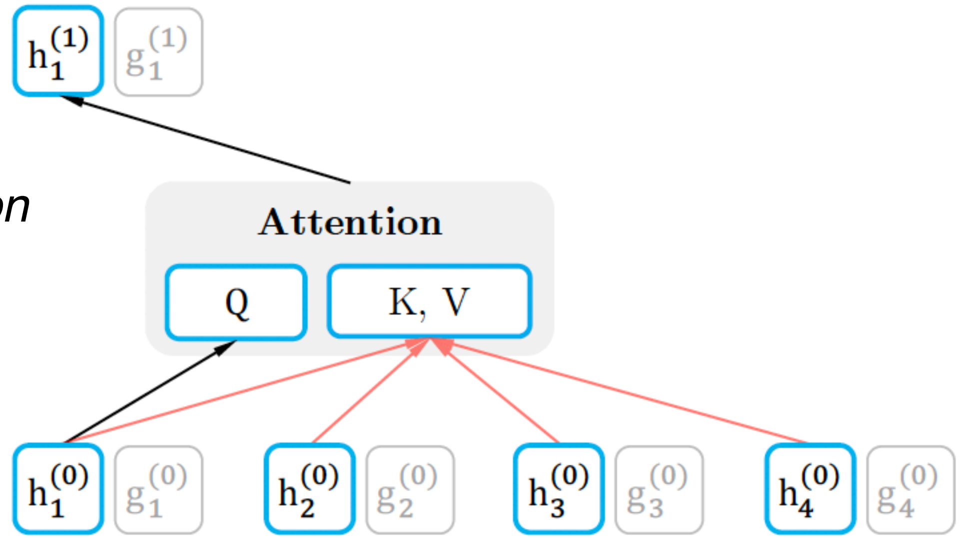
[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Two-Stream Self-Attention

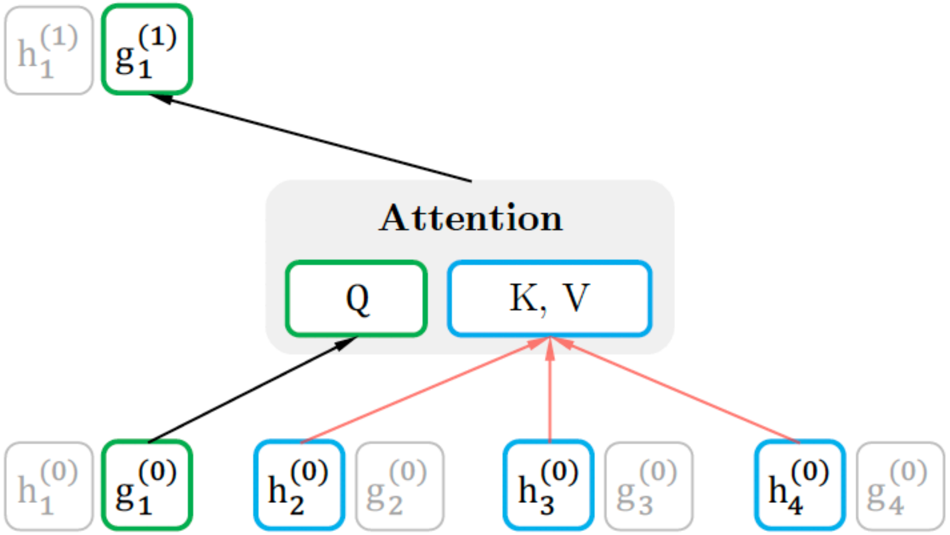
Content stream attention

Query stream attention

h: content attention
g: query attention



(a)



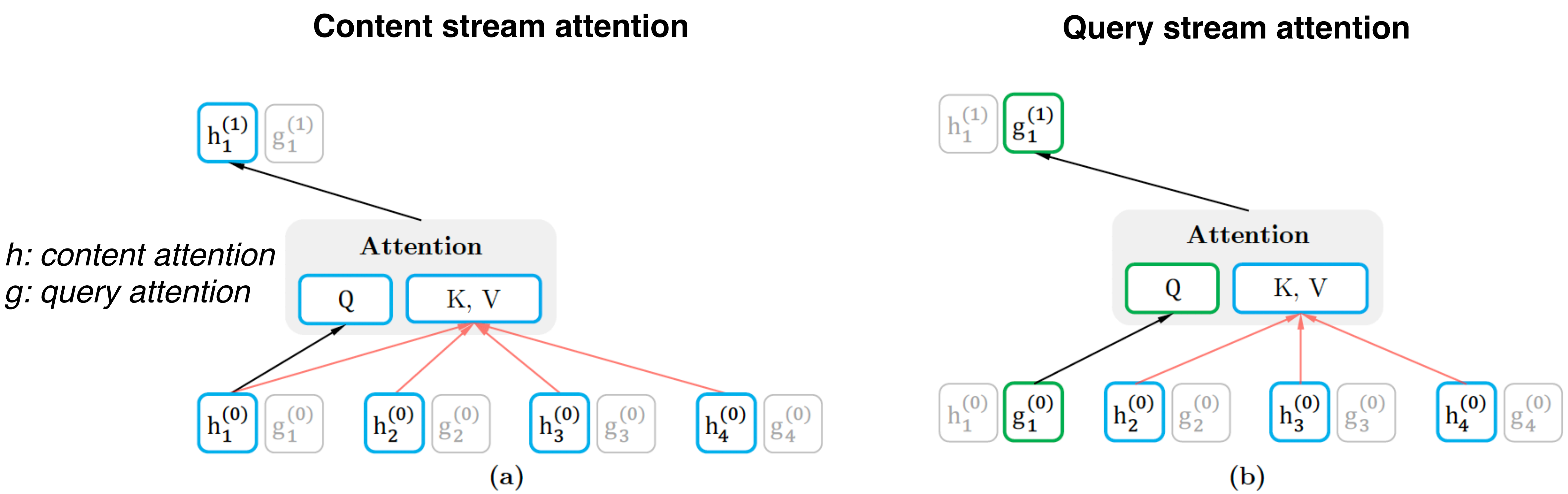
(b)

same as the standard self-attention
(Can see self)

Does not have access to information
about the content x_{zt} (Cannot see self)

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Two-Stream Self-Attention



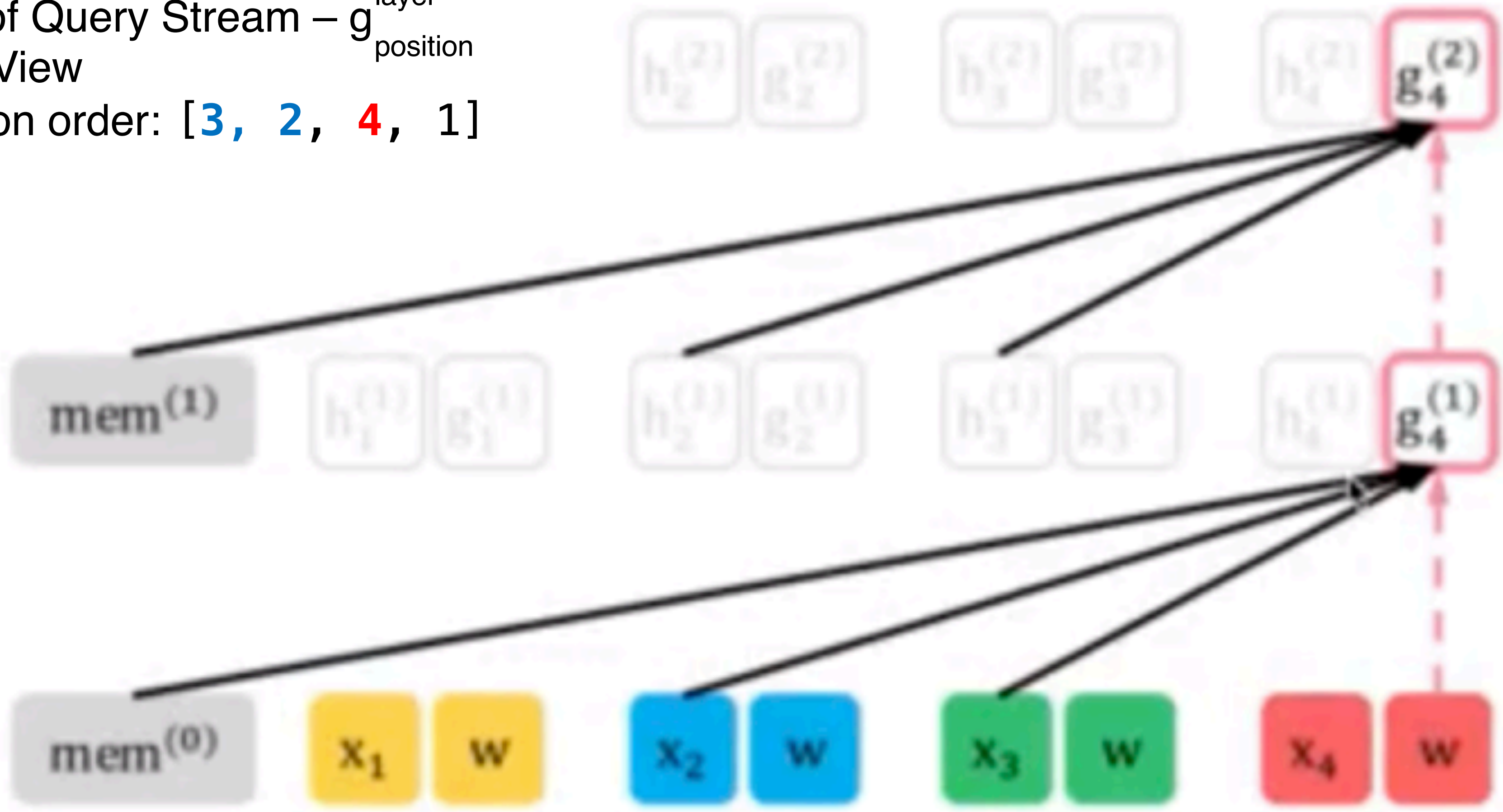
$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = g_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t}\text{)}$$

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = h_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}\text{)}.$$

