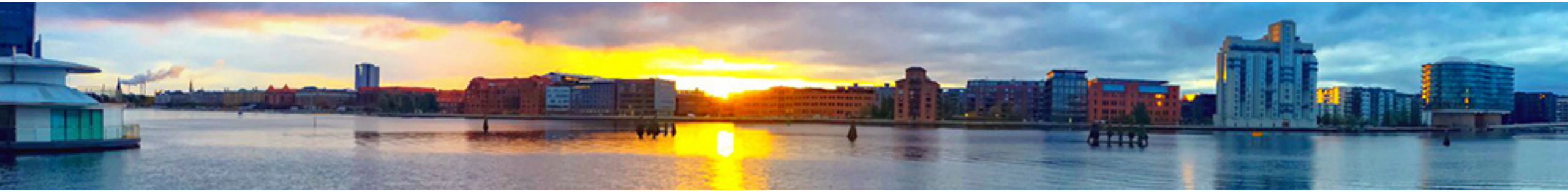
The ACM Conference Series on Recommender Systems
Copenhagen, Denmark Sept 16-20, 2019

Concept to Code: Deep Learning for Multitask Recommendation

Omprakash Sonie
Flipkart, India



Agenda

Session 1: Breadth (45min)

- An Overview of Multi-Task Learning in Deep Neural Networks
- SEMAX: Multi-Task Learning for Improving Recommendations
- Why I like it: Multi-task Learning for Recommendation and Explanation
- Neural Multi-Task Recommendation from Multi-Behavior Data
- Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation
- Code – Jupyter Notebook

Session 2: Depth (45min)

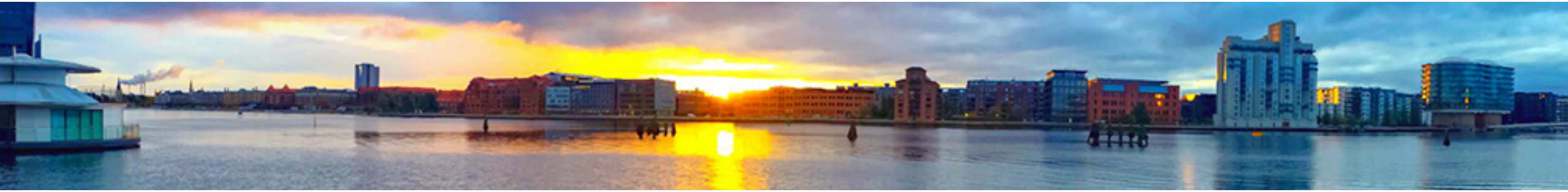
- BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer
- Transformer – Attention is All You Need
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- MT-DNN Multi-Task Deep Neural Networks for Natural Language Understanding
- Code – Jupyter Notebook

Session-1

(45min)

1

[1] Sebastian Ruder. 2017
An Overview of Multi-Task Learning in Deep Neural Networks.



Multi-Task Learning:

- Goal: Improve generalization by leveraging the domain-specific information contained in the training signals of related tasks
- Approach: Share representation between related tasks
- Applications:
  - Natural Language Processing, Speech Recognition, Computer Vision
- Self Driving Car
  - Recognize Traffic lights, Vehicles, Stop sign, Pedestrians, etc.
- Optimizing more than one loss function

[1] Sebastian Ruder. 2017 An Overview of Multi-Task Learning in Deep Neural Networks.

Why does MTL work?

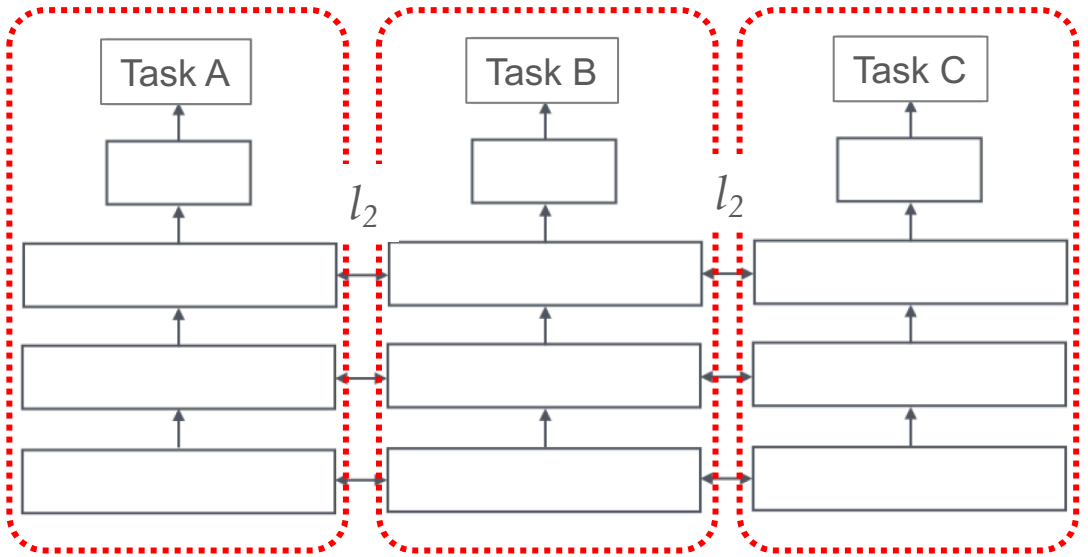
- Data Augmentation:
  - Increases sample size by jointly training - instead of training each task separately
- Attention Focusing:
  - On those features that actually matter, other tasks will provide additional evidence for the relevance or irrelevance of those features
- Representation bias:
  - Biases the model to prefer representation that other tasks also prefer
- Regularization:
  - Reduces risk of overfitting for one task
- Generalize well to a new task and adopt to new domain

[1] Sebastian Ruder. 2017 An Overview of Multi-Task Learning in Deep Neural Networks.

MTL Methods for Deep Learning:

1. Soft parameter Sharing

- Each task has its own model with its own parameters
- Distance between the parameters of the model is regularized
  - Parameters to be similar
  - l_2 distance is used for regularization

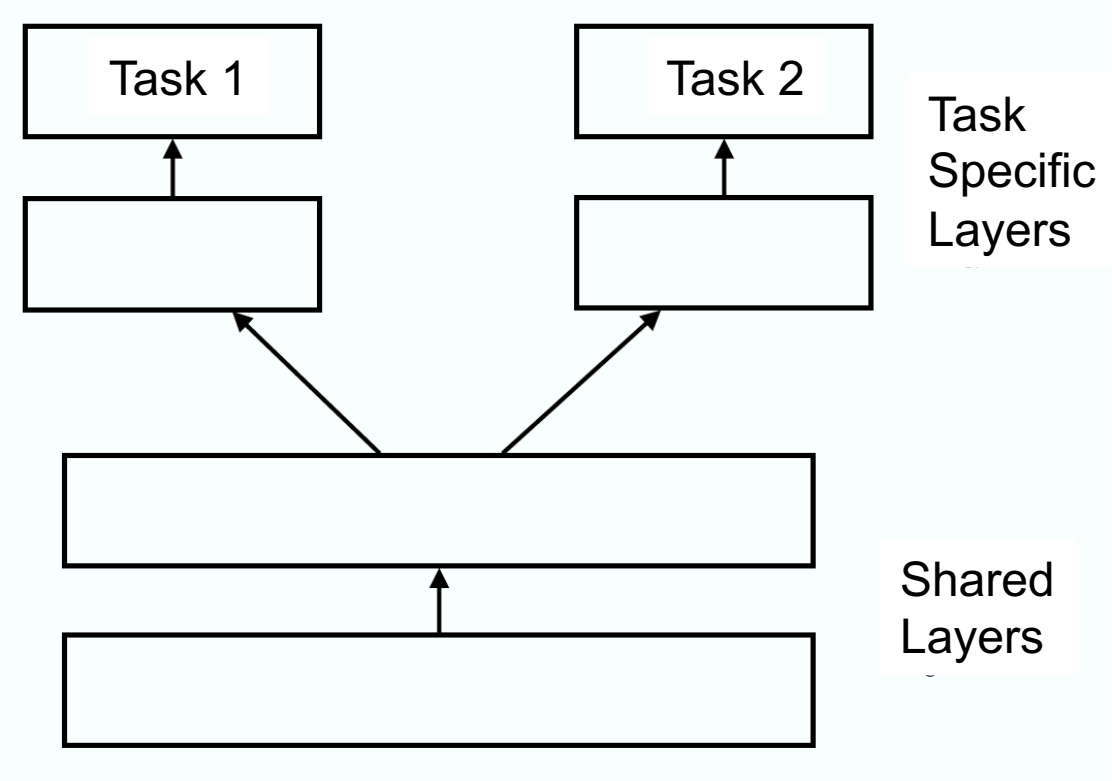


[1] Sebastian Ruder. 2017 An Overview of Multi-Task Learning in Deep Neural Networks.

MTL Methods for Deep Learning:

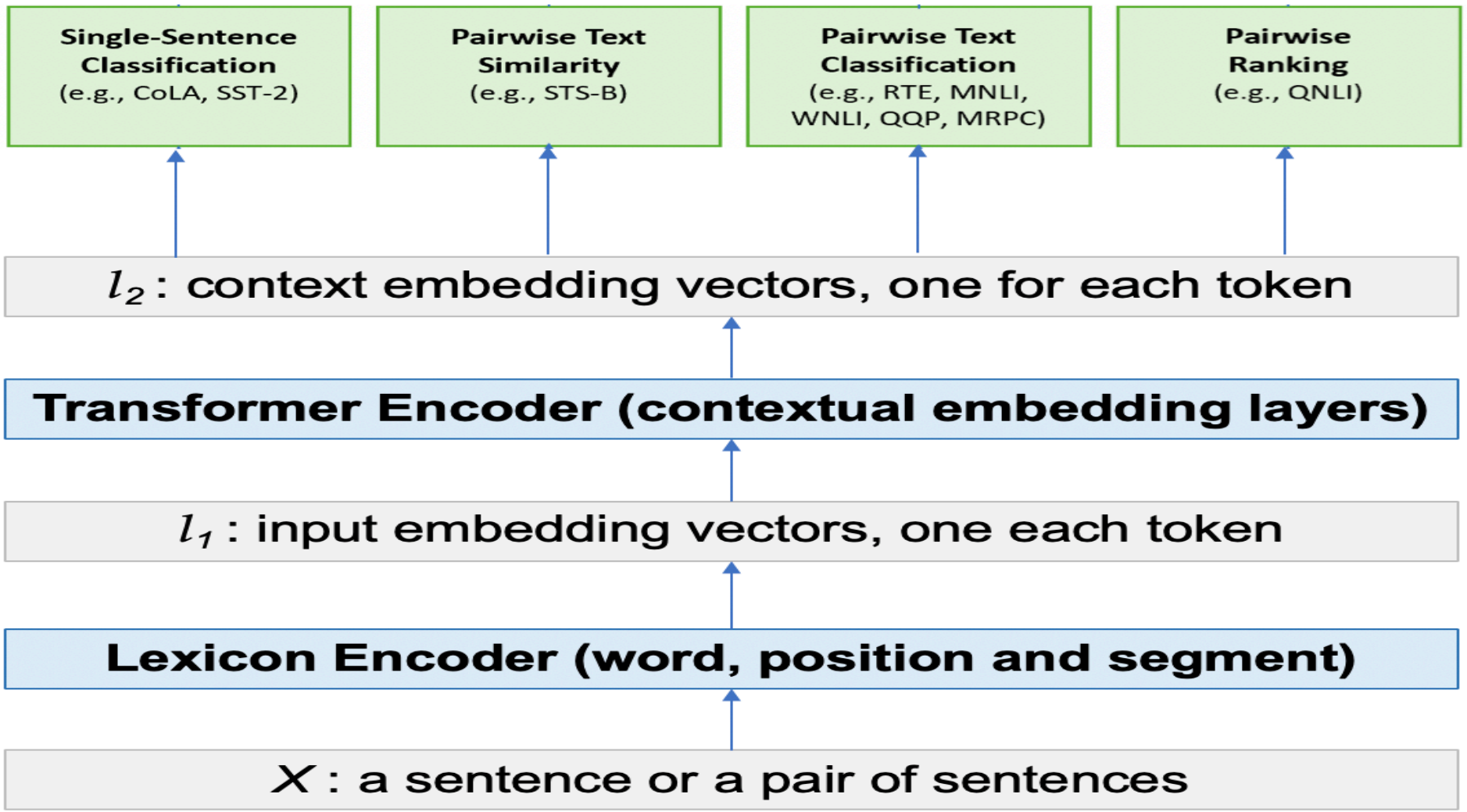
2. Hard parameter Sharing

- Most common approach
- Share hidden layers between all tasks
- Task specific output layers
- Model tries to find a representation that captures all the tasks
- Reduces risk of overfitting



[1] Sebastian Ruder. 2017 An Overview of Multi-Task Learning in Deep Neural Networks.

Hard parameter Sharing: MT-DNN for NLU (1/4)

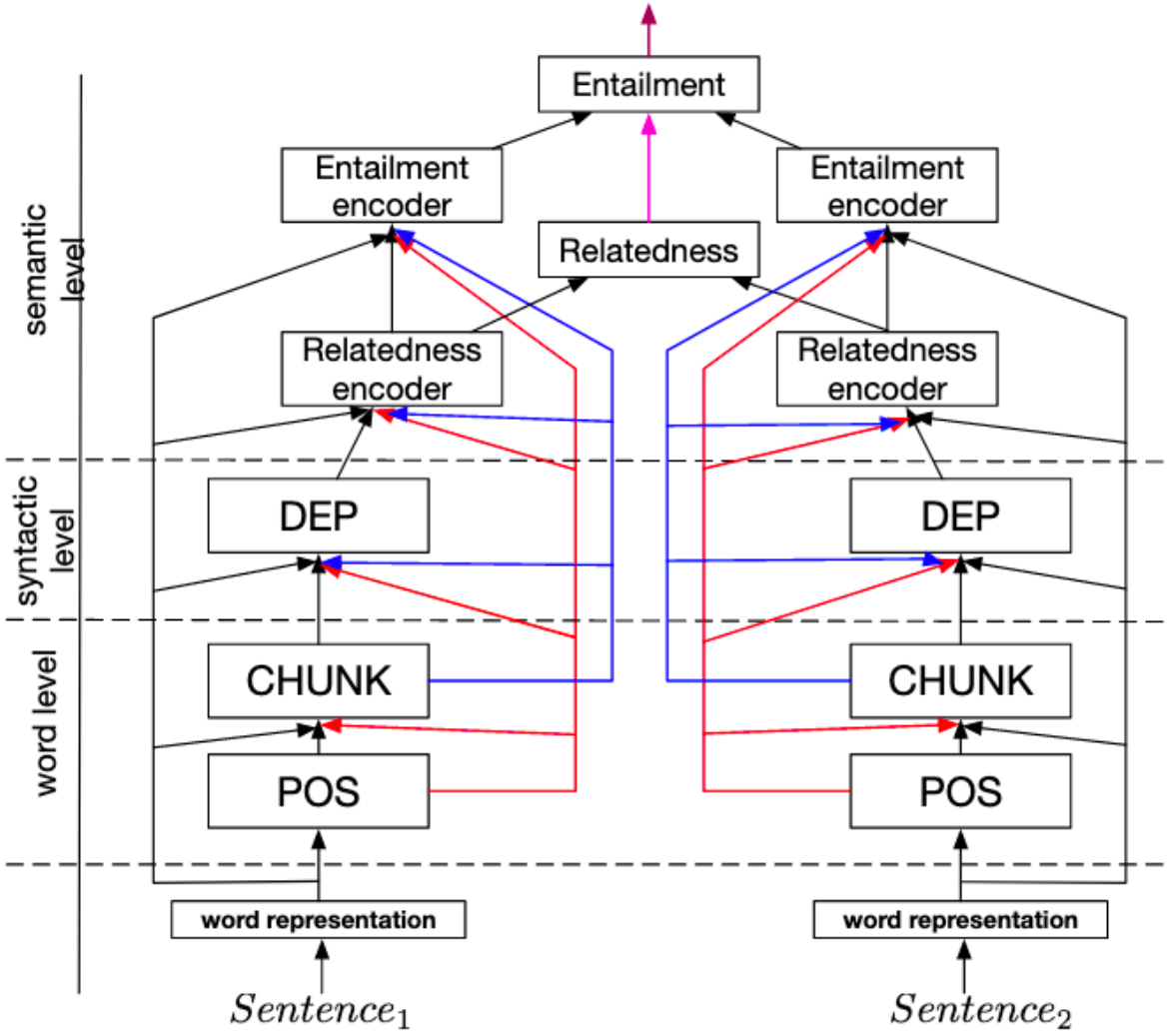


Task Specific Layers

Shared Layers across all tasks

Liu X. , Pengcheng He, Weizhu Chen, Jianfeng Gao (2019)
Multi-Task Deep Neural Networks for Natural Language Understanding

A Joint Many-Task Model (2/4)

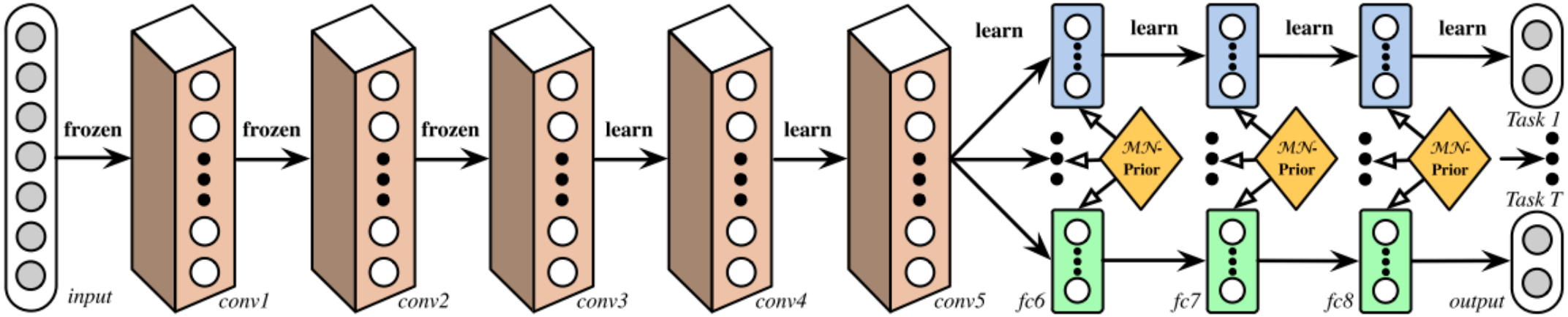


- Pipelined systems:
  - Part-Of-Speech (POS) tags are used for syntactic parsers.
  - The parsers are used to improve higher-level tasks, such as
    - natural language inference and
    - machine translation
- Predict increasingly complex NLP tasks at successively deeper layers
- Can be trained end-to-end for
  - POS tagging
  - Chunking
  - Dependency parsing
  - Demantic relatedness
  - Textual entailment

  by considering linguistic hierarchies

Hashimoto, K., Xiong, C., Tsuruoka, Y., and Socher, R. (2016). A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks.

Deep Relationship Networks: Computer Vision (3/4)



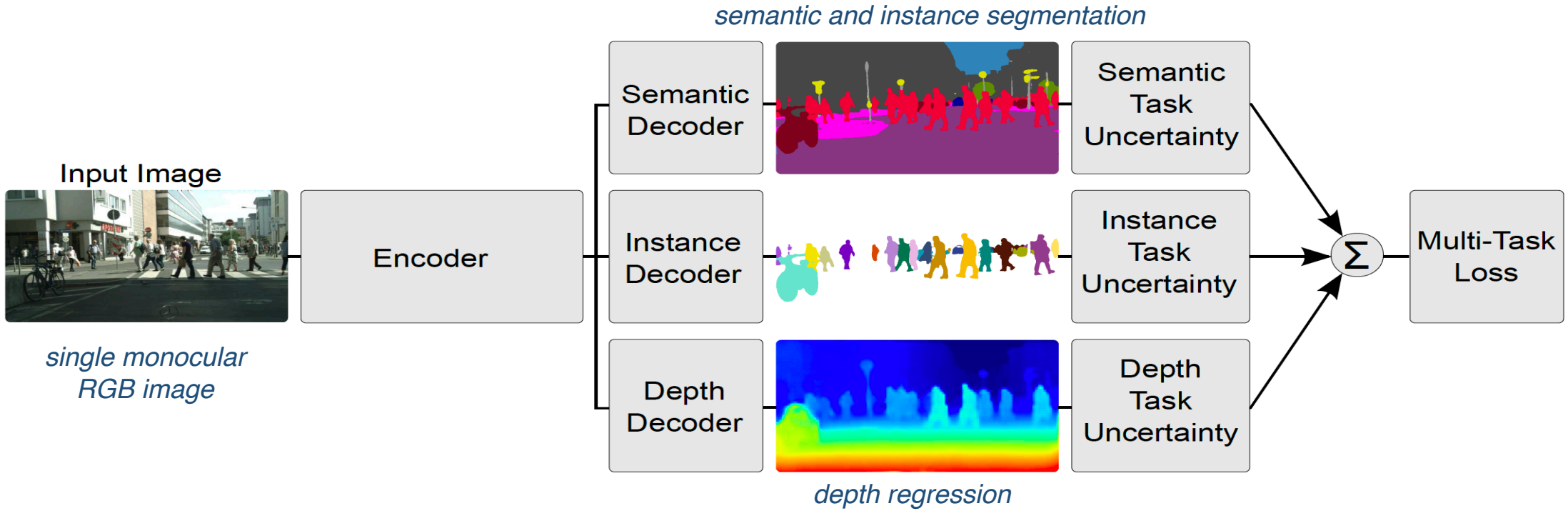
share the convolutional layers

task specific
fully-connected layers

place **matrix priors** on the fully connected layers, which allow the model **to learn the relationship between tasks**

Long, M. and Wang, J. (2015) Learning Multiple Tasks with Deep Relationship Networks

Uncertainty-based loss function weighting for multi-task learning (4/4)



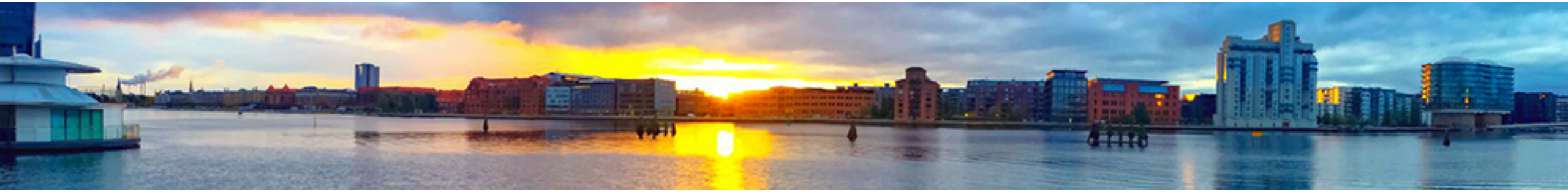
- Considering the uncertainty of each task
- Combining multiple regression and classification loss functions for multi-task learning
- Adjust each task's relative weight in the cost function by deriving a multi-task loss function

- Improve accuracy over separately trained models because cues from one task, such as depth, are used to regularize and improve the generalization of another domain, such as segmentation.

Kendall, A., Gal, Y., and Cipolla, R. (2017). Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.

2

[2] Jia Dong Zhang et al.
SEMAX: Multi-Task Learning for Improving Recommendations



SEMAX: Semantic mEaning and teMporal dynAmic eXtended

- Goal: Multi-task learning framework for reinforcing the
  - *recommendation task* that predicts the rating of users on new items through leveraging the
  - *related task* that predicts the rating of texts
- Recommendation task gets domain information from other task
  - semantic meanings from texts
  - temporal dynamics from text sequences for both users and items
  - predicting rating of a text written by a user
- Learn both semantic meaning and temporal dynamics to reinforce one another
  - Semantic meanings in the reviews should help capture temporal dynamics of users and items
  - Evolution of user interests and item characteristics should be beneficial in learning their semantic meaning in text
- Learn both semantic meaning and temporal dynamics of both users and items simultaneously

[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

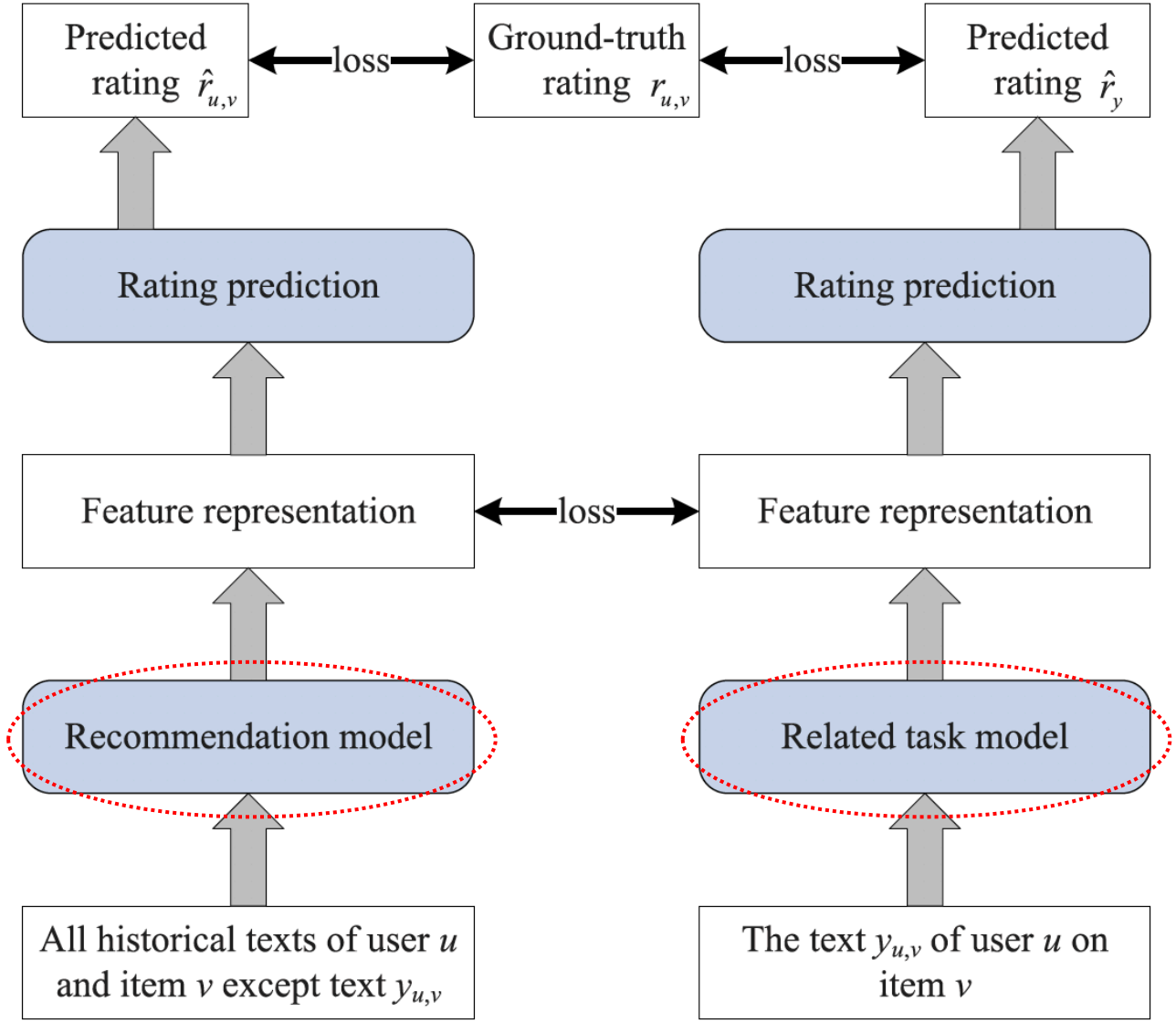
SEMAX: Semantic mEaning and teMporal dynAmic eXtended

- Interaction Records:
  - quadruple user, written text, with rating, for item
- User and Item Latent Factors:
  - represent some stationary components that encode fixed features of users and items,
    - e.g., the profile of users, the categories of items, or the long-term preference of a user on items

[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

SEMAX: Architecture (1/4)

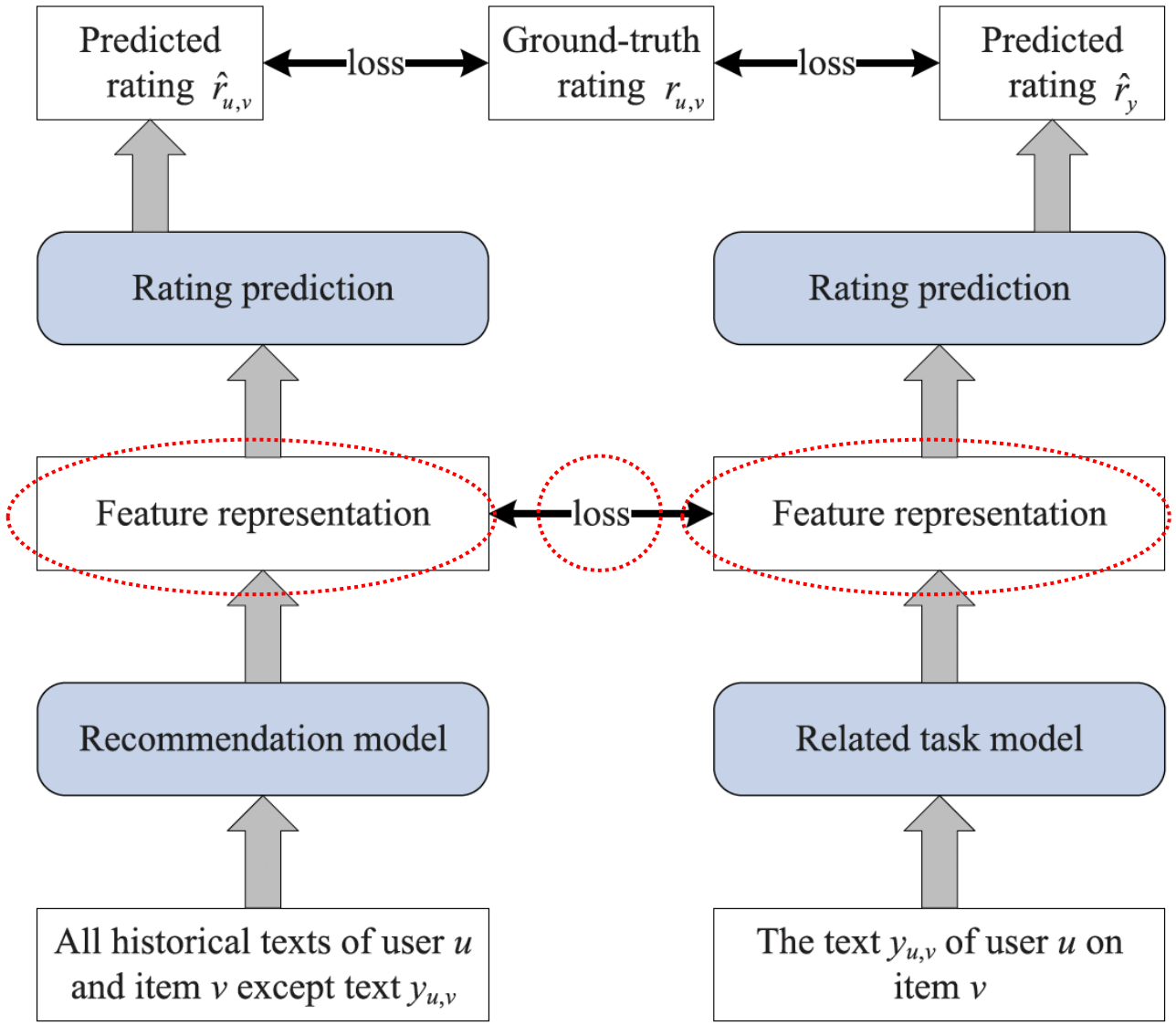
- MTL Framework
  - Recommendation task (left)
    - Predicting Ratings of Users on New Items
    - Extracts features from historical text based on RNNs
  - Related task (right)
    - Predicting Ratings of Texts
    - Generates feature representation based on attention model



[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

SEMAX: Architecture (2/4)