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

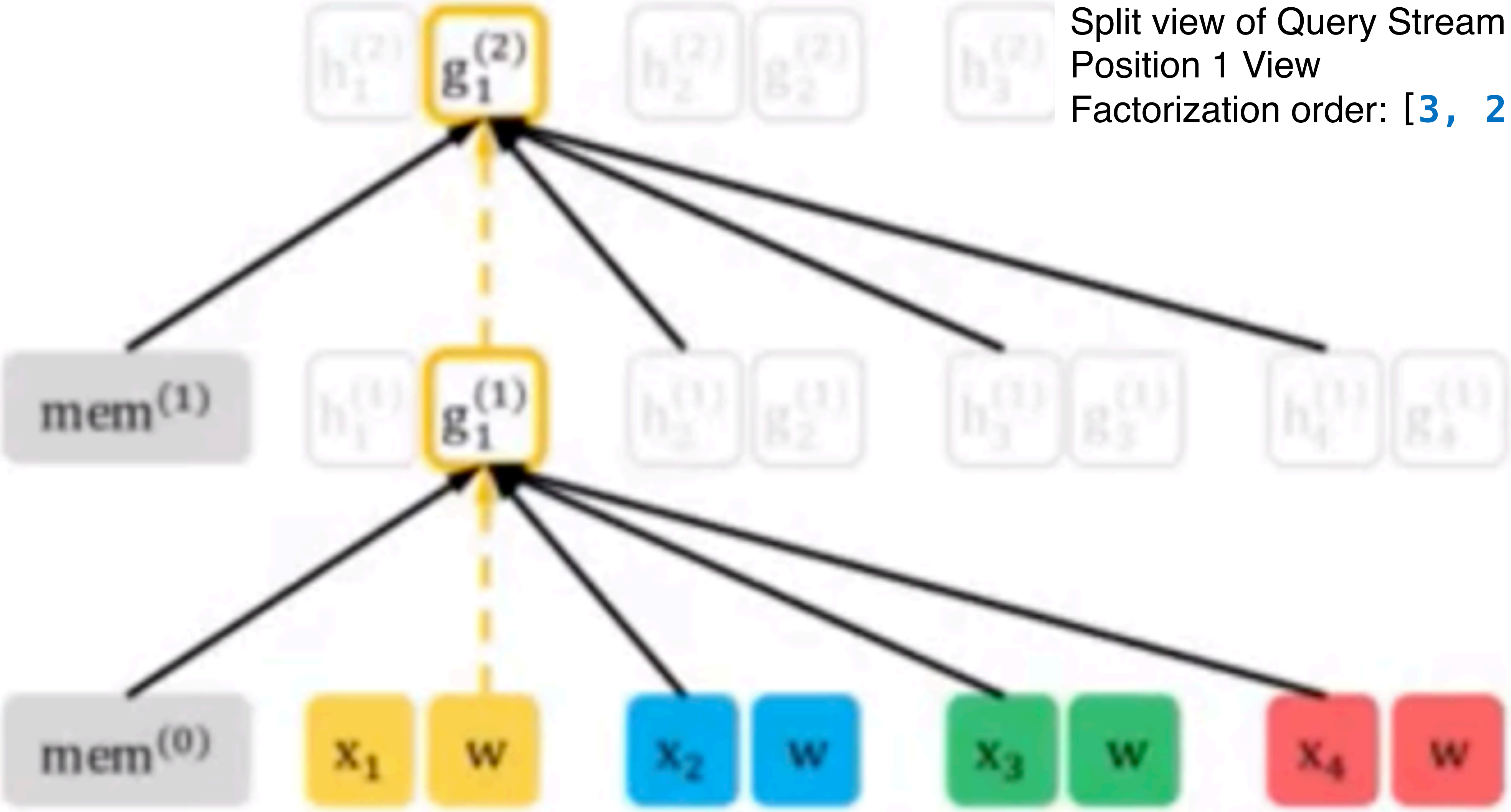
Two-Stream Self-Attention

Split view of Query Stream – $g^{layer}_{position}$
Position 4 View
Factorization order: `[3, 2, 4, 1]`



[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Two-Stream Self-Attention



Split view of Query Stream
Position 1 View
Factorization order: [3, 2, 4, 1]

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Conclusion

XLNet: Generalized Autoregressive Pretraining for Language Understanding

- Pretrain without data corruption (masking)
- Using Permutation LM

- Autoregressive LM
- Utilizes bidirectional content

[4] XLNet: Generalized Autoregressive Pretraining for Language Understanding

Agenda:

- Advanced approaches
  - Transformer
  - BERT
  - Transformer-XL
  - XLNet
  - **MT-DNN**

MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Xiaodong Liu , Jianfeng Gao
Microsoft Research

Pengcheng He , Weizhu Chen
Microsoft Dynamics 365 AI

MT-DNN Objective:

- Learn representations across multiple Natural Language Understanding (NLU) tasks
- Leverages
  - large amount of cross-task data
  - benefits from regularization effects
    - more general representation
    - help adapt new tasks and domains

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Approaches:

1. Multi-task learning

- Learn multiple tasks jointly so that knowledge learned in one task can benefit other tasks

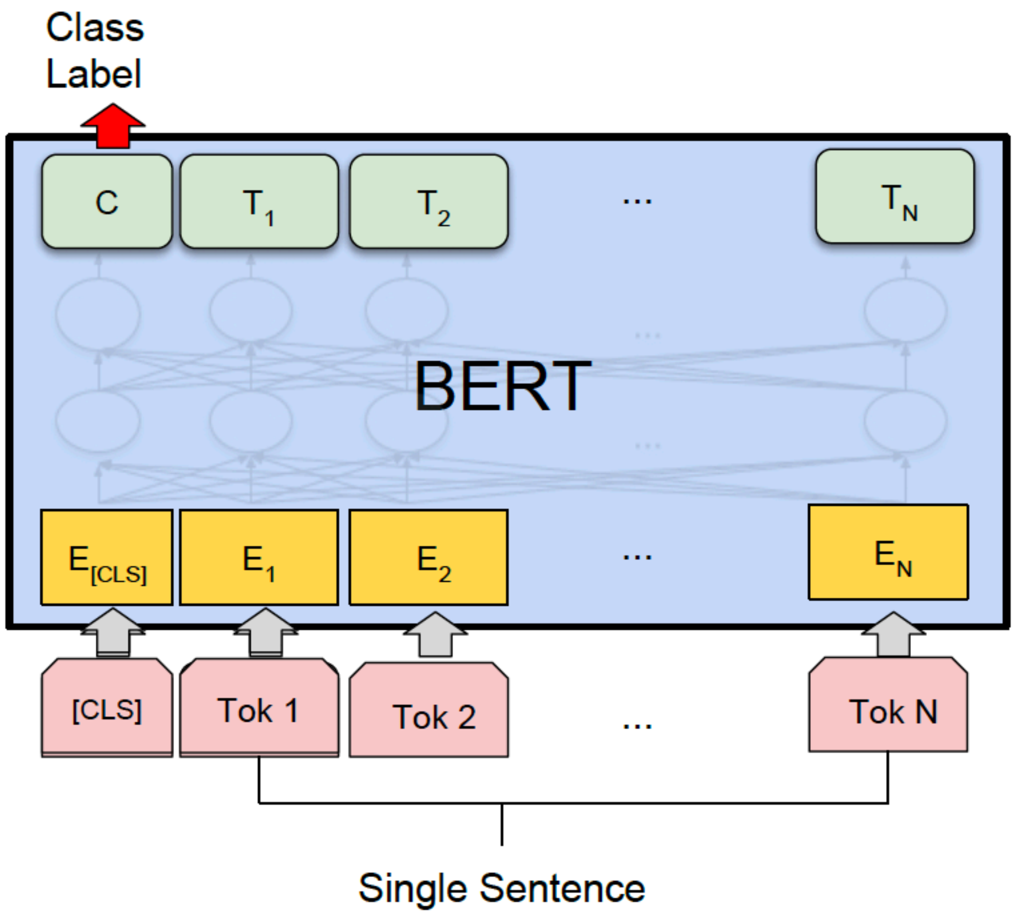
- Addresses:
  - Lack of large amount of supervised data
    - Leverage supervised data from many related tasks
  - Overfitting one specific task
    - Multi-task learning gains from regularization effect
    - Learned Representation is universal across tasks
  - Adoption to New Domain with fewer dataset

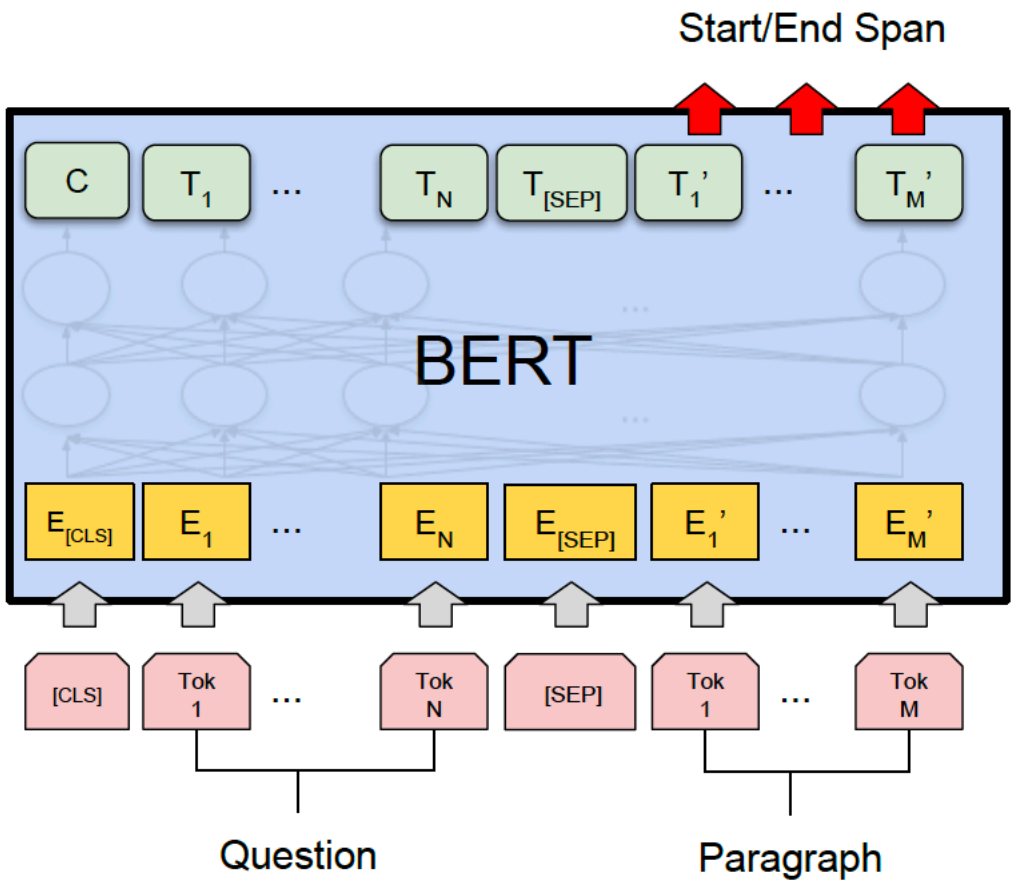
[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Approaches:

2. Language model pre-training

- Utilizes large amount of unlabeled data (eg. BERT)
- Fine tune pre-trained model for specific NLU task



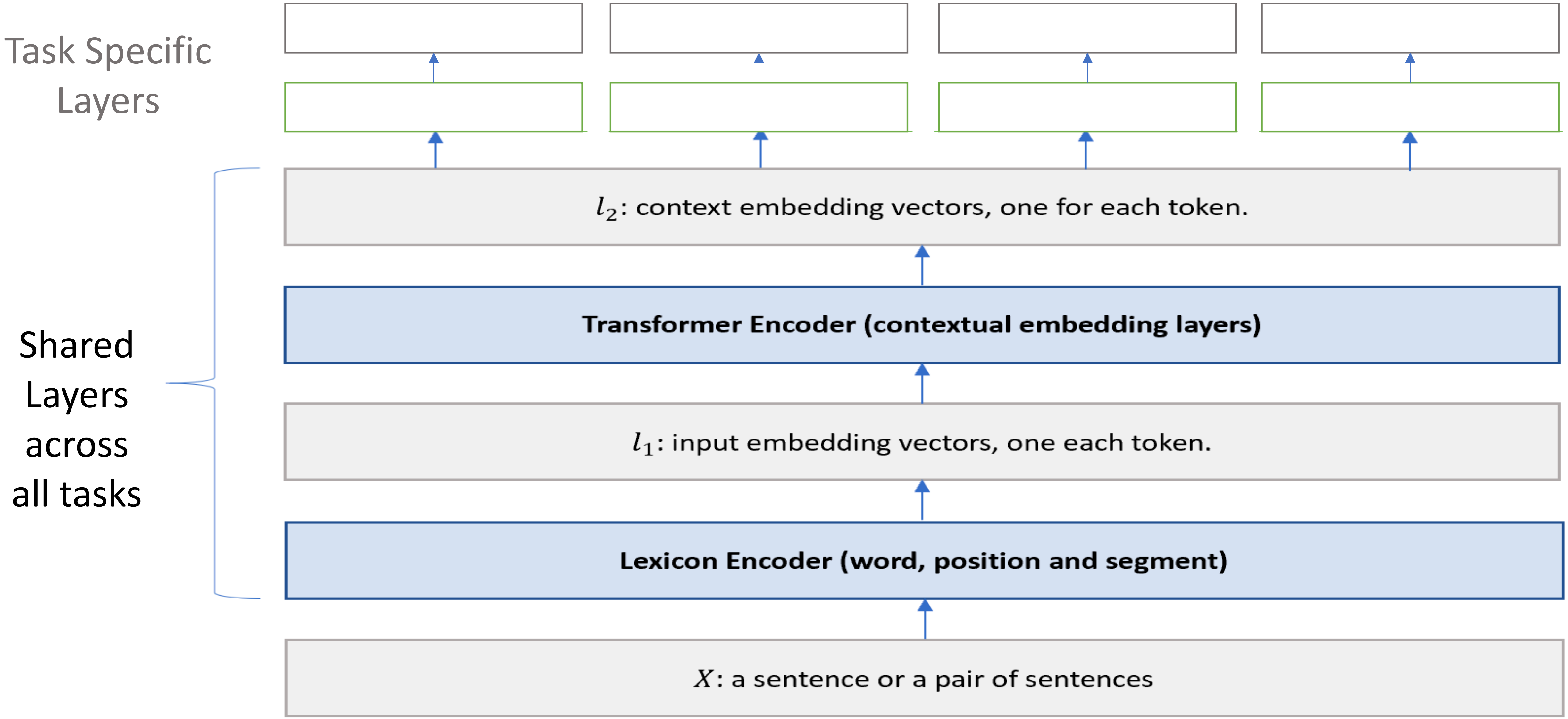


NLU tasks:

- Single Sentence Classification
- Text Similarity Scoring
- Pairwise Text Classification
- Relevance Ranking

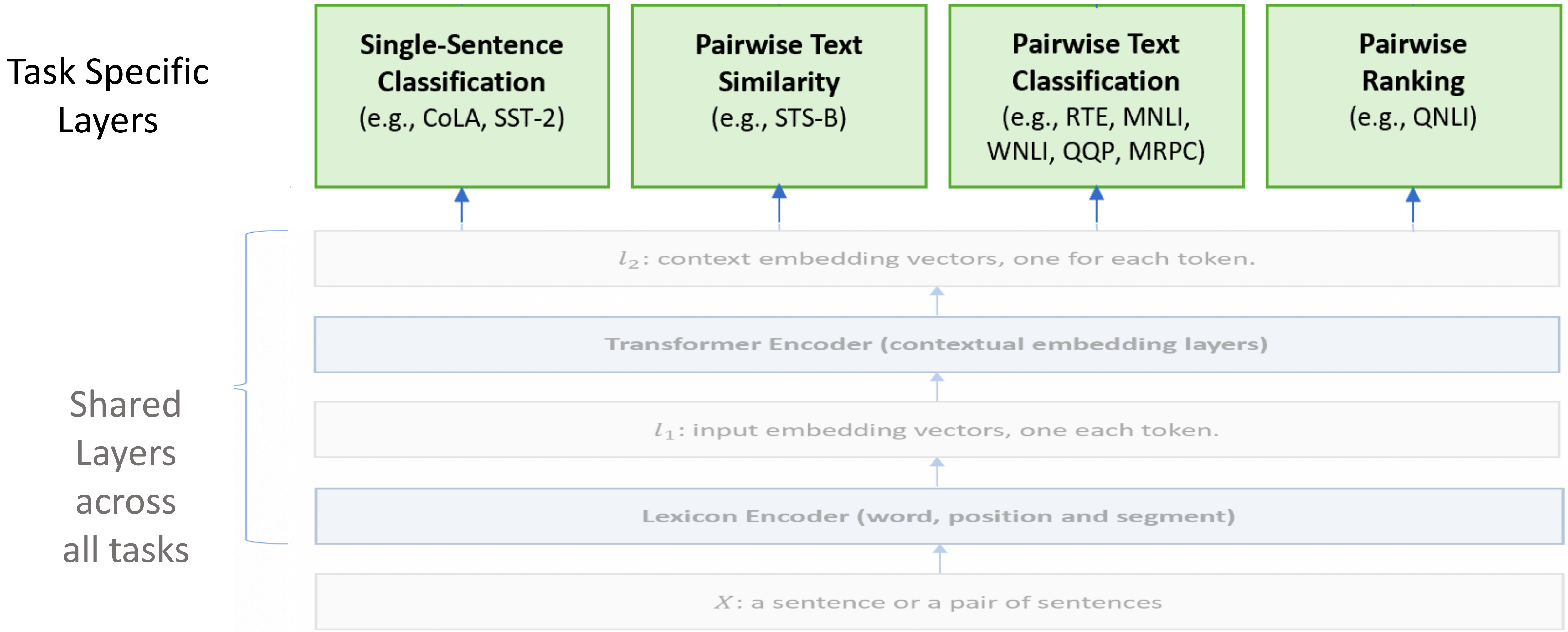
[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Architecture of MT-DNN Model



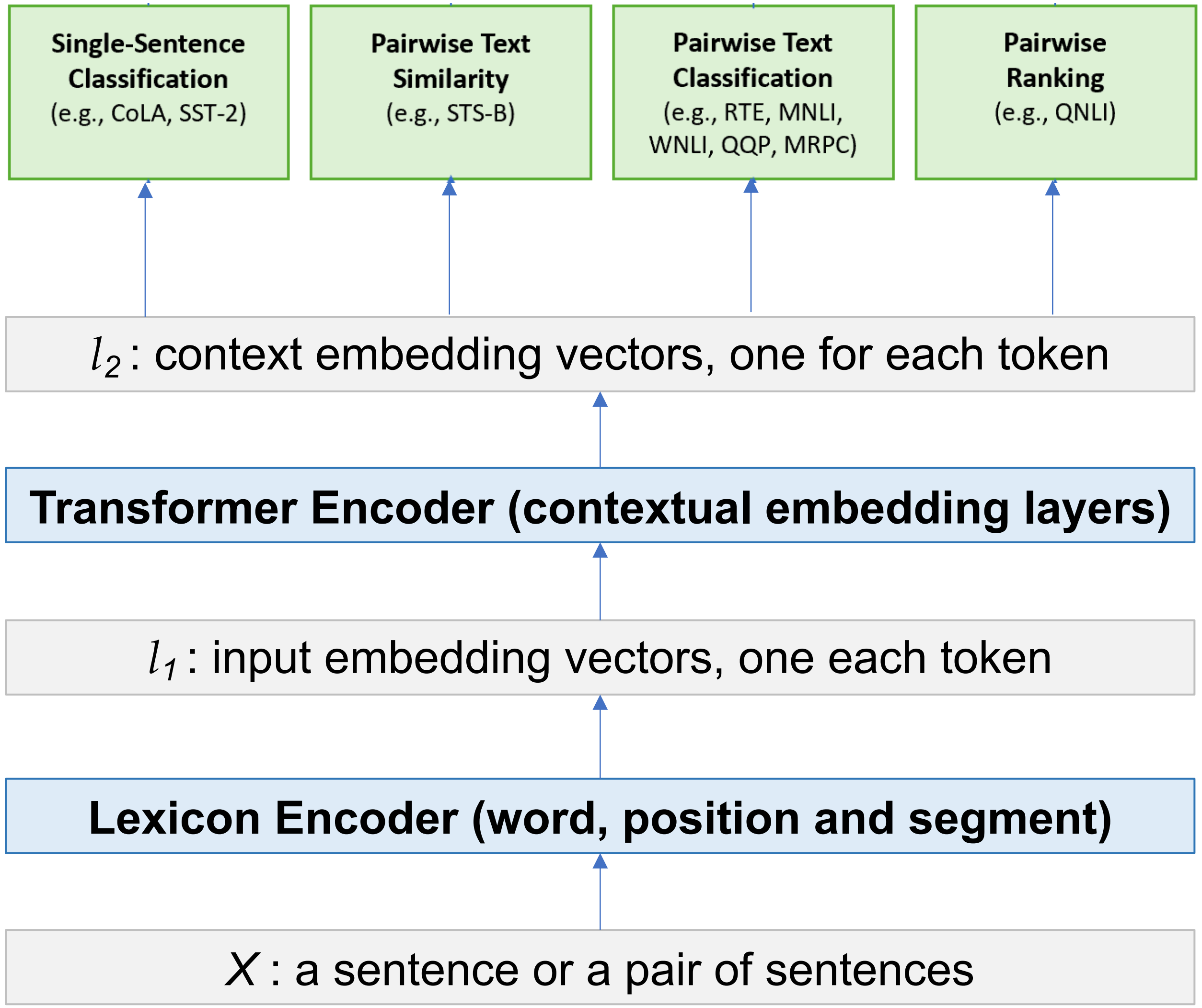
[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Architecture of MT-DNN Model