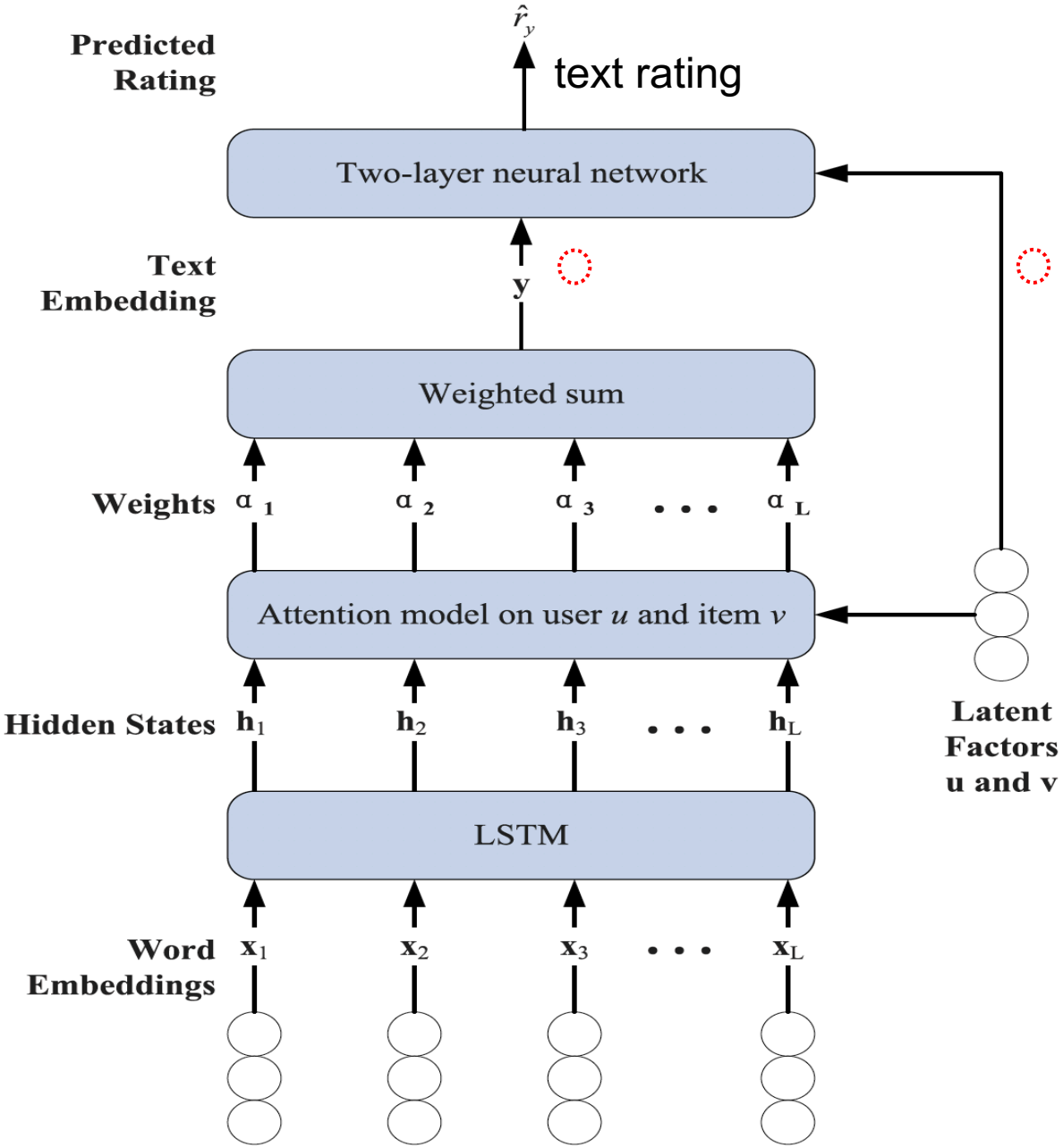
- MTL Framework
  - Shared word embedding
    - to enable them to reinforce each other
  - Regularization loss
    - used to lead the recommendation task to generate an approximation of feature representation of related task
  - Enhances generalization of feature representation and word embeddings in recommendation task



[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

SEMAX: Related Task Predict Rating (3/4)

- Text embedding and latent factors for user u and item v are used to predict text rating $\hat{r}_y$
- Text embedding is weighted sum of hidden states
- Word weights are normalized by softmax function
- Attention model learns importance of each word by considering attention of the user u and item v
- Attention mechanism is used
- Word embedding



[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

SEMAX: Recommendation Task (4/4)

Predicting Ratings of Users on New Items

Latent factors (**u** and **v**) and dynamic representation Yu and Yv are combined and fed to two layer neural network to predict the rating $\hat{r}_{u,v}$ of user u on new item v

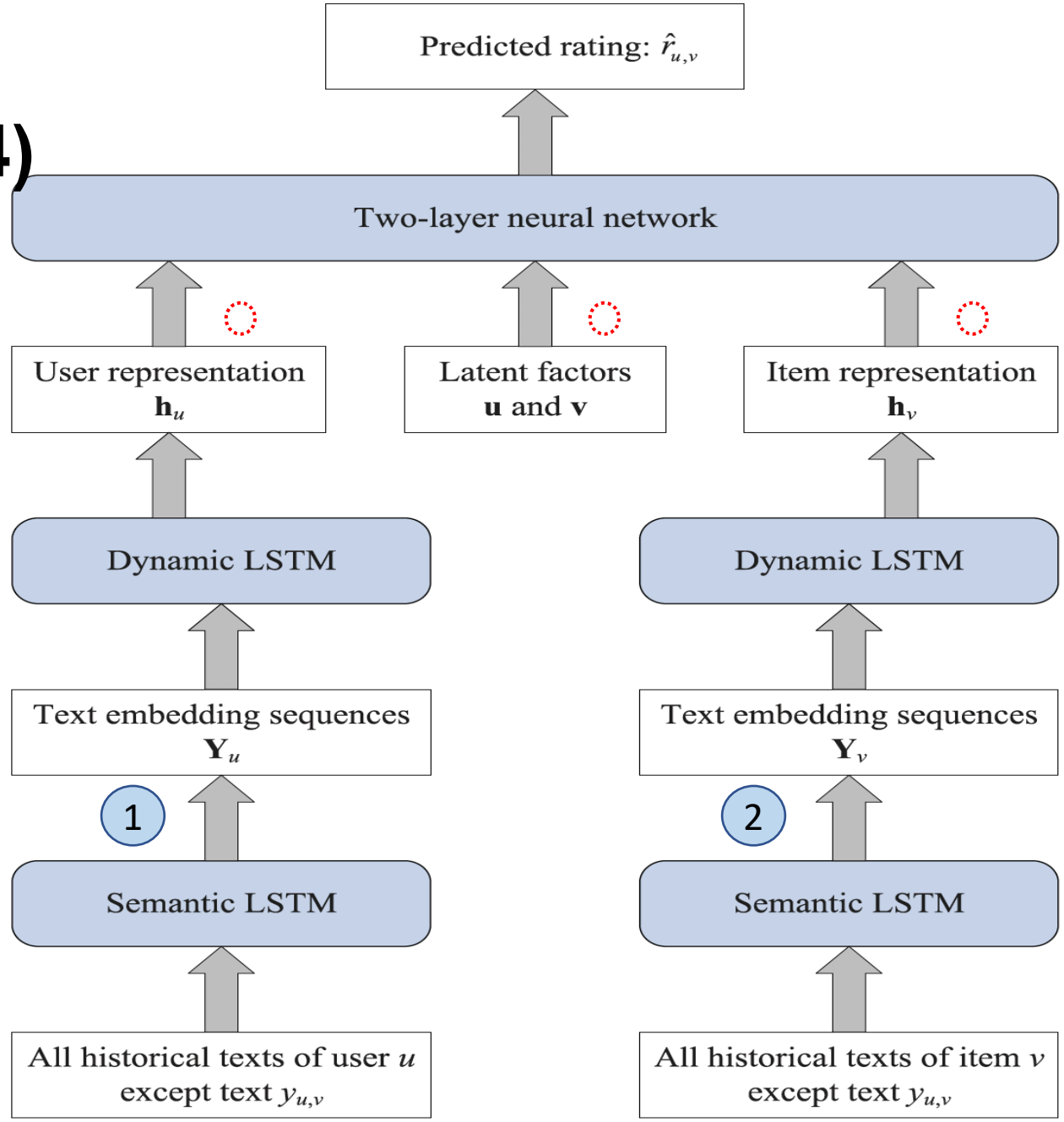
Temporal Dynamic Learning:

- Get dynamic user representation and item representation user hu and item hv
- LSTMs are used to learn dynamic interests of users

Semantic Meaning Learning:

- Take last hidden state as the text embedding
- Get text embedding sequences Yu and Yv of texts of user u and item v
- LSTMs are used to learn hidden state of each word

- Uses learning semantic meanings and temporal dynamics from all historical texts of the user and item



[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

SEMAX: Multi-Task Learning

- Minimise weighted sum of three loss terms:

$$\min_{\Omega} \frac{1}{|R|} \sum_{r_{u,v} \in R} [\beta \mathcal{L}_1 + (1 - \beta)(\mathcal{L}_2 + \lambda \mathcal{L}_3)]$$

  - R -> training rating set, β controls loss weight of recommendation and related task
- Loss of Rating prediction for new item:
  - square error of ground truth rating and rating prediction

$$\mathcal{L}_1 = (r_{u,v} - \hat{r}_{u,v})^2$$

- Loss for Predicted rating in related task:

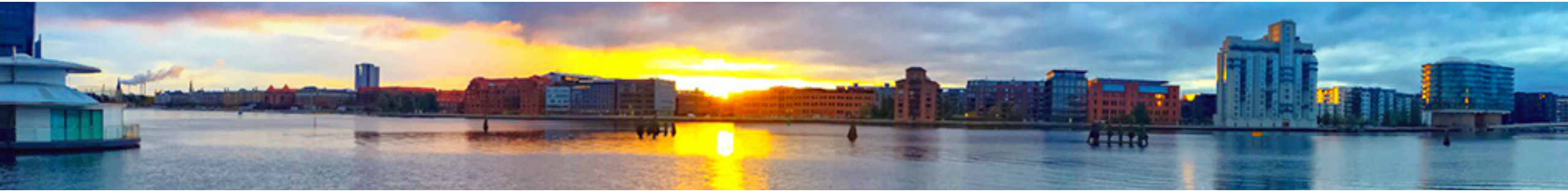
$$\mathcal{L}_2 = (r_{u,v} - \hat{r}_y)^2$$

- Regularization loss: $\mathcal{L}_3 = ||\mathbf{z}_y - \mathbf{z}_{u,v}||^2$

[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations

3

[3] Yichao Lu at el.
Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018



Why I like it: Multi-task Learning for Recommendation and Explanation

Terminology

- User Review document:
  - Set of all reviews written by a user
  - Indicates user preference
  - Example: Price
- Item Review document:
  - Set of all reviews written for an item
  - Describes the features of items
  - Example: Poor Quality
- Same network architecture for modelling user and item review document but with different set of parameters

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

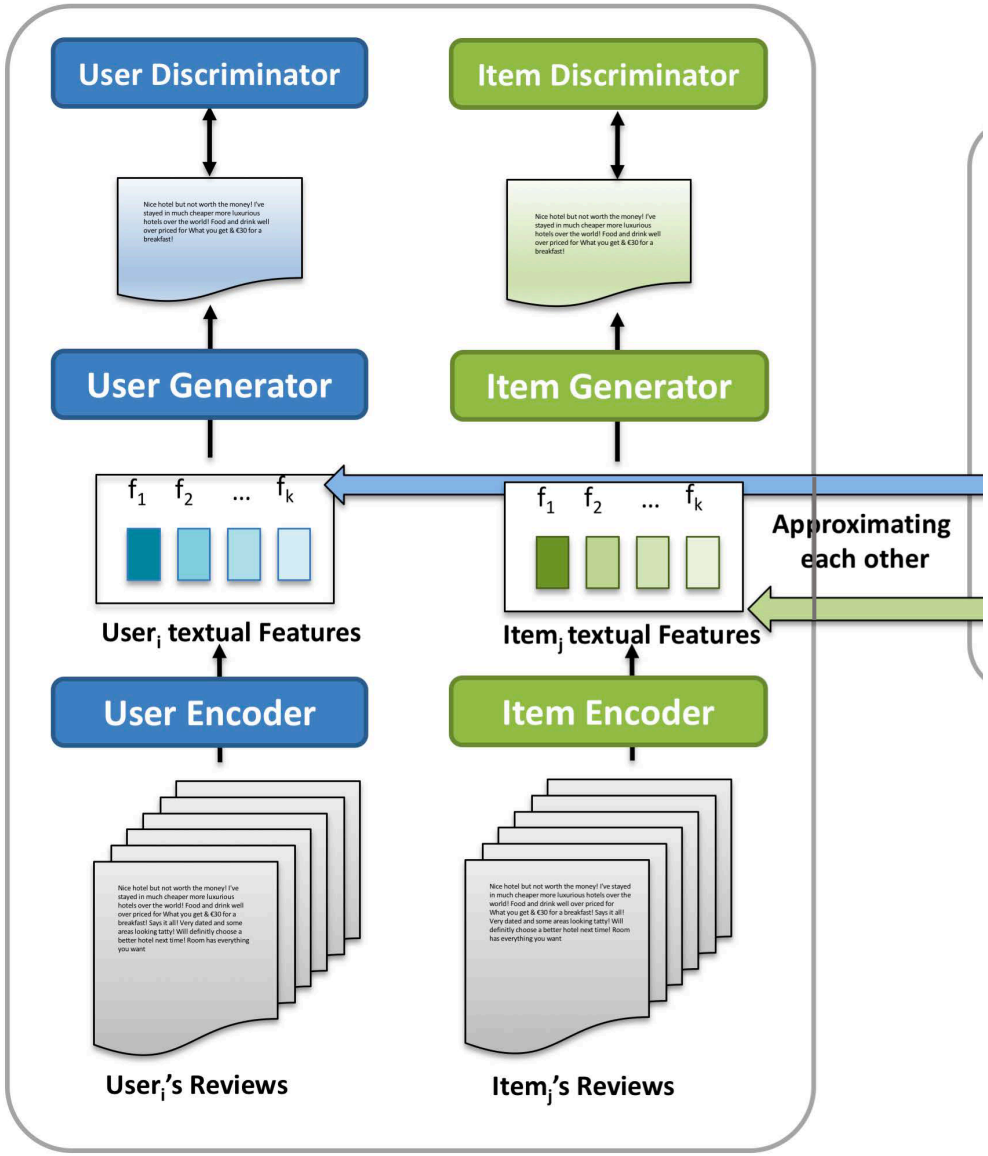
Why I like it: Multi-task Learning for Recommendation and Explanation

- Jointly learn to perform
  - Rating prediction and
  - Recommendation Explanation

  from user-generated reviews
- Rating prediction
  - Matrix factorization
- Recommendation Explanation
  - Sequence to sequence learning model
- Multi-task model simultaneously learns to perform
  - Rating prediction and review generation

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

1. Review Generation (1/4)

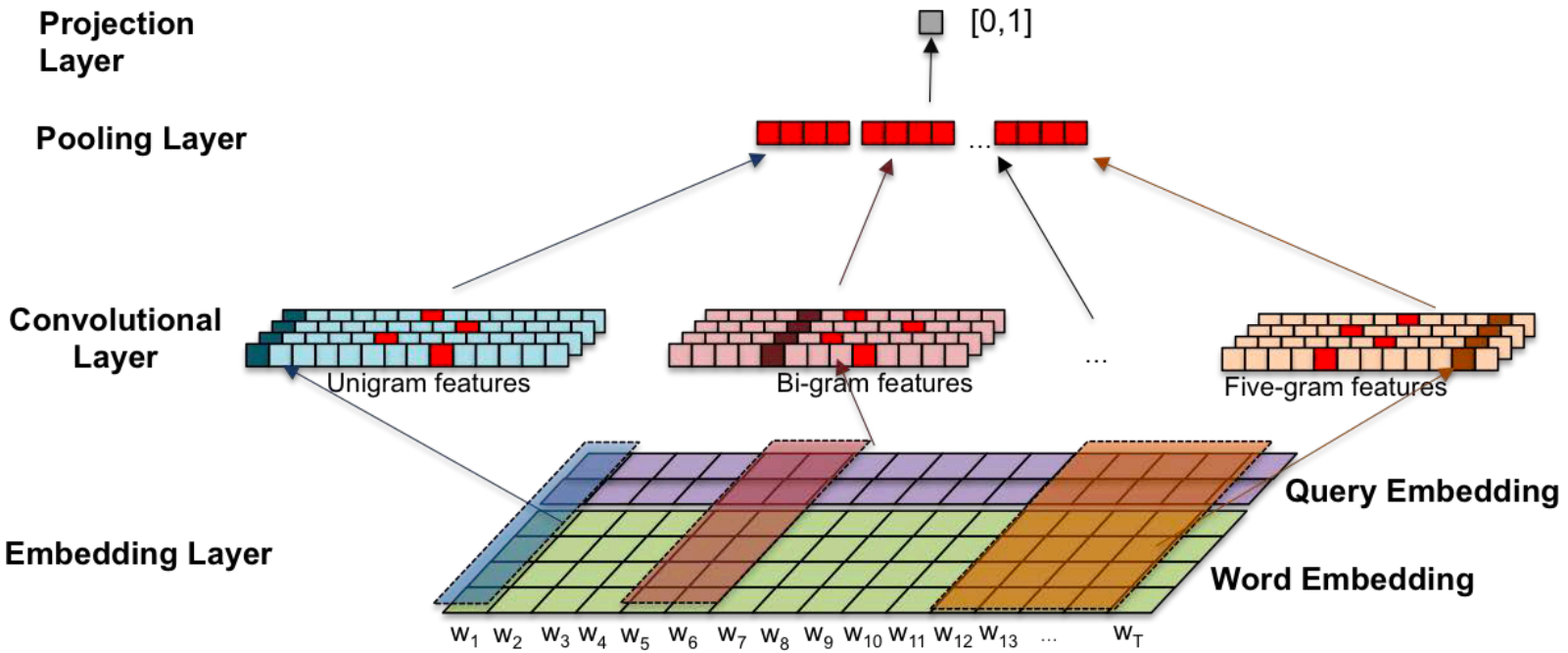


Learning from Reviews

- Use adversarial seq2seq learning model to generate reviews
- User review document Encoder:
  - Maps review document to textual features
- User review decoder/generator:
  - Based on textual features User review decoder generates a review
- User review Discriminator
  - Learns to distinguish the adversarial samples from authentic reviews. Criteria:
    - written by human being
    - review is generated by target user

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

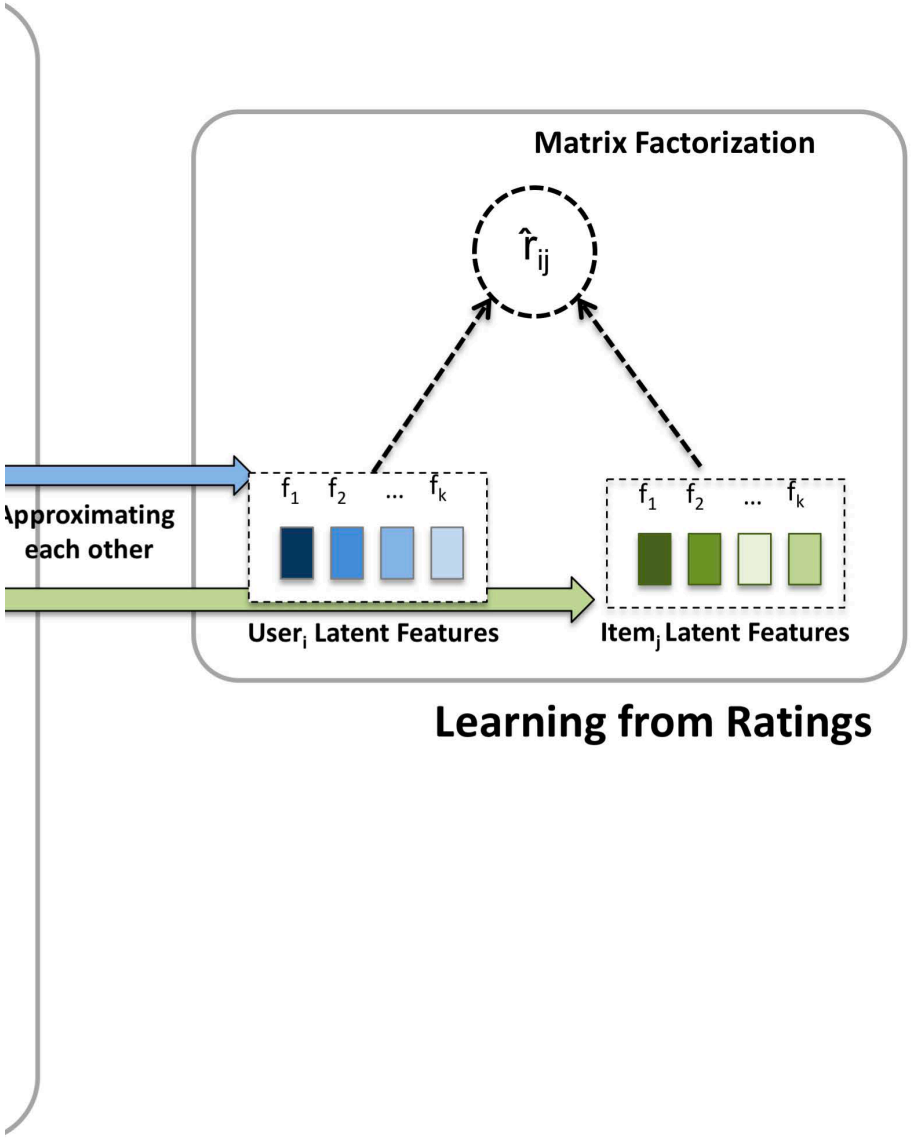
2 Convolutional Review Discriminator (2/4)



- Each word in review is mapped to corresponding word vector
- Concatenated with user-specific vector that identifies user information
- Concatenated word vector representations are processed by convolution layers, max-pooling layers and fully connected projection layer
- Final output is sigmoid non-linearity

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

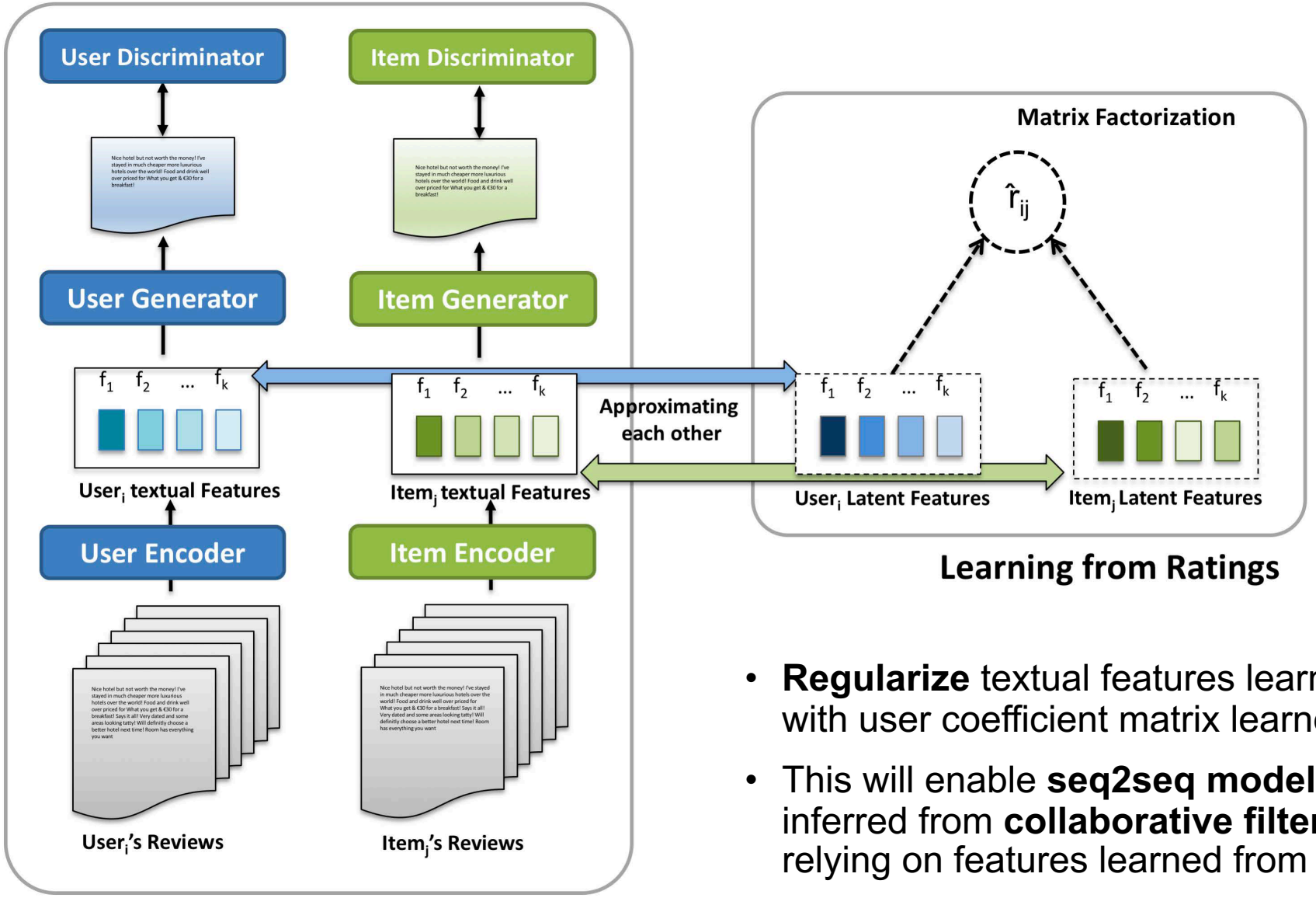
3 Context aware MF for rating prediction (3/4)



- User coefficient matrix learned by matrix factorization
- Extend **Probabilistic MF model** by using textual features extracted from review documents to serve as **regularizes** for the user and item latent vectors
- Introduce the textual features into the **prior** distributions of the latent variables to address sparsity problem in

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

4 System architecture of multi-task learning model (4/4)



user coefficient matrix learned by matrix factorization

- **Regularize** textual features learned by the **document encoder** with user coefficient matrix learned by matrix factorization
- This will enable **seq2seq model** to **exploit** user preferences inferred from **collaborative filtering approach** instead of merely relying on features learned from review documents

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

Optimization Algorithm: Multi-Task Learning

for *epoch* $\leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    Update U_i with least square approximation:

$$U_i \leftarrow (VI_iV^T + \lambda_U I_K)^{-1}(VR_i + \lambda_U \widetilde{U}_i)$$

  end
  for $j \leftarrow 1$ **to** M **do**
    Update V_j with least square approximation:

$$V_j \leftarrow (UI_jU^T + \lambda_V I_K)^{-1}(UR_j + \lambda_V \widetilde{V}_j)$$

  end
  for *iteration* $\leftarrow 1$ **to** $T - step$ **do**
    Update θ_U and θ_V via teacher forcing.
  end
  for *iteration* $\leftarrow 1$ **to** $G - step$ **do**
    Update θ_U and θ_V via policy gradient.
  end
  for *iteration* $\leftarrow 1$ **to** $D - step$ **do**
    Update ϕ_U and ϕ_V via gradient ascent.
  end
end

Inspired by Expectation-Maximization (EM) algorithm alternate between the parameter update of

- Sequence to Sequence review generation model and
- Context aware recommendation model

ALS (Alternate Least Square (ALS) **repeatedly optimizes one of U and V** while temporarily fixing the other to be constant

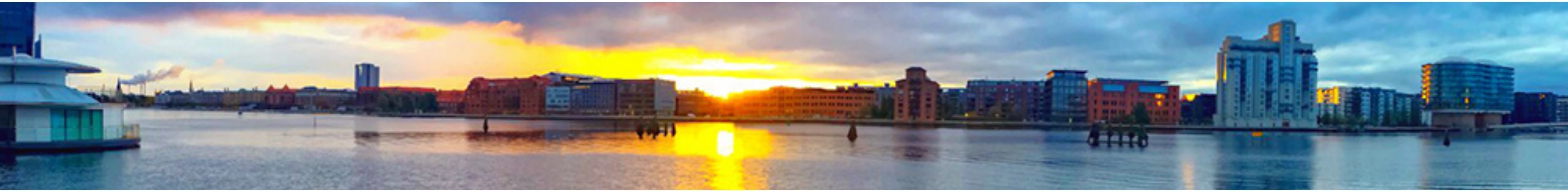
Teacher forcing is used as a supplement to Policy gradient

We feed ground-truth reviews to the generator for model update

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

4

[4] Chen Gao et al.
Neural Multi-Task Recommendation from Multi-Behavior Data.
ICDE-2017



Neural Multi-Task Recommendation from Multi-Behavior Data

- In e-commerce in addition to Purchase data we have more data including view, click etc.
- Can use cascading relationship among different types of behaviors
  - e.g., a user must click on a product before purchasing it
- Each behavior is treated as a task
- Multi-task learning:
  - Learn from multiple behavior data

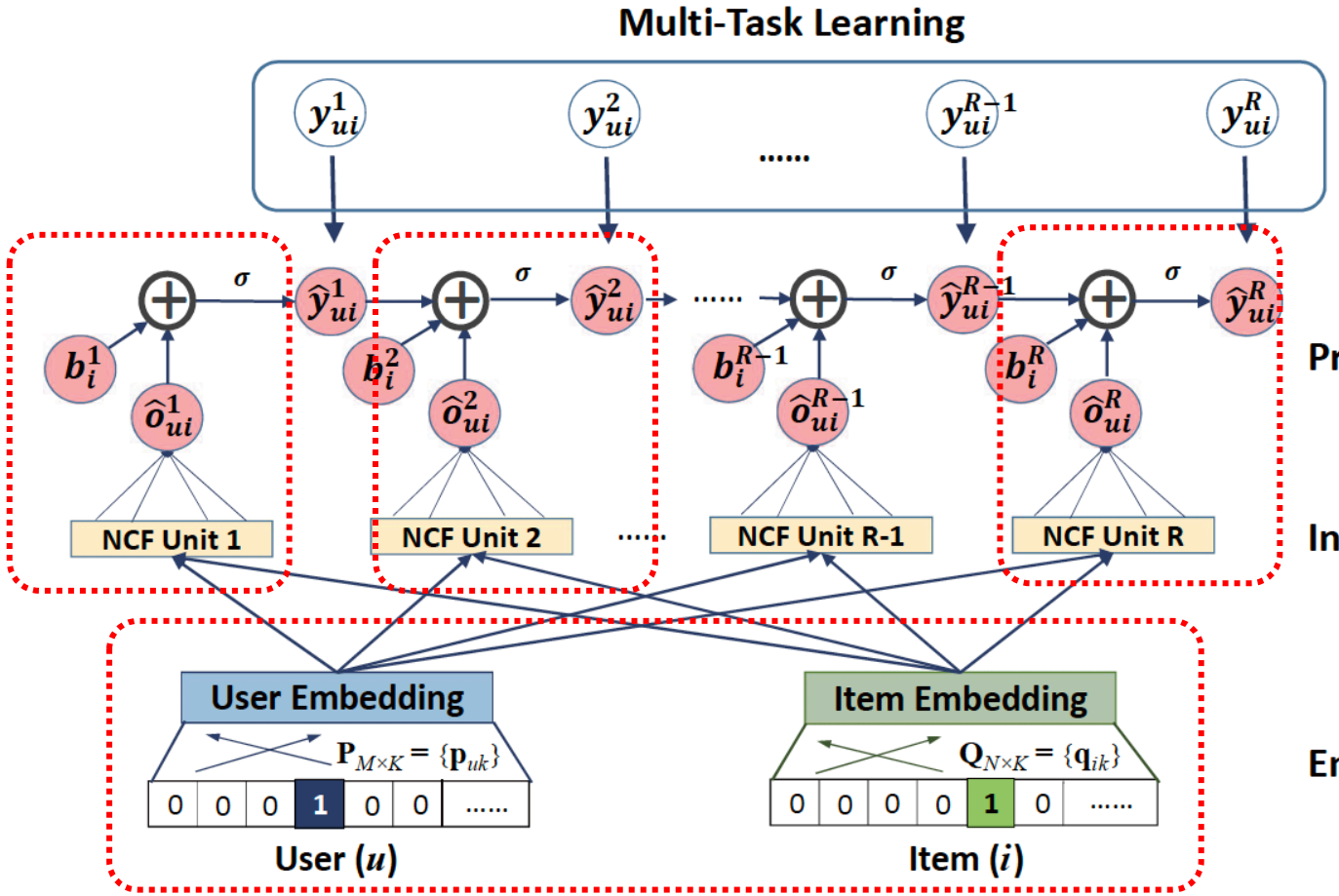
[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

Neural Multi-Task Recommendation (NMTR)

- Many user behaviour types in real-world applications follow an ordinal (or sequential) relationship
  - Task: target behavior is the purchase behavior in E-commerce
  - Related Task: other behaviors can include the click, adding to cart, etc.
- Estimates the likelihood that a user will interact with an item under the target behavior.

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

Neural Multi-Task Recommendation Model (1/5)



Prediction

Interaction

Embedding

Hard Parameter Sharing

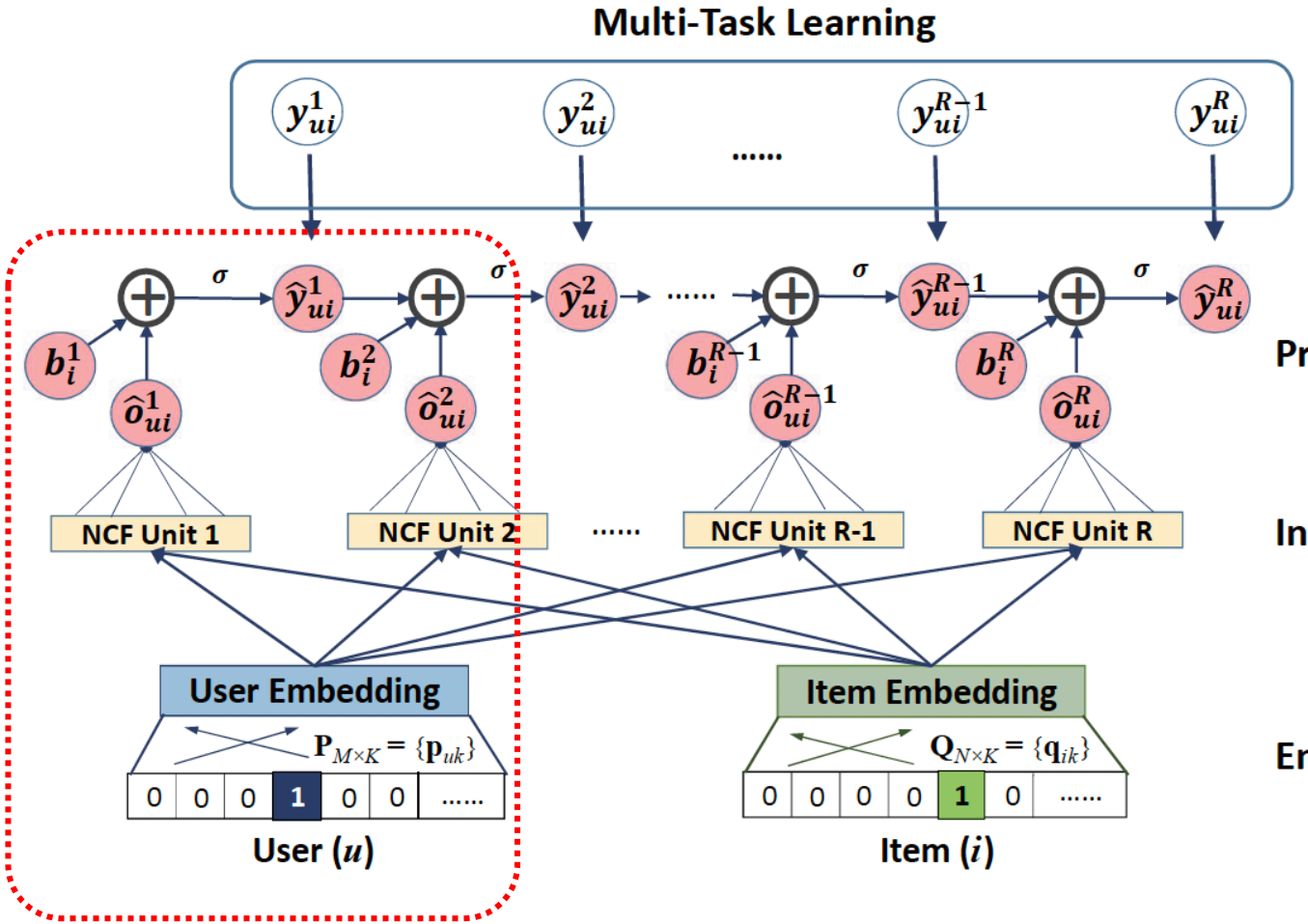
To capture the ordinal relations among behavior types, correlate the model prediction of each behaviour type in a cascaded way.

A data-dependent interaction function is learned for each behavior type.

A user (and an item) has a shared embedding across multiple types of behaviors.

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

Neural Multi-Task Recommendation Model (2/5)



Single Task:

Prediction

Sigmoid function converts the output to a probability

Interaction

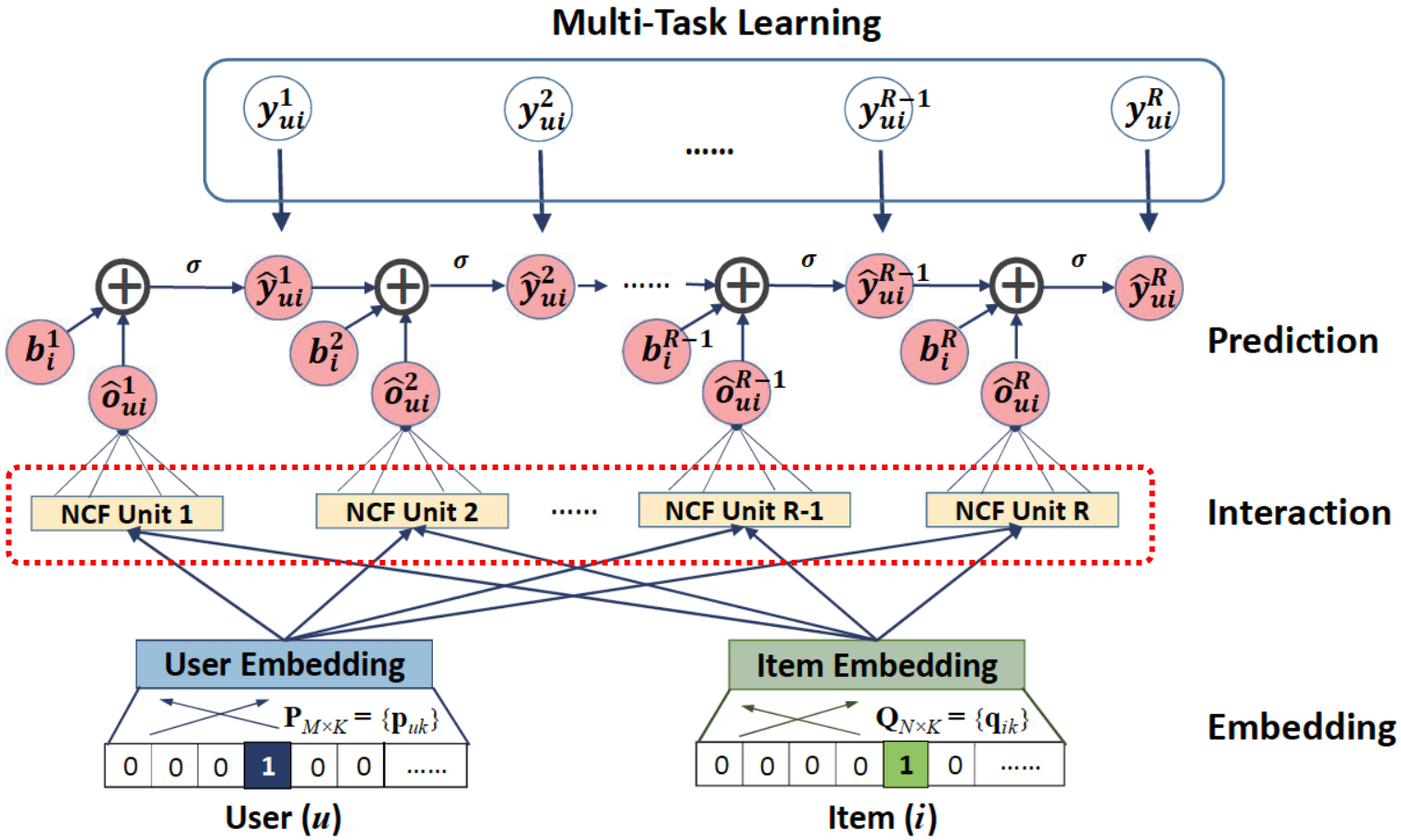
Predict the likelihood of multiple behavior types (R types) with the same input, it is essential to learn a separated interaction function for each type.

Embedding

Apply one-hot encoding to encode the input of user ID and item ID
It can be easily extended to incorporate other features of a user and an item (e.g., user demographics and item attributes)

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

Neural Multi-Task Recommendation Model (3/5)



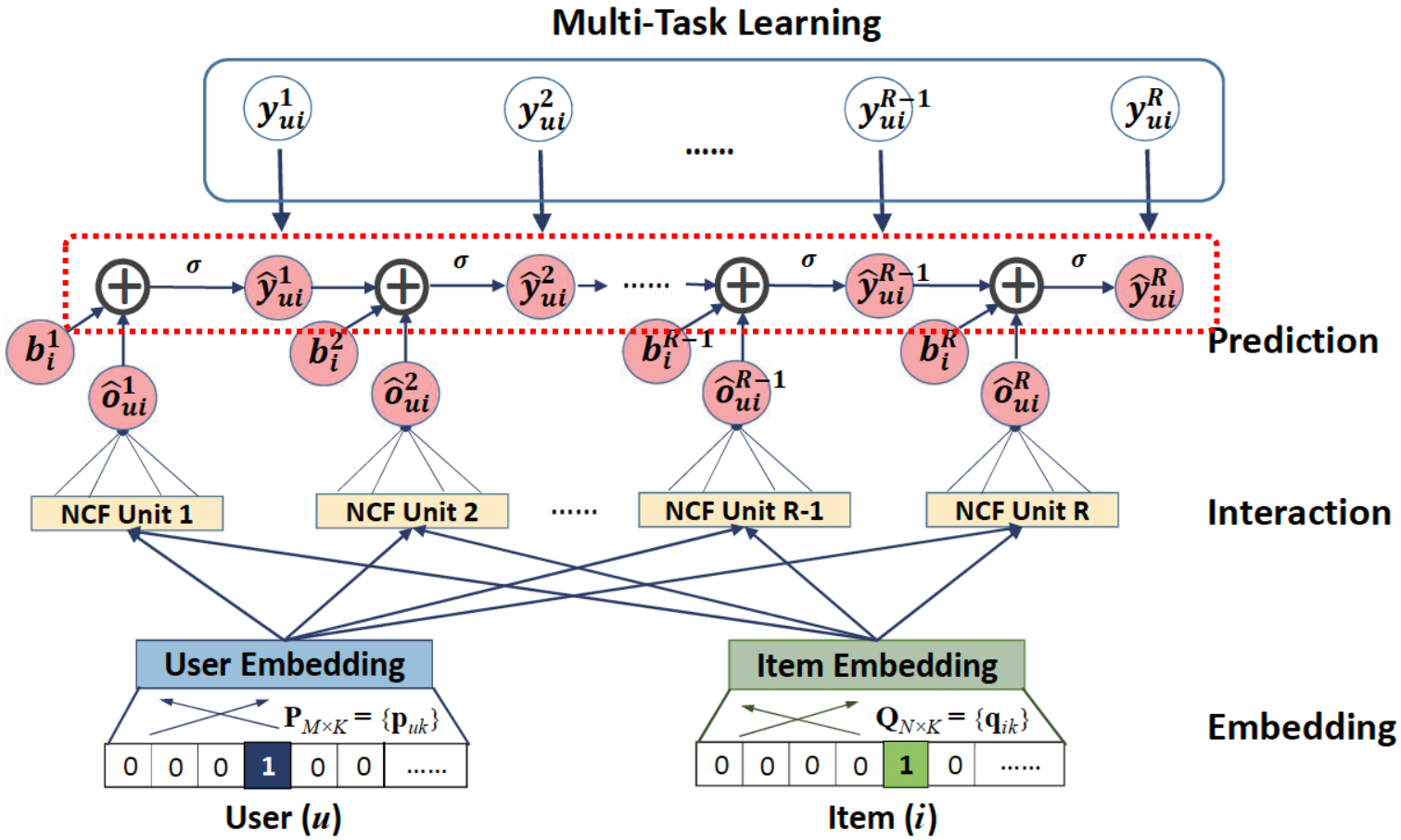
Complex Interaction:

Provide ability and sufficient flexibility to learn the possible complicated patterns.
Three Neural Network Units from NCF: Neural Collaborative Filtering

- GMF (Generalized matrix factorization)
- MLP (multi-layer perceptron)
- NeuMF (neural matrix factorization)

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

Neural Multi-Task Recommendation Model (4/5)



Cascade - intuition:

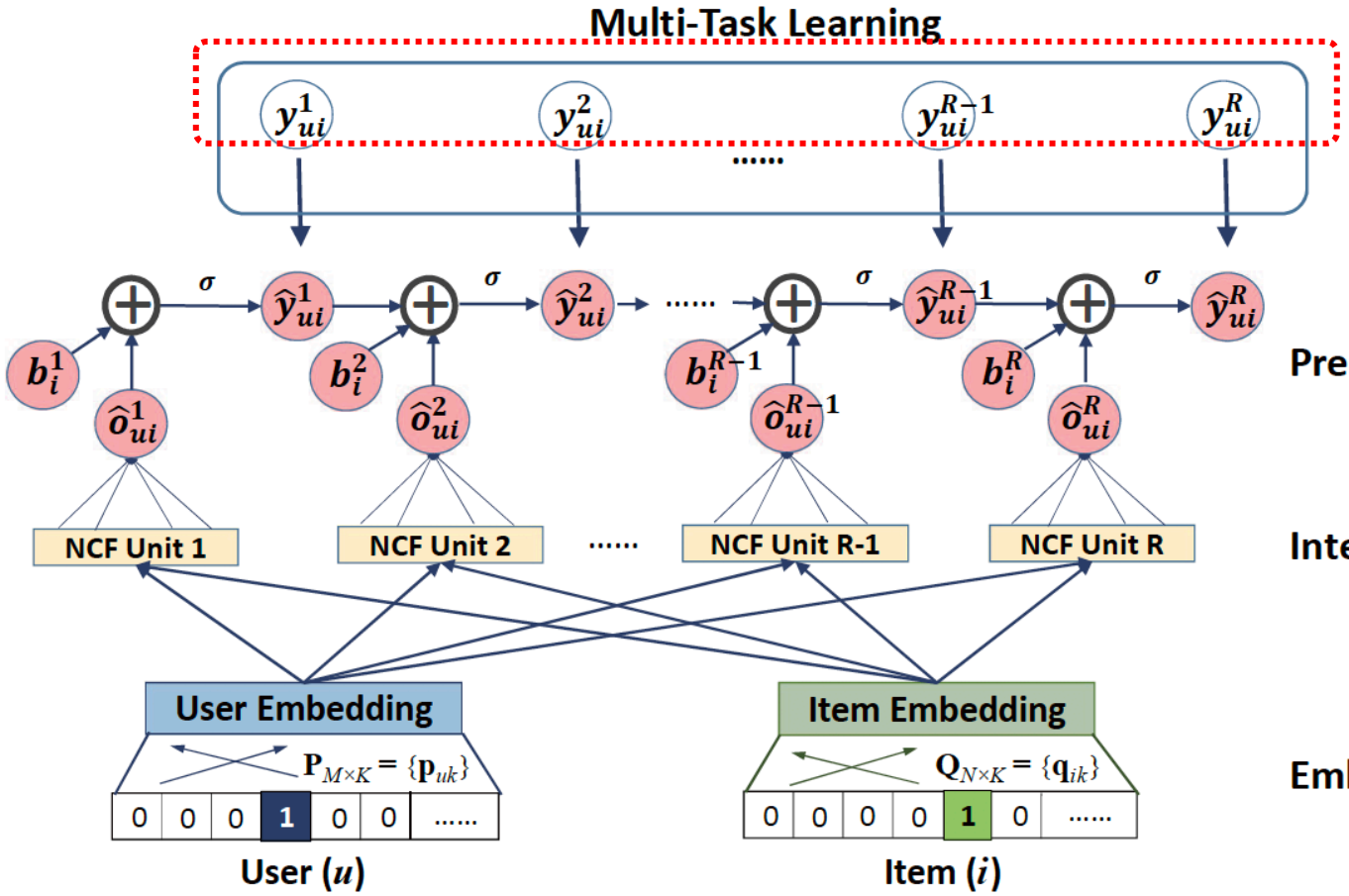
To encode the sequential effect

- prediction on a behavior type depends on the predictions of the precedent behavior types.

y_2 depends on y_1, y_R depends on y_{R-1}

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

Neural Multi-Task Recommendation Model (5/5)



Multi-task learning (MTL):
Performs joint training on the modes of different but correlated tasks, so as to obtain a better model for each task

Train all models simultaneously, where the model learning for each behavior type is treated as a task.

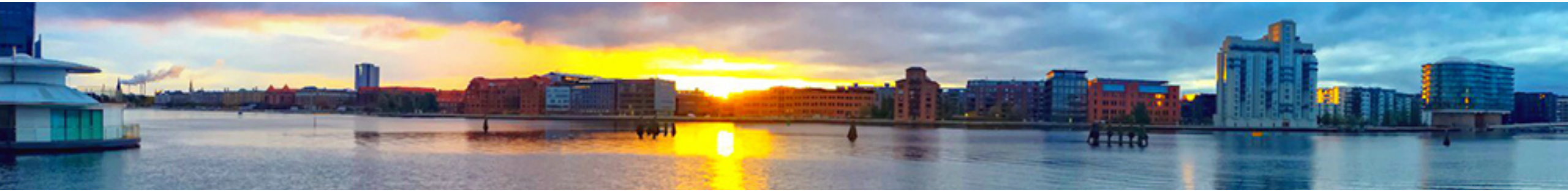
The loss function to be minimized is:

$$L = -\sum_{r=1}^{R} \lambda_r \Big(\sum_{(u,i)\in\mathcal{Y}_r^+} \log \hat{y}_{ui}^r + \sum_{(u,i)\in\mathcal{Y}_r^-} \log(1-\hat{y}_{ui}^r) \Big)$$

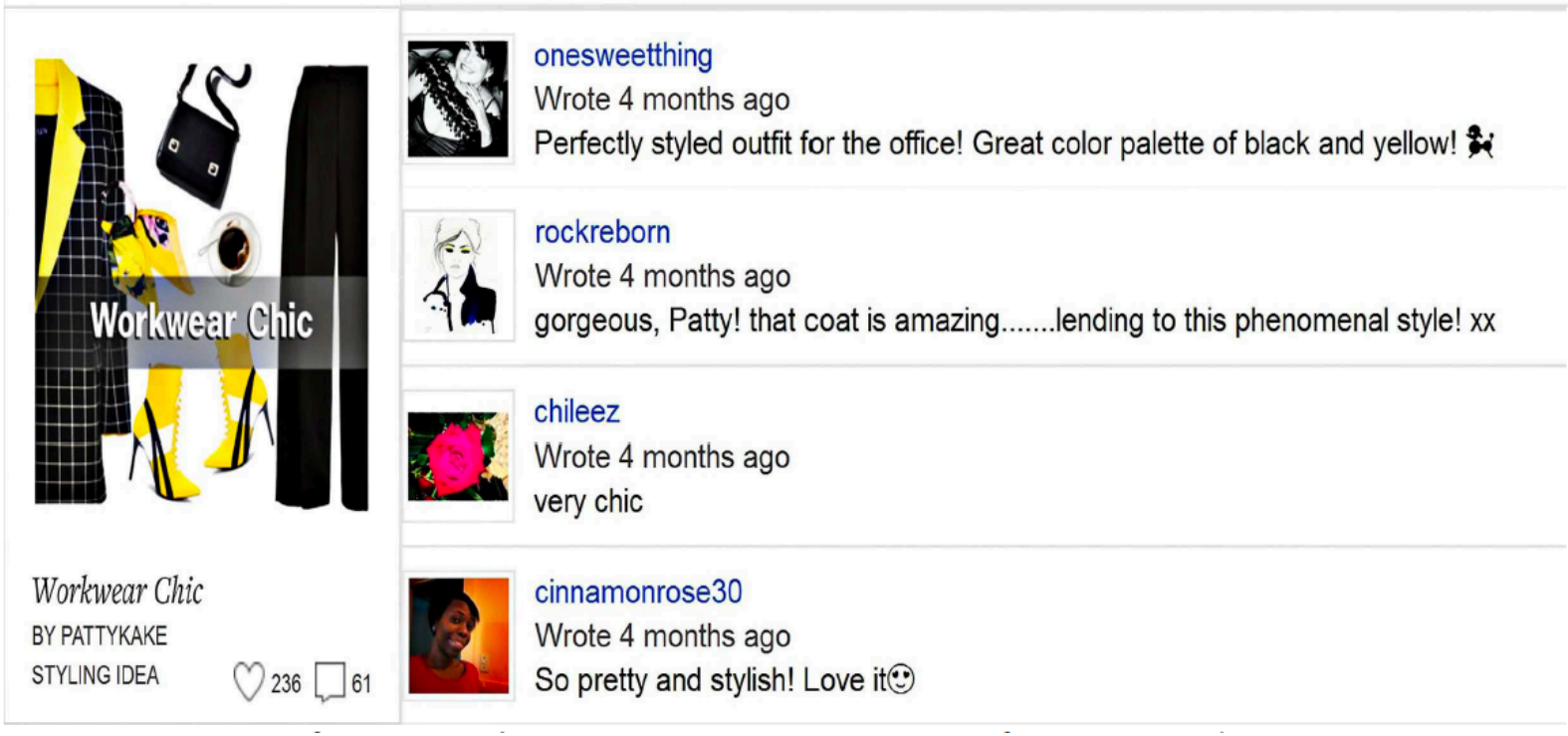
[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

5

[5] Yujie Lin et al,.
Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018



Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation



- Users share their outfit compositions with a broad public (left)
- Others express their comments about the outfit compositions (right).
- Fashion-oriented online communities like Polyvore, Chictopia, etc. exhibits a large collection of outfit composition data, enriched with valuable user comments.

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

Neural Outfit Recommendation (NOR)

Goal:

- Bottom recommendation task:
  - Given a top recommend a ranked list of bottoms
- Top recommendation task:
  - Given a bottom recommend a ranked list of tops
- Comment generation task:
  - Generate a natural-sounding comment for each recommended outfit
  - Generated comments can be regarded as explanations for each recommended outfit

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation

- Previous recommenders use only visual features to enhance recommendations
- Explainable outfit recommendation challenges:
  - Model the compatibility of fashion factors, e.g., color, material, pattern, shape, etc.
  - Model transformations between visual and textual information, which involves mappings from the visual to the textual space
- Neural Outfit Recommendation (NOR)
  - Outfit matching and comment generation
  - Simultaneously provides outfit recommendations and generates abstractive comments

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

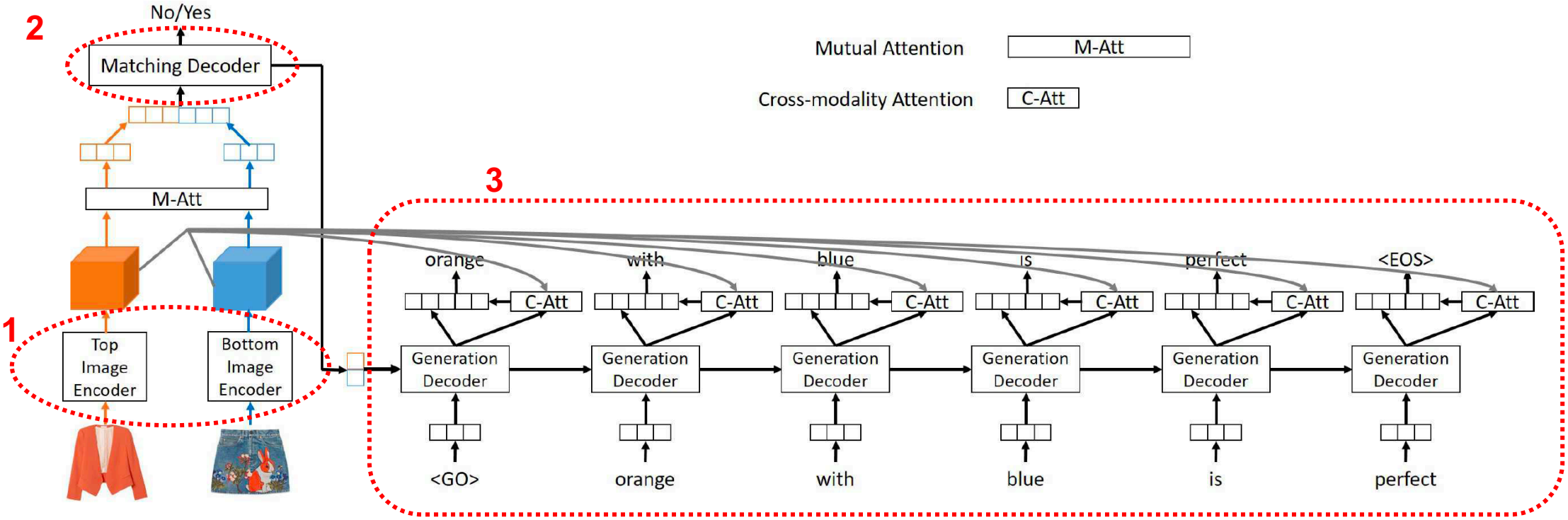
Neural Outfit Recommendation (NOR)

- Extract Visual Features
  - Convolutional Neural Network (CNN) with a mutual attention mechanism
  - The visual features are then decoded into a rating score for the matching prediction.
- Abstractive comment generation
  - A gated recurrent neural network with a cross-modality attention mechanism transforms visual features into a concise sentence.
- Multi-Task Learning:
  - Two parts are jointly trained based on a multi-task learning framework in an end-to-end back-propagation paradigm.

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

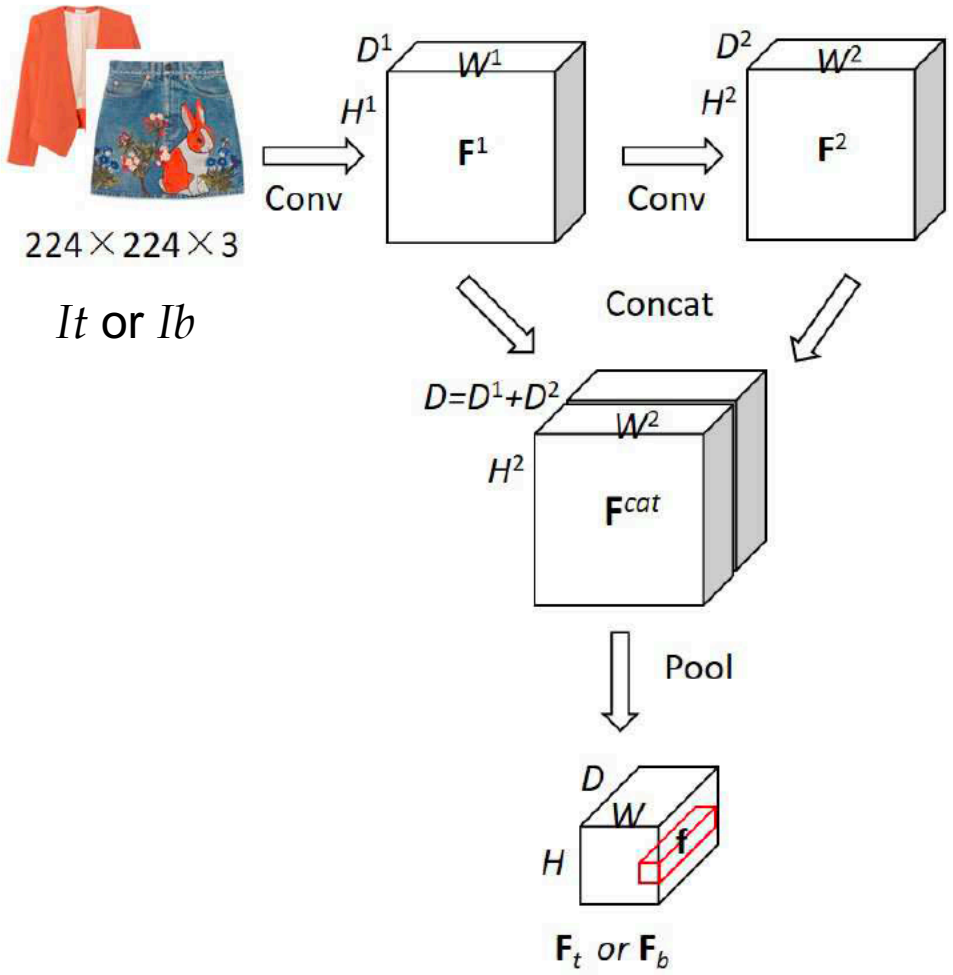
Neural Outfit Recommendation (NOR) – Architecture (1/5)

1 Top and Bottom Image Encoder
2 Matching Decoder
3 Generation Decoder



[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

1 Top and Bottom Image Encoder (2/5)



- Two-layer CNNs with mutual attention as the top and bottom image encoder
  - Extract visual features from It or Ib
- F^1 Primary Visual Features
- F^2 Advanced Visual Features

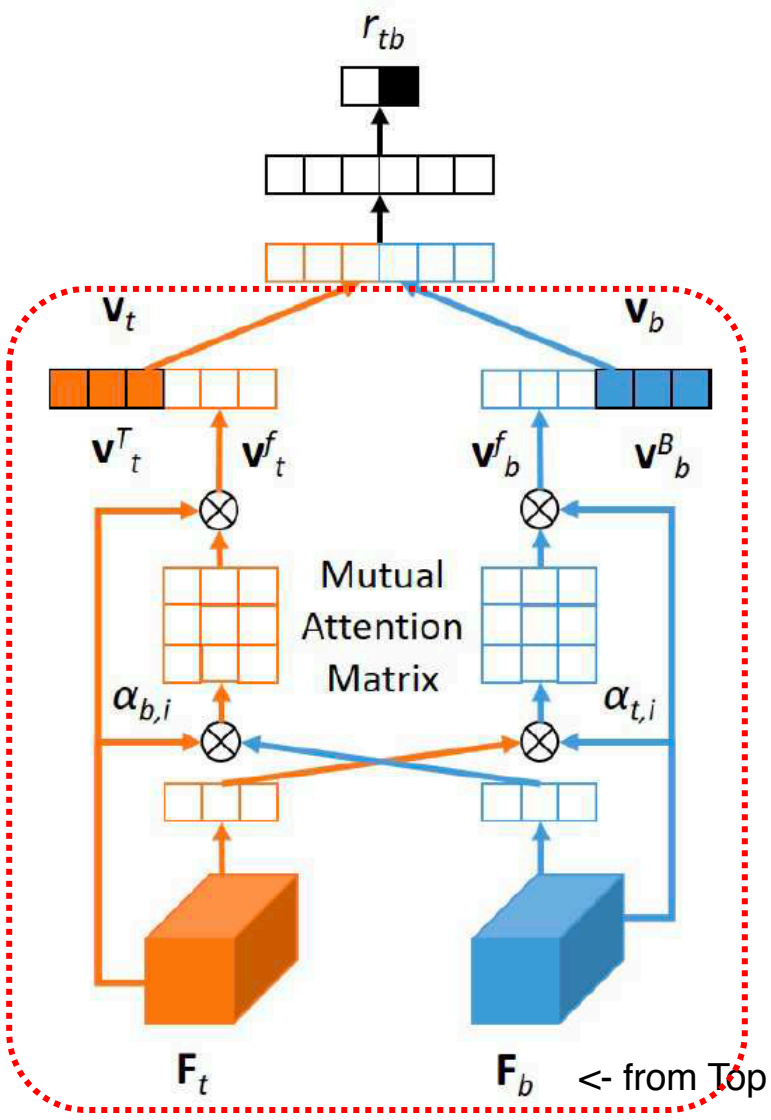
- F^{cat} : Concatenate F^1 and F^2

- F : Max-pooling to get final visual features

- Flatten to get F_t and F_b

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

2 Mutual attention and matching decoder (3/5)

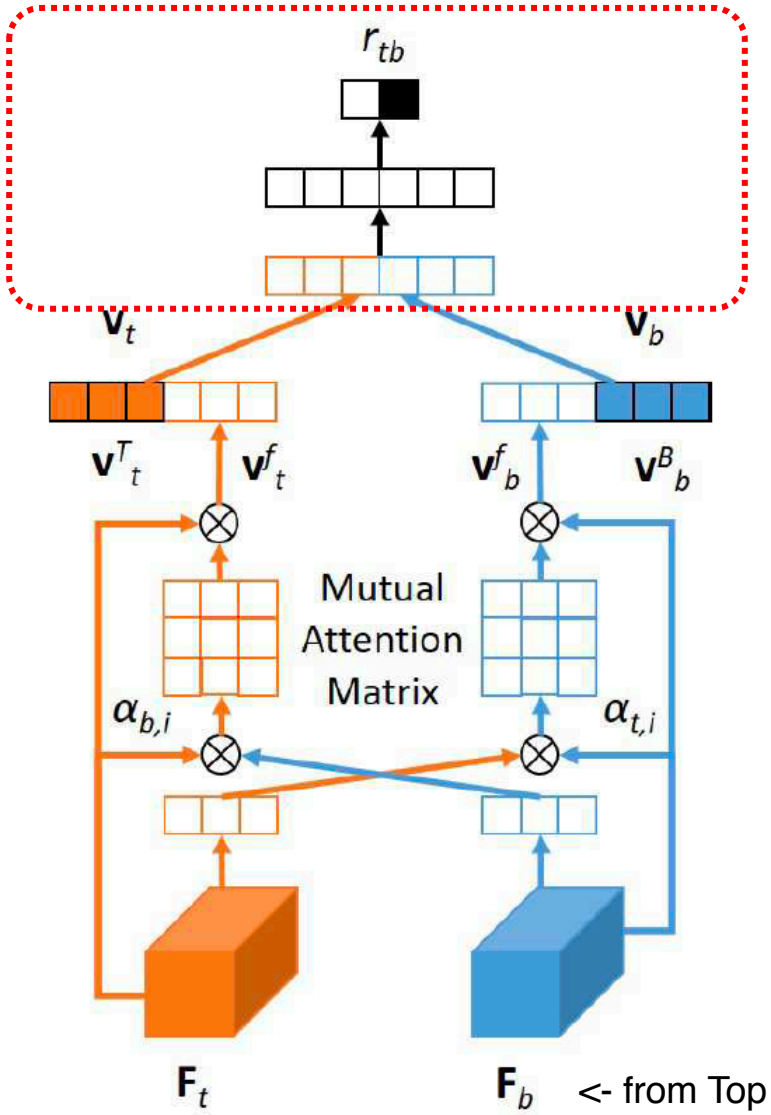


<- from Top and Bottom Image Encoder

- Mutual attention mechanism:
  - to evaluate the correlation and alignment between each local region of *It* or *Ib*
- Model the matching relation from two sides
  - bottom images to top images and
  - top images to bottom images
- Global Average pooling F_t
- Find Attention weights
- Find Visual Feature Vectors
  - For Top $\mathbf{v}_t^f$ and for Bottom $\mathbf{v}_b^f$
- Using Matrix factorization find latent factors for
  - top $\mathbf{v}_t^T$ and $\mathbf{v}_b^B$ bottom
- Concatenate to get Latent Vector Representation $\mathbf{v}_t$ and $\mathbf{v}_b$

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

2 Mutual attention and matching decoder (4/5)

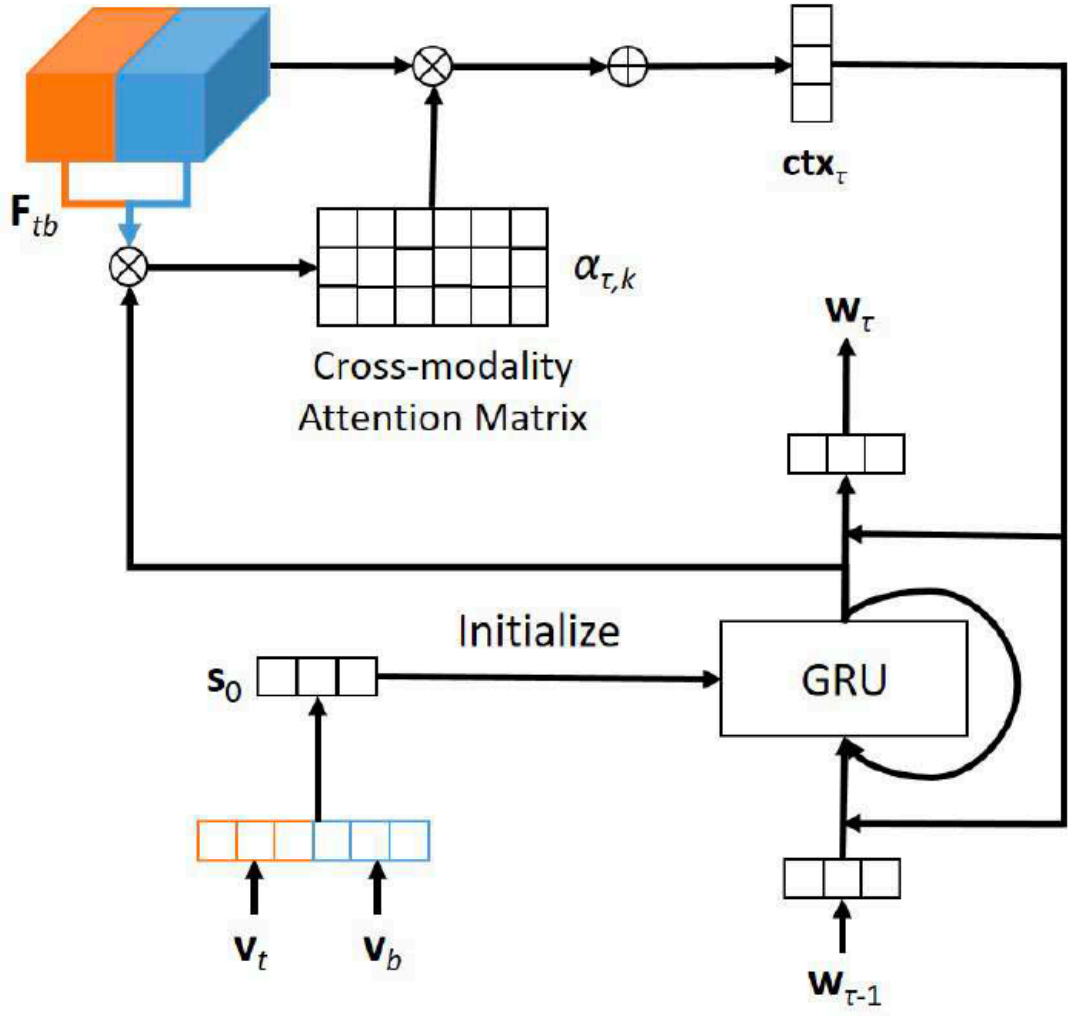


- Multi-layer neural network to calculate the matching probability of *t* and *b*
- Map in shared space
- Find matching probability using softmax
- Recommend Tops or Bottoms according to probability $p(r_{tb})$

<- from Top and Bottom Image Encoder

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

3 Cross-modality attention and generation decoder (5/5)



- Use GRU with cross-modality attention as the generation decoder
- Compute the initial hidden state
- For each timestamp GRU computes new hidden state based on previous
  - hidden state, context vector and embedding
- Context Vector
  - Weighted sum of visual features of F_t and F_b
  - computed through cross-modality attention
- Matches the current state s_t with each element of F_t and F_b to get an importance score
- makes use of the extracted visual features to generate comments by paying attention to particularly effective visual features.