[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

MT-DNN Model Layers



Operations necessary for classification, similarity scoring, or relevance ranking

For each task, additional task-specific layers generate task-specific representations

Shared semantic representation that is trained by multi-task objectives

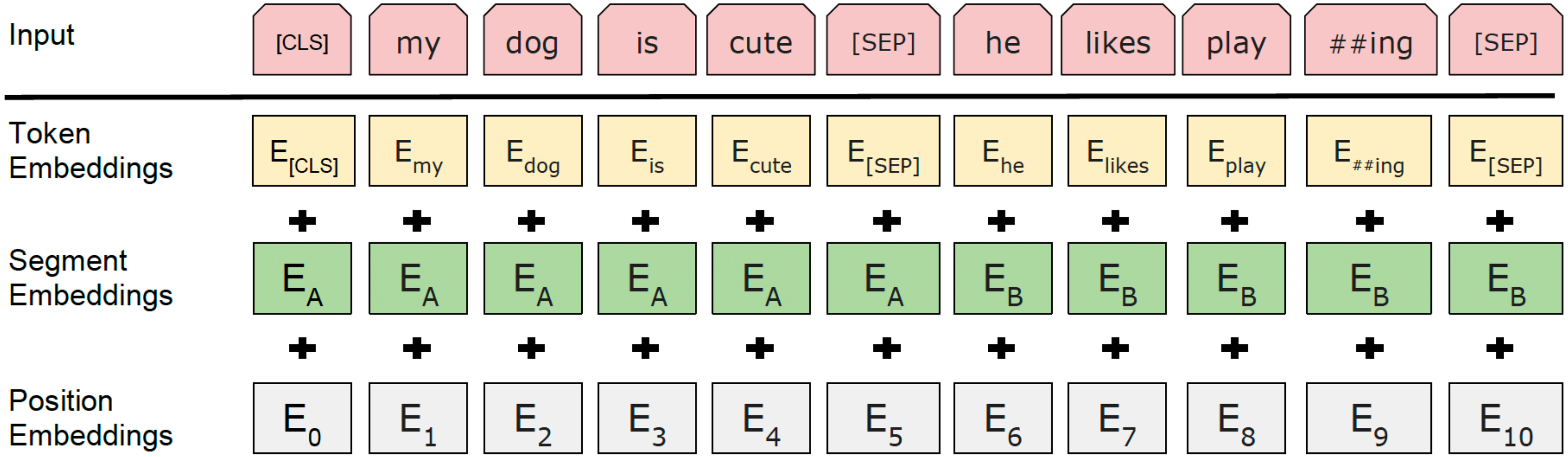
Generates a sequence of contextual embeddings in $l2$

Transformer encoder captures the contextual information for each word via self-attention

Sequence of embedding vectors, one for each word, in $l1$

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Lexicon Encoder ($l1$):



Single Sentence:
The input $X = f\{ x_1; \ldots ; x_m \}$ is a sequence of tokens of length m.
First token x_1 is always the [CLS] token.

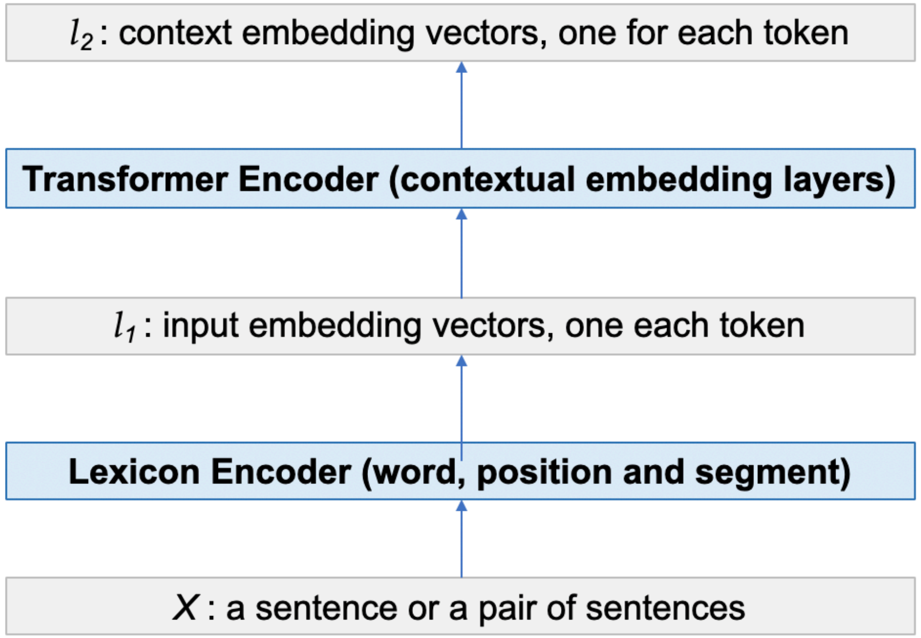
Sentence Pair:
Separate two X_1 and X_2 sentences with a special token [SEP]

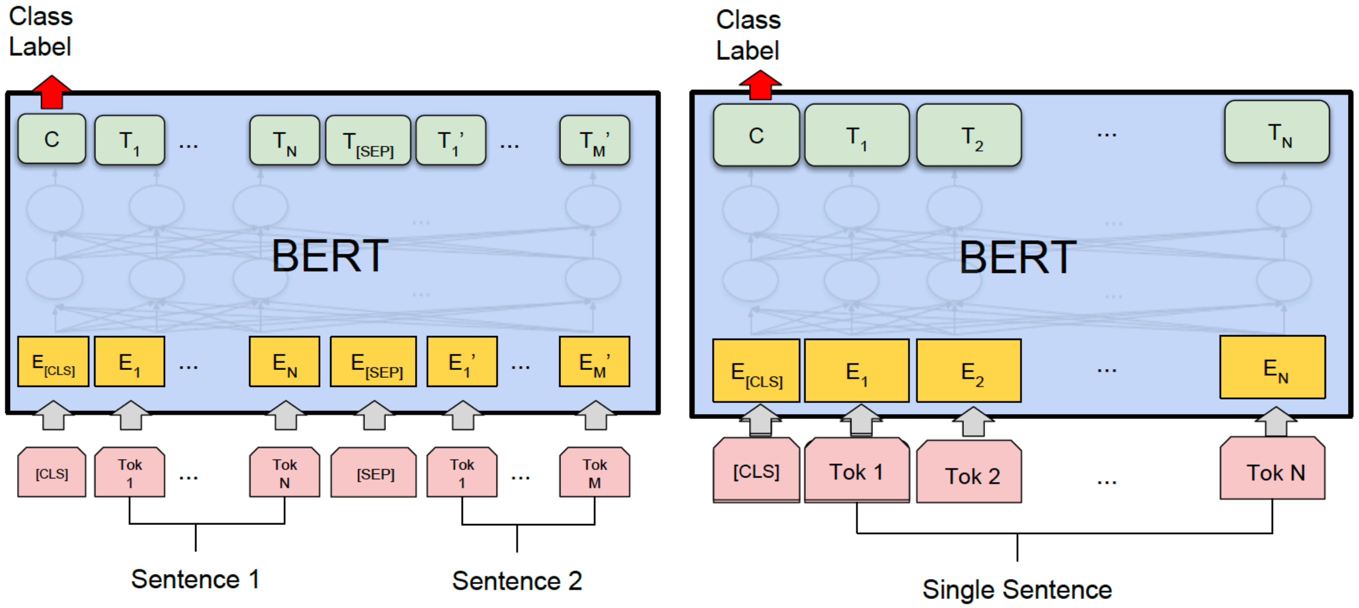
Maps X into a sequence of input embedding vectors, one for each token, constructed by summing

- the corresponding word
- segment and
- positional embeddings

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Transformer Encoder ($l2$):





(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

Uses a multilayer bidirectional Transformer encoder to map the input representation vectors ($l1$) into a sequence of contextual embedding vectors $\mathbf{C} \in \mathbb{R}^{d\times m}$

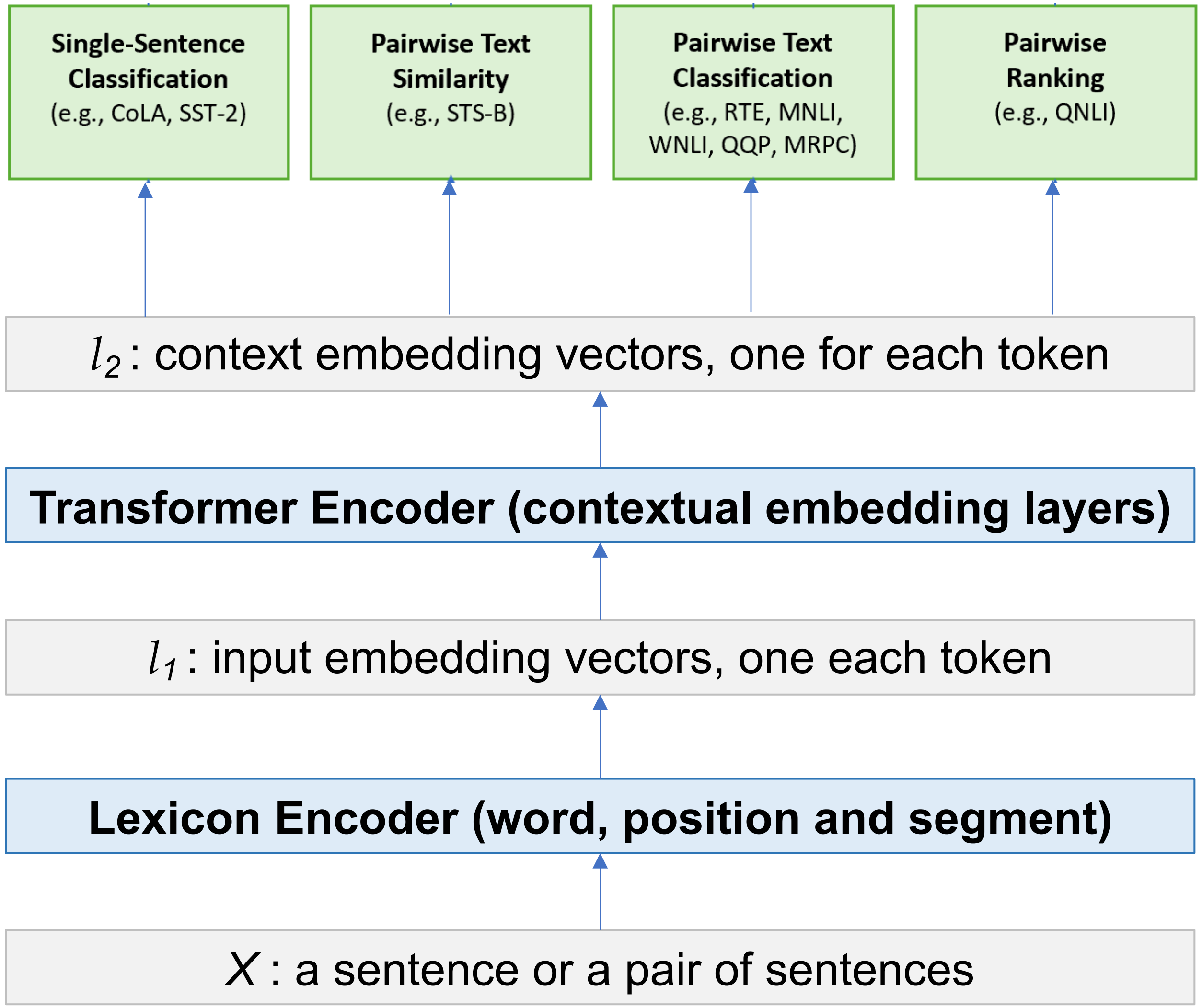
- MT-DNN learns the representation using multi-task objectives