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

Multi-Task Learning Framework

The whole framework is trained using back-propagation in an end-to-end paradigm

- **Multi-task Objective Function:**
  - Linear combination
  - adjust regularization weight

$$L = L_{mat} + L_{gen} + \lambda_{reg} L_{reg}$$

- **Matching Task** Loss Function:
  - Negative Log Likelihood
  - Positive combinations: prob=1
  - Negative combinations: prob=0

$$L_{mat} = \sum_{\{r_{tb} | (t,b) \in \mathcal{P}^+ \cup \mathcal{P}^-\}} - \log p(r_{tb})$$

- **Generation Task** Loss Function:
  - Negative Log Likelihood
  - Use generation loss for positive combinations, ignore it for negative combinations.
- **L2 loss as Regularization**

$$L_{gen} = \sum_{\{c_k^{tb} | c_k^{tb} \in \mathcal{C}^{tb} \wedge (t,b) \in \mathcal{P}^+\}} - \log p(c_k^{tb})$$

[8] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

REFERENCES

REFERENCES

Session 1:

[1] Sebastian Ruder. 2017 An Overview of Multi-Task Learning in Deep Neural Networks.

[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations IEEE-2018

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

[5] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

Session 2:

[6] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

[7] Vasvani et. al. Transformer – Attention is All You Need

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

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

Code- Notebook

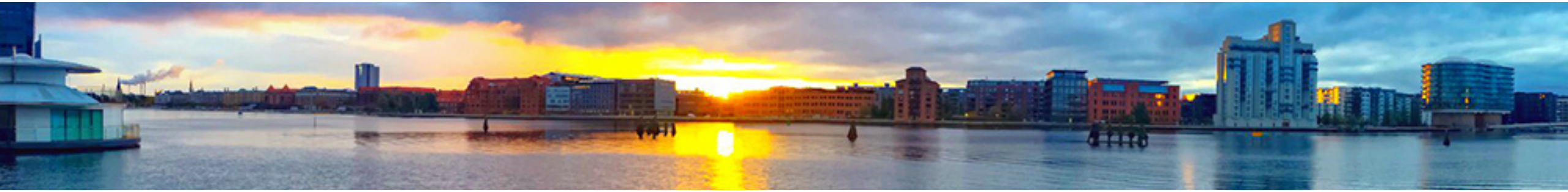
Ref: Documents/RecSys2019 MT-KG/Final/

Explainable Outfit Recommendation - Tutorial.ipynb

Session-2

(45min)

[8] Fei Sun, et. al.
BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019



BERT4Rec

- BERT – State Of The Art mechanism used for various problems
- Challenge
  - Model users' dynamic and evolving preferences from historical behaviors
- Limitations of previous approaches:
  - Assume rigid ordered sequence
  - Have only left to right (unidirectional) architectures
- Conditioning on both left and right context in bidirectional model
  - Predict masked items jointly conditioning on both left and right context
  - This way can generate more training data

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

BERT4Rec

Foundation for BERT4Rec:

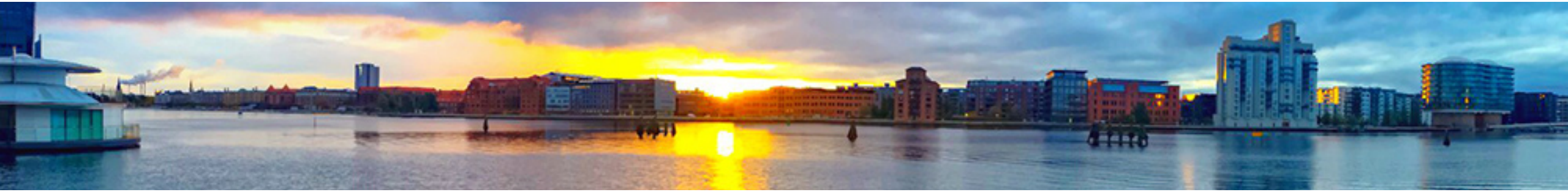
- Transformer – Attention is All You Need
- BERT – Bidirectional Encoder Representation from Transformer

Can come up with other ideas for Recommendation System

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

6

[6] Vasvani et. al Attention Is All You Need (Transformer)

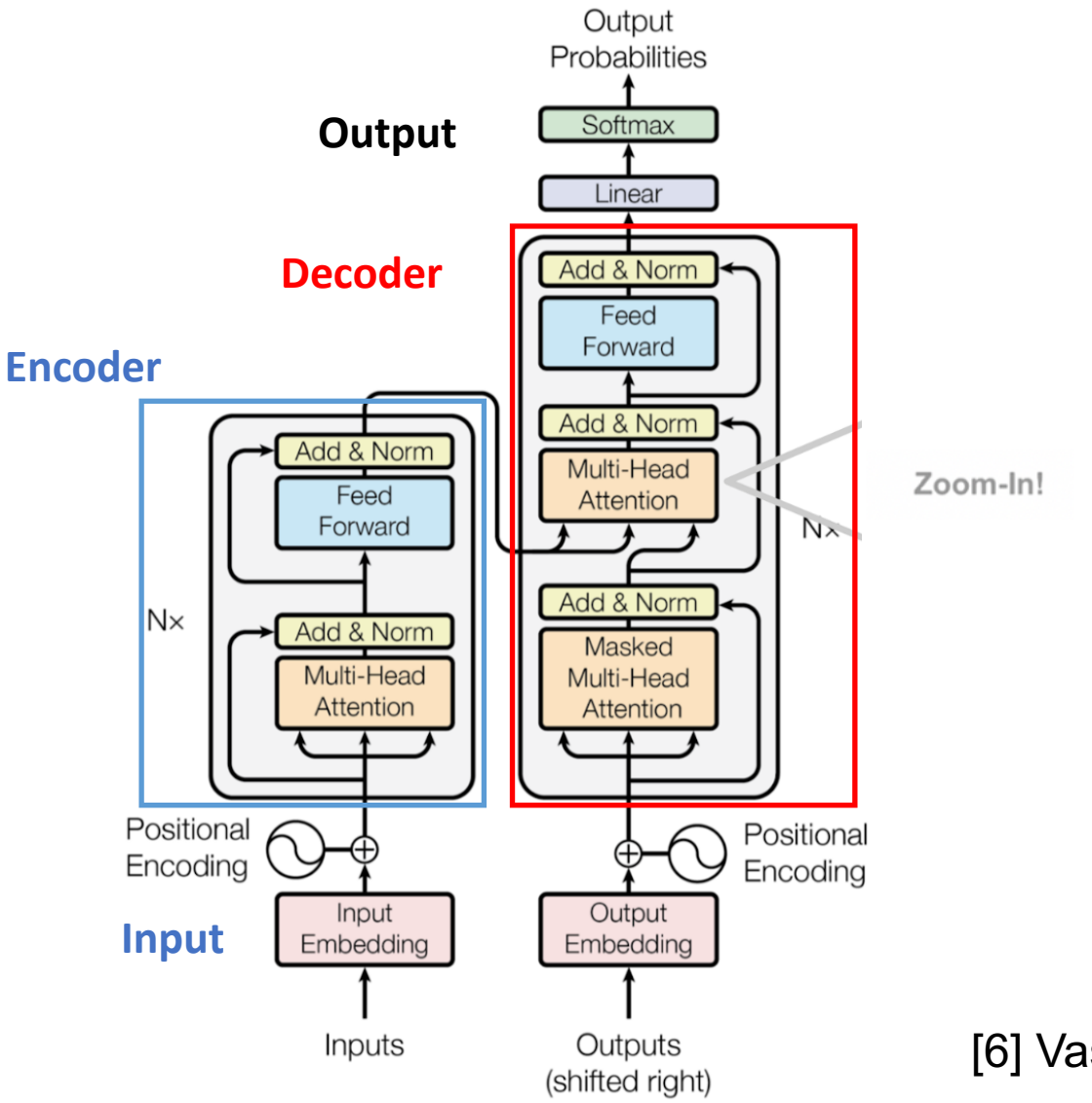


Why Transformer Network?

- **Addresses drawbacks of RNN (seq2seq) based architecture**
  - Hard to parallelize
  - Difficulty in learning long range dependencies
  - Complex
- **It uses only attention – No RNN or CNN**

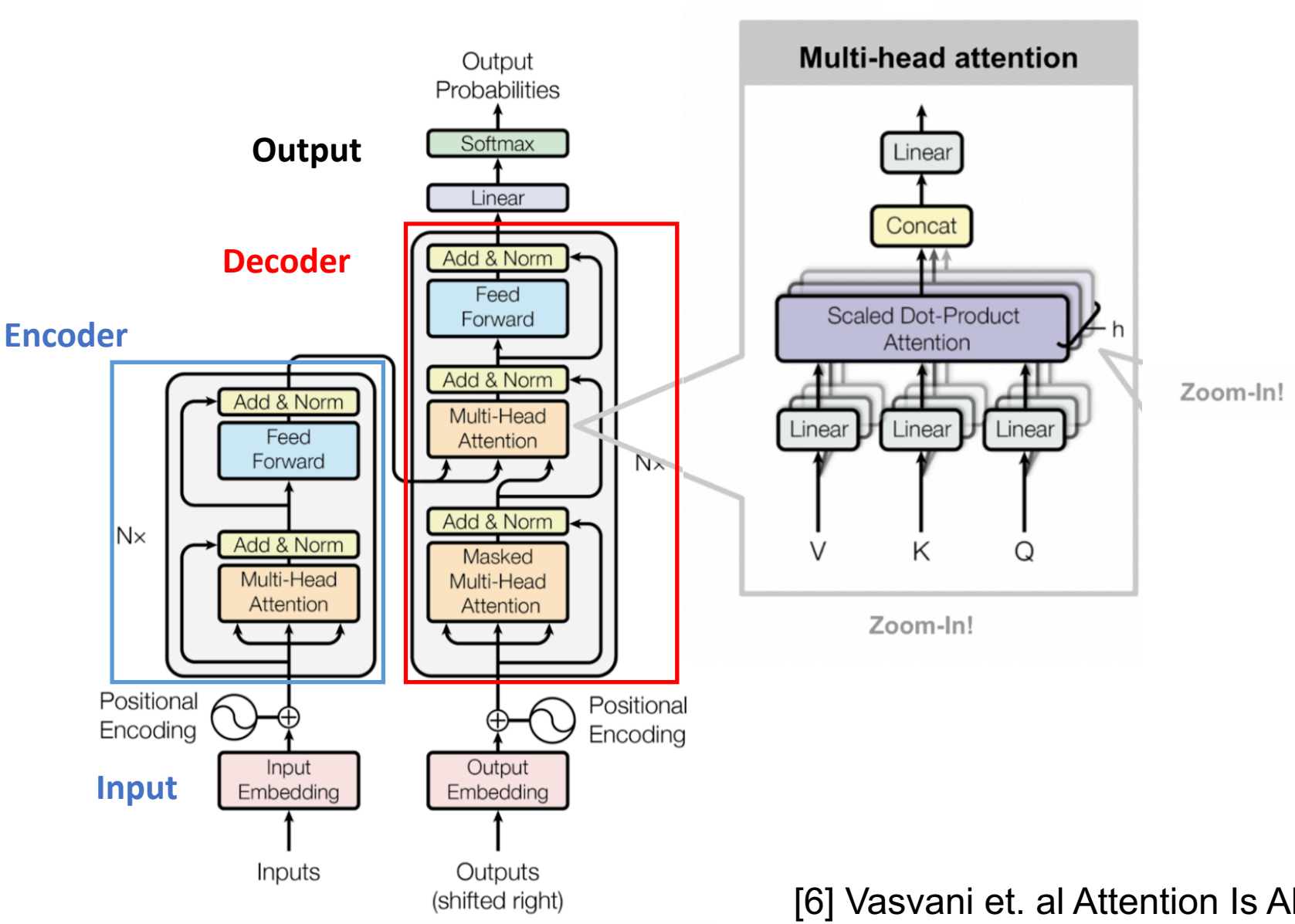
[6] Vasvani et. al Attention Is All You Need (Transformer)

Full Model Architecture



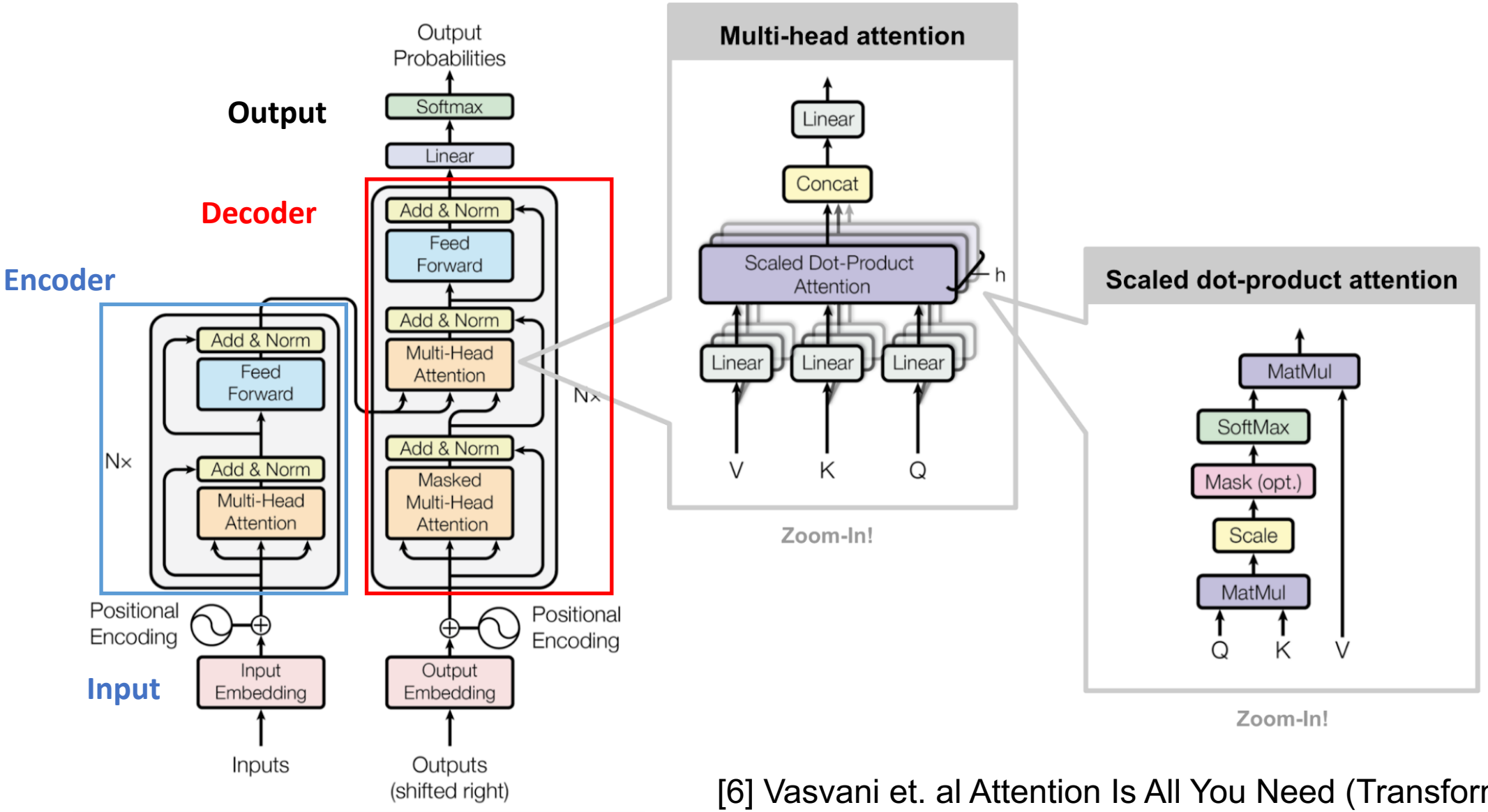
[6] Vasvani et. al Attention Is All You Need (Transformer)

Full Model Architecture



[6] Vasvani et. al Attention Is All You Need (Transformer)

Full Model Architecture

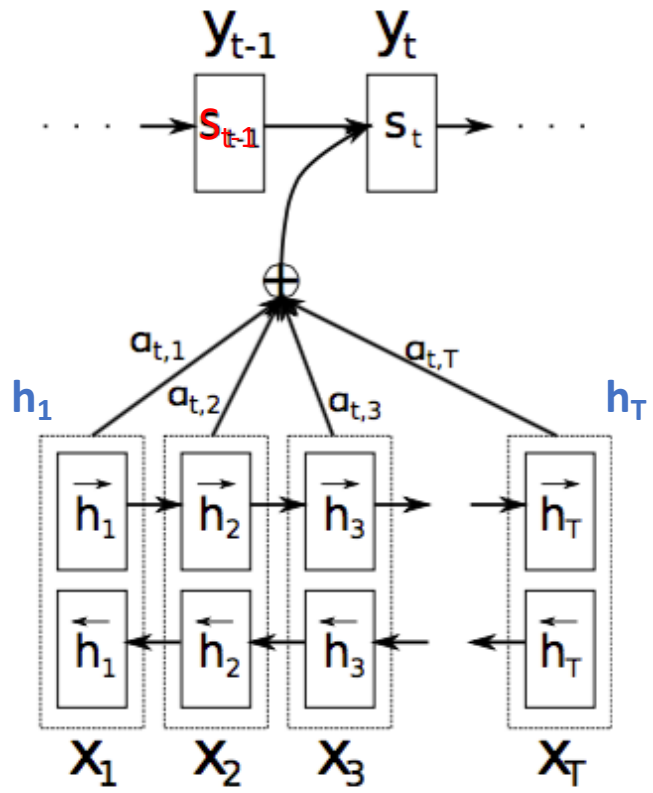


[6] Vasvani et. al Attention Is All You Need (Transformer)

1 Scaled Dot Product Attention

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

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



[6] Vasvani et. al Attention Is All You Need (Transformer)

1 Scaled Dot Product Attention

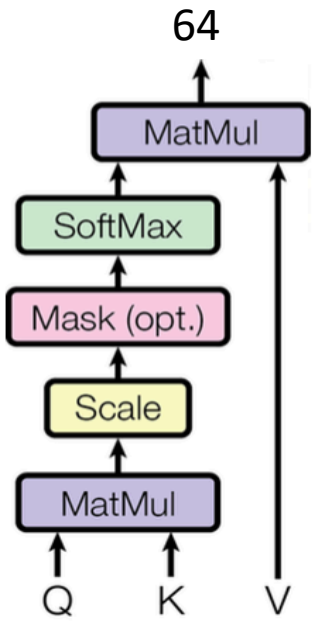
- **Dot-Product Attention**
  - Query(Q): S_{t-1}
  - Keys(K): $[h_1, h_2, \ldots, h_T]$
  - Values(V): $[h_1, h_2, \ldots, h_T]$

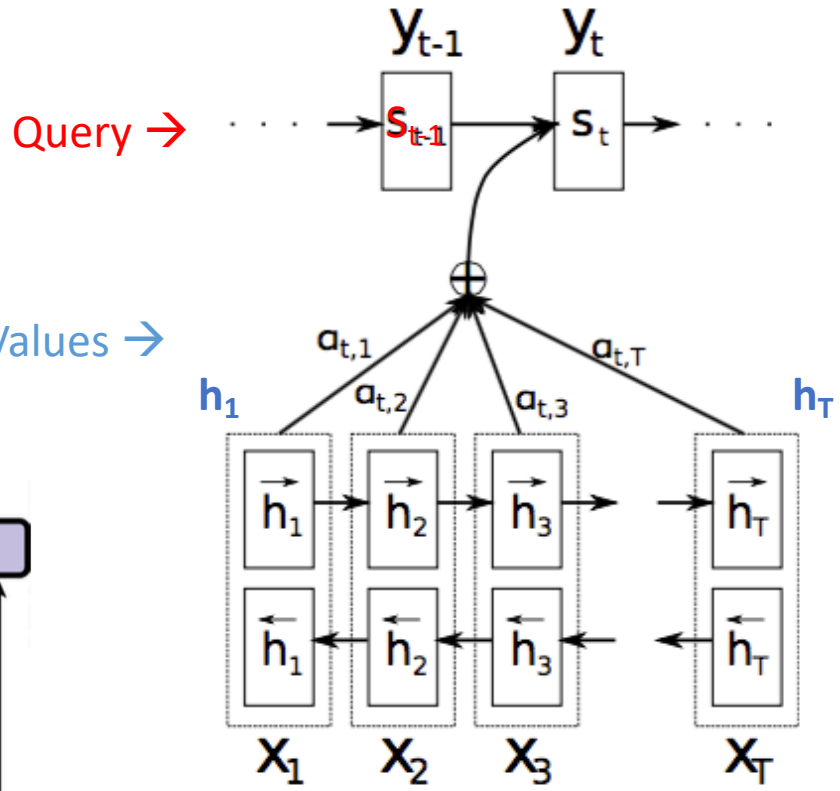
$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V$$

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

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

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





[6] Vasvani et. al Attention Is All You Need (Transformer)

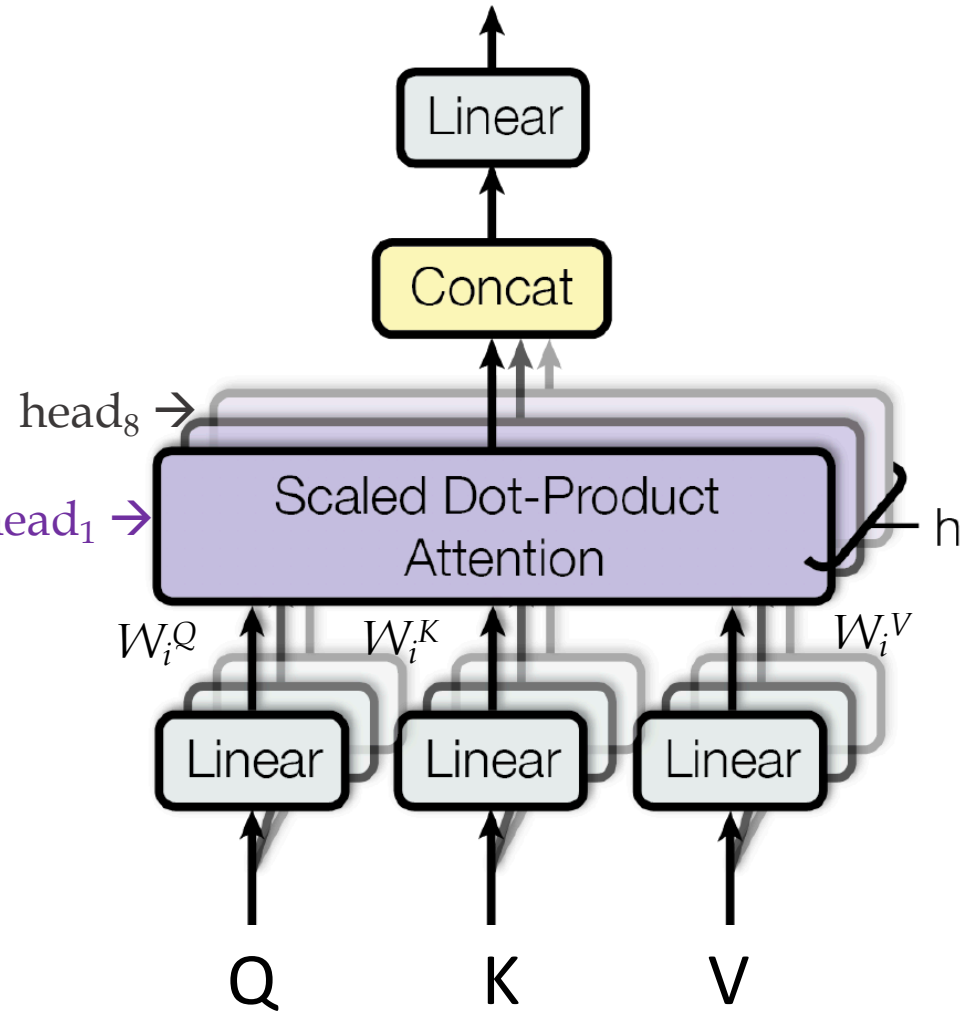
2 Multi Head Attention

- 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

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

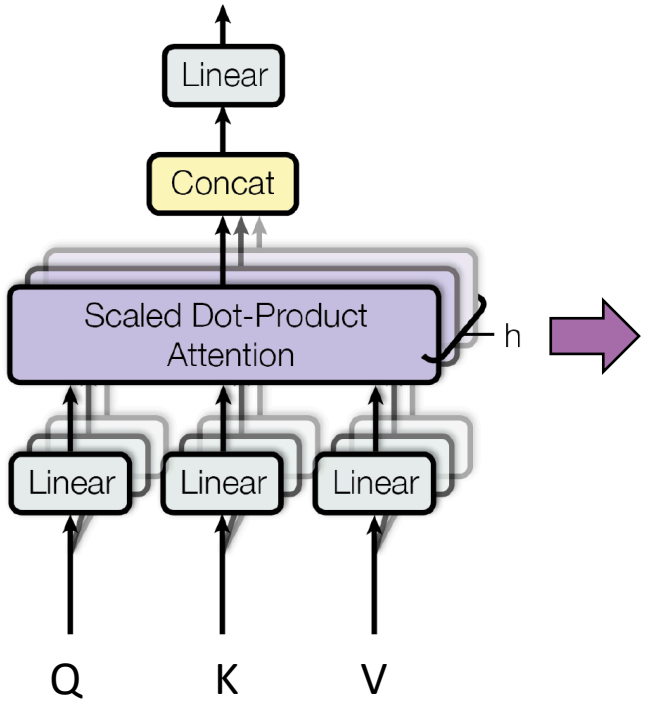
$$\text{head}_1 = \text{Attention}(QW_1^Q, KW_1^K, VW_1^V)$$

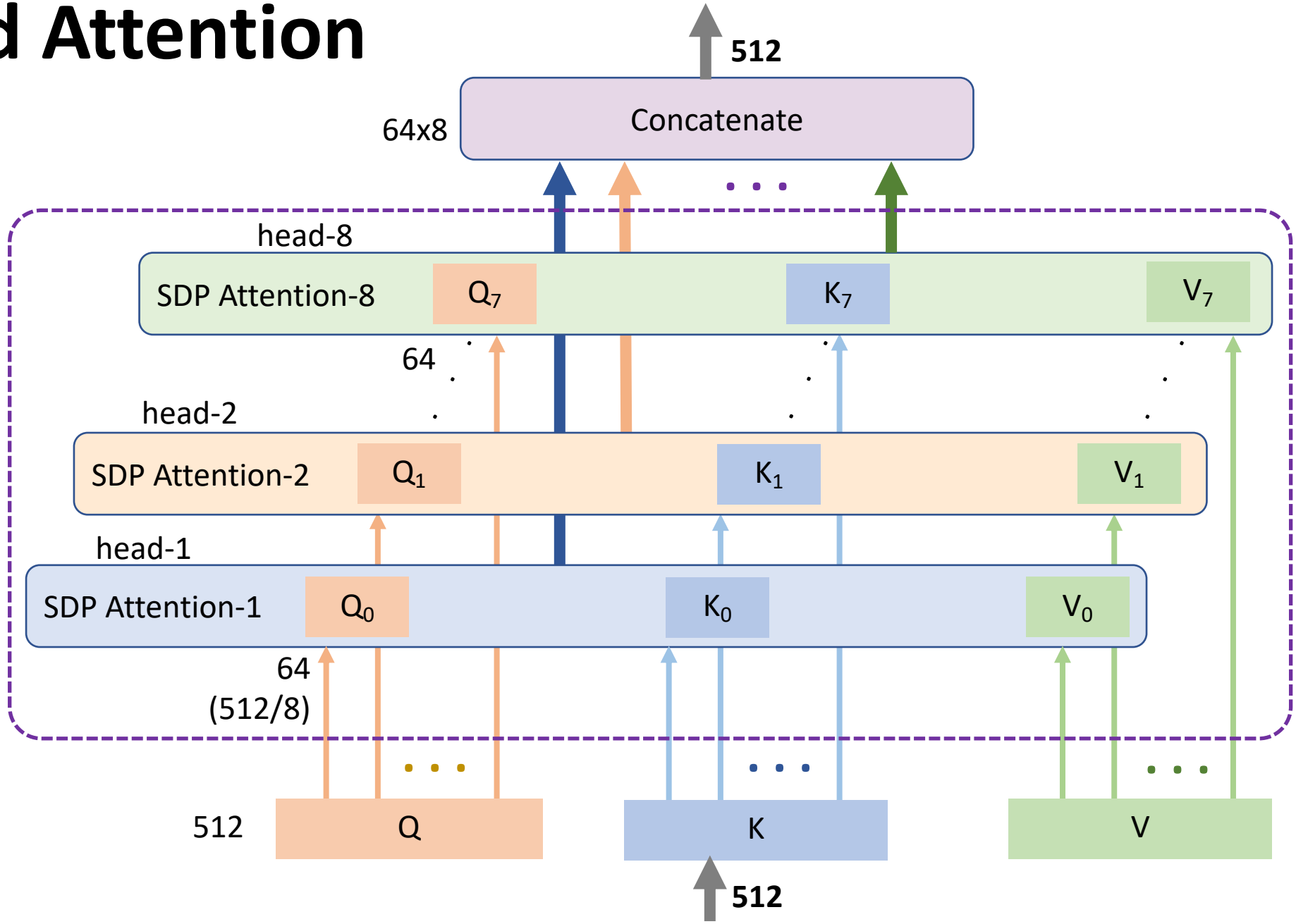
Attention on
- Projection of Query with Weight W_1^Q
- Projection of Key with Weight W_1^K
- Projection of Value with Weight W_1^V



[6] Vasvani et. al Attention Is All You Need (Transformer)