- BERT model learns the representation via pre-training and adapts it to each individual task via fine-tuning.

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

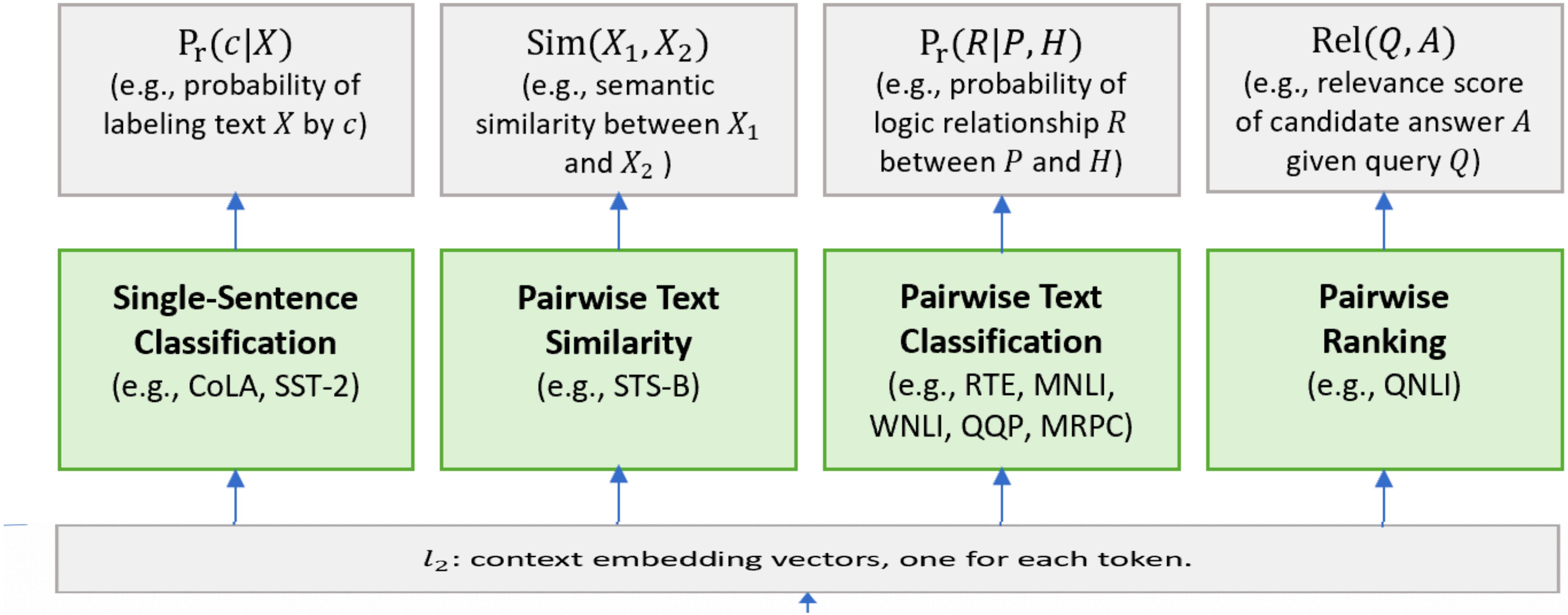
MT-DNN Model Layers

Again for Ref:



[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

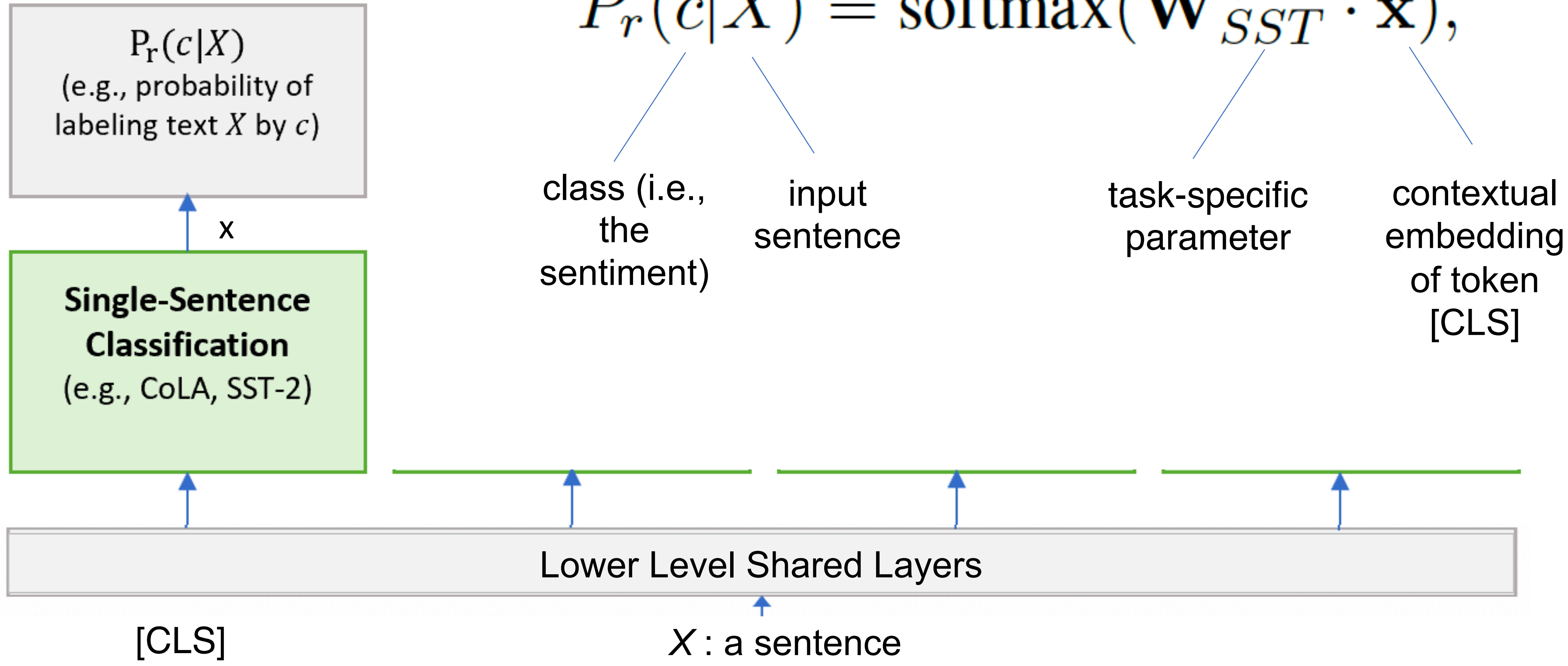
4 NLU tasks



[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

1 Single Sentence Classification:

$$P_r(c|X) = \text{softmax}(\mathbf{W}_{SST}^{\top} \cdot \mathbf{x}), \qquad (1)$$



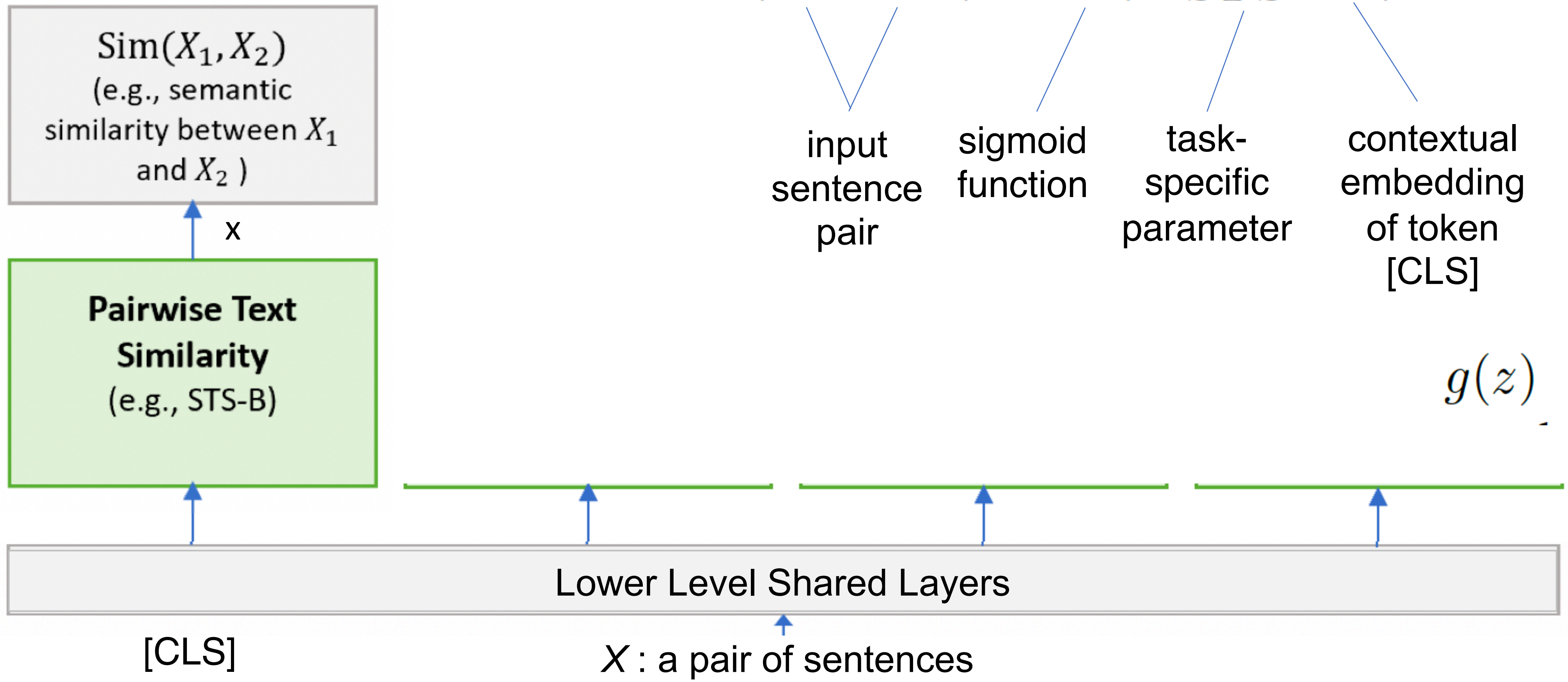
[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

2 Pairwise Text Similarity

$$\mathrm{Sim}(X_1, X_2) = g(\mathbf{w}_{STS}^{\top} \cdot \mathbf{x}), \qquad (2)$$

input sentence pair — sigmoid function — task-specific parameter — contextual embedding of token [CLS]

$$g(z) = \frac{1}{1+\exp(-z)}$$



[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

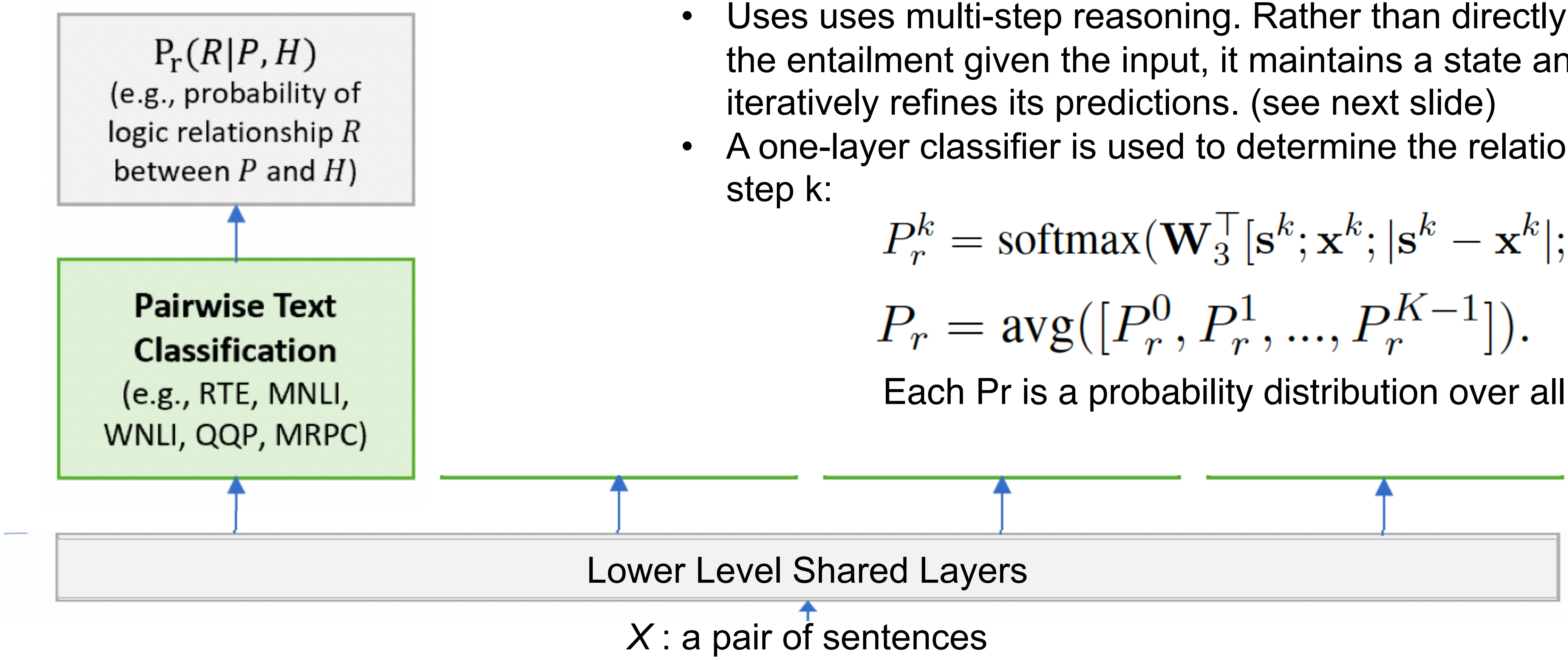
3 Pairwise Text Classification

- Premise P = (p_1, . . . , p_m) of m words
- Hypothesis H = (h_1, . . . , h_n) of n words
- Find a logical relationship R between P and H.
- **Stochastic Answer Network (SAN):**
  - Uses uses multi-step reasoning. Rather than directly predicting the entailment given the input, it maintains a state and iteratively refines its predictions. (see next slide)
  - A one-layer classifier is used to determine the relation at each step k:

$$P_r^k = \mathrm{softmax}(\mathbf{W}_3^\top[\mathbf{s}^k; \mathbf{x}^k; |\mathbf{s}^k - \mathbf{x}^k|; \mathbf{s}^k \cdot \mathbf{x}^k]).$$

$$P_r = \mathrm{avg}([P_r^0, P_r^1, ..., P_r^{K-1}]). \qquad (4)$$

Each Pr is a probability distribution over all the relations R

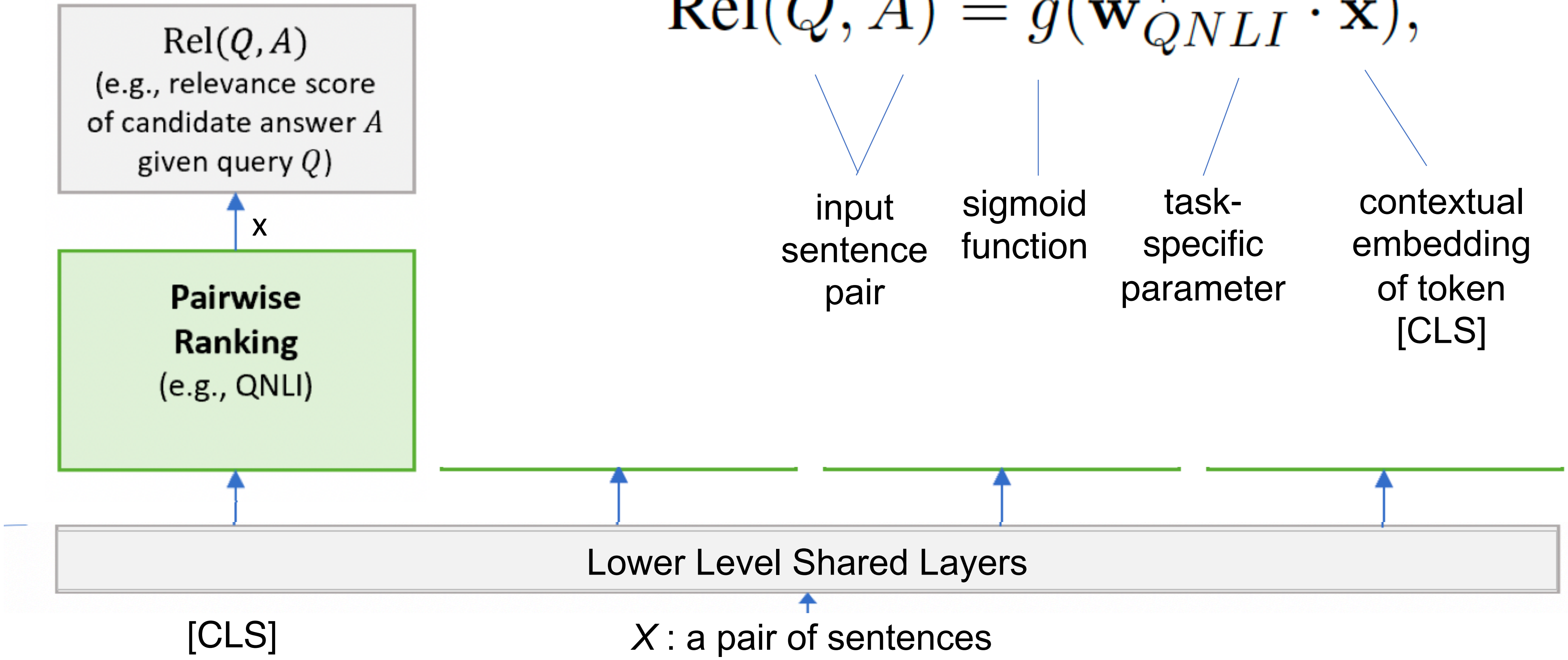


[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

4 Pairwise Ranking

- For a given Q, rank all of its candidate answers based on their relevance scores.

$$\mathrm{Rel}(Q, A) = g(\mathbf{w}_{QNLI}^{\top} \cdot \mathbf{x}), \quad (5)$$



[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Objectives for tasks (1/2):

For the **classification tasks** (i.e., single-sentence or pairwise text classification), use the **cross entropy loss** as the objective:

$$\boxed{-\sum_{c} \mathbb{1}(X, c) \log(P_r(c|X)), \qquad (6)}$$

where 1 (X, c) is the binary indicator (0 or 1) if class label c is the correct classification for X, and Pr(.) is defined by e.g., Equation 1 or 4.

$$P_r(c|X) = \text{softmax}(\mathbf{W}_{SST}^{\top} \cdot \mathbf{x}), \qquad (1)$$

$$P_r = \text{avg}([P_r^0, P_r^1, ..., P_r^{K-1}]). \qquad (4)$$

For the **text similarity tasks** where each sentence pair is annotated with a real valued score y, we use the **mean squared error** as the objective:

$$\boxed{(y - \text{Sim}(X_1, X_2))^2, \qquad (7)}$$

where Sim(.) is defined by Equation 2.

$$\text{Sim}(X_1, X_2) = g(\mathbf{w}_{STS}^{\top} \cdot \mathbf{x}), \qquad (2)$$

$$g(z) = \frac{1}{1+\exp(-z)}$$

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Objectives for tasks (2/2):

- The objective for the **relevance ranking** tasks follows the pairwise **learning-to-rank** paradigm
- Given a query Q, obtain a list of candidate answers A which contains a positive example A+ that includes the correct answer and IAI-1 negative examples.
- Minimize the **negative log likelihood** of the positive example given queries across the training data

$$\boxed{-\sum_{(Q,A^+)} P_r(A^+|Q), \qquad (8)}$$

$$P_r(A^+|Q) = \frac{\exp(\gamma \mathrm{Rel}(Q, A^+))}{\sum_{A' \in \mathcal{A}} \exp(\gamma \mathrm{Rel}(Q, A'))}, \qquad (9)$$

where Rel(.) is defined by Equation 5

$$\mathrm{Rel}(Q, A) = g(\mathbf{w}_{QNLI}^\top \cdot \mathbf{x}), \qquad (5)$$

γ is a tuning factor determined on held-out data. In experiment, it is set to 1.