2 Multi Head Attention





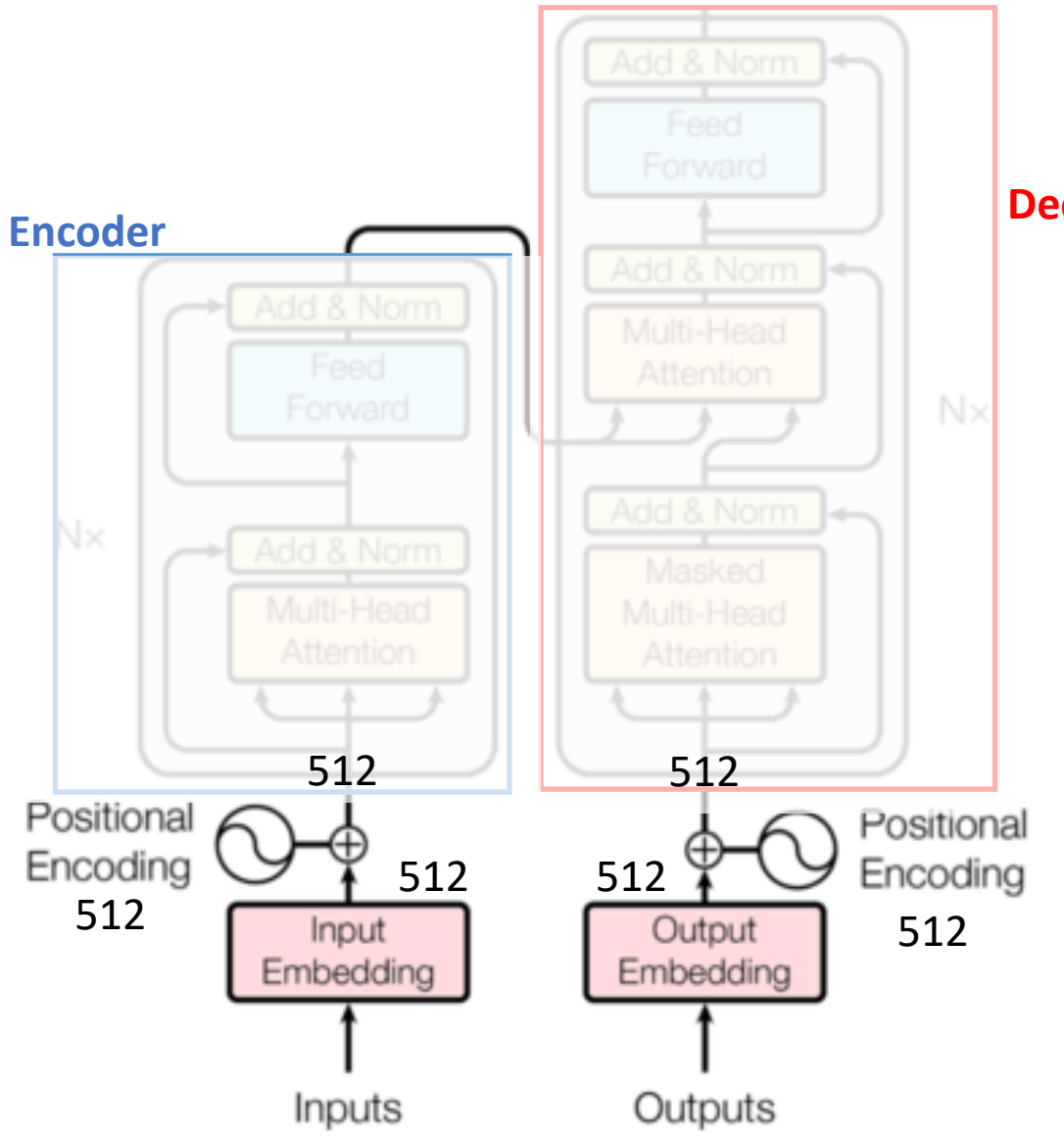
[6] Vasvani et. al Attention Is All You Need (Transformer)

4 Embedding and Positional Encoding

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

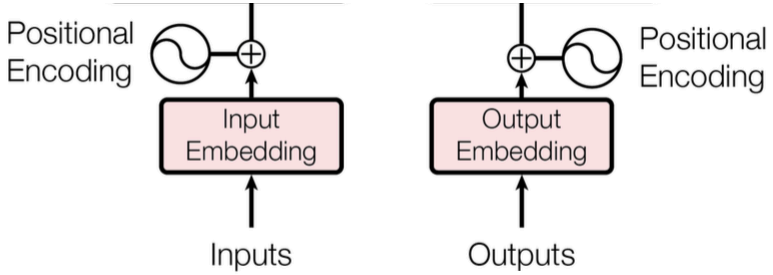
- **Positional Encoding (fixed):**
  - Need to include position information
  - Use sine and cosine functions of different frequencies
  - Dimension = 512

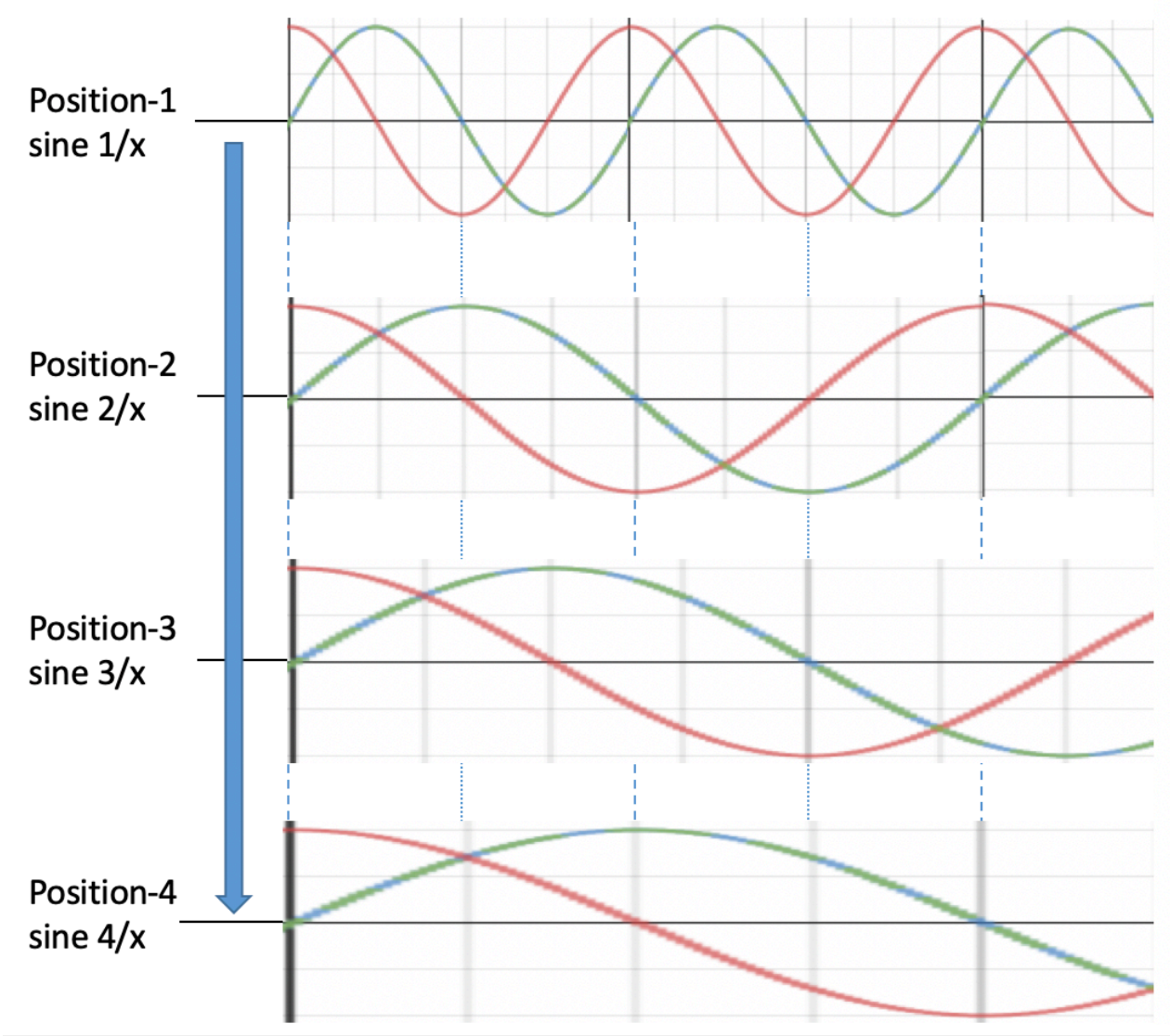
- **Embedding and Positional Encoding**
  - Are added and fed to Encoder and Decoder



[6] Vasvani et. al Attention Is All You Need (Transformer)

4 Positional Encoding



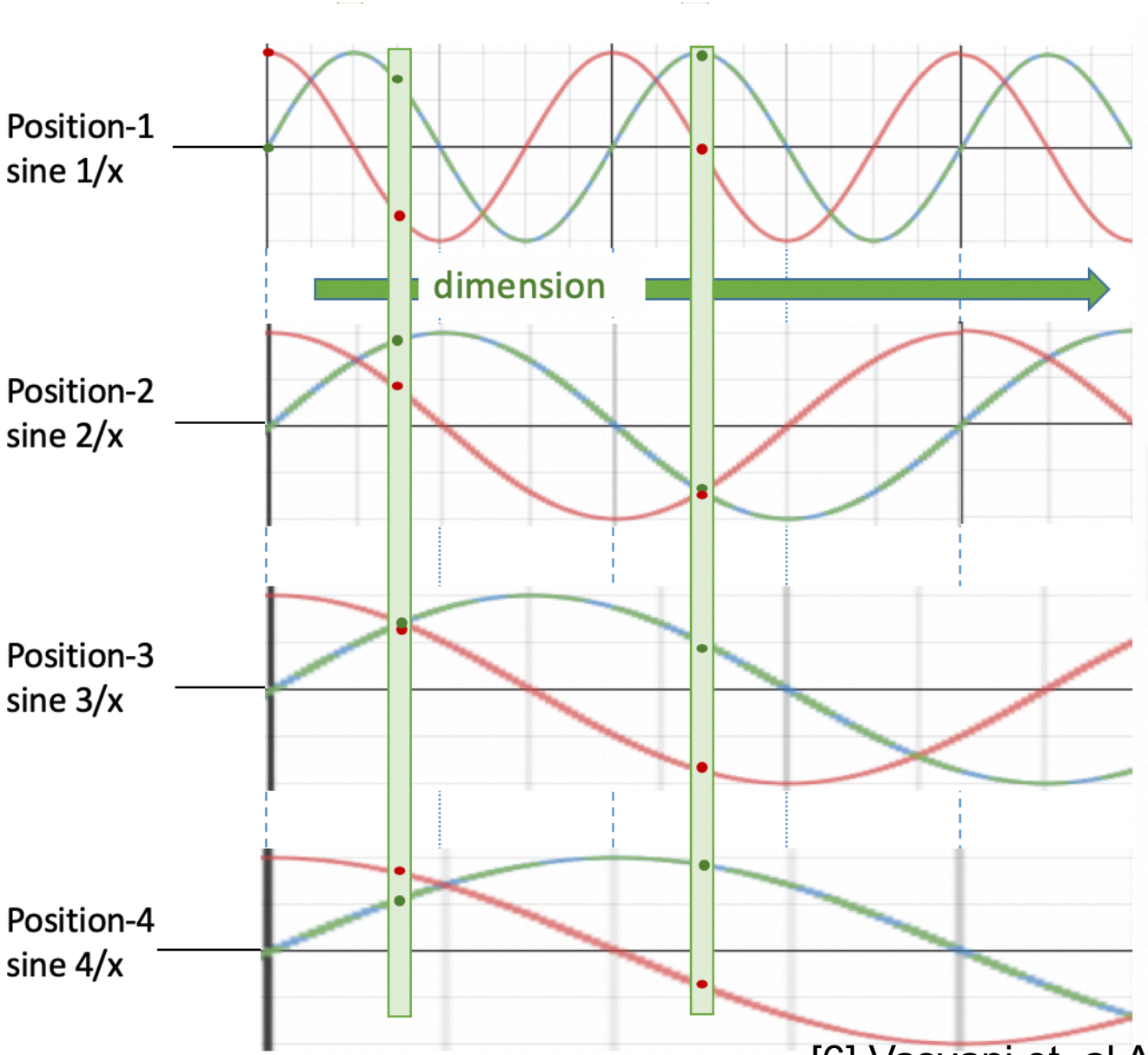


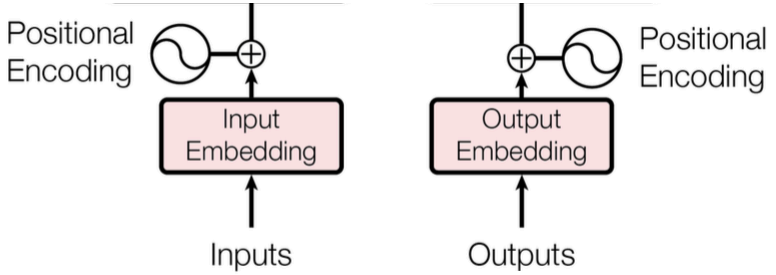
- Use sine and cosine functions of different frequencies

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

[6] Vasvani et. al Attention Is All You Need (Transformer)

4 Positional Encoding

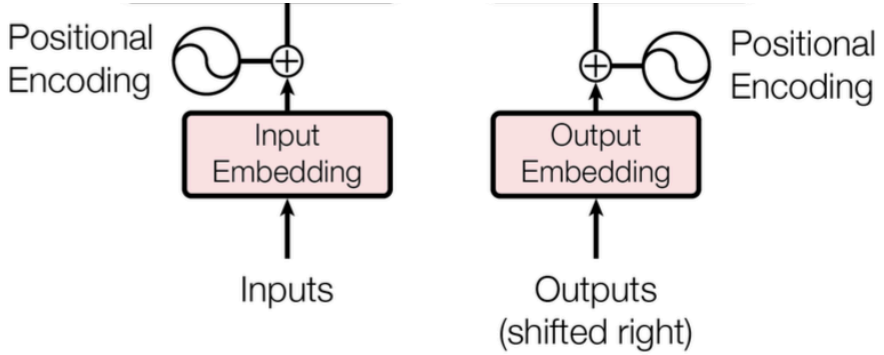




$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

[6] Vasvani et. al Attention Is All You Need (Transformer)

4 Positional Encoding

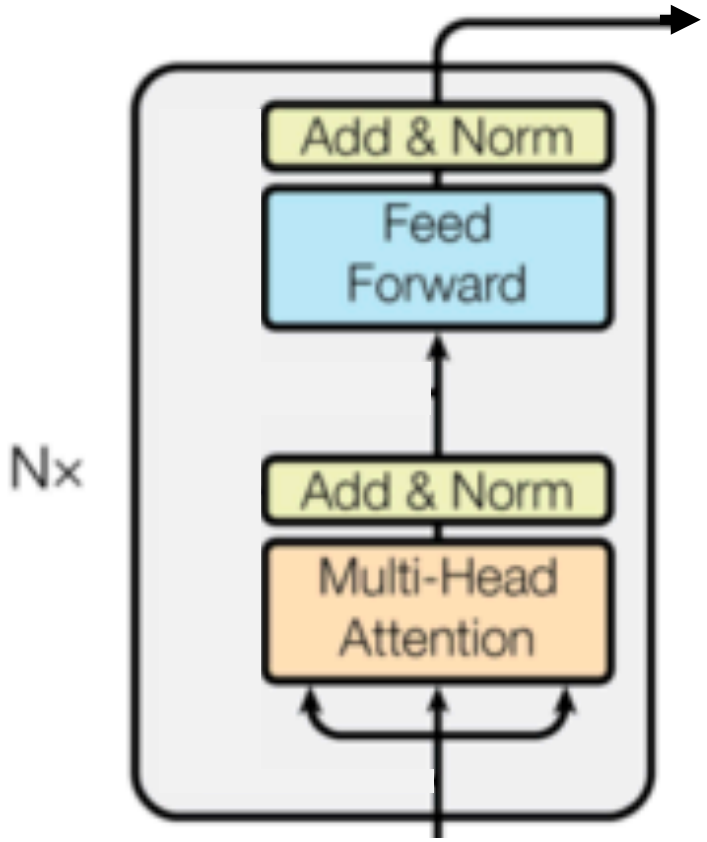


- Easily learn to attend by relative positions
- Any fixed offset k, PE(pos+k) can be represented as a linear function of PE(pos)
- $\begin{bmatrix} \sin(pos+k) \\ \cos(pos+k) \\ \dots \end{bmatrix} = \begin{bmatrix} \sin(pos)\cos(k) + \cos(pos)\sin(k) \\ \cos(pos)\cos(k) - \sin(pos)\sin(k) \\ \dots \end{bmatrix}$
- Model will extrapolate to sequence lengths longer than the ones seen during training

[6] Vasvani et. al Attention Is All You Need (Transformer)

Encoder (1/3)

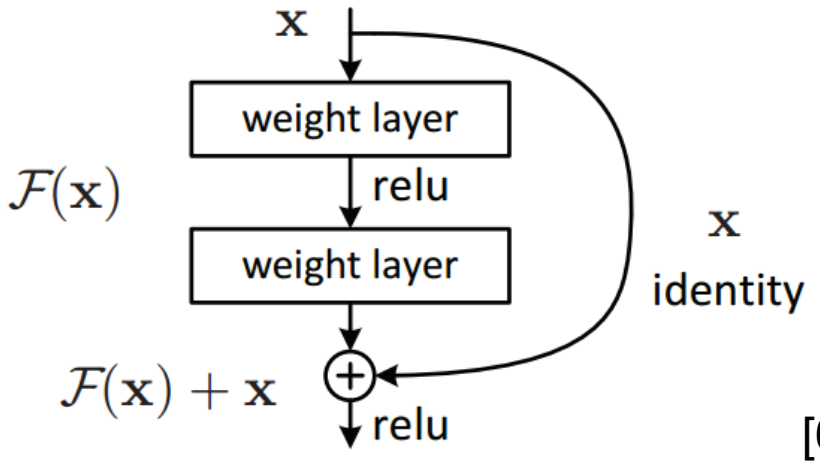
- **Six Layers (stacked)**
- **Each layer has two sub-layers**
  - Multi-head attention (self-attention)
  - Feed-Forward
    - hidden layer d_{ff} = 2048, input and output = 512
    - $FFN(x) = ReLU(xW1 + b1)\ W2 + b2$

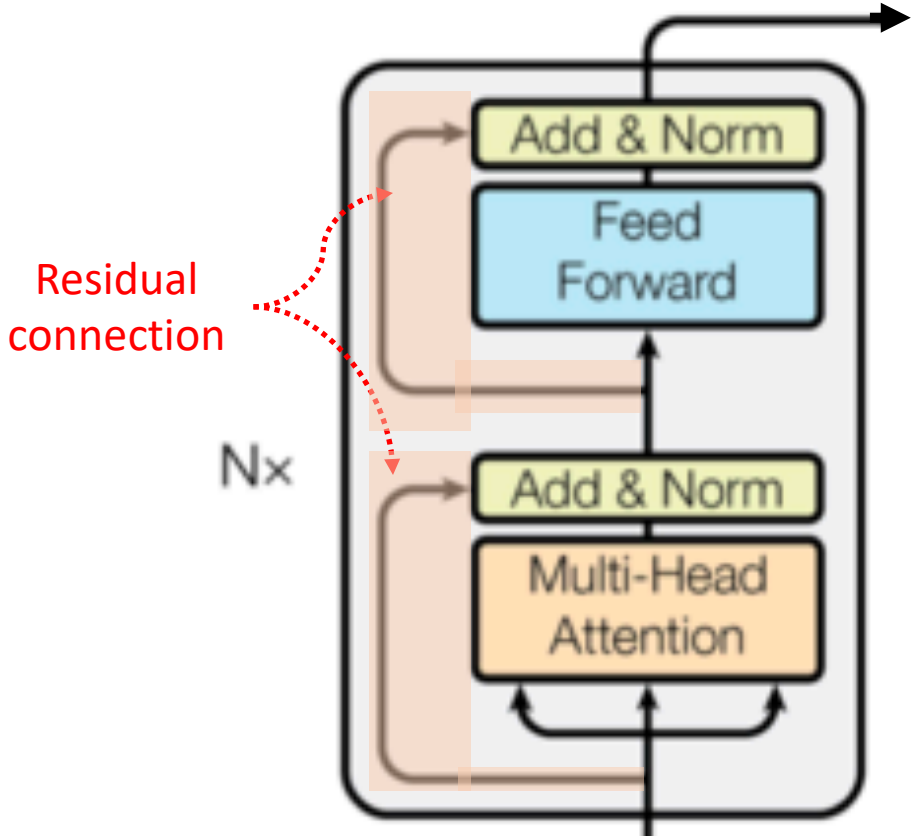


[6] Vasvani et. al Attention Is All You Need (Transformer)

Encoder (2/3)

- **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





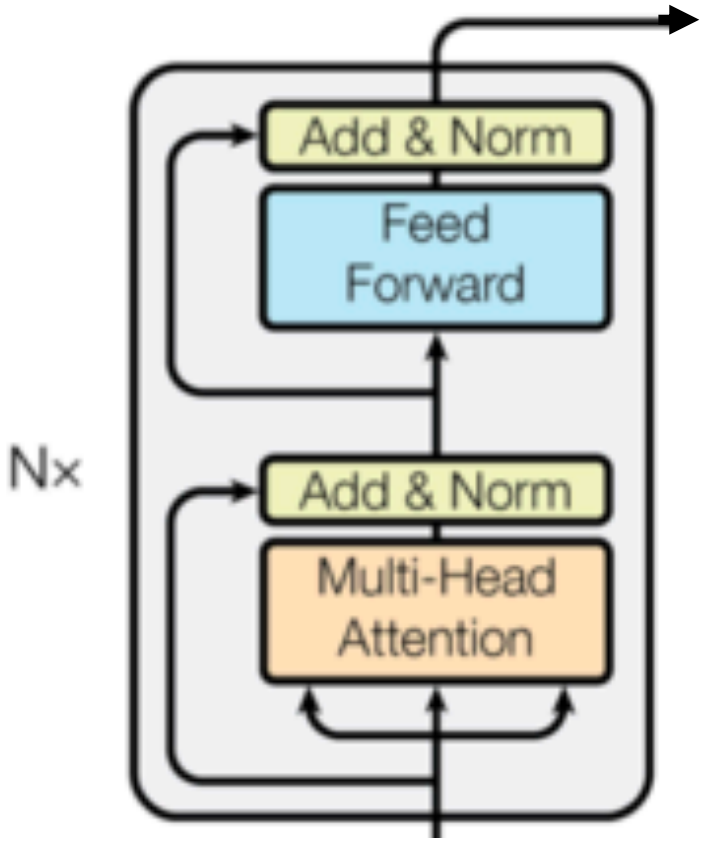
[6] Vasvani et. al Attention Is All You Need (Transformer)

Encoder (3/3)

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

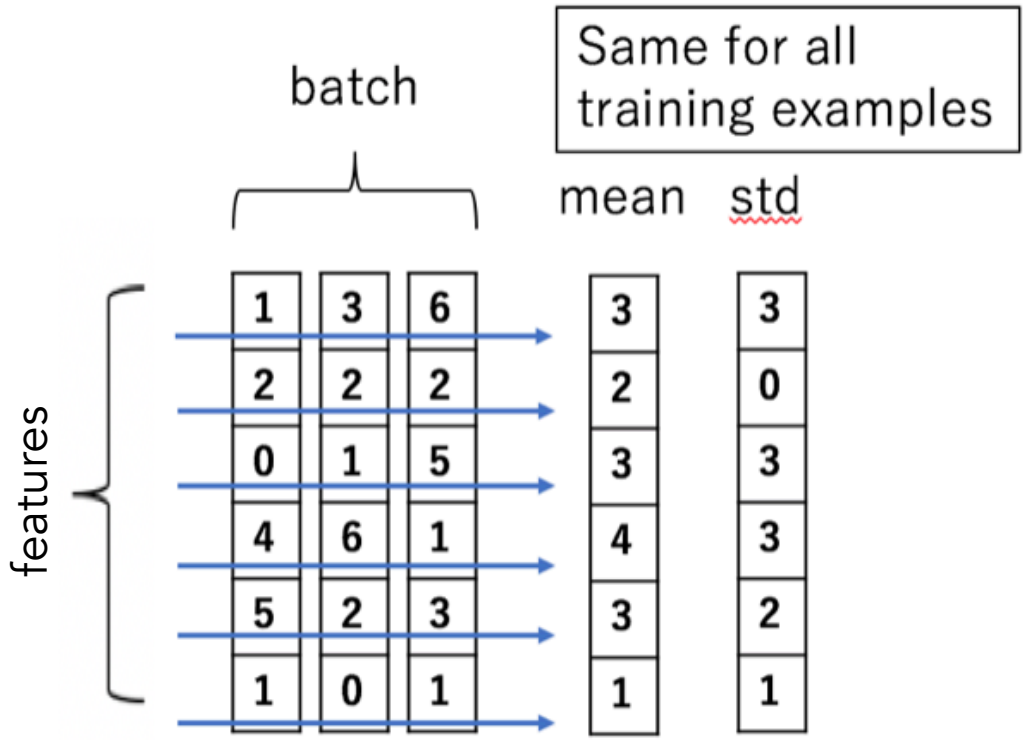


[6] Vasvani et. al Attention Is All You Need (Transformer)

Batch Normalization

Ref

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



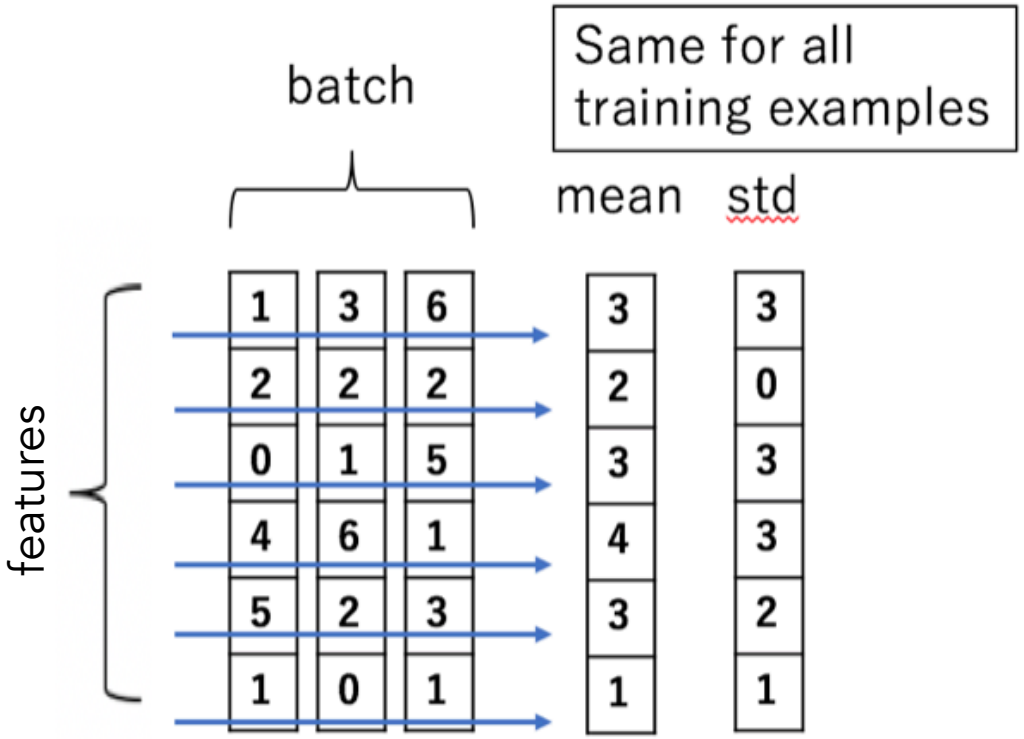
the statistics are computed across the **batch** and are the same for each example in the batch

- Difficult to apply to recurrent connections

[6] Vasvani et. al Attention Is All You Need (Transformer)

Batch Normalization

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

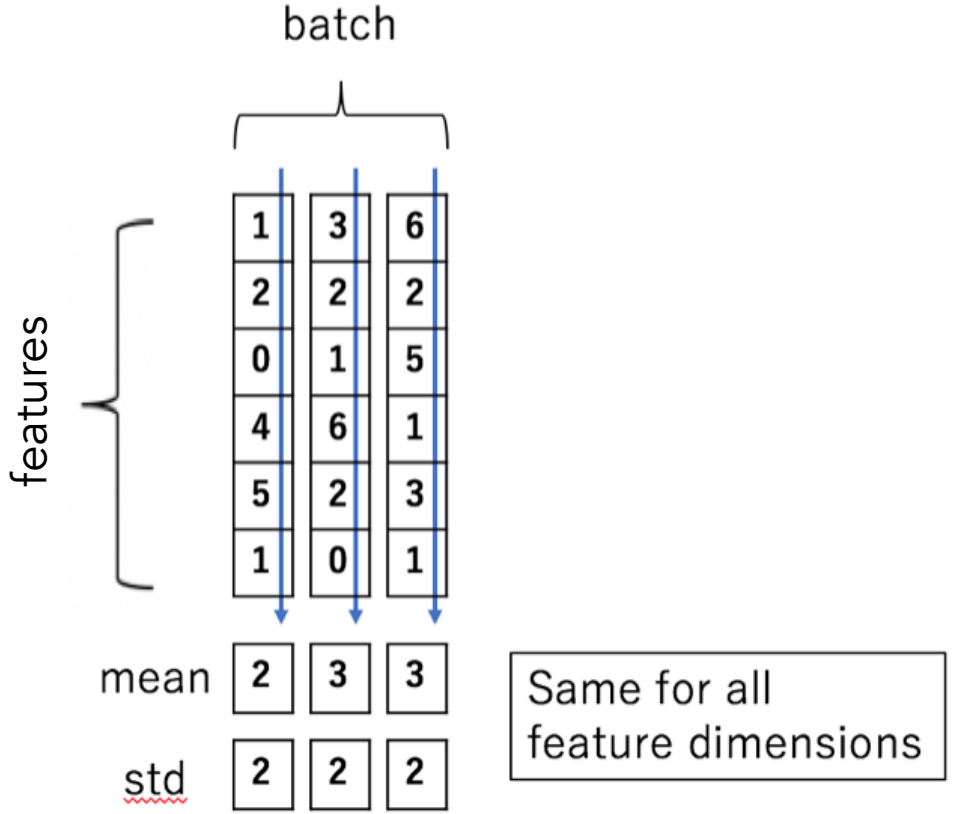


the statistics are computed across the **batch** and are the same for each example in the batch

Layer Normalization

Ref

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

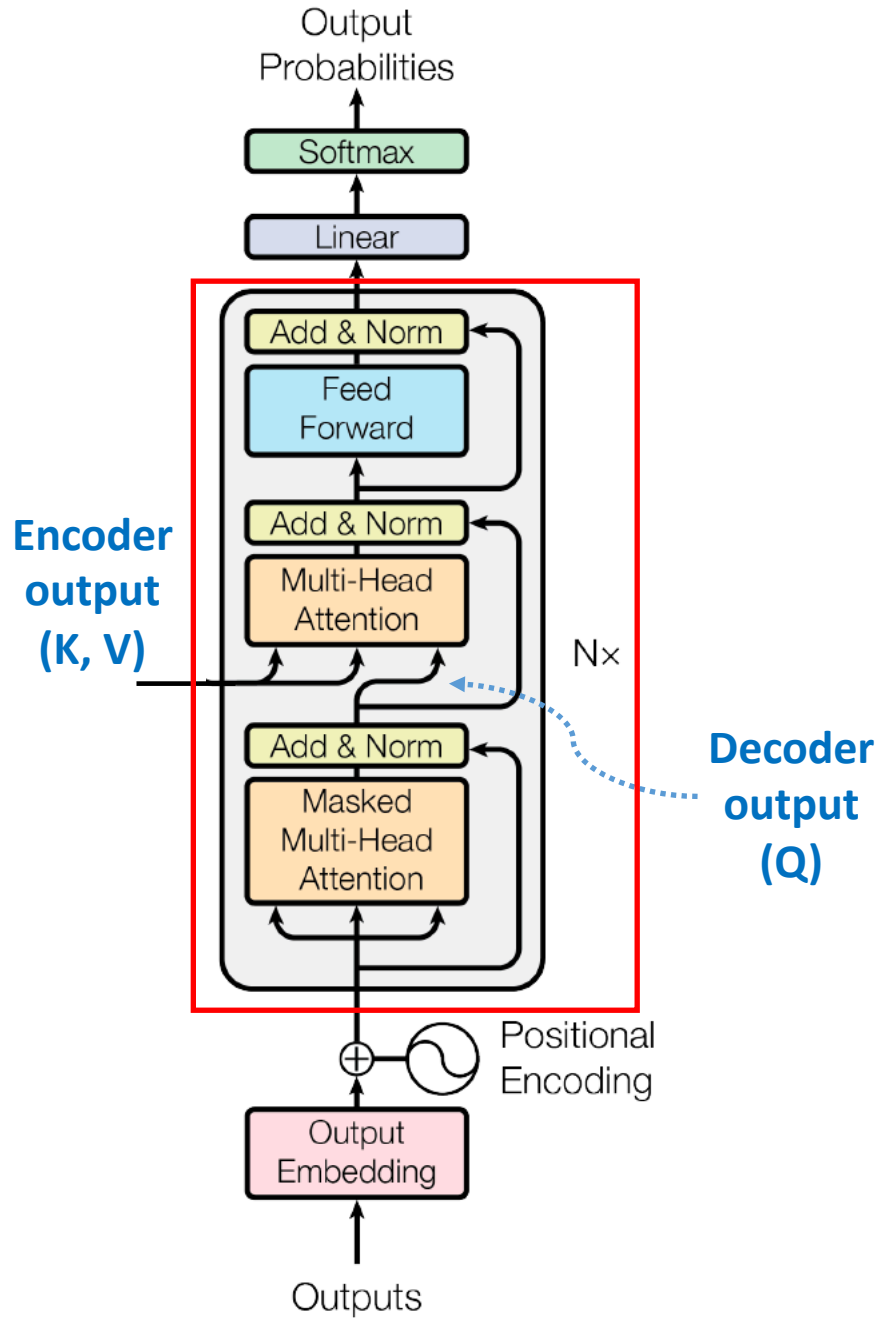


statistics are computed across **each feature** and are independent of other examples