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

1. Training: Pre-Training

BERT:

- The parameters of the lexicon encoder and Transformer encoder are learned using two unsupervised prediction tasks:
  - masked language modelling
  - next sentence prediction.

Algorithm 1: Training a MT-DNN model.

Initialize model parameters Θ randomly.
Pre-train the shared layers (i.e., the lexicon encoder and the transformer encoder).
Set the max number of epoch: $epoch_{max}$.
//Prepare the data for T tasks.
for t *in* $1, 2, ..., T$ **do**
  Pack the dataset t into mini-batch: D_t.
end
for *epoch in* $1, 2, ..., epoch_{max}$ **do**
  1. Merge all the datasets:
  $D = D_1 \cup D_2 ... \cup D_T$
  2. Shuffle D

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

2. Multi-Task Fine tuning stage

for *epoch in* $1, 2, ..., epoch_{max}$ **do**

1. Merge all the datasets:

   $D = D_1 \cup D_2 ... \cup D_T$

2. Shuffle D

   for b_t *in* D **do**

   //b_t is a mini-batch of task t.

   3. Compute loss : $L(\Theta)$

      $L(\Theta) =$ Eq. 6 for classification

      $L(\Theta) =$ Eq. 7 for regression

      $L(\Theta) =$ Eq. 8 for ranking

   4. Compute gradient: $\nabla(\Theta)$

   5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$

   end

end

$$-\sum_c \mathbb{1}(X, c) \log(P_r(c|X)), \quad (6)$$

$$(y - \text{Sim}(X_1, X_2))^2, \quad (7)$$

$$-\sum_{(Q,A^+)} P_r(A^+|Q), \quad (8)$$

$$P_r(A^+|Q) = \frac{\exp(\gamma \text{Rel}(Q, A^+))}{\sum_{A' \in \mathcal{A}} \exp(\gamma \text{Rel}(Q, A'))}, \quad (9)$$

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

Summary:

- Advanced approaches for Neural Conversational AI
  - Transformer – Attention Is All You Need
  - BERT
  - Transformer-XL
  - XLNet
  - MT-DNN
- Application of BERT for Intent Detection and Slot filling

References:

[1] Vasvani A., et. al Attention Is All You Need

[2] Devlin J. et. al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[3] Dai Z. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

[4] Yang Z. et. al XLNet: Generalized Autoregressive Pretraining for Language Understanding

[5] Liu X. et. al. MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Thank You.

Contact:
omprakash.s@flipkart.com
omsonie@gmail.com

Refer: www.DeepThinking.ai

BERT for Joint Intent Classification and Slot Filling

Qian Chen, Zhu Zhuo,Wen Wang
Speech Lab, DAMO Academy, Alibaba Group
ftanqing.cq, zhuozhu.zz, w.wangg@alibaba-inc.com

Problem Statement – Customer Support Chatbot

Create a conversational agent (aka Chatbot) for customer service where the agent should be able to converse with customers on their issues (e.g., delivery, return, refund, cancellation etc.). A realistic conversation should be able to handle

Essential task: Multiple intents and Slot filling.

Core Problem: Intent Detection and Slot Filling

Query	Find me a movie by Steven Spielberg	
Frame	**Intent**	find_movie
	Slot	genre = movie directed_by = Steven Spielberg

An example from user query to semantic frame.

Core Problem: Multiple Intent Detection and Slot Filling

Book flight from <departure city> to <arrival city> on <date>

Single Intent:
Book Flight (intent)
Arrival city, Departure city, date (slots)

Yes, book the flight

Multiple intent - Confirmation and book flight

BERT for Joint Intent Classification and Slot Filling

- Poor Generalisation capability:
    - Suffer from small-scale human-labeled training data especially for rare words.
- BERT facilitates pre-training deep bidirectional representations on large-scale unlabelled corpora
- Created state-of-the-art models for a wide variety of natural language processing tasks after simple fine-tuning

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Approaches

- Slot is at word level
- Intent is at Sentence level

Separately learn:

- Slot filling
- Intent Detection

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Jointly learn Intent and Slot filling

Recent approaches based on:

- BERT
- Sequence
- Transformer and Sequence

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

BERT for Joint Intent Classification and Slot Filling

To jointly model intent classification and slot filling, the objective is formulated as:

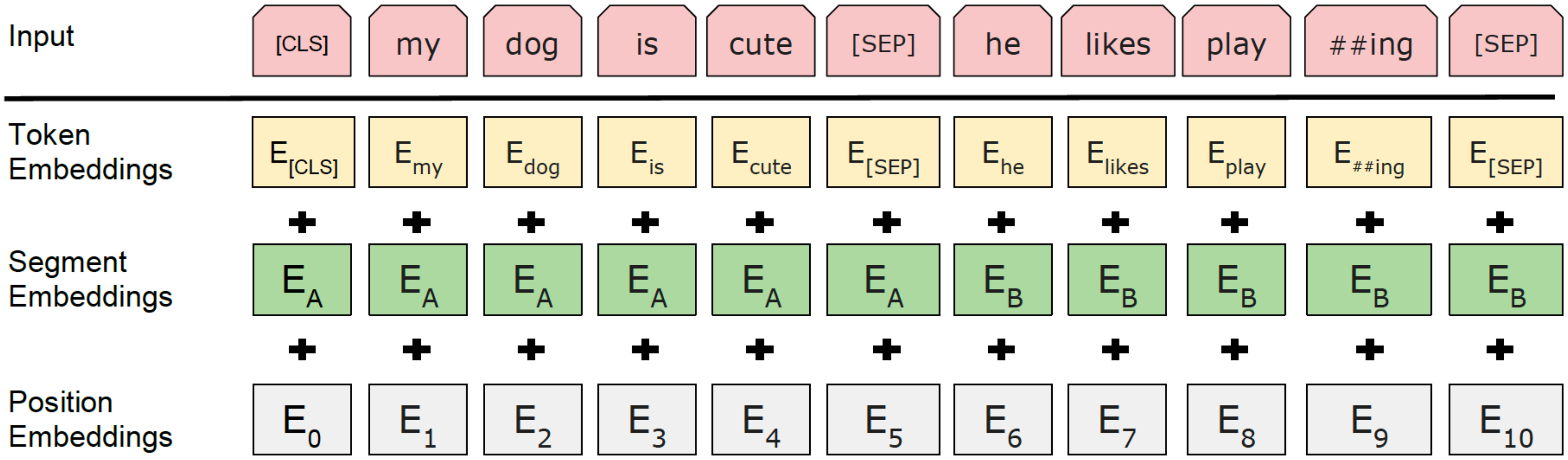
$$p(y^i, y^s|\boldsymbol{x}) = p(y^i|\boldsymbol{x}) \prod_{n=1}^{N} p(y_n^s|\boldsymbol{x})$$

The learning objective is to maximize the conditional probability $p(y^i, y^s|\boldsymbol{x})$

The model is finetuned end-to-end via minimizing the cross-entropy loss.

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

BERT for Joint Intent Classification and Slot Filling

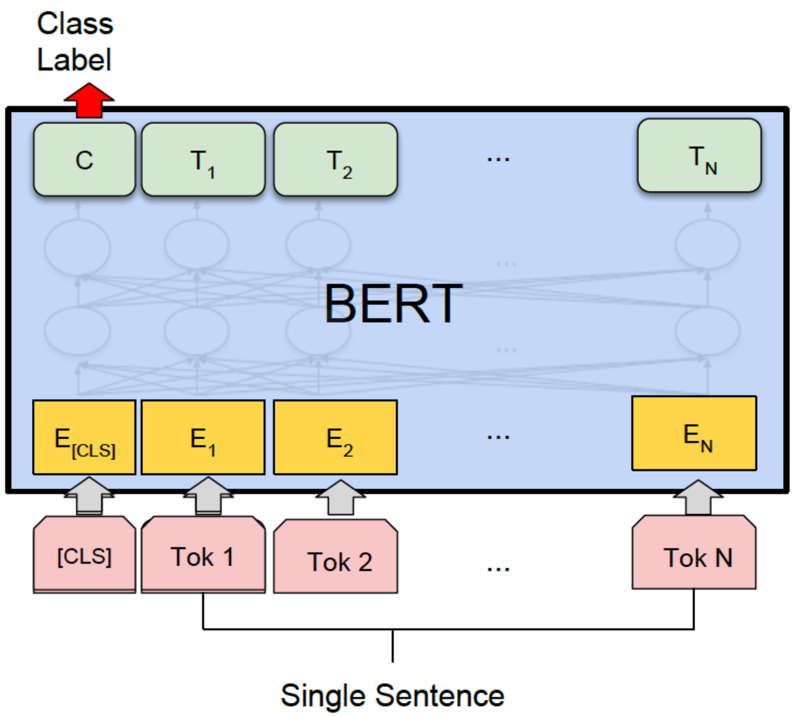


BERT uses: Word embedding, Segment Embedding and Position Embedding

For single sentence classification and tagging tasks, the segment embedding has no discrimination

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

BERT for Joint Intent Classification and Slot Filling

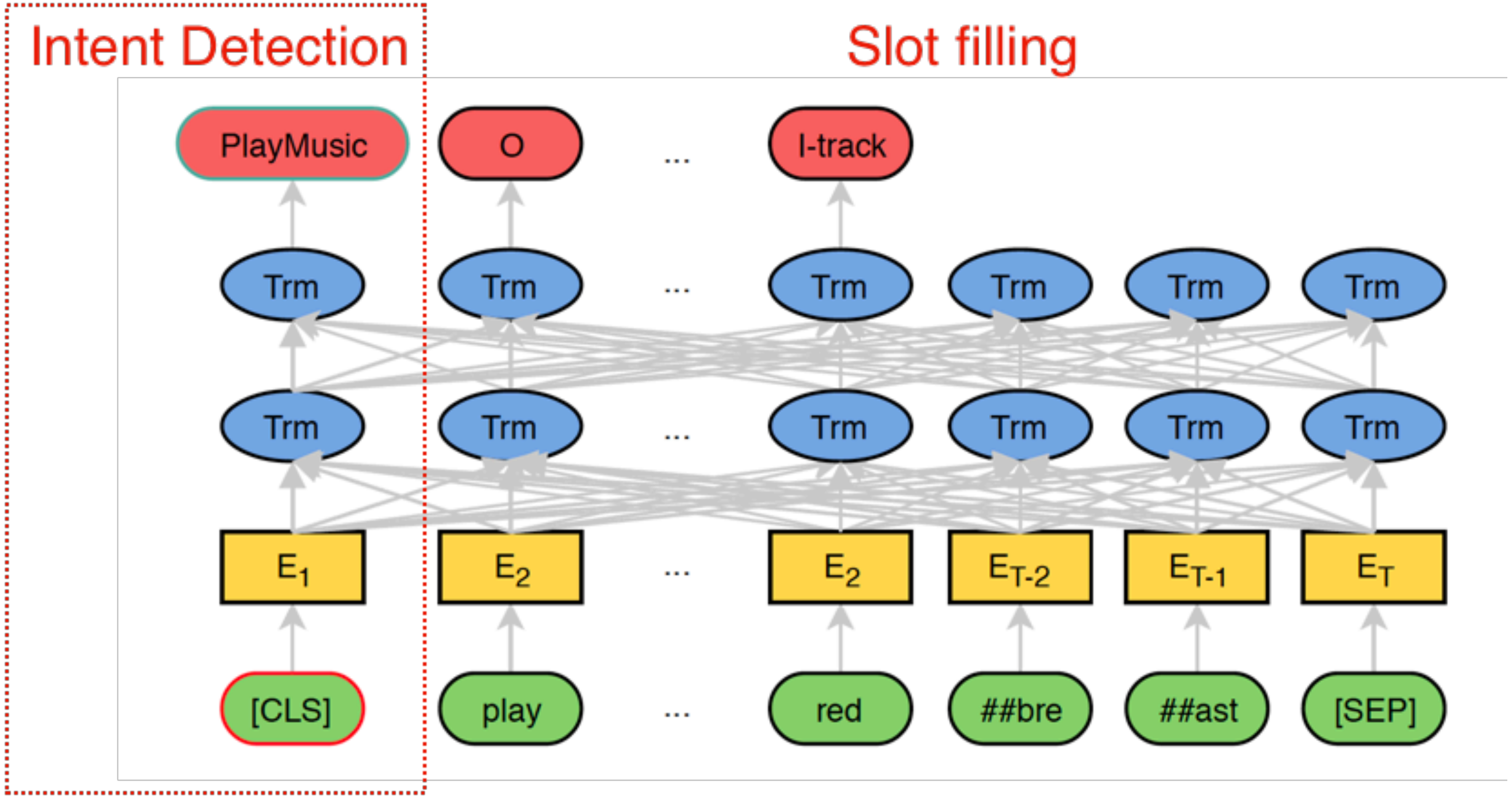


A special classification embedding ([CLS]) is inserted as the first token and a special token ([SEP]) is added as the final token.
Given an input token sequence x = (x1; : : : ; xT), the output of BERT is H = (h1; : : : ;hT).

The pre-trained BERT model provides a powerful context-dependent sentence representation and can be used for various target tasks, i.e., intent classification and slot filling, through the finetuning procedure.

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

BERT for Joint Intent Classification and Slot Filling

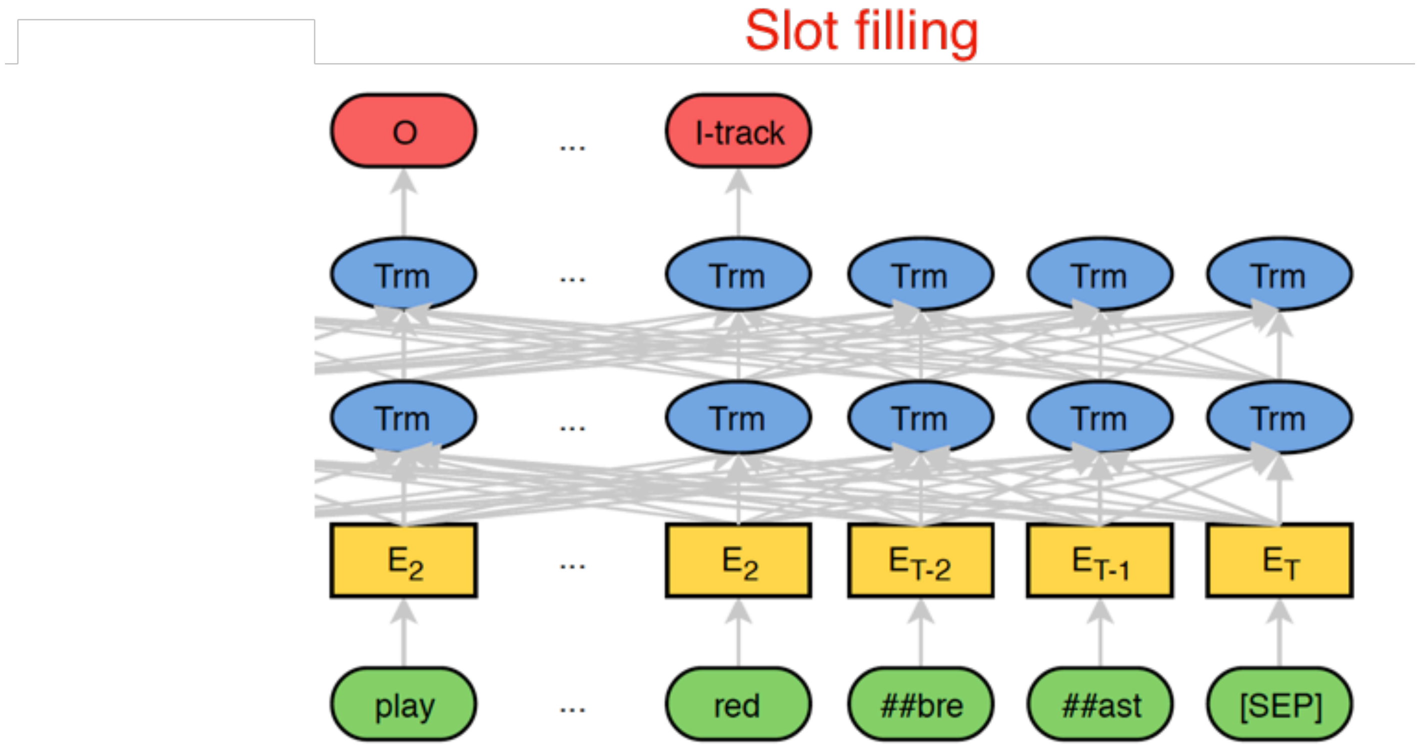


input: play the song little robin redbreast

Based on the hidden state of the first special token ([CLS]), denoted h1, the intent is predicted as: $y^i = \mathrm{softmax}(W^i h_1 + b^i)$;

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

BERT for Joint Intent Classification and Slot Filling



input: play the song little robin redbreast

For slot filling, we feed the final hidden states of other tokens h2; : : : ;hT into a softmax layer to classify over the slot filling labels.

y^s_n = softmax($W^s h_n$ + b^s) ; n belongs to 1 to N

where h_n is the hidden state corresponding to the first sub-token of word x_n.

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Datasets:

ATIS dataset:
Includes audio recordings of people making flight reservations.

Utterances:

- Training set: 4,478
- Development set: 500
- Test set 893

For training set:
- Slots labels: 120
- Intent types: 21

Snips dataset:
Snip personal voice assistant.

Utterances:

- Training set: 13,084
- Development set: 700
- Test set 700

For training set:
- Slots labels: 72
- Intent types: 7

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Model: BERT for Joint Intent Classification and Slot Filling

Used English uncased BERT-Base model:

- 12 layers
- 768 hidden states
- 12 heads

BERT is pre-trained on BooksCorpus (800M words)

Fine-tuning, all hyper-parameters are tuned on the development set.
The maximum length is 50.
The batch size is 128.
Adam is used for optimization with an initial learning rate of 5e-5.
The dropout probability is 0.1.
The maximum number of epochs is selected from [1, 5, 10, 20, 30, 40].

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Result: BERT for Joint Intent Classification and Slot Filling

Models	Snips			ATIS		
	Intent	Slot	Sent	Intent	Slot	Sent
RNN-LSTM (Hakkani-Tür et al., 2016)	96.9	87.3	73.2	92.6	94.3	80.7
Atten.-BiRNN (Liu and Lane, 2016)	96.7	87.8	74.1	91.1	94.2	78.9
Slot-Gated (Goo et al., 2018)	97.0	88.8	75.5	94.1	95.2	82.6
Joint BERT	**98.6**	**97.0**	**92.8**	97.5	**96.1**	88.2
Joint BERT + CRF	98.4	96.7	92.6	**97.9**	96.0	**88.6**

NLU performance on Snips and ATIS datasets.
The metrics:

- Intent classification accuracy
- slot filling F1
- sentence-level semantic frame accuracy (%)

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

A case in the Snips dataset.

Query	need to see **mother joan of the angels** in one second
Gold, predicted by joint BERT correctly	
Intent	SearchScreeningEvent
Slots	O O O B-movie-name I-movie-name I-movie-name I-movie-name I-movie-name B-timeRange I-timeRange I-timeRange

Predicted by Slot-Gated Model (Goo et al., 2018)	
Intent	BookRestaurant
Slots	O O O B-object-name I-object-name I-object-name I-object-name I-object-name B-timeRange I-timeRange I-timeRange

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Summary:

- Advanced approaches for Neural Conversational AI
  - Transformer – Attention Is All You Need
  - BERT
  - Transformer-XL
  - XLNet
  - MT-DNN
- Application of BERT for Intent Detection and Slot filling

[6] Chen Q. BERT for Joint Intent Classification and Slot Filling

Thank You.

Contact:
omprakash.s@flipkart.com
omsonie@gmail.com

Refer: www.DeepThinking.ai