[6] Vasvani et. al Attention Is All You Need (Transformer)

Decoder (1/3)

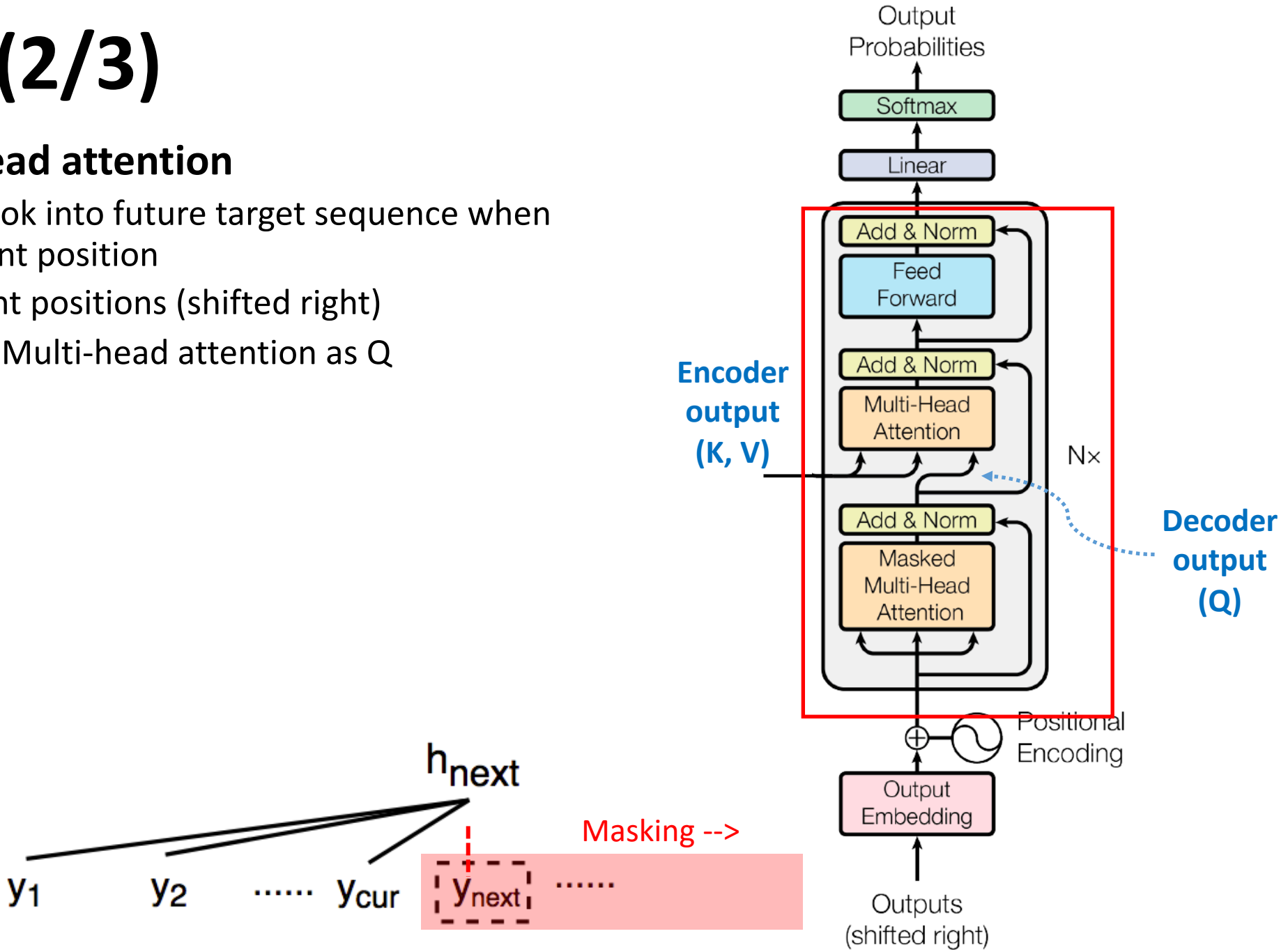
- **Six Layers (stacked)**
- **Each layer has three sub-layers**
  - Masked Multi-head attention
  - Multi-head attention
  - Feed-Forward
- **Multi-head attention**
  - Output of Encoder is fed as K and V
  - Output of Masked Multi Head is fed as Q



[6] Vasvani et. al Attention Is All You Need (Transformer)

Decoder (2/3)

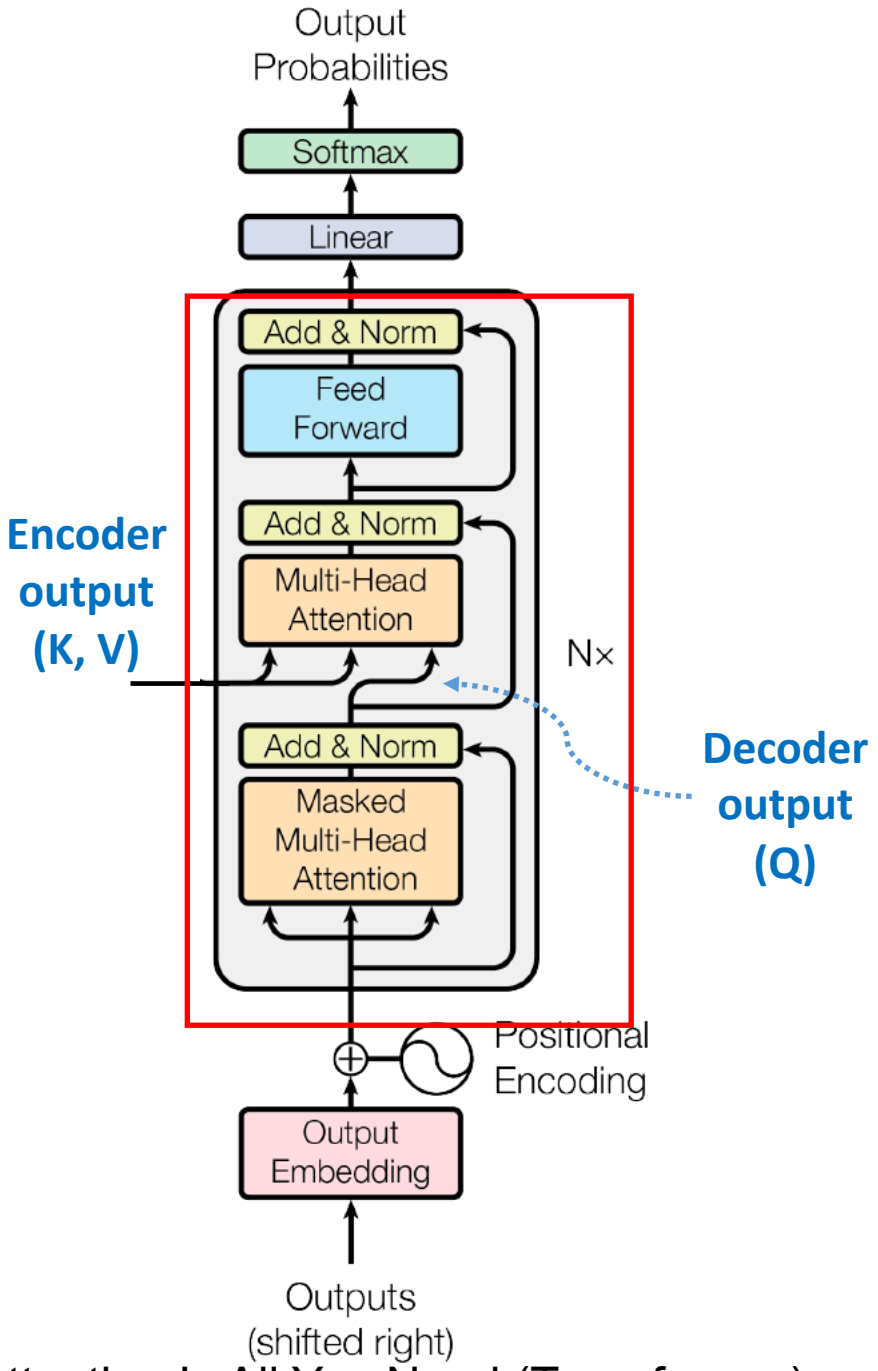
- **Masked Multi-head attention**
  - Don’t want to look into future target sequence when predicting current position
  - Mask subsequent positions (shifted right)
  - Output is fed to Multi-head attention as Q



[6] Vasvani et. al Attention Is All You Need (Transformer)

Decoder & Output (3/3)

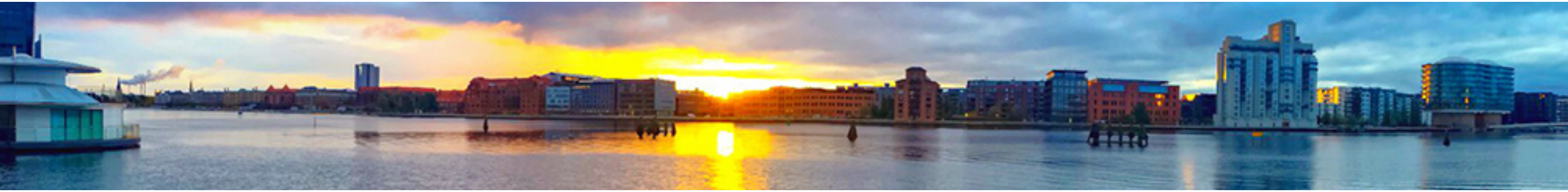
- **Masked Multi-head attention**
  - Don’t want to look into future target sequence when predicting current position
  - Mask subsequent positions (shifted right)
  - Output is fed to Multi-head attention as Q
- **Output**
  - **Fully connected layer**
  - **Softmax**



[6] Vasvani et. al Attention Is All You Need (Transformer)

7

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



Language Modeling

Approaches:

- Unidirectional vs Bidirectional
- **Pre-Trained Model**
- Task specific or multi-task training
- Unsupervised Pre-Training and Supervised Fine-Tuning
- Context free or context specific, word-level or higher

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

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.

[7] Devlin J. et. al 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)

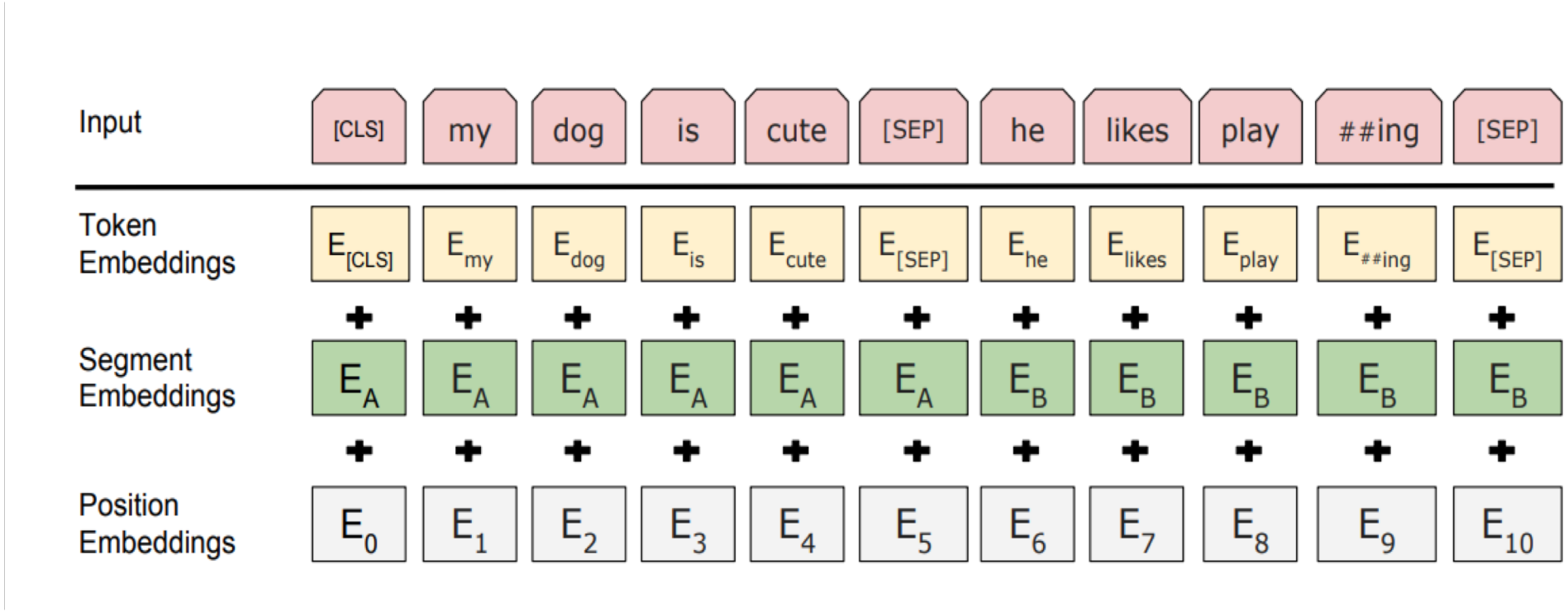
Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext

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

[7] Devlin J. et. al 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.



[7] Devlin J. et. al 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

[7] Devlin J. et. al 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

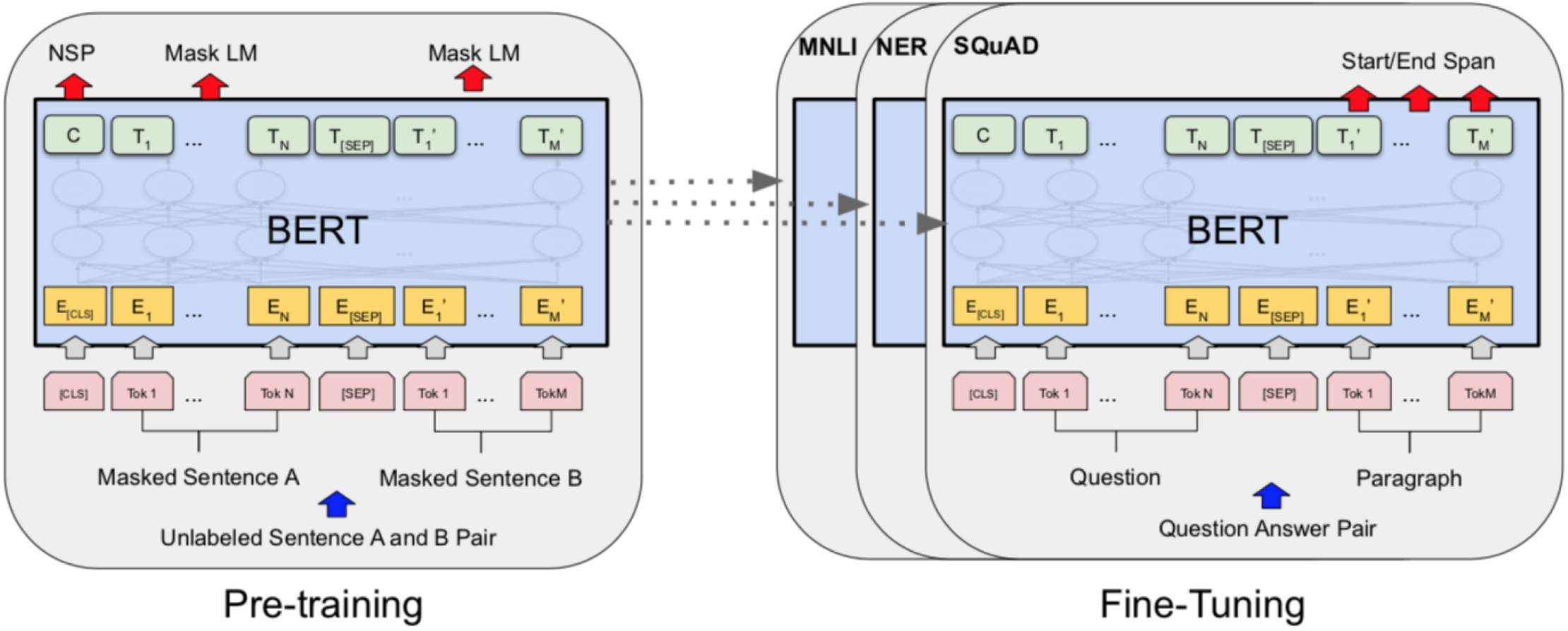
$$P = \text{softmax}(CW^T)$$

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

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

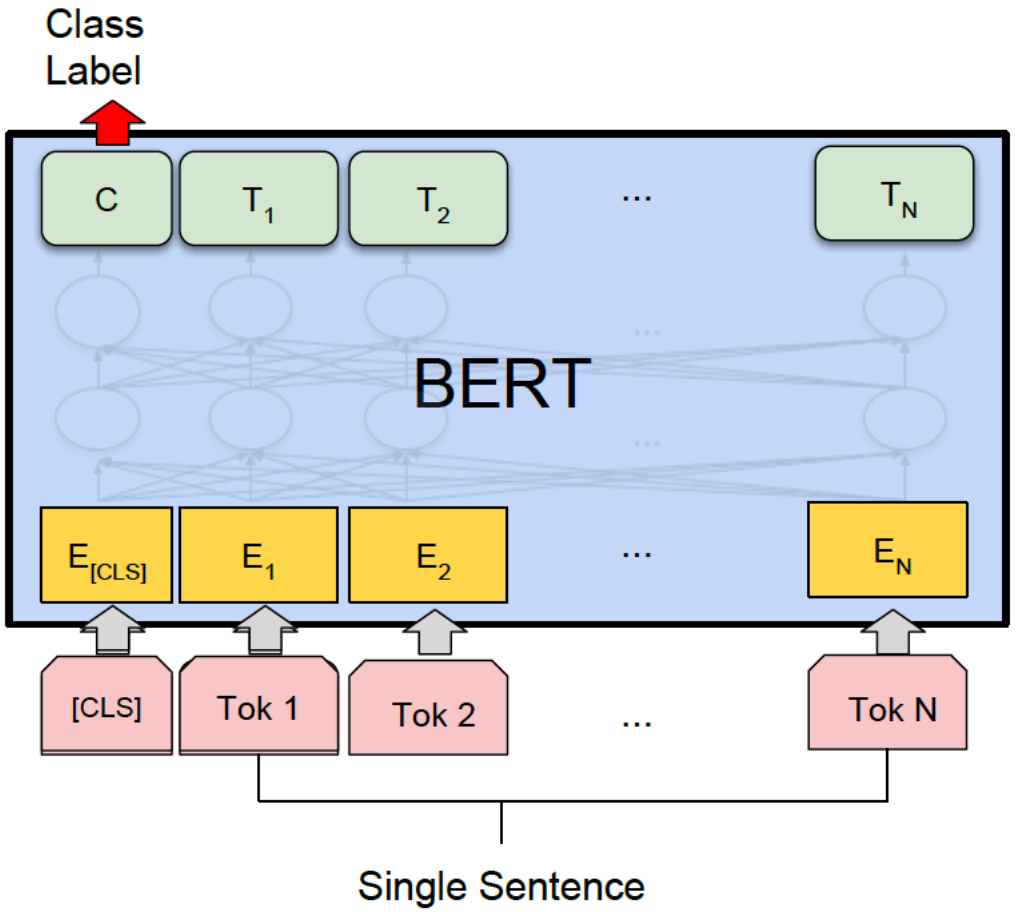
BERT: Fine Tuning

- Sentiment Analysis
- Tagging (NER-Named Entity Recognition)
- Sentence Pair Classification
- Question Answer



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

1 Single Sentence Classification (e.g Sentiment)

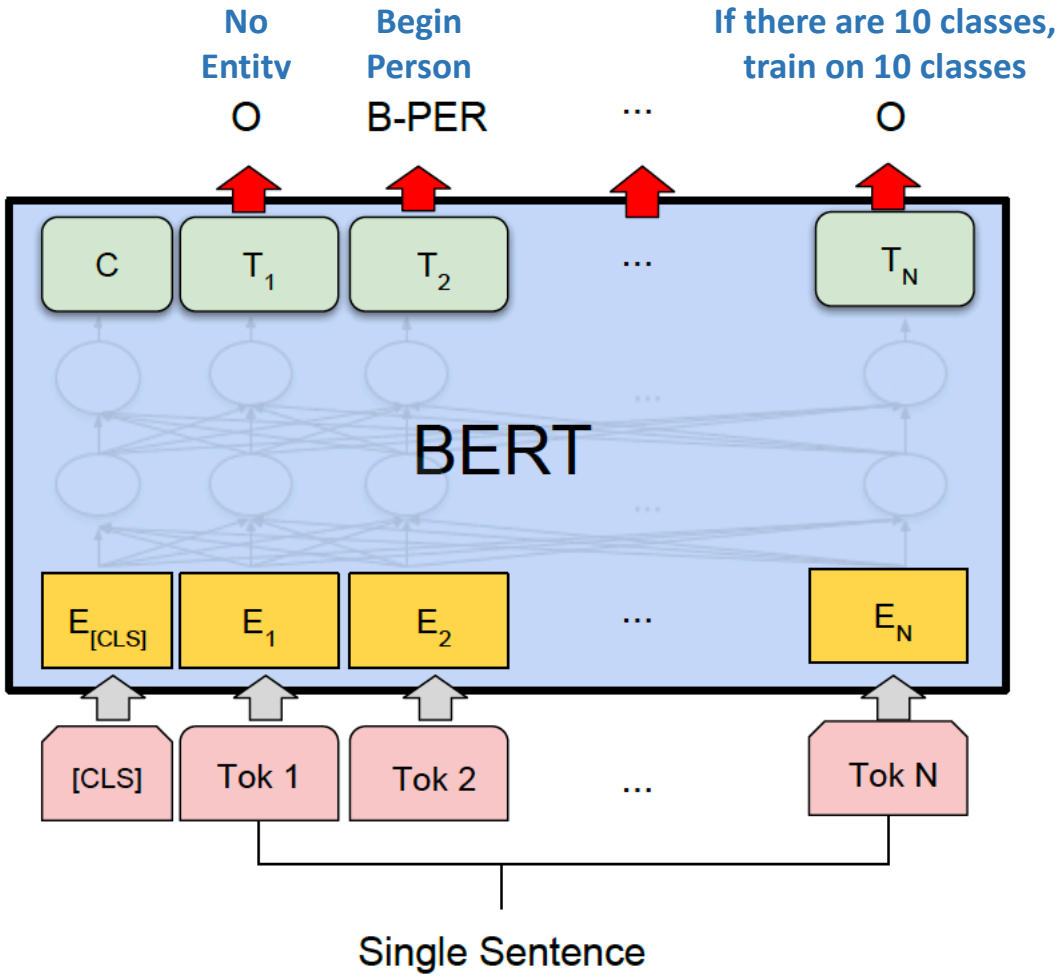


(b) Single Sentence Classification Tasks:
SST-2, CoLA

Is Movie Review Positive or Negative?

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 Single Sentence Tagging Tasks (e.g NER)



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
```

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

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

3 Sentence Pair Classification (1/3)

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

question1	question2	is_duplicate
What are natural numbers?	What is a least natural number?	0
Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
How do you start a bakery?	How can one start a bakery business?	1
Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

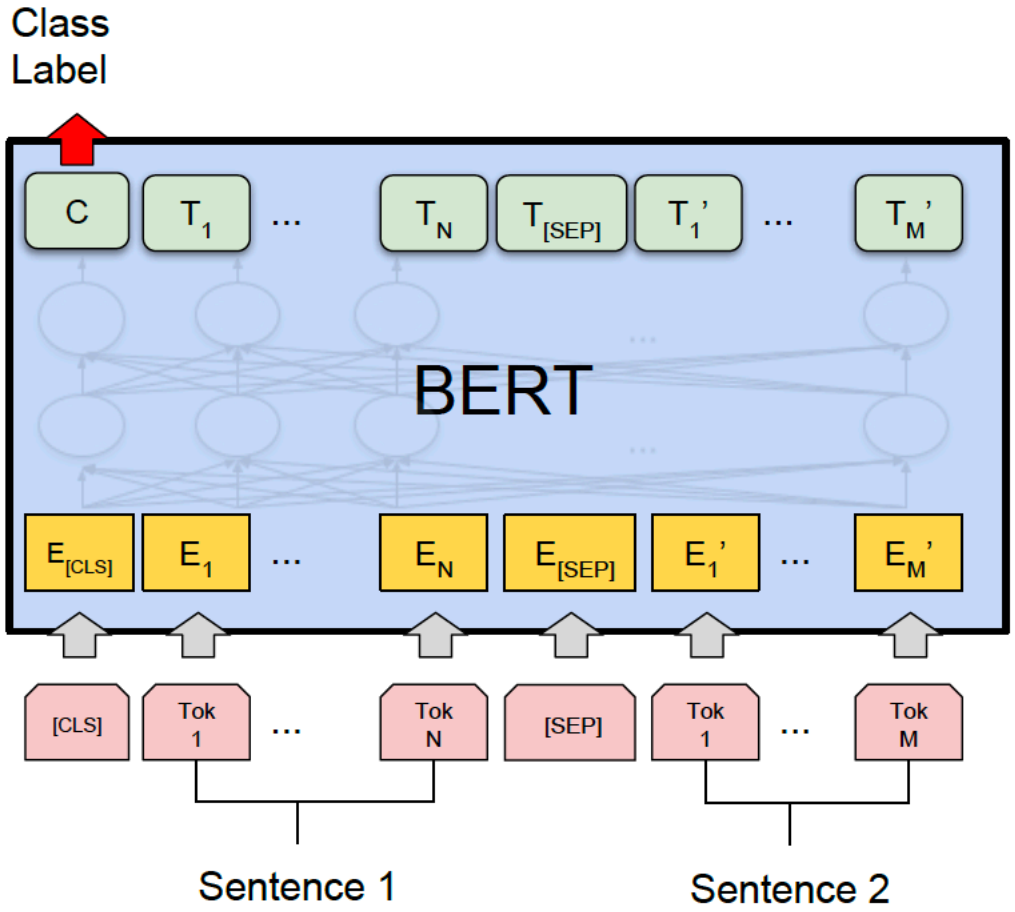
3 Natural language inference (2/3)

- Determine whether a “hypothesis” is
  - true (entailment),
  - false (contradiction), or
  - undetermined (neutral)

  given a “premise”.

Premise	Label	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction	The man is sleeping.
An older and younger man smiling.	neutral	Two men are smiling and laughing at the cats playing on the floor.
A soccer game with multiple males playing.	entailment	Some men are playing a sport.

3 Sentence Pair Classification (3/3)



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

4 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

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

4 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?`

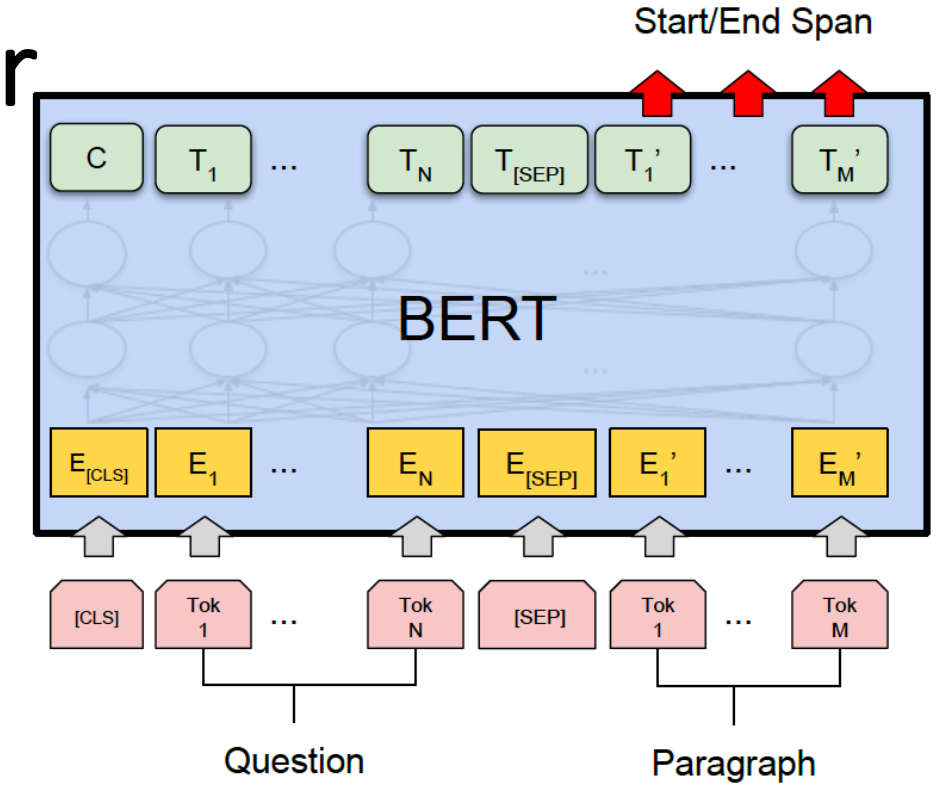
- Input Paragraph:

  `... 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.

Code- Notebook

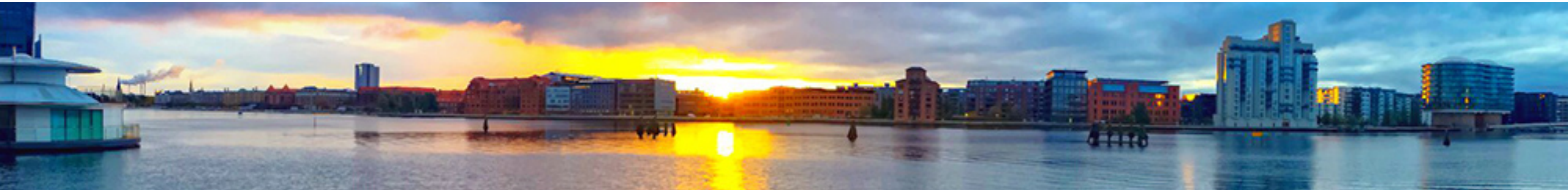
Ref: Documents/RecSys2019 MT-KG/Final/

2 Transformer_Torch_Tutorial.ipynb

3 BERT_Torch_Tutorial.ipynb

8

[8] Fei Sun, et. al.
BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

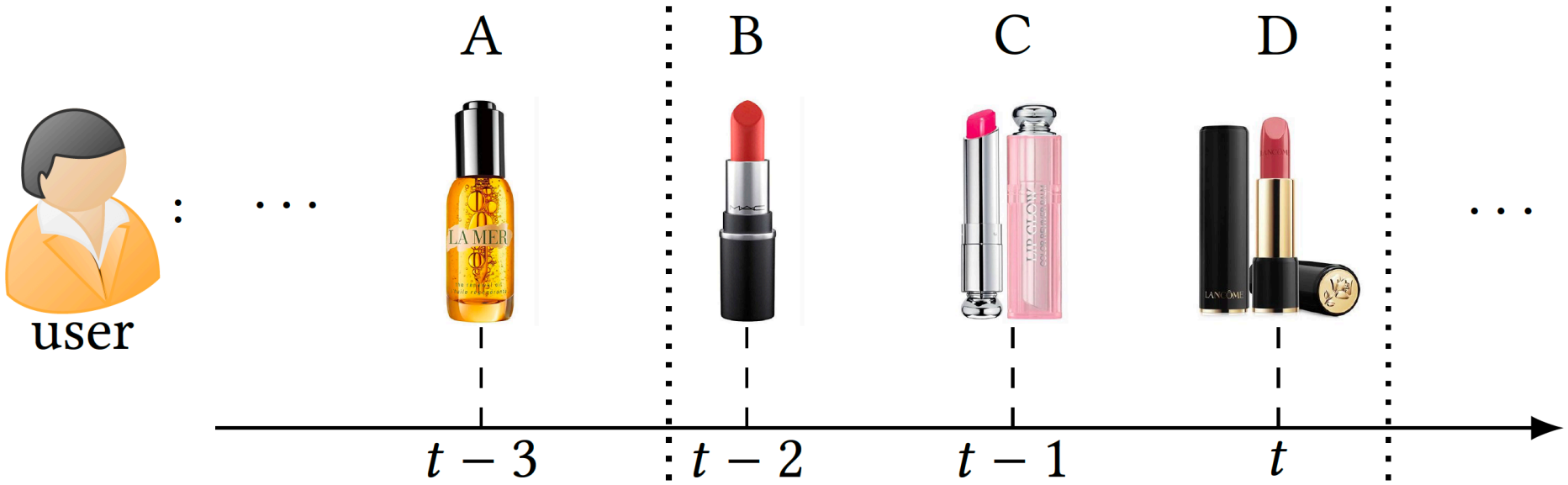


BERT4Rec

- BERT – State Of The Art mechanism used for various problems
- Challenge
  - Model users' dynamic and evolving preferences from historical behaviors
- Limitations of previous approaches:
  - Assume rigid ordered sequence
  - Have only left to right (unidirectional) architectures
- Conditioning on both left and right context in bidirectional model
  - Predict masked items jointly conditioning on both left and right context
  - This way can generate more training data

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

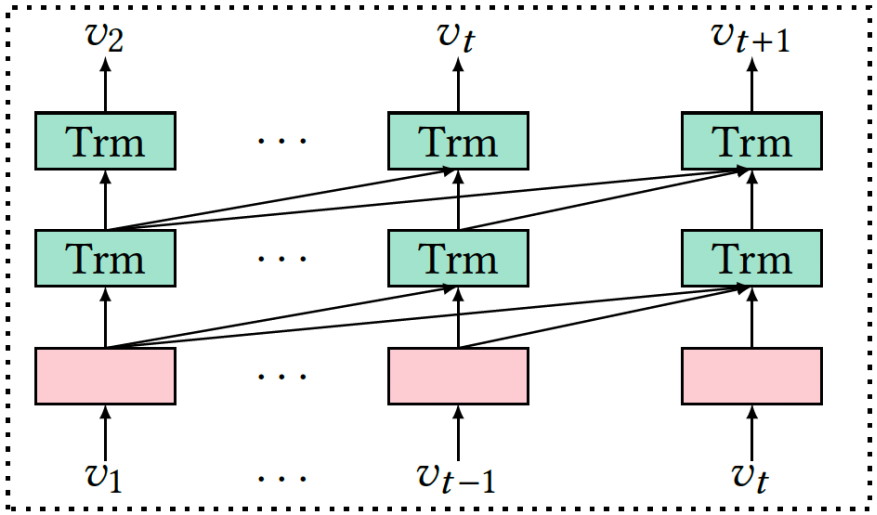
An example of anonymous user click records



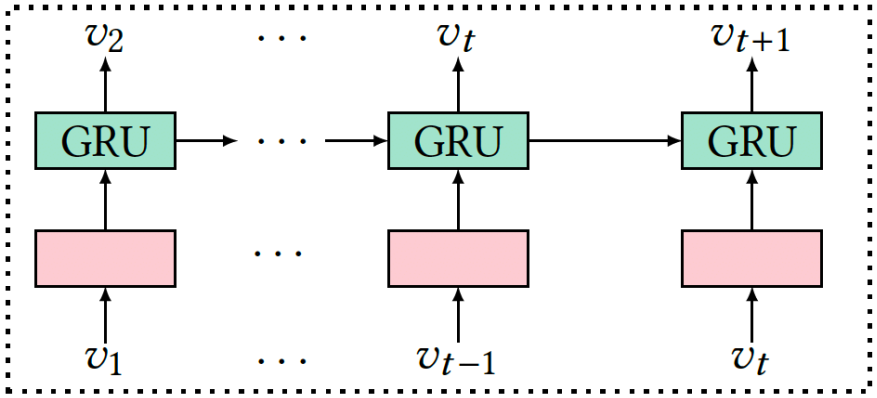
Users' click sequence can't be rigid
A user's actions in a short period are often random (e.g., the order in [B,C,D]).

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

BERT4Rec



(c) SASRec model architecture.

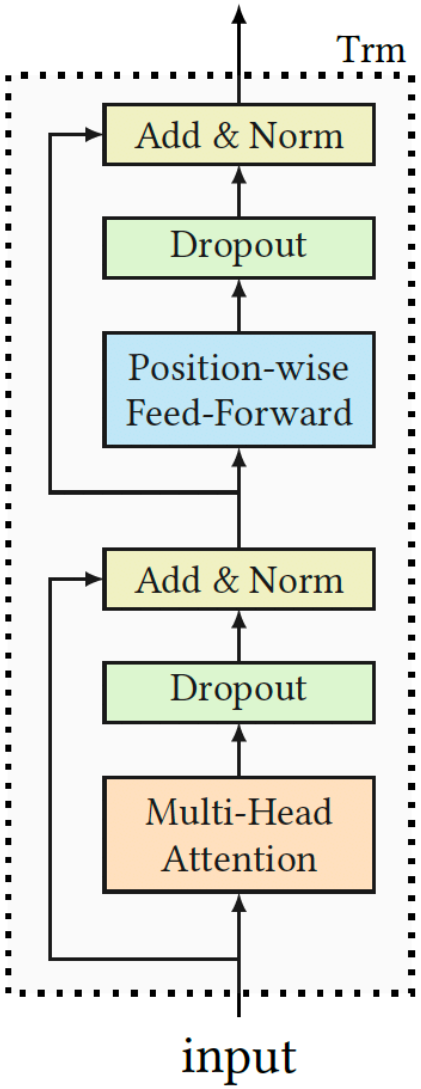


(d) RNN based sequential recommendation methods.

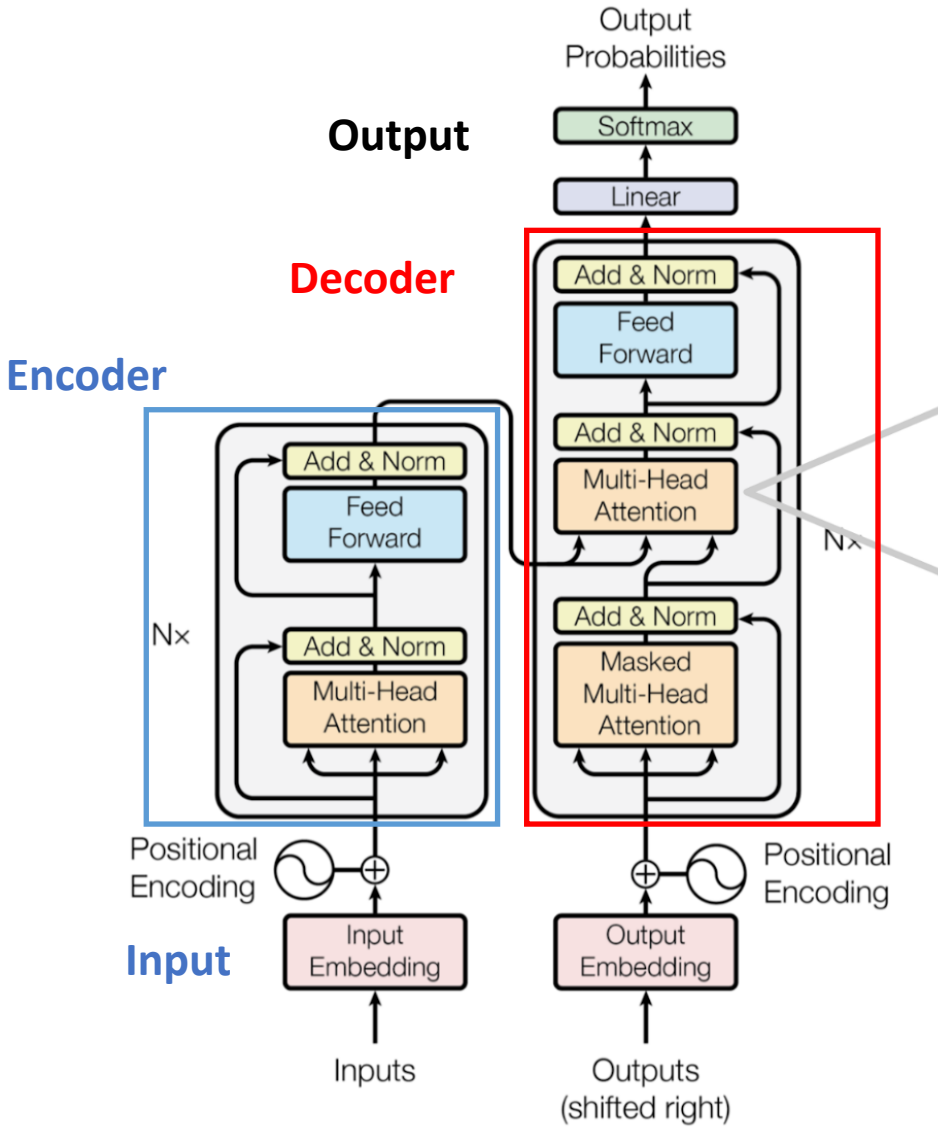
Self-Attentive Sequential Recommendation (SASRec) and RNN based methods are all left-to-right unidirectional model which predicts next item sequentially.

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

Transformer Architecture

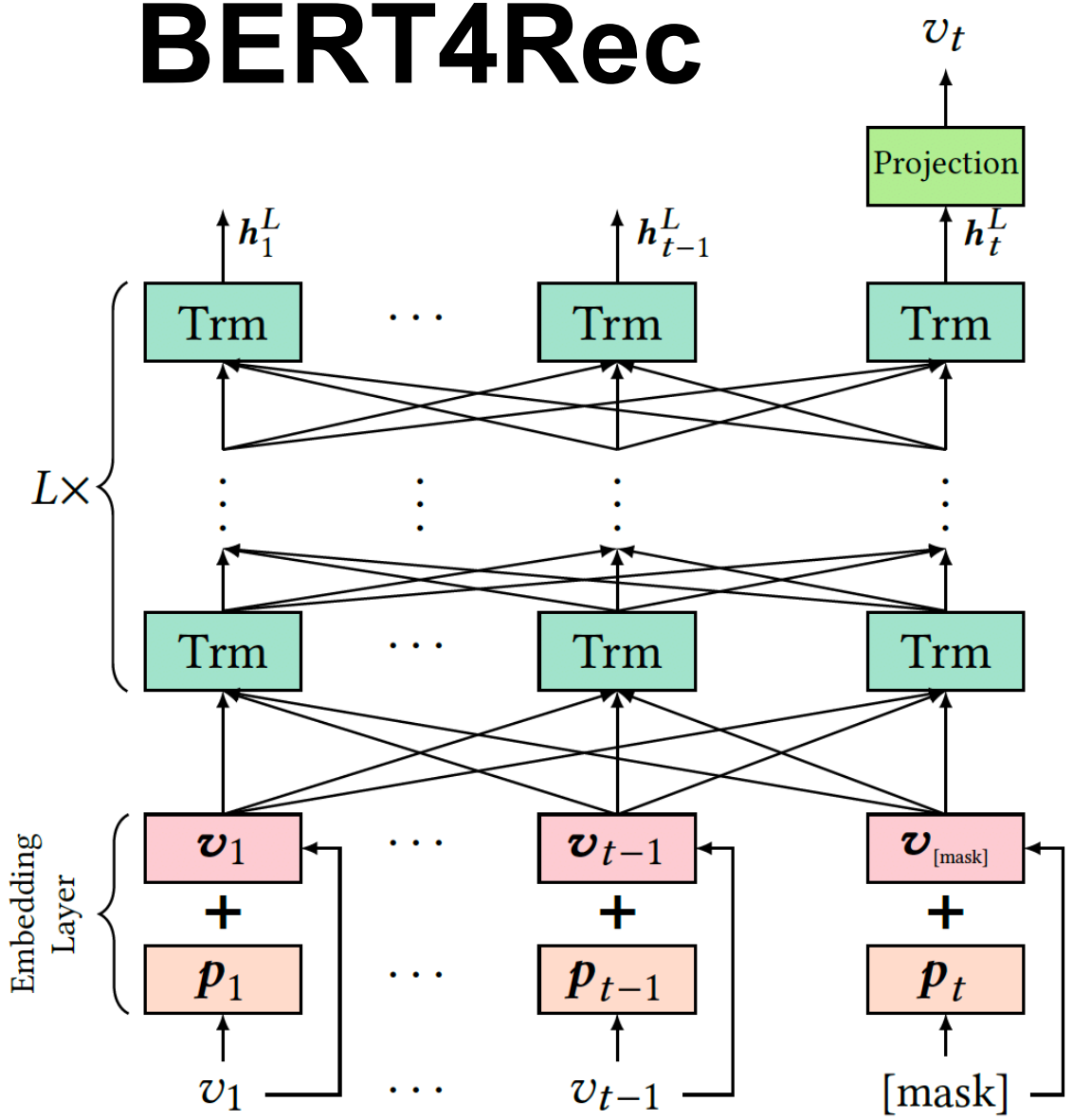


(a) Transformer Layer.

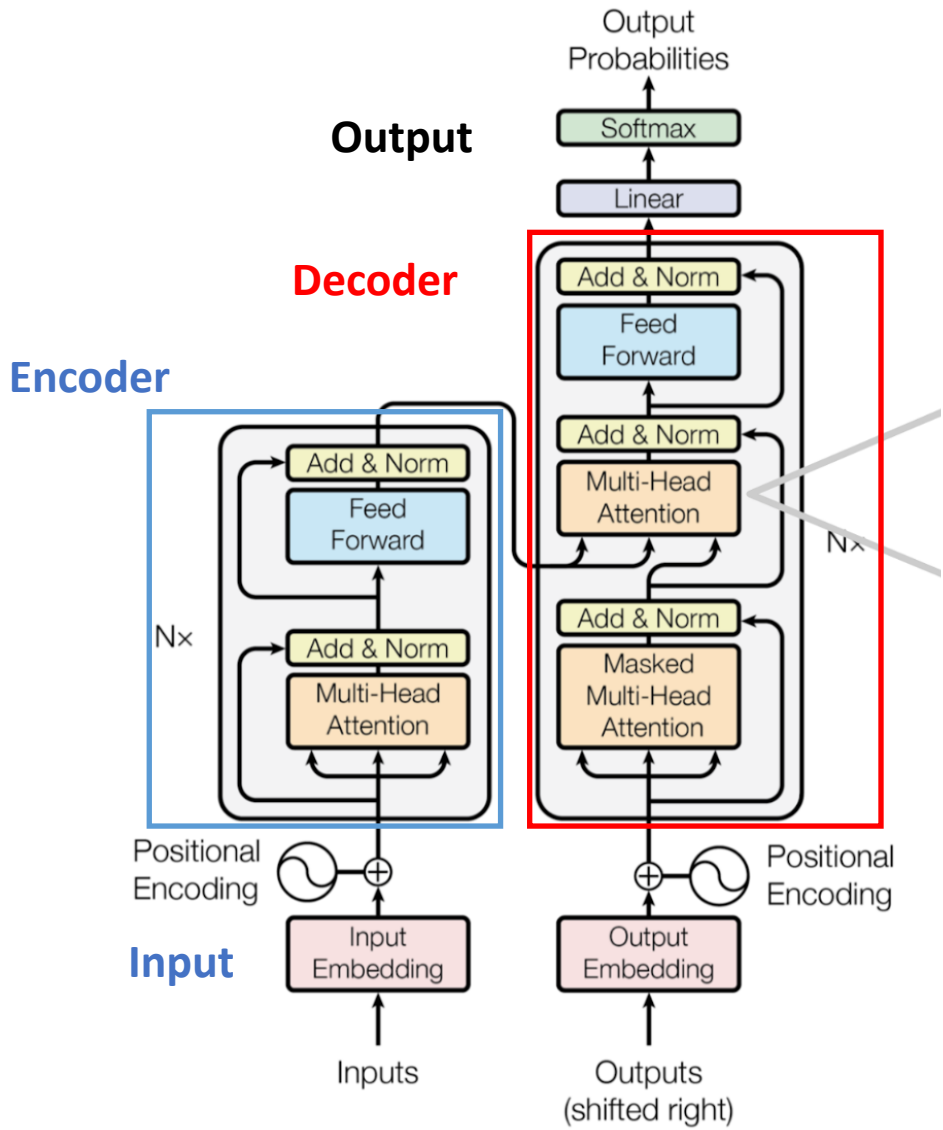


[6] Vasvani et. al. Transformer; Attention is All You Need

BERT4Rec



(b) BERT4Rec model architecture.



[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

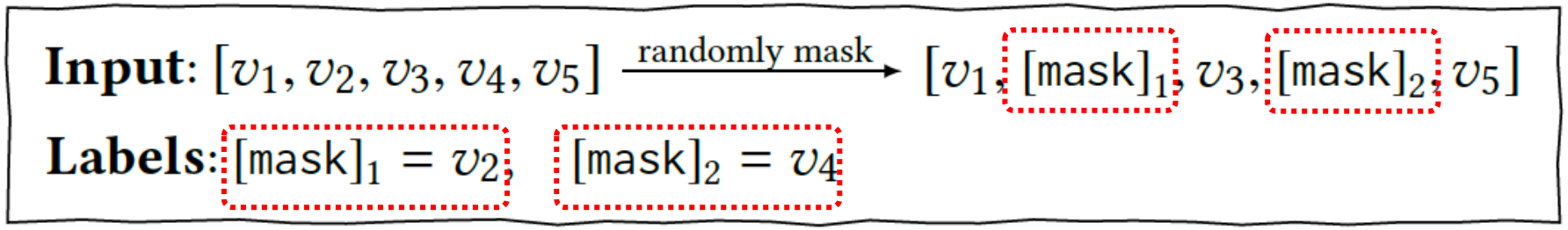
Model Training

Unidirectional Model Training:

- Predict the next item for each position in the input sequence
- The Target of the input sequence is shifted version

Bidirectional Model Training:

- Uses Cloze task – Masked Language Model.
- For each training step, randomly mask ρ proportion of all items in the input sequence (i.e., replace with special token “[mask]”)
- then predict the original ids of the masked items based solely on its left and right context.
- For example:



[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

Model Training

- As in conventional sequential recommendation:
  - The final hidden vectors corresponding to “[mask]” are fed into an output softmax over the item set
- The loss for each masked input S′u as the negative log likelihood of the masked targets:

$$\mathcal{L} = \frac{1}{|\mathcal{S}_u^m|} \sum_{v_m \in \mathcal{S}_u^m} -\log P(v_m = v_m^* | \mathcal{S}_u') \qquad (8)$$

where S'_u is the masked version for user behavior history Su
S^m_u is the random masked items in it
v^*_m is the true item for the masked item v_m
probability is defined as $P(v) = \text{softmax}\big(\text{GELU}(\boldsymbol{h}_t^L \boldsymbol{W}^P + \boldsymbol{b}^P)\boldsymbol{E}^\top + \boldsymbol{b}^O\big)$
GELU: Gaussian Error Linear Unit

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

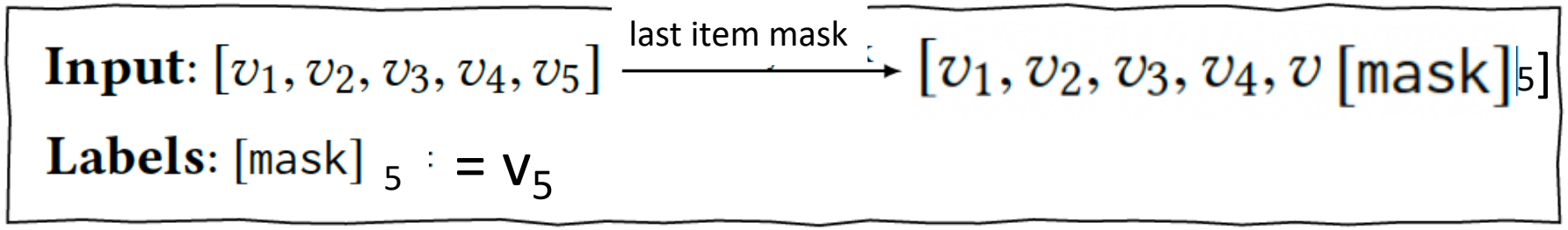
More sample data

- An additional advantage for Cloze task is that it can generate more samples to train the model.
- Assuming a sequence of length n, conventional sequential predictions produce n unique samples for training, while BERT4Rec can obtain $\binom{n}{k}$ samples (if we randomly mask k items).
- This allows us to learn a more powerful bidirectional representation model

[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

Model Testing

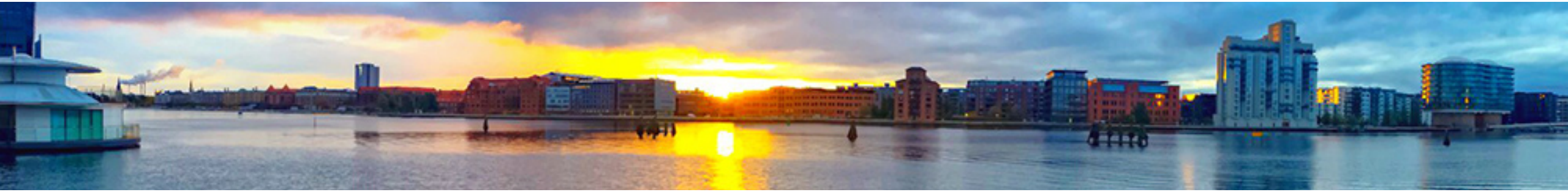
- Mismatch between the training and the final sequential recommendation task since the Cloze objective is to predict the current masked items while sequential recommendation aims to predict the future.
- To address this, append the special token “[mask]” to the end of user’s behaviour sequence, and then predict the next item based on the final hidden representation of this token.
- To better match the sequential recommendation task (i.e., predict the last item), **produce samples that only mask the last item** in the input sequences during training.
- It works like fine-tuning for sequential recommendation and can further improve the recommendation performances.



[8] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

9

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



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

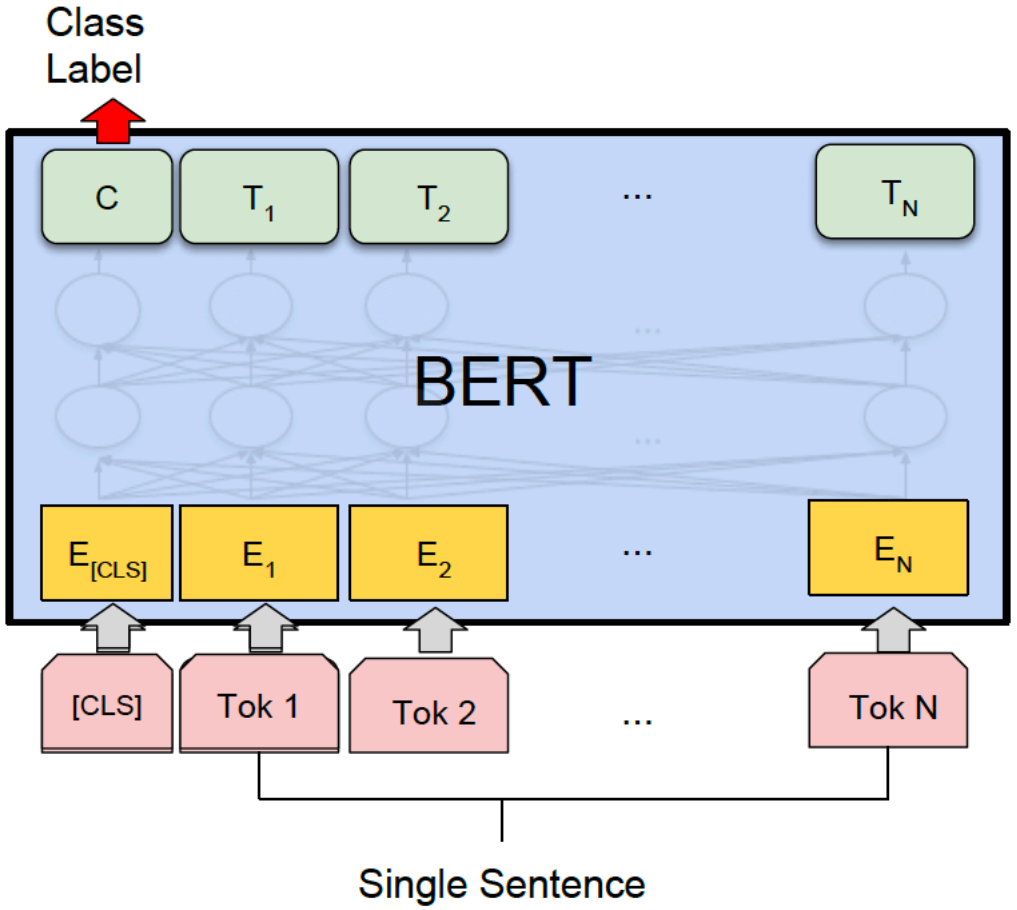
Approaches:

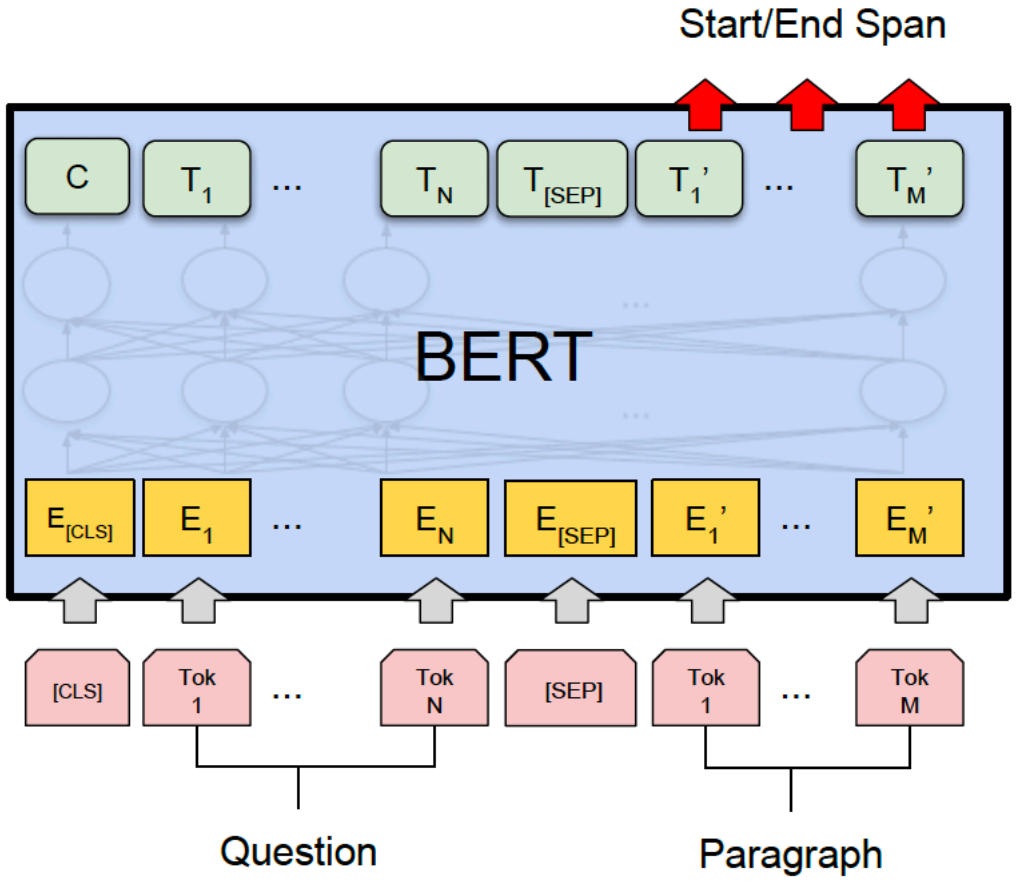
1. Multi-task learning
   - Learn multiple tasks jointly so that knowledge learned in one task can benefit other tasks
- Addresses:
   - Lack of large amount of supervised data
     - Leverage supervised data from many related tasks
   - Overfitting one specific task
     - Multi-task learning gains from regularization effect
     - Learned Representation is universal across tasks
   - Adoption to New Domain with fewer dataset

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**

Relevance Ranking (1/2):

Task: (in the paper)

- *Given a query and a list of candidate answers, the model ranks all the candidates in the order of relevance to query.*
- *Pairwise ranking – rank the candidate that contains the correct answer higher than candidate that does not.*

Relevance Ranking (2/2):

- **QNLI**:
- *Task: Determine whether the context sentence contains the answer to the question.*
  - The Stanford Question Answering Dataset (Rajpurkar et al. 2016) is a question-answering dataset consisting of question-paragraph pairs, where one of the sentences in the paragraph (drawn from Wikipedia) contains the answer to the corresponding question (written by an annotator).
  - We convert the task into **sentence pair classification** by forming a pair between **each question** and **each sentence** in the corresponding context, and filtering out pairs with low lexical overlap between the question and the context sentence.

SNLI:

- **SNLI:**
- SNLI The Stanford Natural Language Inference (SNLI) dataset contains 570k human annotated sentence pairs, in which the premises are drawn from the captions of the Flickr30 corpus and hypotheses are manually annotated (Bowman et al., 2015b).
- This is the most widely used entailment dataset for NLI.
- The dataset is used only for domain adaptation in this study.

Used for Domain Adaptation.

Premise: A boat sank in the Pacific Ocean.

Hypothesis: A boat sank in the Atlantic Ocean.

Label: Contradiction

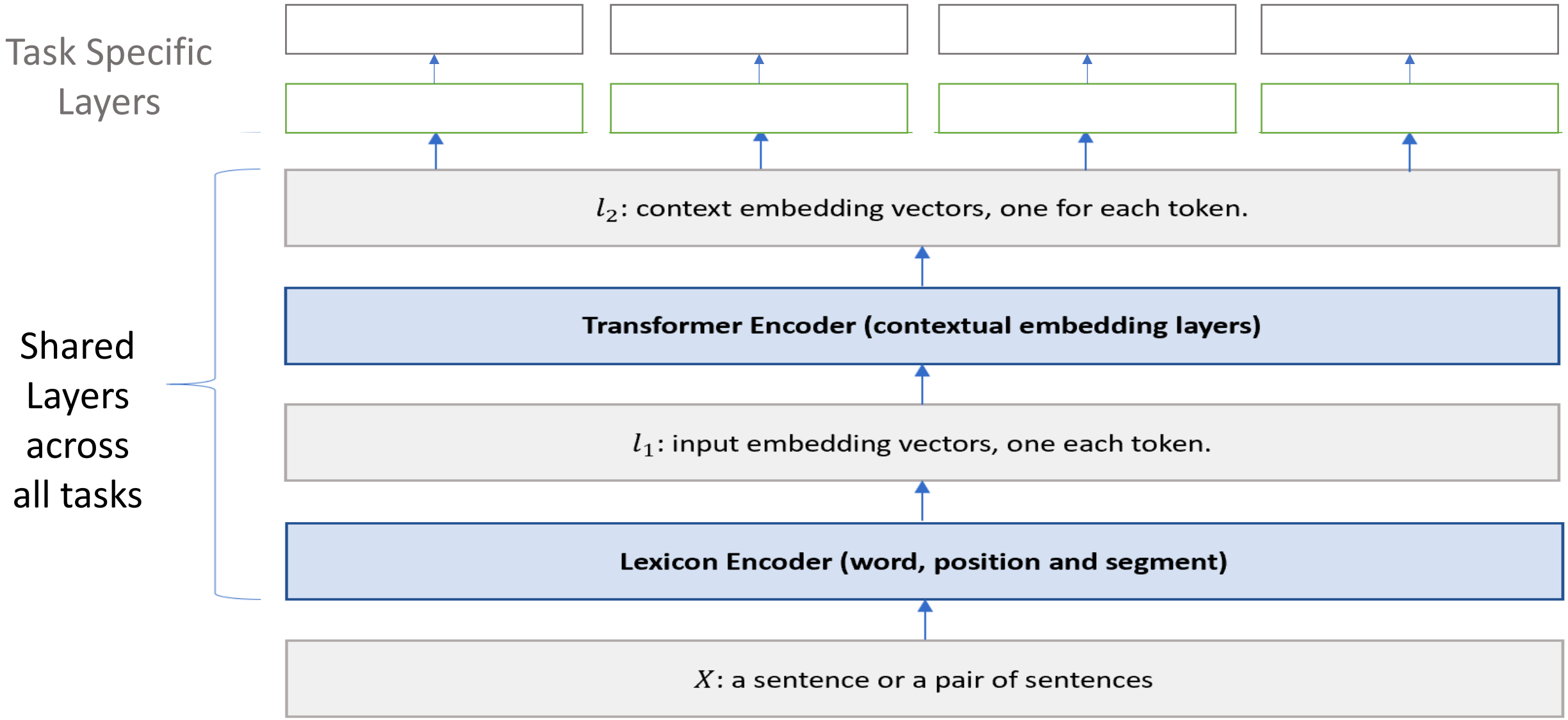
Premise: A boat sank in the Pacific Ocean.

Hypothesis: A boat sank in the Atlantic Ocean.

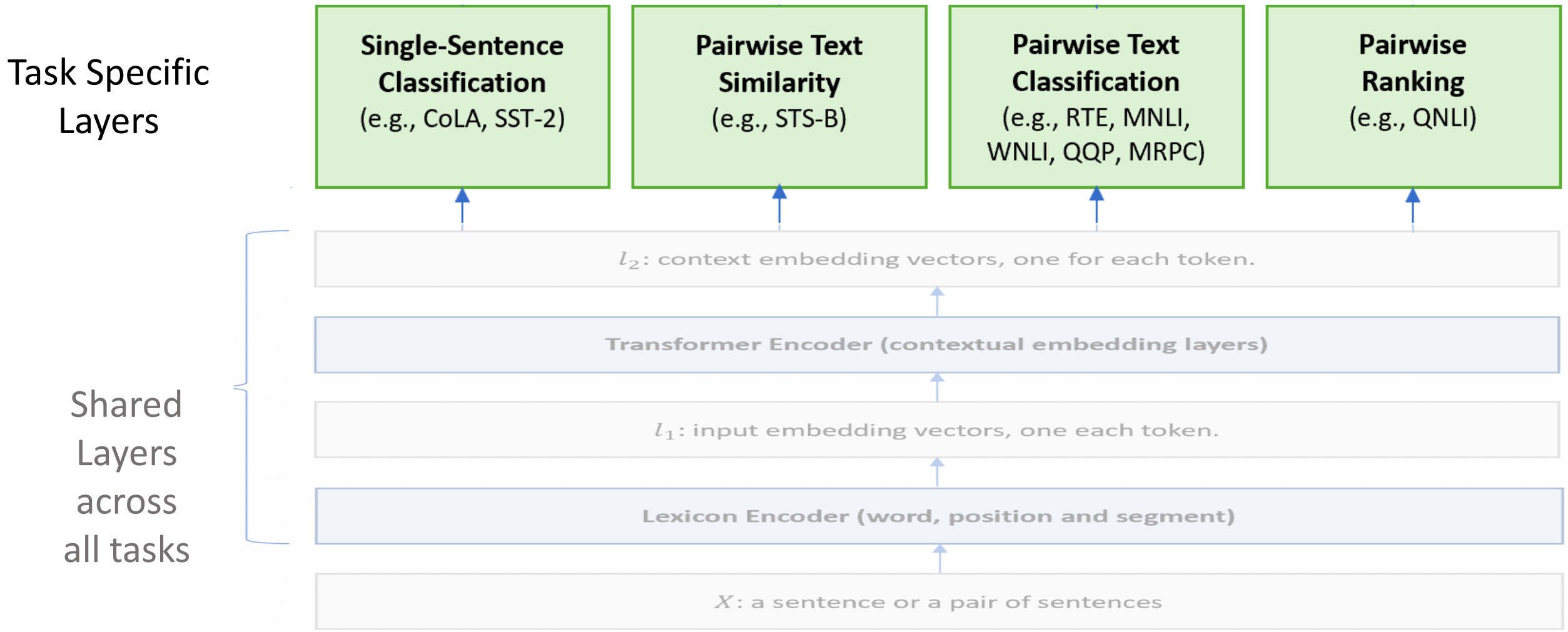
Label: Neutral (two different events)

No contradictions at all in typical natural sentences.

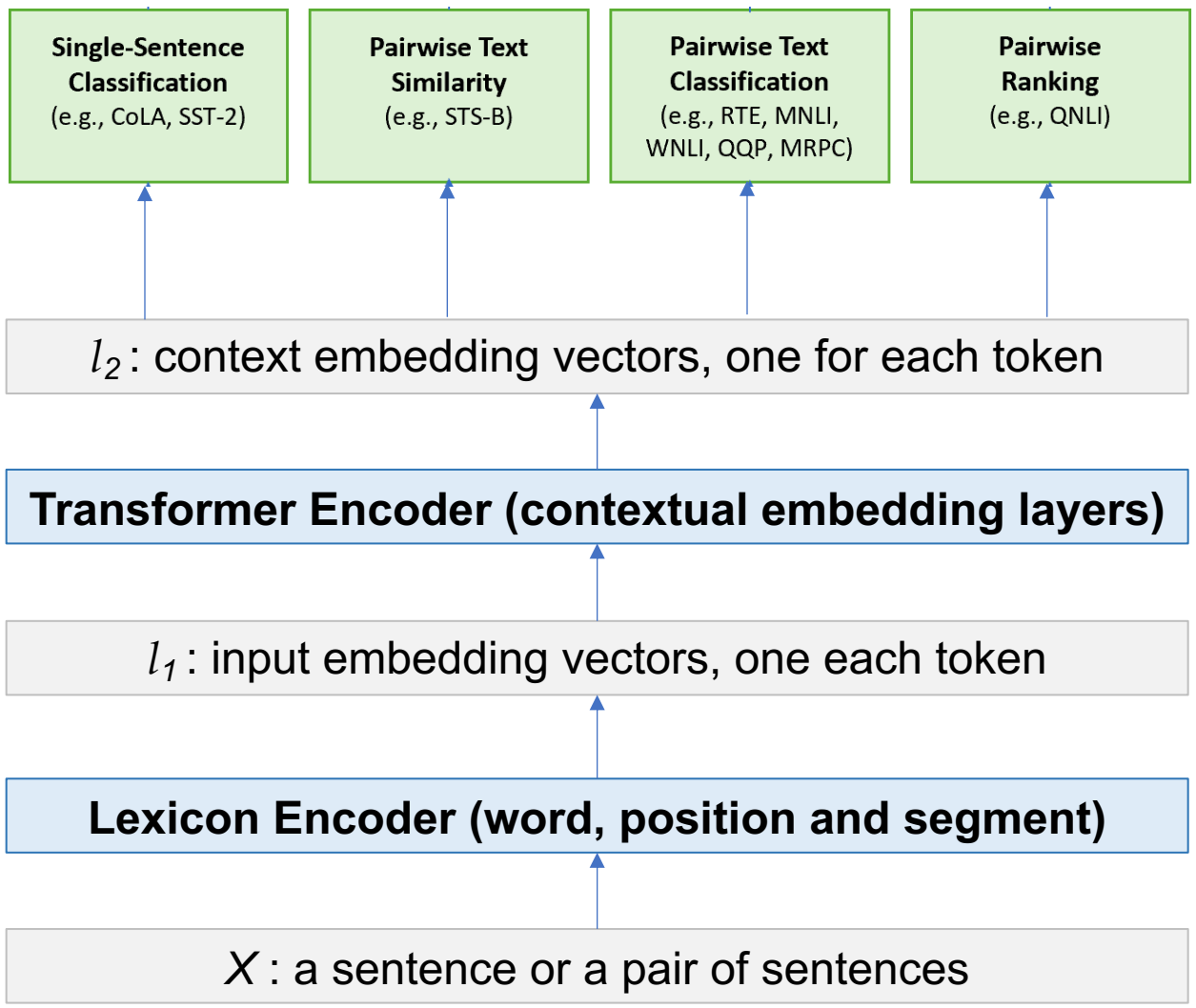
Architecture of MT-DNN Model



Architecture of MT-DNN Model



MT-DNN Model Layers



Operations necessary for classification, similarity scoring, or relevance ranking

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

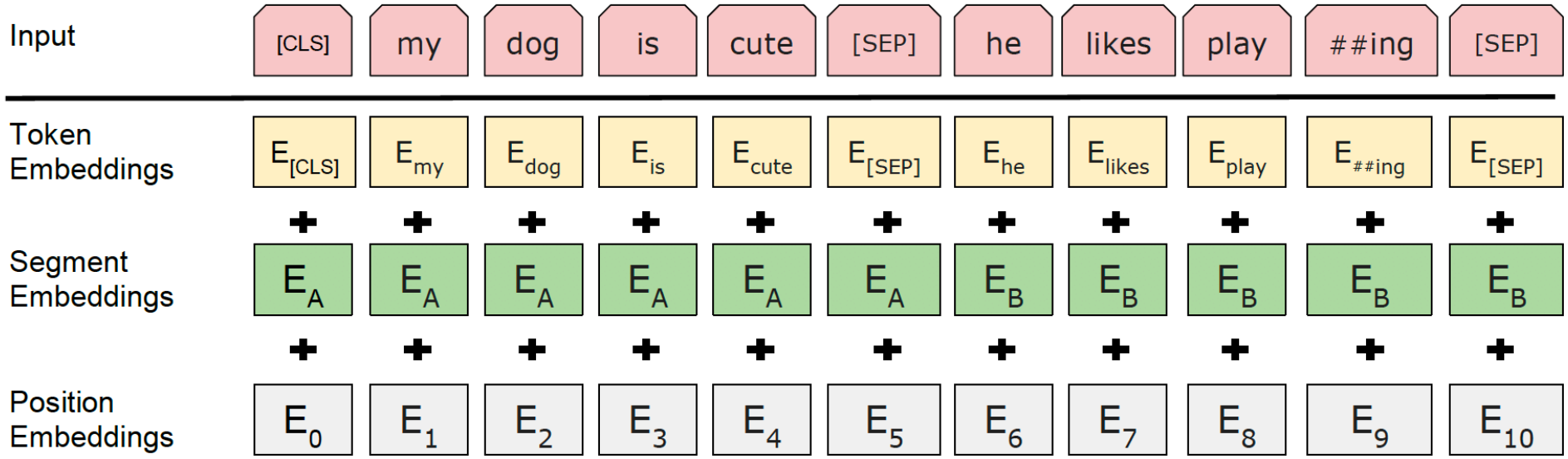
Shared semantic representation that is trained by multi-task objectives

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 $l1$

Lexicon Encoder ($l1$):



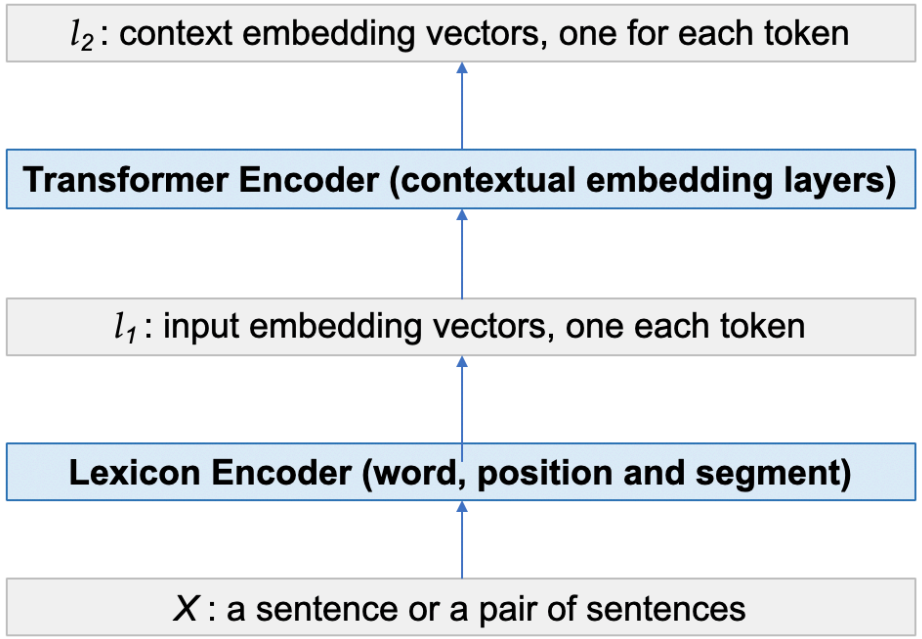
Single Sentence:
The input X = f { x_1; . . . ; x_m } is a sequence of tokens of length m.
First token x_1 is always the [CLS] token.

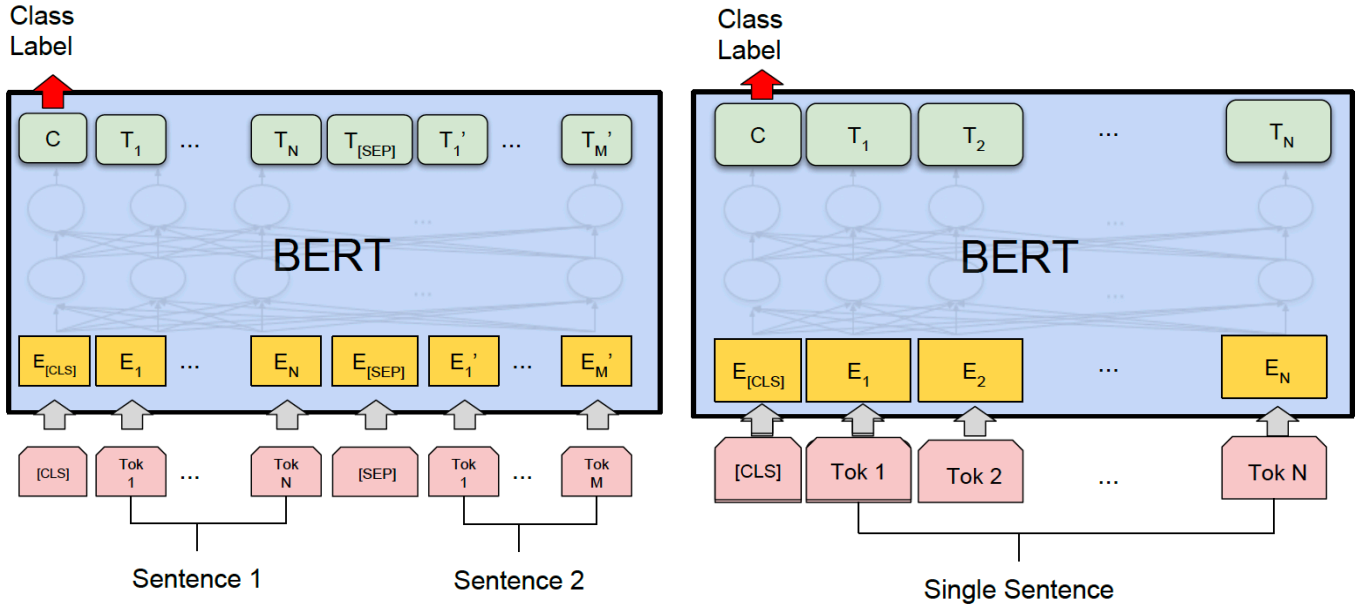
Sentence Pair:
Separate two X_1 and X_2 sentences with a special token [SEP]

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

- the corresponding word
- segment and
- positional embeddings

Transformer Encoder ($l2$):





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

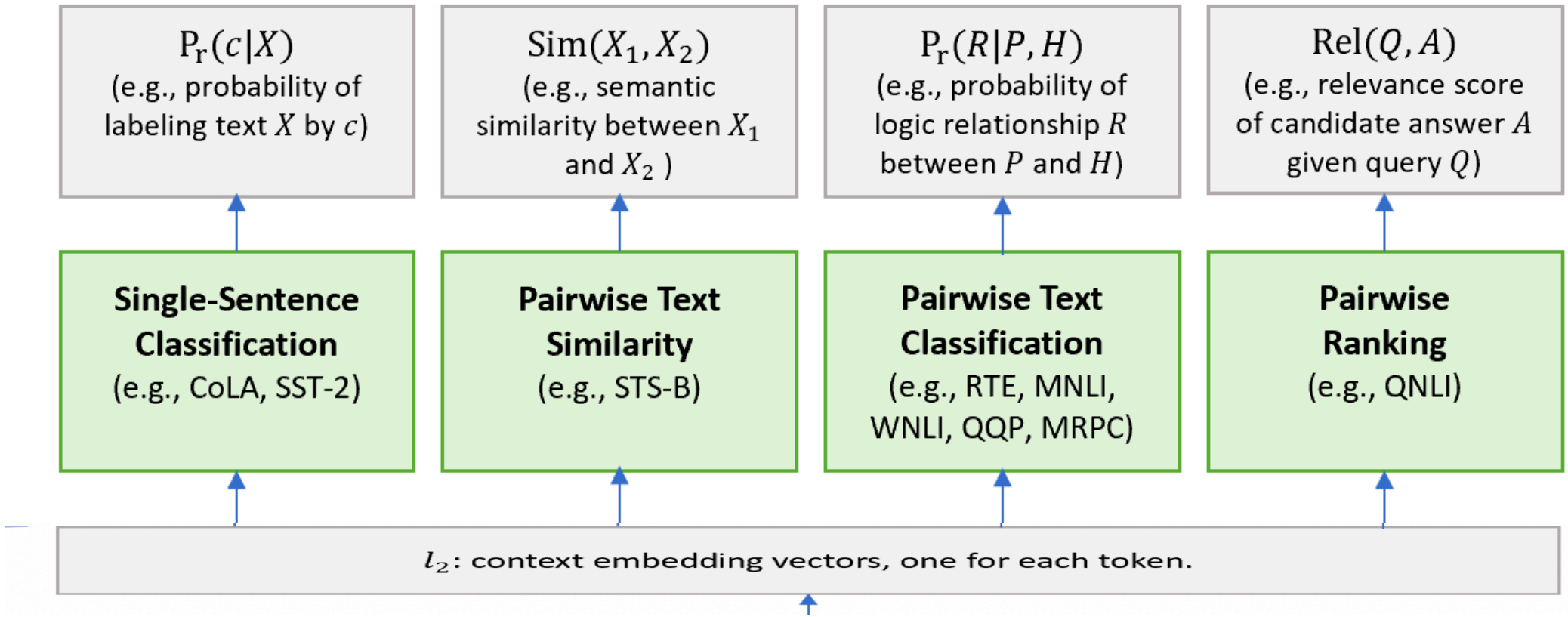
(b) Single Sentence Classification Tasks: SST-2, CoLA

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.

4 NLU tasks


$\mathrm{P_r}(c|X)$
(e.g., probability of labeling text X by c)
$\mathrm{Sim}(X_1, X_2)$
(e.g., semantic similarity between X_1 and X_2)
$\mathrm{P_r}(R|P, H)$
(e.g., probability of logic relationship R between P and H)
$\mathrm{Rel}(Q, A)$
(e.g., relevance score of candidate answer A given query Q)
Single-Sentence Classification
(e.g., CoLA, SST-2)
Pairwise Text Similarity
(e.g., STS-B)
Pairwise Text Classification
(e.g., RTE, MNLI, WNLI, QQP, MRPC)
Pairwise Ranking
(e.g., QNLI)
l_2: context embedding vectors, one for each token.

1 Single Sentence Classification:

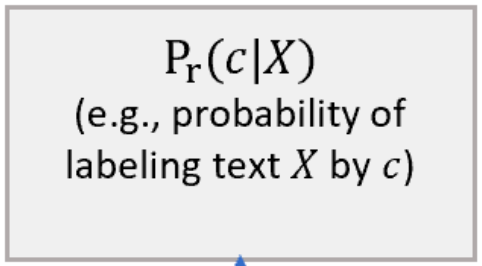
$$P_r(c|X) = \text{softmax}(\mathbf{W}_{SST}^{\top} \cdot \mathbf{x}), \qquad (1)$$

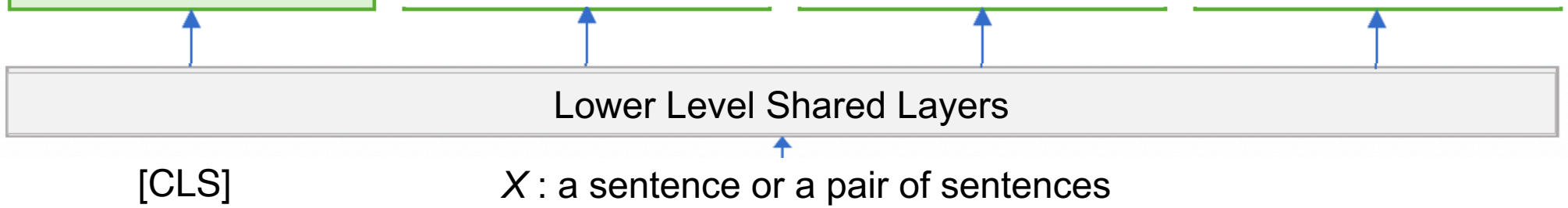
class (i.e., the sentiment)

input sentence

task-specific parameter

contextual embedding of token [CLS]





2 Pairwise Text Similarity

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

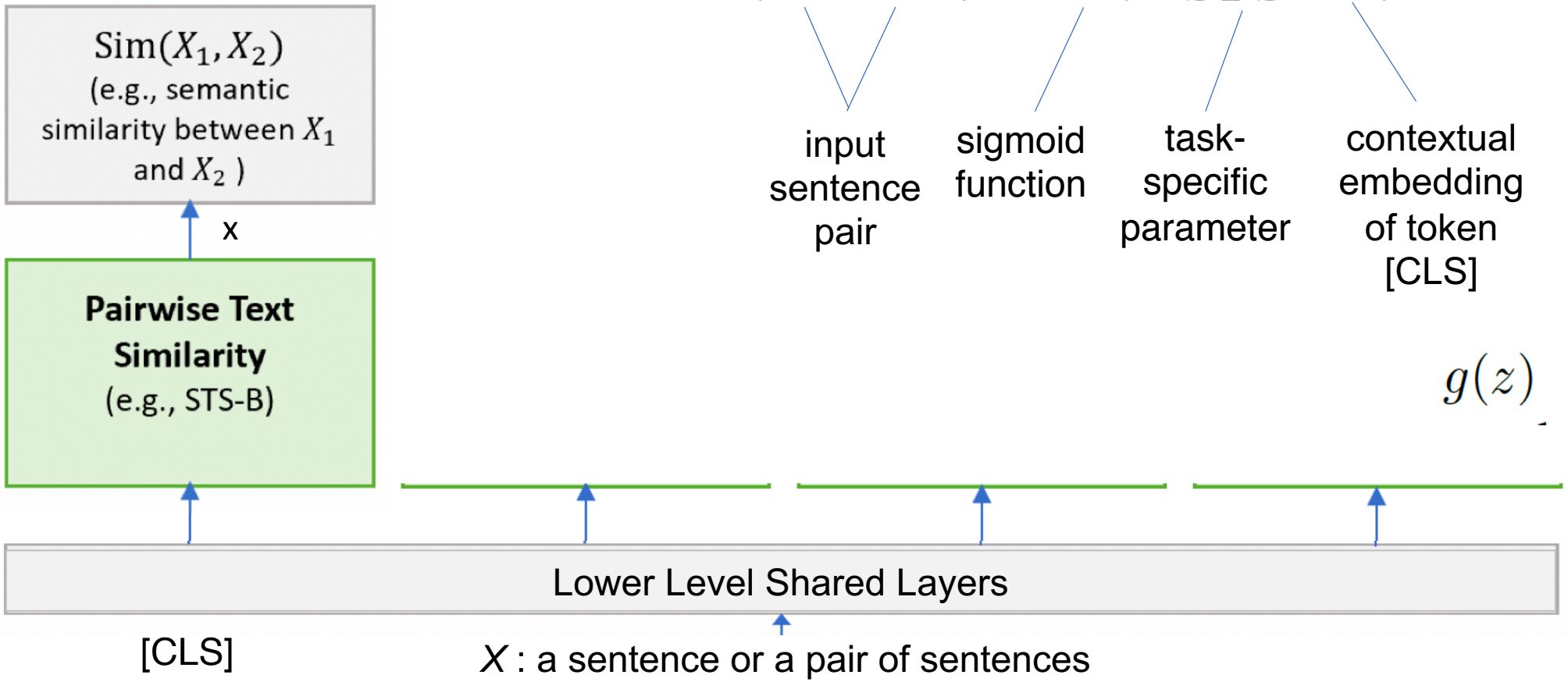
input sentence pair

sigmoid function

task-specific parameter

contextual embedding of token [CLS]

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



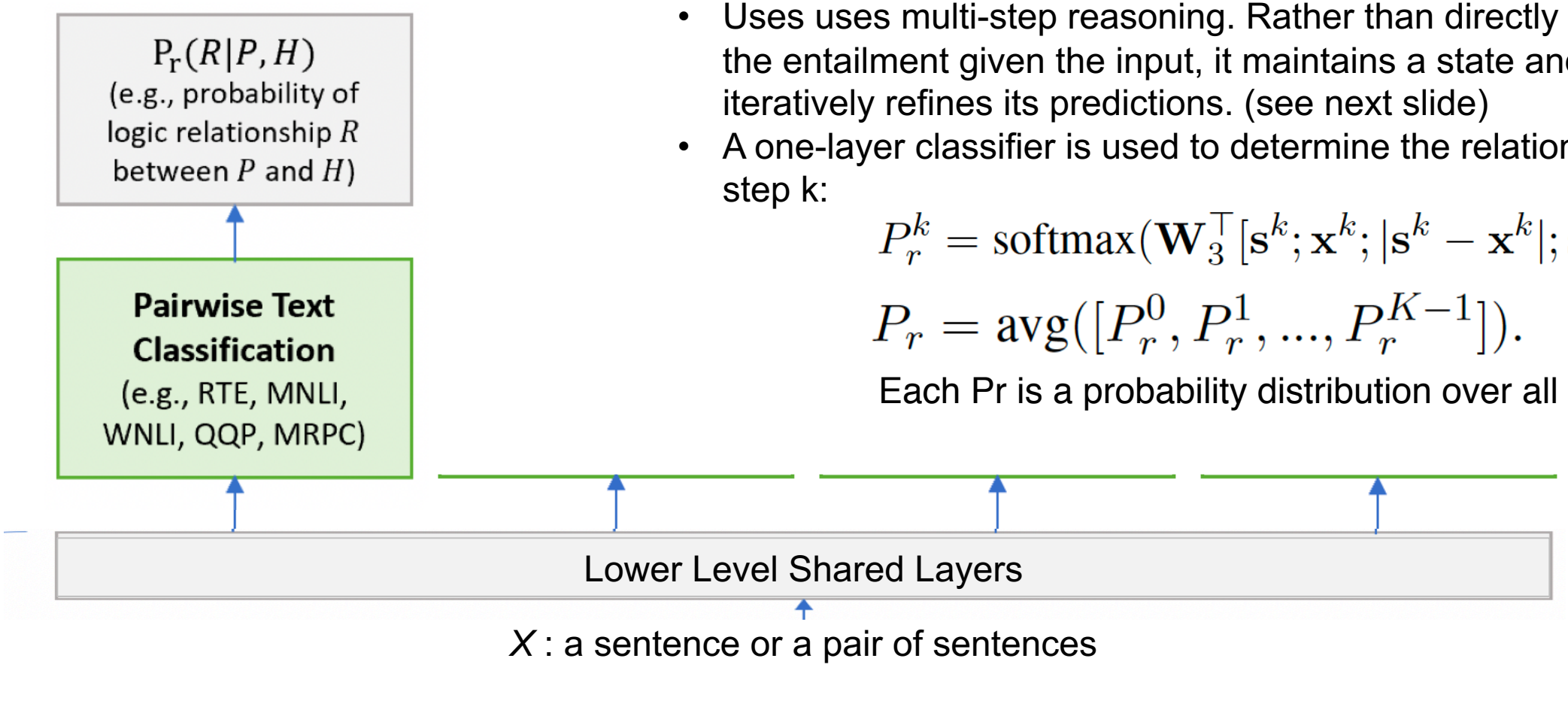
3 Pairwise Text Classification

- 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:

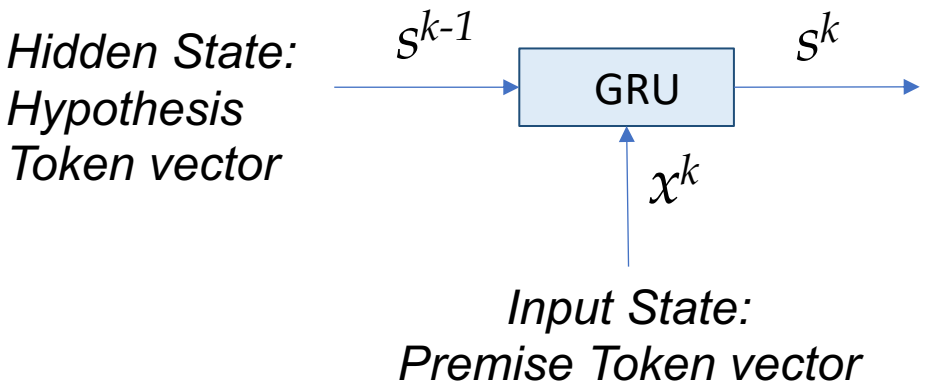
$$P_r^k = \text{softmax}(\mathbf{W}_3^\top [\mathbf{s}^k; \mathbf{x}^k; |\mathbf{s}^k - \mathbf{x}^k|; \mathbf{s}^k \cdot \mathbf{x}^k]).$$

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

Each Pr is a probability distribution over all the relations R



Stochastic Answer Network Module (Ref)



- Construct the working memory of premise P by concatenating the contextual embeddings of the words in P, which are the output of the transformer encoder, denoted as M^p
- Similarly the working memory of hypothesis H, denoted as M^h
- Perform K-step reasoning on the memory to output the relation label, where K is a hyperparameter.
- Initial state s^0 is the summary of M^h:

$$\mathbf{s}^0 = \sum_j \alpha_j \mathbf{M}_j^h, \text{ where } \alpha_j = \frac{\exp(\mathbf{w}_1^\top \cdot \mathbf{M}_j^h)}{\sum_i \exp(\mathbf{w}_1^\top \cdot \mathbf{M}_i^h)}.$$

- At time step k in the range of {1, 2,…,K-1 }, the state is defined by $s^k = GRU(s^{k-1}, x^k)$.
- x^k is computed from the previous state s^{k-1} and memory M^p:

$$\mathbf{x}^k = \sum_j \beta_j \mathbf{M}_j^p$$

$$\beta_j = \text{softmax}(\mathbf{s}^{k-1}\mathbf{W}_2^\top \mathbf{M}^p)$$

- A one-layer classifier is used to determine the relation at each step k:

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

- Utilize all of the K outputs by averaging the scores:

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

4 Pairwise Ranking

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

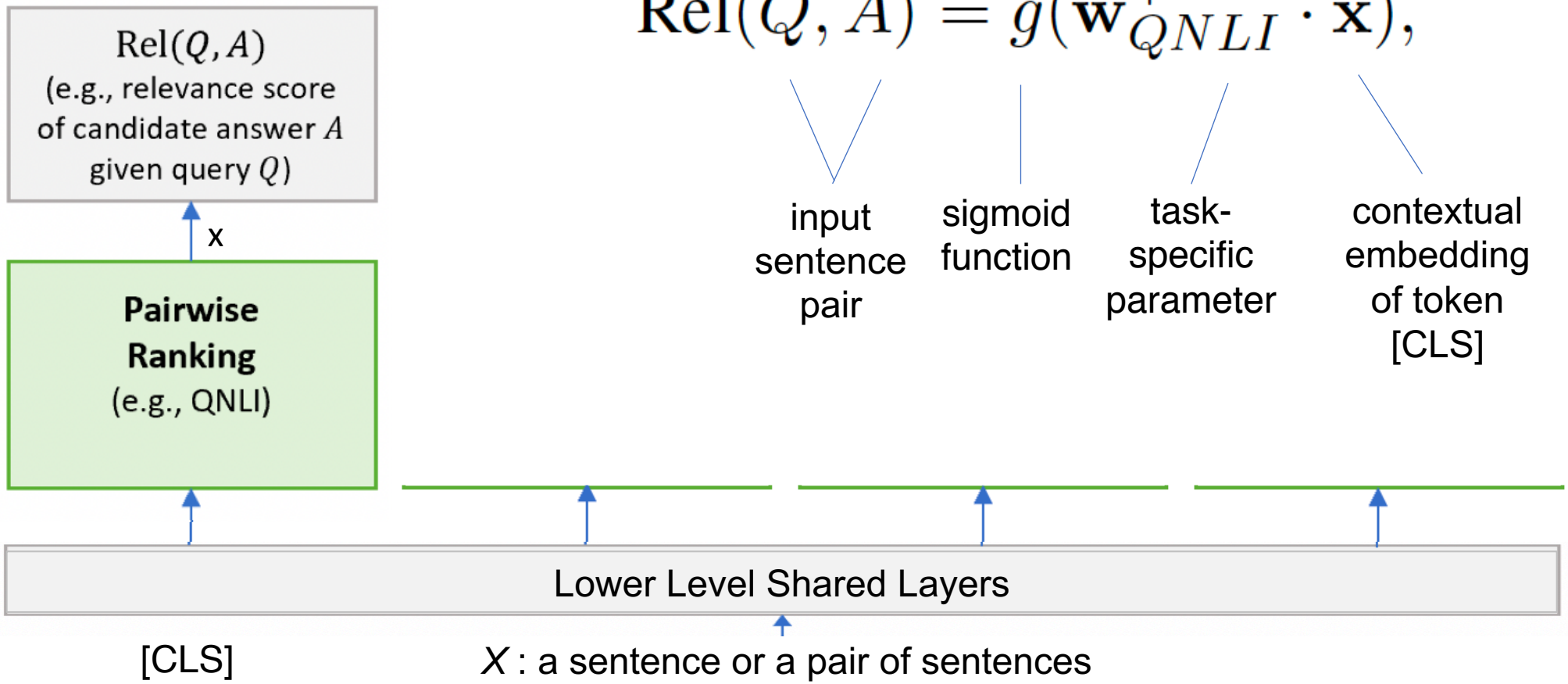
$$\mathrm{Rel}(Q, A) = g(\mathbf{w}_{QNLI}^{\top} \cdot \mathbf{x}), \quad (5)$$

input sentence pair

sigmoid function

task-specific parameter

contextual embedding of token [CLS]



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:

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

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:

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

where Sim(.) is defined by Equation 2.

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

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

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 |A|-1 negative examples.
- Minimize the negative log likelihood of the positive example given queries across the training data

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

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

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.

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

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

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

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

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

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

REFERENCES

REFERENCES

Session 1:

[1] Sebastian Ruder. 2017 An Overview of Multi-Task Learning in Deep Neural Networks.

[2] Jia Dong Zhang et al. SEMAX: Multi-Task Learning for Improving Recommendations IEEE-2018

[3] Yichao Lu at el. Why I like it: Multi-task Learning for Recommendation and Explanation. RecSys-2018

[4] Chen Gao et al. Neural Multi-Task Recommendation from Multi-Behavior Data. ICDE-2017

[5] Yujie Lin et al,. Explainable Outfit Recommendation with Joint Outfit Matching and Comment Generation. 2018

Session 2:

[6] Fei Sun, et. al. BERT4Rec- Sequential Recommendation with Bidirectional Encoder Representations from Transformer. WOODSTOCK 2019

[7] Vasvani et. al. Transformer – Attention is All You Need

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

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

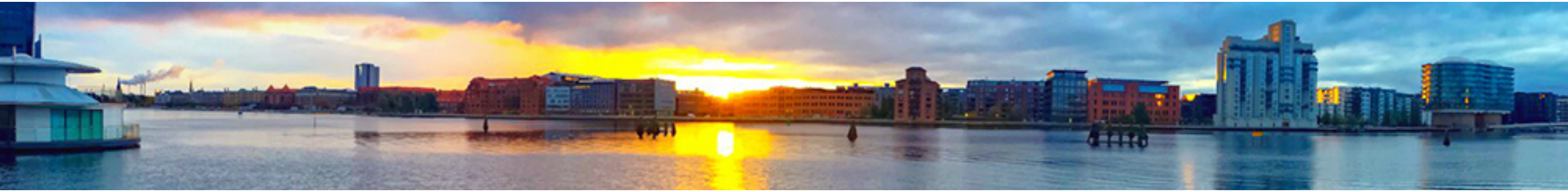
The ACM Conference Series on
Recommender Systems

Copenhagen, Denmark Sept 16-20, 2019

Thank You.

www.DeepThinking.AI

omsonie@gmail.com



Code- Notebook

Ref: