SONY

Recommendation Systems and Big Data

Brijraj Singh
Senior Research Scientist
Sony Research India (Sony AI)

Outline:

- Recommendation Systems
  - ✓ Collaborative Filtering
  - ✓ Content Similarity
  - ✓ Neural Model based
- Fundamental of creating Representations through NLP
- Transformer Architecture
  - ✓ Self Attention
  - ✓ Positional Encoding
  - ✓ Layer Normalization
- Applications
- Session Based Recommendation
- Cold Start Problem and Meta Learning
- LLM in Recommendations
- Agents for Recommendation

Recommendation Systems

Basic of RecSys:

Interaction matrix

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

- A user consumes the content from the online platform which could be e-commerce (Amazon, Flipkart), music (Spotify, you tube), video (Netflix, Hotstar, SonyLiv).
- Users provide feedback either Directly (through ratings) or Indirectly (consumption ratio).
- Platform keeps huge list of items and each user interacts just a small fraction of total items.
- Hence, user item interaction matrix is extremely sparse in nature.
- Any recommendation model is employed to accurately fill the sparse entries of interaction matrix.

Basic of RecSys:

- A user consumes the content from the online platform which could be e-commerce (Amazon, Flipkart), music (Spotify, you tube), video (Netflix, Hotstar, SonyLiv).
- Users provide feedback either Directly (through ratings) or Indirectly (consumption ratio).
- Platform keeps huge list of items and each user interacts just a small fraction of total items.
- Hence, user item interaction matrix is extremely sparse in nature.
- Any recommendation model is employed to accurately fill the sparse entries of interaction matrix.

Interaction matrix

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

Train: If users set is U and item set is I and if we reserve the test data as unseen data and train the model using training data $U_{tr}, I_{tr} \Rightarrow M(U_{tr}, I_{tr})$.

Test: $M(U_{ts}, I_{ts}) \Rightarrow R$,

Evaluation: $MAE = Performance(R, R_{ts})$

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

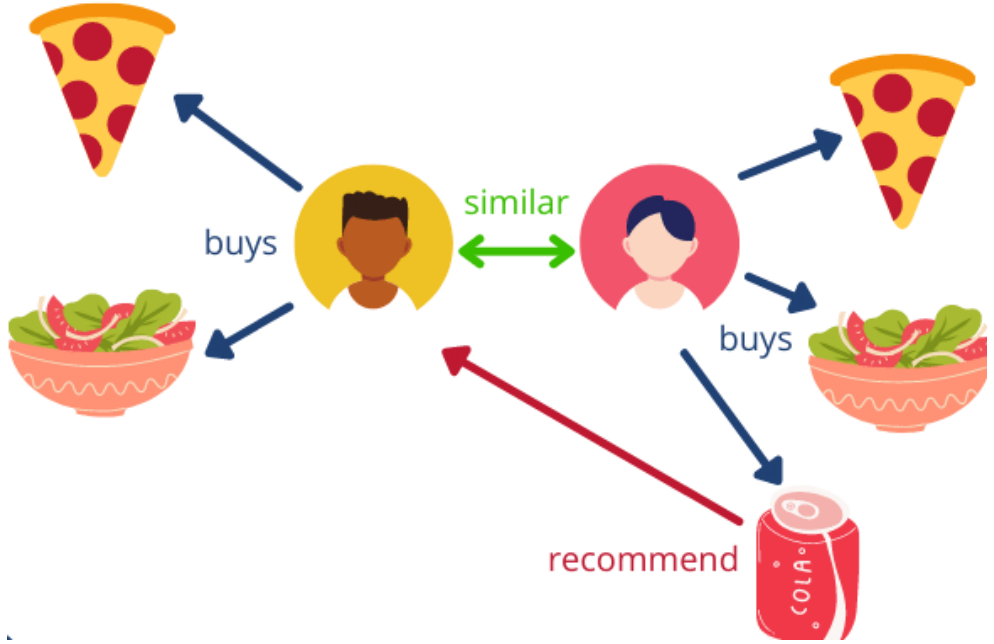
$M(user2, item2)$
$M(user2, item3)$
$M(user2, item5)$

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	$R1$	3	$R2$	3	4
User 2	4	$R3$	$R4$	2	$R5$
User 3	$R6$	$R7$	3	$R8$	$R9$
User 4	3	$R10$	4	$R11$	3
User 5	4	3	$R12$	4	4

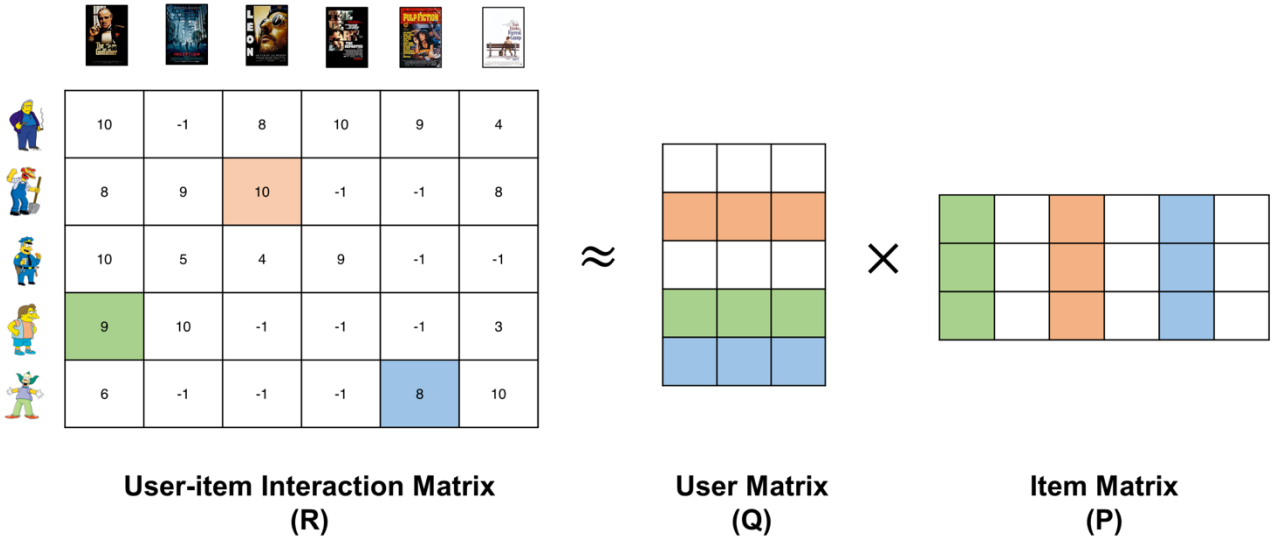
What to recommend

?

Collaborative Filtering



Matrix Factorization: (SVD)



Step 1: Start with the User–Item Matrix:

- Let's say you have a matrix **R** of size m × n:
  m = number of users
  n = number of items

- Each cell R[u, i] is the **rating** that user *u* gave to item *i*.
  If no rating exists → the cell is empty (or negative).

Step 2: Matrix Factorization (SVD Concept)

- The **Singular Value Decomposition (SVD)** decomposes `R` into three matrices:

$$R = U \Sigma V^T$$

Where:

U (m × k) → user latent features

Σ (k × k) → diagonal matrix of singular values (importance of each latent factor)

V^T (k × n) → item latent features

- Usually, we only keep the **top k** singular values (and corresponding vectors) to reduce dimensionality and remove noise.

This gives us an approximate reconstruction:

$$R' = U_k \, \Sigma_k \, V_k^T$$

Now, **R′** contains estimated values for all missing entries — the predicted ratings!

Step 3: Intuition

- Each user is represented as a k-dimensional vector (in U).
- Each item is represented as a k-dimensional vector (in V).
- The dot product between user and item vectors gives the predicted rating:

$$\hat{\boldsymbol{r}}_{\boldsymbol{ui}} = (\boldsymbol{U}_{\boldsymbol{k}}\boldsymbol{\Sigma}_{\boldsymbol{k}}^{\boldsymbol{1/2}})_{\boldsymbol{u}} \cdot (\boldsymbol{V}_{\boldsymbol{k}}\boldsymbol{\Sigma}_{\boldsymbol{k}}^{\boldsymbol{1/2}})_{\boldsymbol{i}}$$

So, the model learns *hidden factors* (like genre preference, price sensitivity, etc.) that explain the observed ratings.

Step 4: How It's Trained

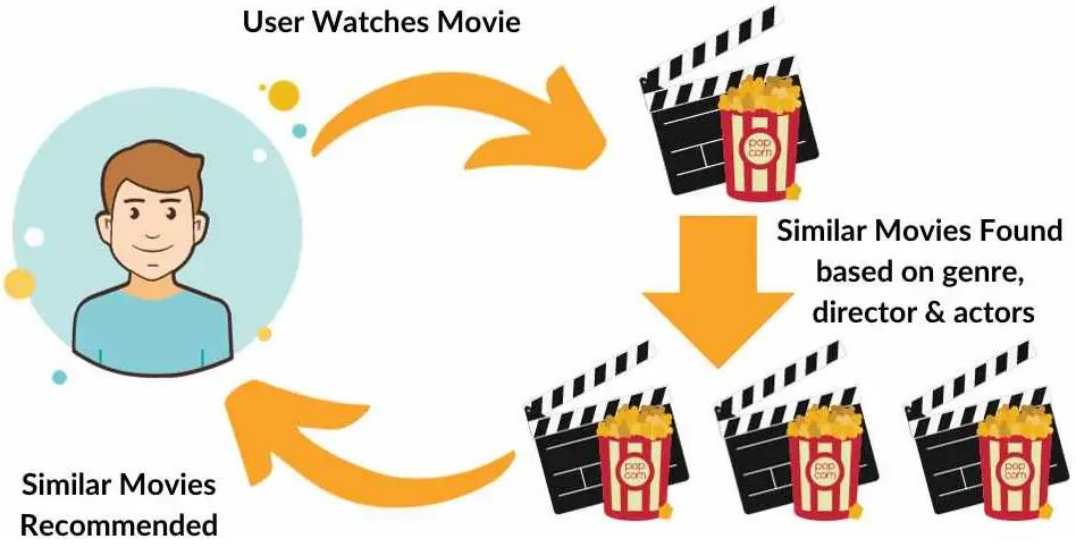
- In real recommender systems, we usually don't compute SVD directly (because R is sparse).

$$\min_{P,Q} \sum_{(u,i)\in K} (R_{ui} - P_u^T Q_i)^2 + \lambda(||P_u||^2 + ||Q_i||^2)$$

- Instead, we **optimize the latent factors** by minimizing the error between known ratings and predicted ratings. Where:

  P= user feature matrix (m × k)
  Q= item feature matrix (n × k)
  λ= regularization parameter (to prevent overfitting)
  K= set of known (user, item) pairs

Content Similarity



1. Compute Similarity Between Items

- The model computes **content similarity** between items using a similarity metric such as:

$$\text{similarity}(i, j) = \frac{V_i \cdot V_j}{\| V_i \| \quad \| V_j \|}$$

- That's **cosine similarity**, which measures how close two feature vectors point in direction (independent of magnitude). 1 → identical, 0 → no similarity, -1 → opposite.

2. Build User Profile

For each user, we aggregate the features of items they have liked or rated highly.

$$U_u = \frac{1}{| L_u |} \sum_{i \in L_u} V_i$$

where:
L_u =items liked by user u, V_i =feature vector of item i

3. Recommend New Items

Now for each new item j:

$$\text{score}(u, j) = \mathit{similarity}(U_u, V_j)$$

Recommend the items with highest similarity scores that the user hasn't seen yet.

Correspondence between both domain

NLP	Vs	RecSys

Wiki _id# 12345	user _id# 145
In modern physics, the double-slit experiment demonstrates that light and matter can exhibit…	<item1, item2, item3, …itemk>

Sentences are formed by sequences of words	Interaction history is created using sequences of clicks

GRU ⟷ GRU4Rec

BERT ⟷ BERT4Rec

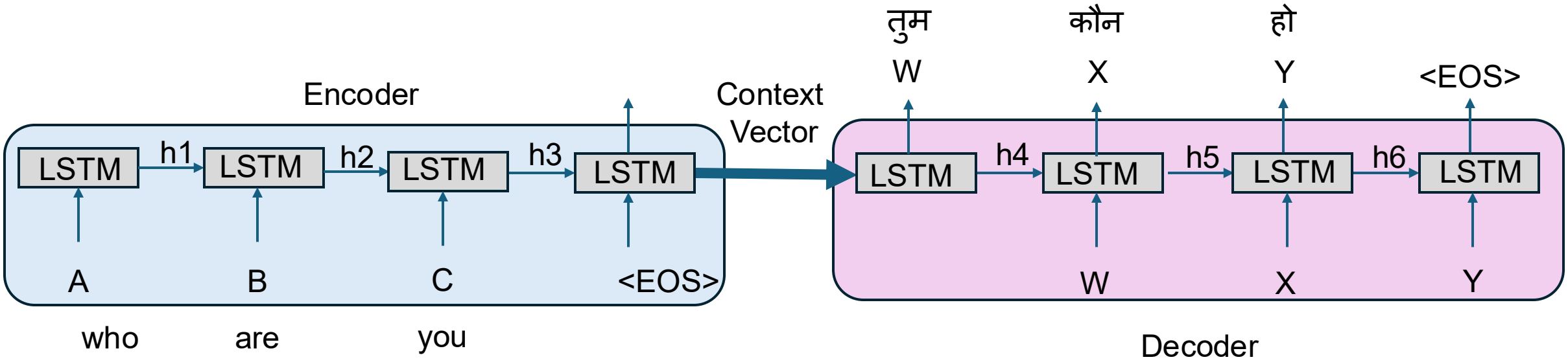
Transformer ⟷ Transformer4Rec

LLM ⟷ LLM4Rec ?

Year: 2014

Creating Representation:

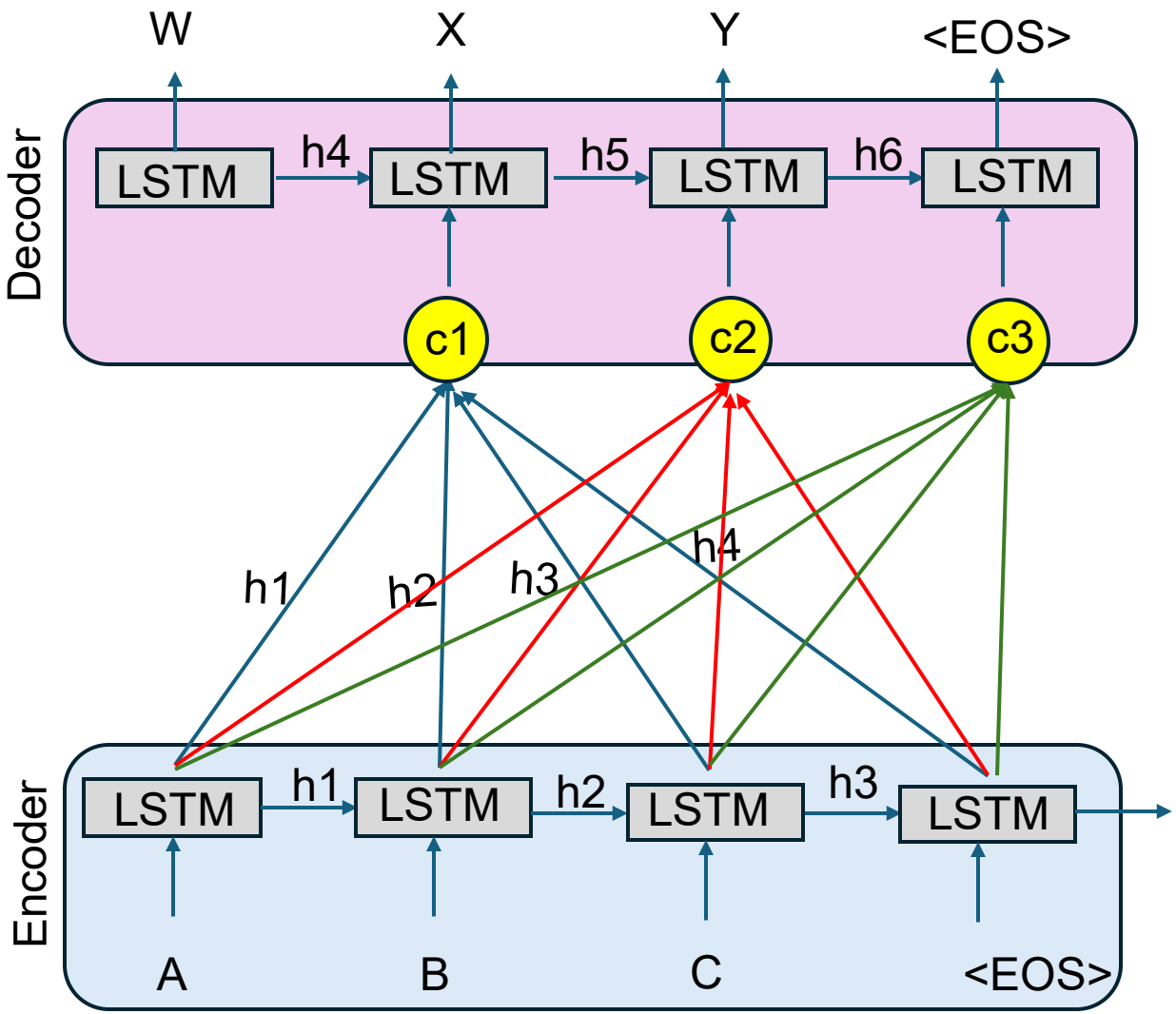
Sequence to Sequence Learning *A case of Machine Translation*



- Context vector plays an extremely important role of transferring summary from *Encoder to Decoder.*
- What If sentence length is large?

Sutskever, I. (2014). Sequence to Sequence Learning with Neural Networks. arXiv preprint arXiv:1409.3215.

Machine Translation by Jointly Learning to Align and Translate



Introducing Attention

Sequential Training

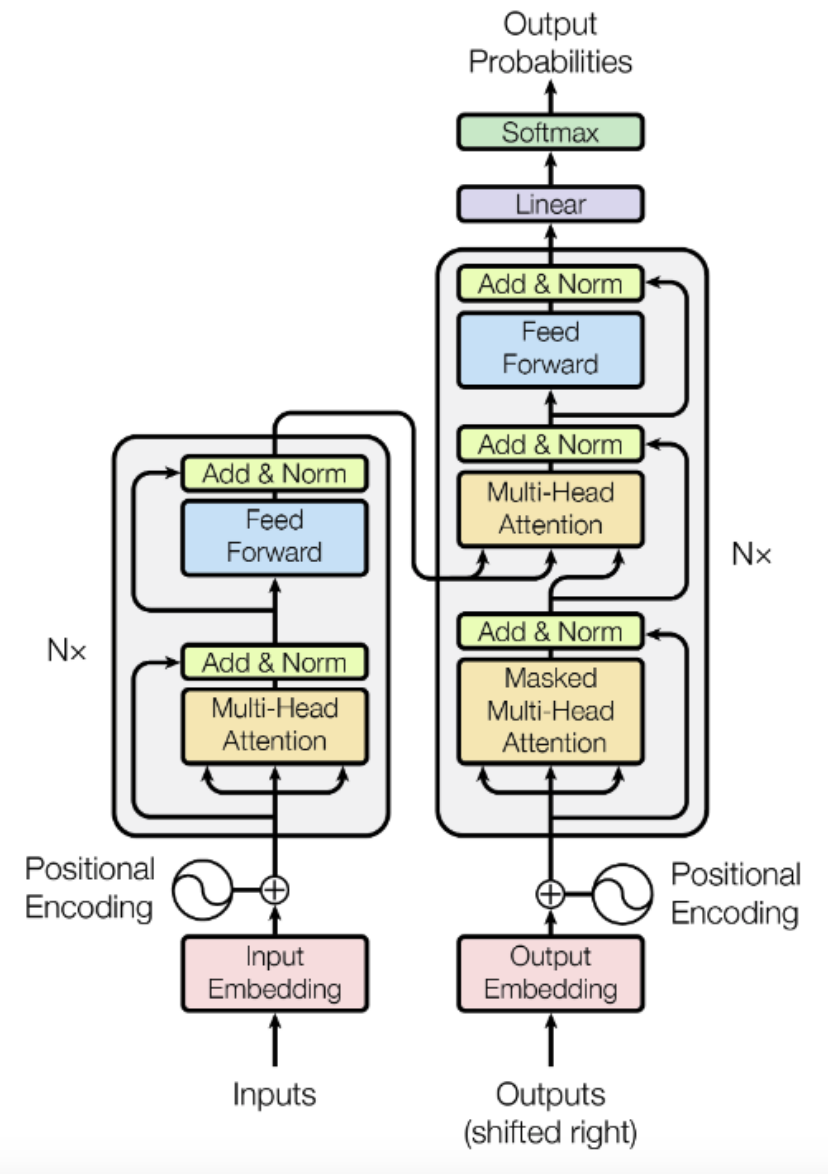
Good for Long Sentences

Bahdanau, D. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Year: 2017

The Paradigm Shift

Attention is All You Need: Transformer



- No RNN Block so NO sequential Learning.
- Suitable for large sentences.
- Operations are Parallel
- Supports Multimodality

- Self Attention
- Positional Encoding
- Multi Head Attention
- Layer Normalization

Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

The requirements of Self Attention

One hot encoding: “This is cat”

Vocab size

This	0	1	0	0	0
Is	1	0	0	0	0
cat	0	0	1	0	0

The requirements of Self Attention

One hot encoding: "This is cat"

Vocab size

This	0	1	0	0	0
Is	1	0	0	0	0
cat	0	0	1	0	0

Word2Vec

Word Embedding

Context-1

"Light Bulb"

Light

0.2	0.8	0.6	0.1	0.3

Bulb

0.1	0.6	0.5	0.7	0.2

Context-2

"Light Shoes"

Light

0.2	0.8	0.6	0.1	0.3

Shoes

0.3	0.6	0.9	0.1	0.5

{Light is for illumination}

Context-3

"Light Shoes"

Light

0.2	0.8	0.6	0.1	0.3

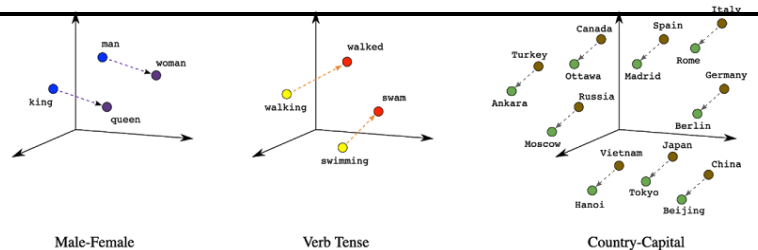
Shoes

0.3	0.6	0.9	0.1	0.5

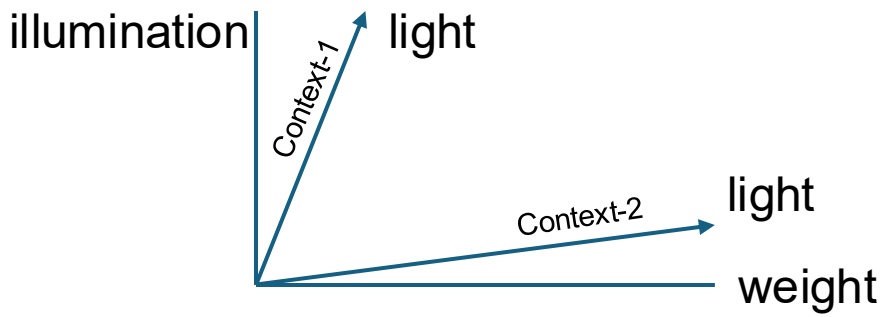
{Light is for weight}

- Captures semantic meaning

- Static in nature
- Avg. meaning of word



Need Contextual Embedding: Self Attention

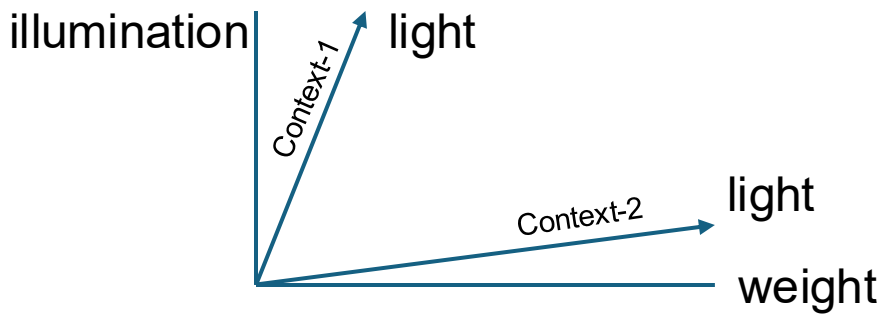


Context-1

"Light Shoes are exciting for kids"

Light	= w11 x (Light) + w12 x (Shoes) + w13 x (are) + w14 x (exciting) + w15 x (for) + w16 x (kids)
Shoes	= w21 x (Light) + w22 x (Shoes) + w23 x (are) + w24 x (exciting) + w25 x (for) + w26 x (kids)
are	= w31 x (Light) + w32 x (Shoes) + w33 x (are) + w34 x (exciting) + w35 x (for) + w36 x (kids)
exciting	= w41 x (Light) + w42 x (Shoes) + w43 x (are) + w44 x (exciting) + w45 x (for) + w46 x (kids)
for	= w51 x (Light) + w52 x (Shoes) + w53 x (are) + w54 x (exciting) + w55 x (for) + w56 x (kids)
kids	= w61 x (Light) + w62 x (Shoes) + w63 x (are) + w64 x (exciting) + w65 x (for) + w66 x (kids)

Need Contextual Embedding: Self Attention



Context-1

"Light Shoes are exciting for kids"

Light	= w11 x (Light) + w12 x (Shoes) + w13 x (are) + w14 x (exciting) + w15 x (for) + w16 x (kids)
Shoes	= w21 x (Light) + w22 x (Shoes) + w23 x (are) + w24 x (exciting) + w25 x (for) + w26 x (kids)
are	= w31 x (Light) + w32 x (Shoes) + w33 x (are) + w34 x (exciting) + w35 x (for) + w36 x (kids)
exciting	= w41 x (Light) + w42 x (Shoes) + w43 x (are) + w44 x (exciting) + w45 x (for) + w46 x (kids)
for	= w51 x (Light) + w52 x (Shoes) + w53 x (are) + w54 x (exciting) + w55 x (for) + w56 x (kids)
kids	= w61 x (Light) + w62 x (Shoes) + w63 x (are) + w64 x (exciting) + w65 x (for) + w66 x (kids)

Context-2

"Light shoes are comfortable for marathon"

Light = w71 x (Light) + w72 x (Shoes) + w73 x (are) + w74 x (comfortable) + w75 x (for) + w76 x (marathon)

embedding: $e_{Light} = w71 \times (e_{Light}) + w72 \times (e_{Shoes}) + w73 \times (e_{are}) + w74 \times (e_{comfortable}) + w75 \times (e_{for}) + w76 \times (e_{marathon})$

Weight is the similarity

"Light shoes are comfortable for marathon"

$$e^{Light} = w71 \times (e_{Light}) + w72 \times (e_{Shoes}) + w73 \times (e_{are}) + w74 \times (e_{comfortable}) + w75 \times (e_{for}) + w76 \times (e_{marathon})$$

$$e_{Light} = [e_{Light} \cdot e_{Light}^{T}] \times (e_{Light}) + [e_{Light} \cdot e_{Shoes}^{T}] \times (e_{Shoes}) + [e_{Light} \cdot e_{are}^{T}] \times (e_{are}) + [e_{Light} \cdot e_{comfortable}^{T}] \times (e_{comfortable}) + [e_{Light} \cdot e_{for}^{T}] \times (e_{for}) + [e_{Light} \cdot e_{marathon}^{T}] \times (e_{marathon})$$

1 $query_{Light}$

2 key_{Light}

3 $value_{Light}$

Weight is the similarity

"Light shoes are comfortable for marathon"

$$e^{Light} = w71 \times (e_{Light}) + w72 \times (e_{Shoes}) + w73 \times (e_{are}) + w74 \times (e_{comfortable}) + w75 \times (e_{for}) + w76 \times (e_{marathon})$$

$$e_{Light} = [e_{Light} \cdot e_{Light}^{T}] \times (e_{Light}) + [e_{Light} \cdot e_{Shoes}^{T}] \times (e_{Shoes}) + [e_{Light} \cdot e_{are}^{T}] \times (e_{are}) + [e_{Light} \cdot e_{comfortable}^{T}] \times (e_{comfortable}) + [e_{Light} \cdot e_{for}^{T}] \times (e_{for}) + [e_{Light} \cdot e_{marathon}^{T}] \times (e_{marathon})$$

1 $query_{Light}$

2 key_{Light}

3 $value_{Light}$

Softmax

W71 w72 w73 w74 w75 w76

Not a trainable parameter

$$e^{Light} = w71 \times (e_{Light}) + w72 \times (e_{Shoes}) + w73 \times (e_{are}) + w74 \times (e_{comfortable}) + w75 \times (e_{for}) + w76 \times (e_{marathon})$$

Weight is the similarity

"Light shoes are comfortable for marathon"

$$e^{Light} = w71 \times (e_{Light}) + w72 \times (e_{Shoes}) + w73 \times (e_{are}) + w74 \times (e_{comfortable}) + w75 \times (e_{for}) + w76 \times (e_{marathon})$$

$$e_{Light} = [e_{Light} \cdot e_{Light}^{T}] \times (e_{Light}) + [e_{Light} \cdot e_{Shoes}^{T}] \times (e_{Shoes}) + [e_{Light} \cdot e_{are}^{T}] \times (e_{are}) + [e_{Light} \cdot e_{comfortable}^{T}] \times (e_{comfortable}) + [e_{Light} \cdot e_{for}^{T}] \times (e_{for}) + [e_{Light} \cdot e_{marathon}^{T}] \times (e_{marathon})$$

1 $query_{Light}$ 2 key_{Light} 3 $value_{Light}$

Softmax

W71 w72 w73 w74 w75 w76 ← Not a trainable parameter

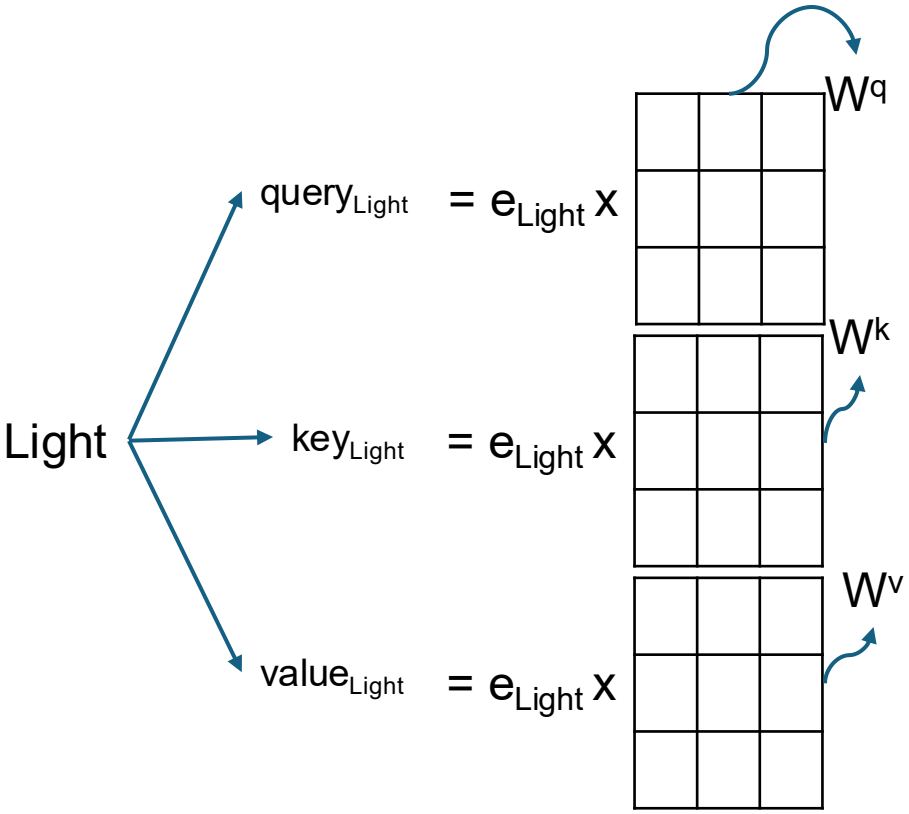
$$e^{Light} = w71 \times (e_{Light}) + w72 \times (e_{Shoes}) + w73 \times (e_{are}) + w74 \times (e_{comfortable}) + w75 \times (e_{for}) + w76 \times (e_{marathon})$$

→ General Contextual Embedding

Fails with idioms, because it is data independent

"Can be performed in parallel for all the words"

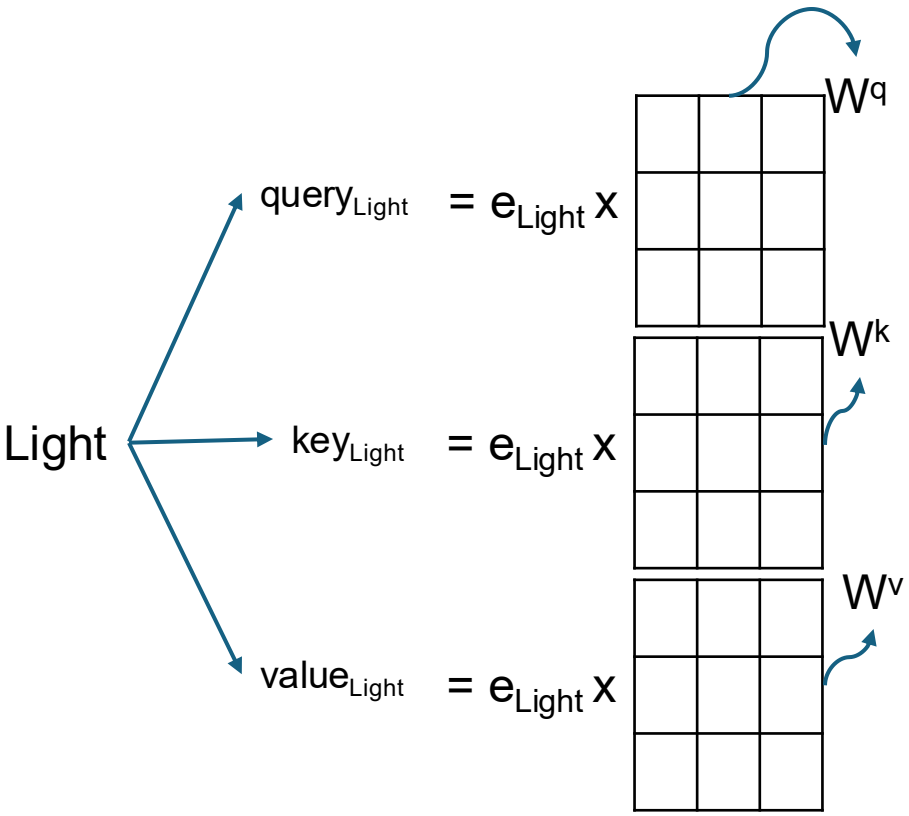
Self Attention…



Bahdanau attention and Luong Attention

- W^q, W^k, W^v could be the trainable weight, which is trained through data.
- Now q, k, v are contextual embedding which are specific to data (Task Specific).

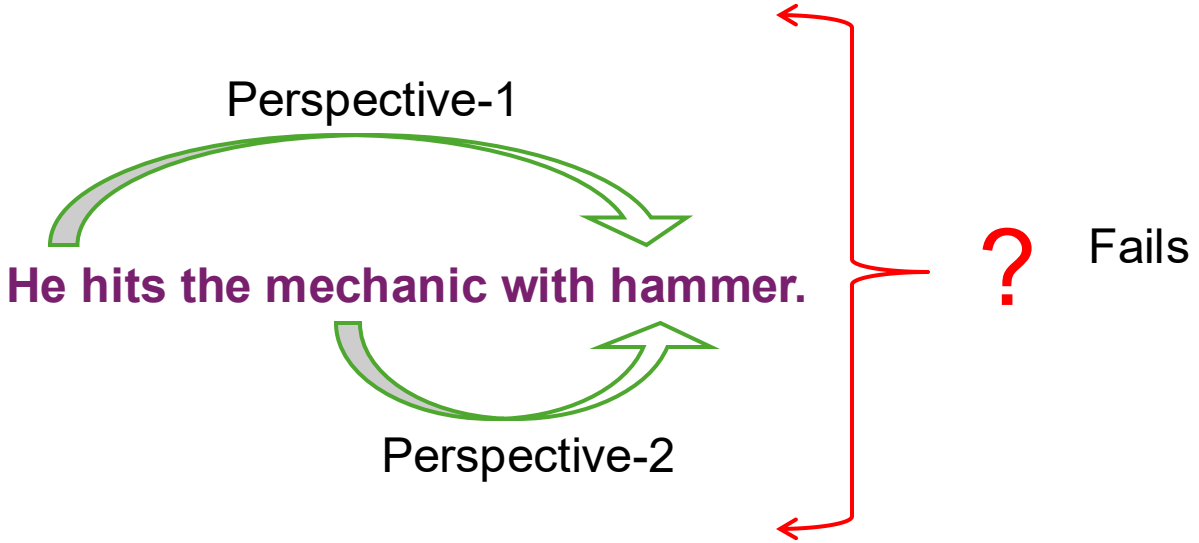
Self Attention…



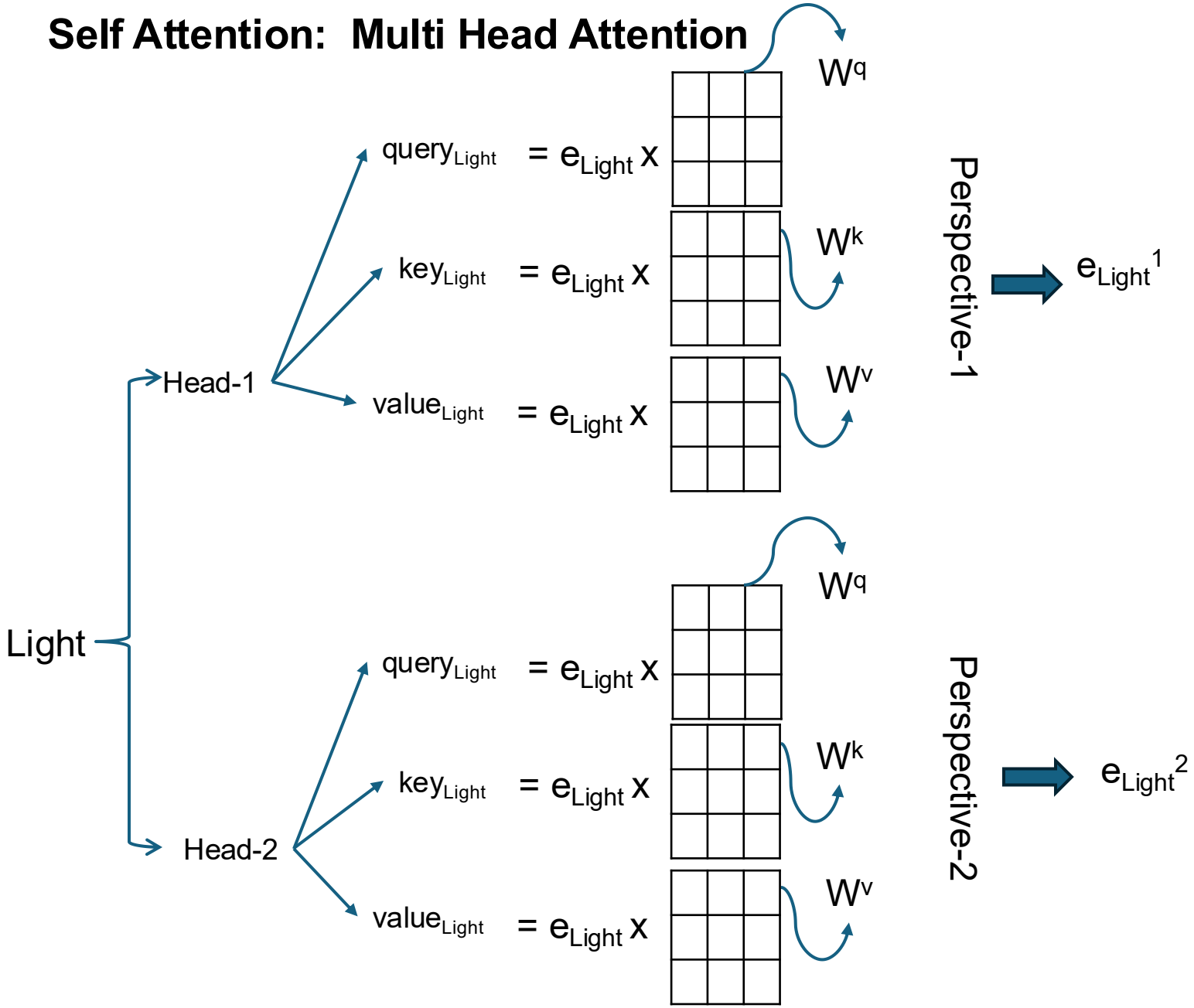
Bahdanau attention and Luong Attention

- W^q, W^k, W^v could be be the trainable weight, which is trained through data.
- Now q, k, v are contextual embedding which are specific to data (Task Specific).

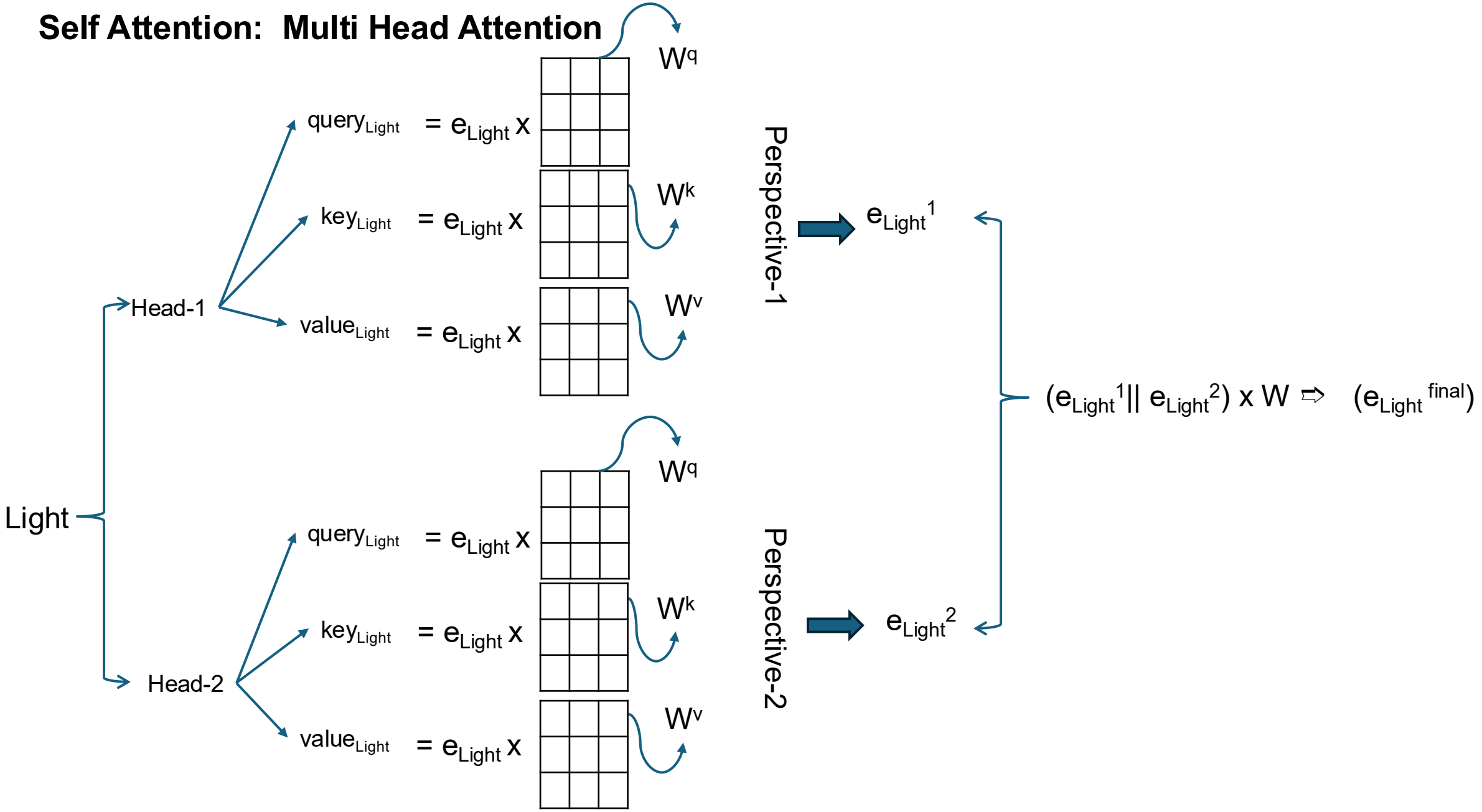
It can capture just one perspective



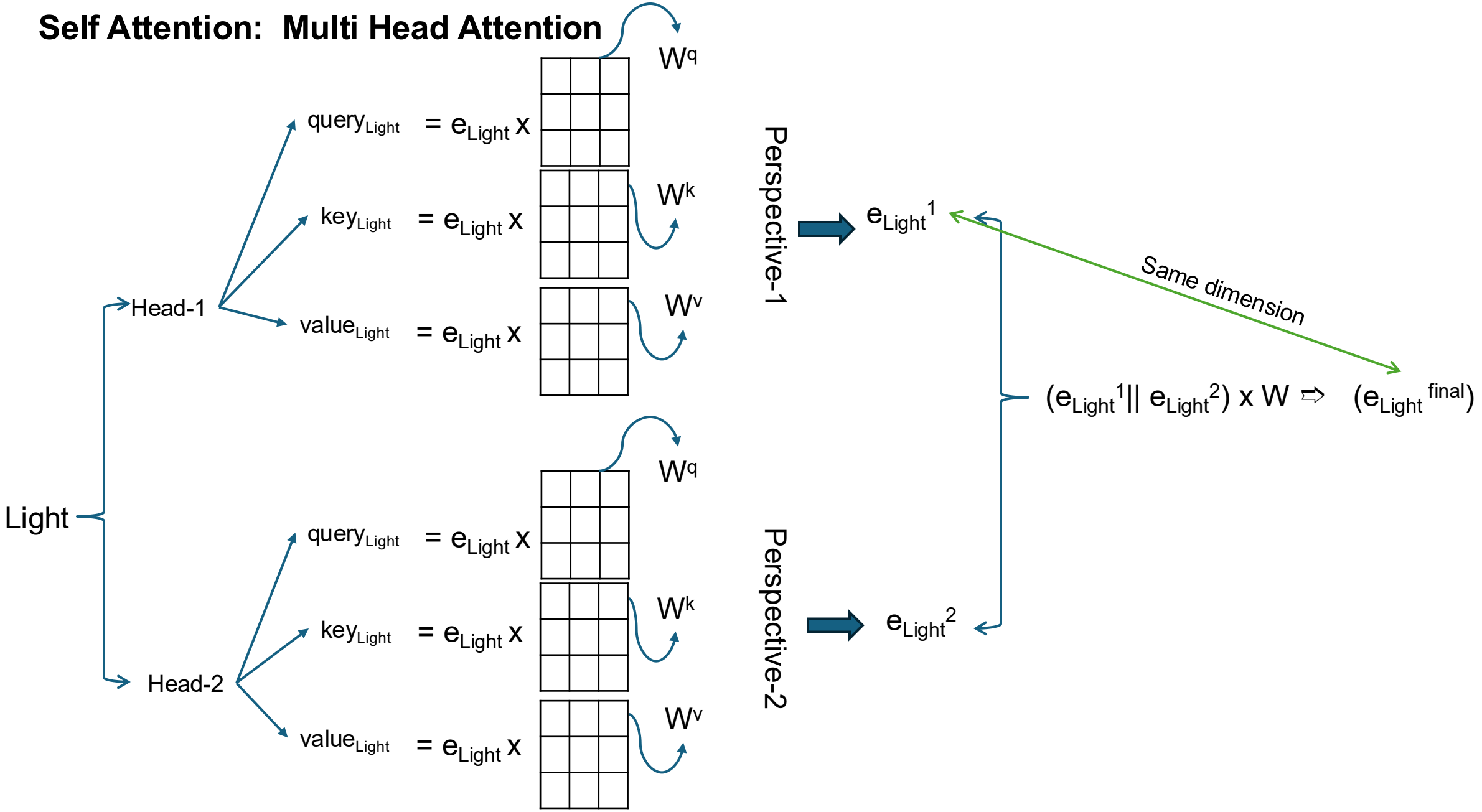
Self Attention: Multi Head Attention



Self Attention: Multi Head Attention



Self Attention: Multi Head Attention



Positional Encoding:

Sentence:	We	are	attending	a	Conference.
Position:	1 ➔	2 ➔	3 ➔	4 ➔	5

- For RNN like cases, it was easy to follow the sequence

- Can not use the unbounded number like: 1, 2, 3,4…. Will get huge with increasing data.
- Can not use discrete number for NN, will cause problem with gradient flow.
- Can not capture relative position as 10-3=> *how many points to consider as NN answers in continuous?*

Positional Encoding:

Sentence:	We	are	attending	a	Conference.
Position:	1 ➔	2 ➔	3 ➔	4 ➔	5

- For RNN like cases, it was easy to follow the sequence

- Can not use the unbounded number like: 1, 2, 3,4…. Will get huge with increasing data.
- Can not use discrete number for NN, will cause problem with gradient flow.
- Can not capture relative position as 10-3=> *how many points to consider as NN answers in continuous?*

It should be:

- Bounded
- Continuous
- Periodic

Positional Encoding:

Sentence:	We	are	attending	a	Conference.
Position:	1 ➔	2 ➔	3 ➔	4 ➔	5

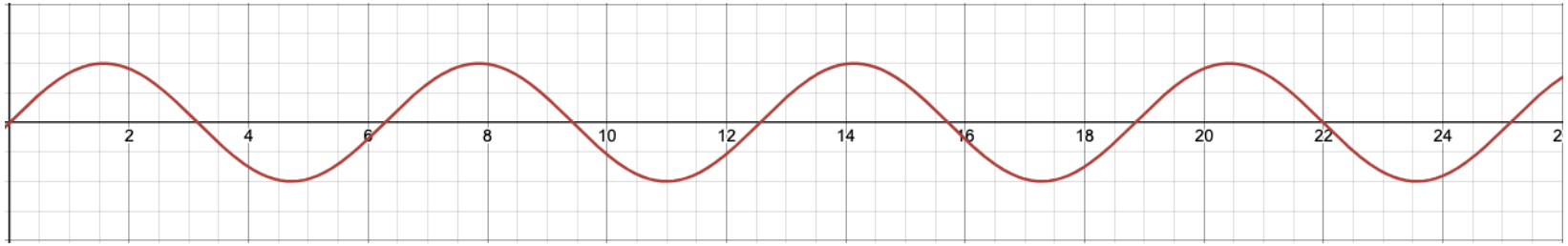
- For RNN like cases, it was easy to follow the sequence

- Can not use the unbounded number like: 1, 2, 3,4…. Will get huge with increasing data.
- Can not use discrete number for NN, will cause problem with gradient flow.
- Can not capture relative position as 10-3=> *how many points to consider as NN answers in continuous?*

It should be:

- Bounded
- Continuous
- Periodic

Y= sin(position)



Non unique values of encoding

Positional Encoding:

Sentence:	We	are	attending	a	Conference.
Position:	1 ➔	2 ➔	3 ➔	4 ➔	5

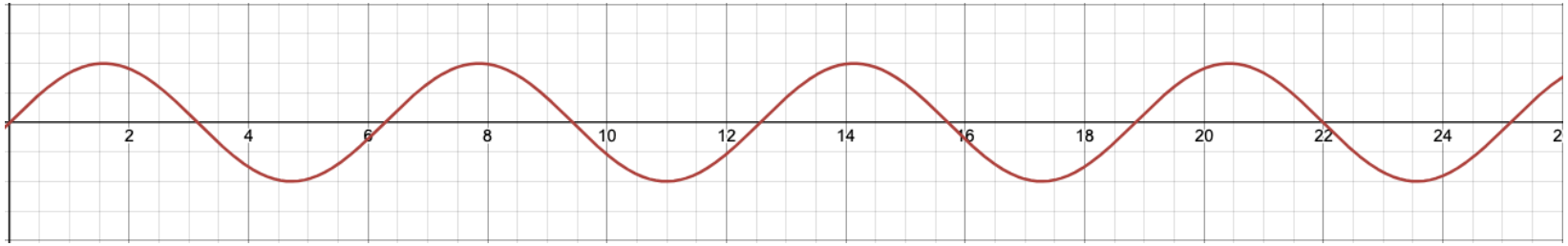
- For RNN like cases, it was easy to follow the sequence

- Can not use the unbounded number like: 1, 2, 3,4…. Will get huge with increasing data.
- Can not use discrete number for NN, will cause problem with gradient flow.
- Can not capture relative position as 10-3=> *how many points to consider as NN answers in continuous?*

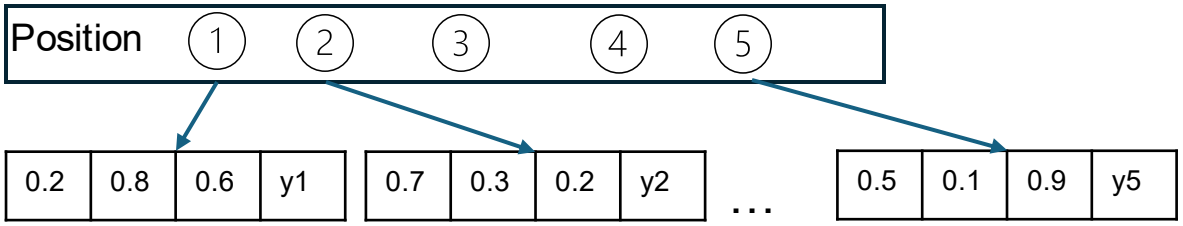
It should be:

- Bounded
- Continuous
- Periodic

Y= sin(position)



You are attending a conference

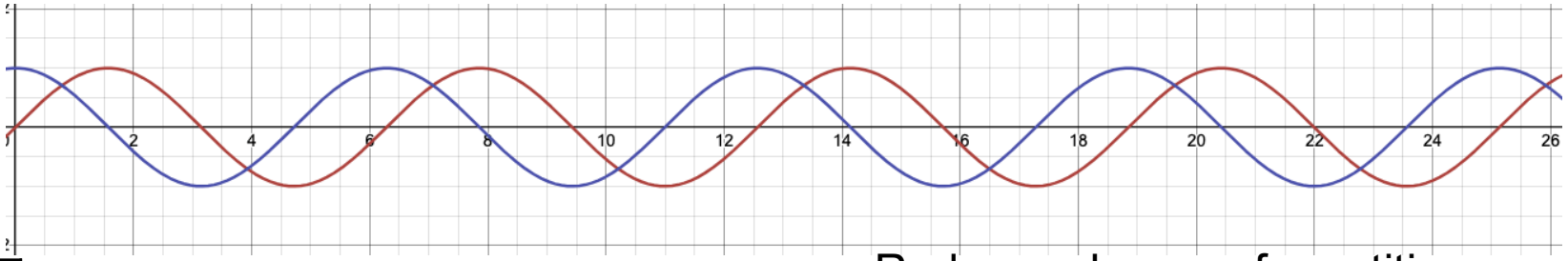


$y1 = \sin(1)$
$y2 = \sin(2)$

$y5 = \sin(5)$

Non unique values of encoding

Positional Encoding…



y11	y12

Reduces chance of repetition

$y11 = \sin(1),\ y12 = \cos(1)$
$y21 = \sin(2),\ y12 = \cos(2)$

$y51 = \sin(5),\ y12 = \cos(5)$

You are attending a conference

Position	1	2	3	4	5

0.2	0.8	0.6	y11	y12

You

0.7	0.3	0.2	y21	y22

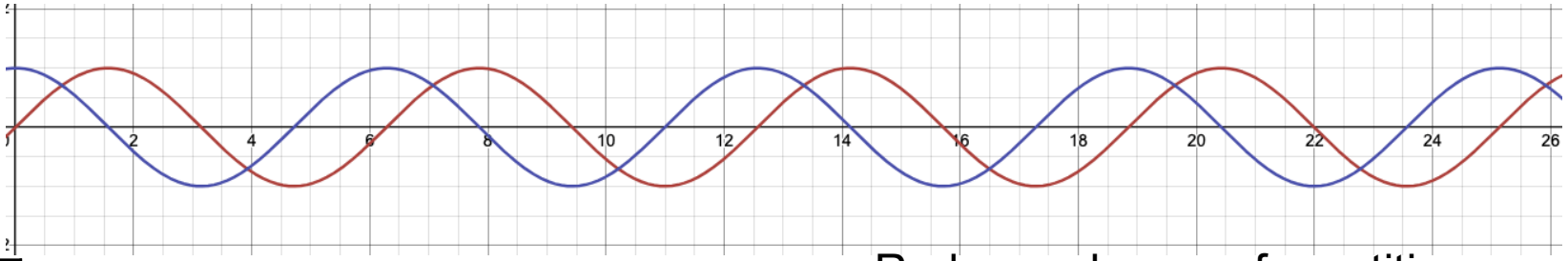
are

…

0.5	0.1	0.9	y51	y52

conference

Positional Encoding…



y11	y12

Reduces chance of repetition

$y11 = \sin(1),\ y12 = \cos(1)$
$y21 = \sin(2),\ y12 = \cos(2)$

$y51 = \sin(5),\ y12 = \cos(5)$

You are attending a conference

Position	1	2	3	4	5

0.2	0.8	0.6	y11	y12

0.7	0.3	0.2	y21	y22

…

0.5	0.1	0.9	y51	y52

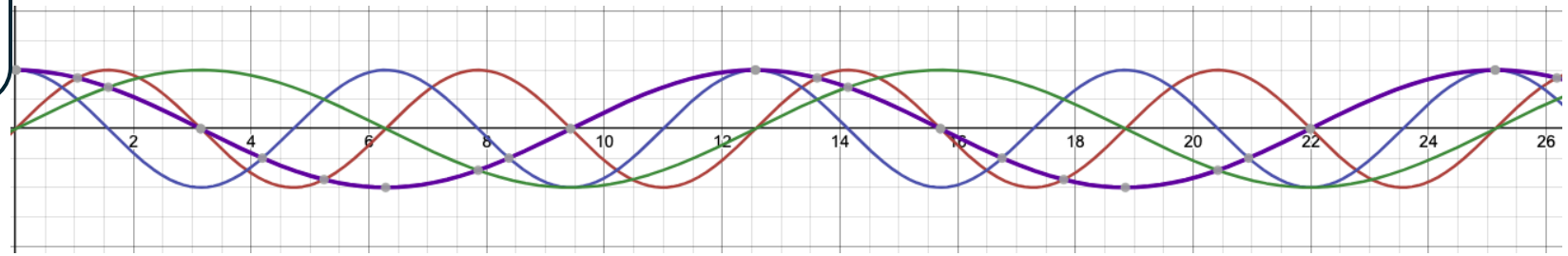
You **are** **conference**

$y11 = \sin(1),\ y12 = \cos(1),\ y13 = \sin(1/2),\ y14 = \cos(1/2)$

$y21 = \sin(2),\ y22 = \cos(2)\ ,\ y23 = \sin(2/2),\ y24 = \cos(2/2)$

$y51 = \sin(5),\ y52 = \cos(5)\ ,\ y53 = \sin(5/2),\ y54 = \cos(5/2)$

Reduces chance of repetition



Increasing Dimensions

sin (x), cos (x), sin (x/2), cos (x/2)

0.2	0.8	0.6	y11	y12	y13	y14

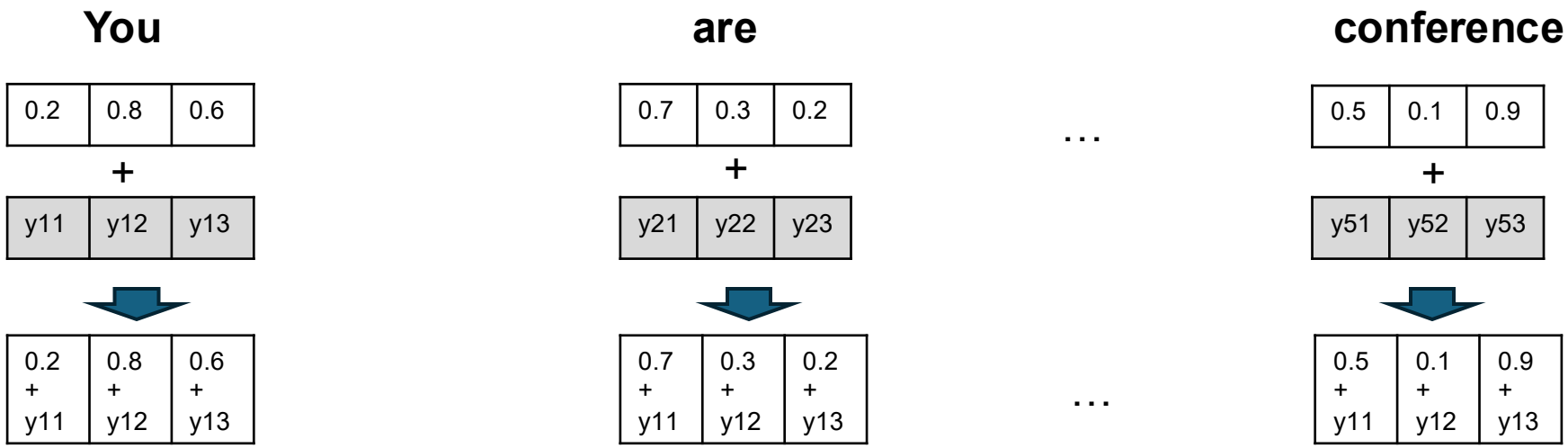
0.7	0.3	0.2	y21	y22	y23	y24

…

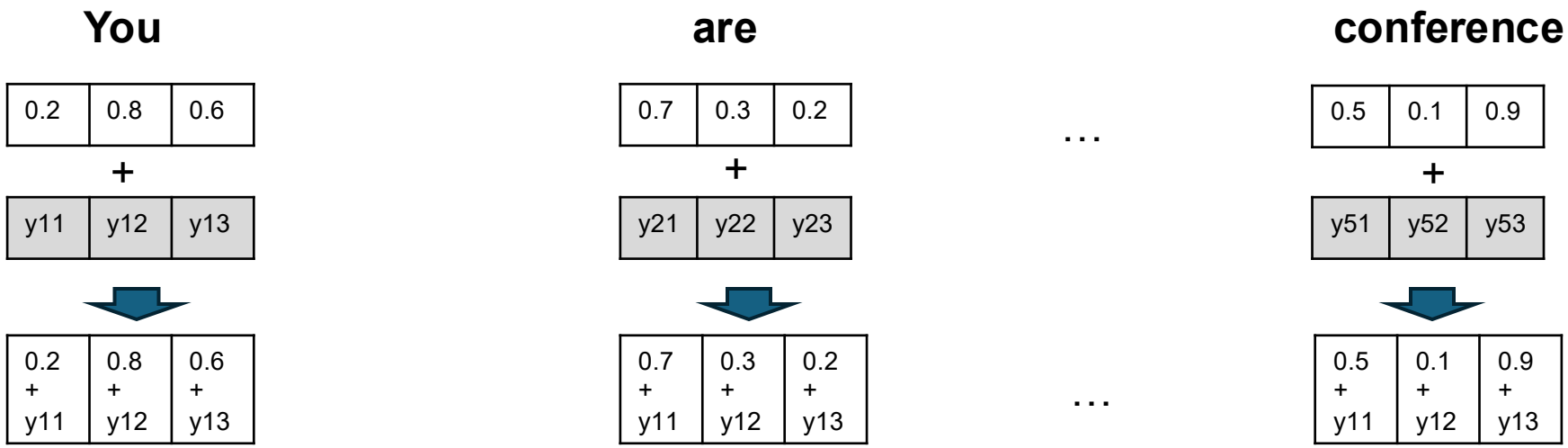
0.5	0.1	0.9	y51	y52	y53	y54

You **are** **conference**

Positional Encoding…



Positional Encoding…



Original Paper:

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

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

Where d_{model}: dimension of embedding

Rotary Positional Encoding (LAMA)

Let x be the token embedding. RoPE applies a rotation matrix R_θ, where:

$$R_\theta(p) = \begin{bmatrix} \cos(\theta_p) & -\sin(\theta_p) \\ \sin(\theta_p) & \cos(\theta_p) \end{bmatrix}$$

where θ_p is determined by:

$$\theta_p = 10000^{-2(i-1)/d}$$

for each embedding dimension i in the model.

- Suppose your embedding (for query or key) is a vector $x \in \mathbb{R}^d$.
- We divide it into pairs of dimensions:

$$(x_1, x_2), (x_3, x_4), \ldots, (x_{d-1}, x_d)$$

- For each pair (x_{2i-1}, x_{2i}) ,we apply a **2D rotation** based on the token's position p:

so each position p applies a **rotation matrix** parameterized by θ_i.

Layer Normalization

Assumption: Max length of a sentence is 5, and embedding dimension is of 4.

Input: We are attending conference. It is an online event.

Layer Normalization

Assumption: Max length of a sentence is 5, and embedding dimension is of 4.

Input: We are attending conference. It is an online event.

We	e11	e12	e13	e14
are	e21	e22	e23	e24
Attending	e31	e32	e33	e34
conference	e41	e42	e43	e44
<padding>	e51	e52	e53	e54
It	e61	e62	e63	e64
is	e71	e72	e73	e74
an	e81	e82	e83	e84
online	e91	e92	e93	e94
event	e101	e102	e103	e104

Matrix: 5x4

Matrix: 5x4

Tensor: 2x5x4

μ, ϑ

Batch Normalization

Layer Normalization

Assumption: Max length of a sentence is 5, and embedding dimension is of 4.

Input: We are attending conference. It is an online event.

We	e11	e12	e13	e14
are	e21	e22	e23	e24
Attending	e31	e32	e33	e34
conference	e41	e42	e43	e44
<padding>	e51	e52	e53	e54
It	e61	e62	e63	e64
is	e71	e72	e73	e74
an	e81	e82	e83	e84
online	e91	e92	e93	e94
event	e101	e102	e103	e104

μ, ϑ

Layer Normalization

Matrix: 5x4

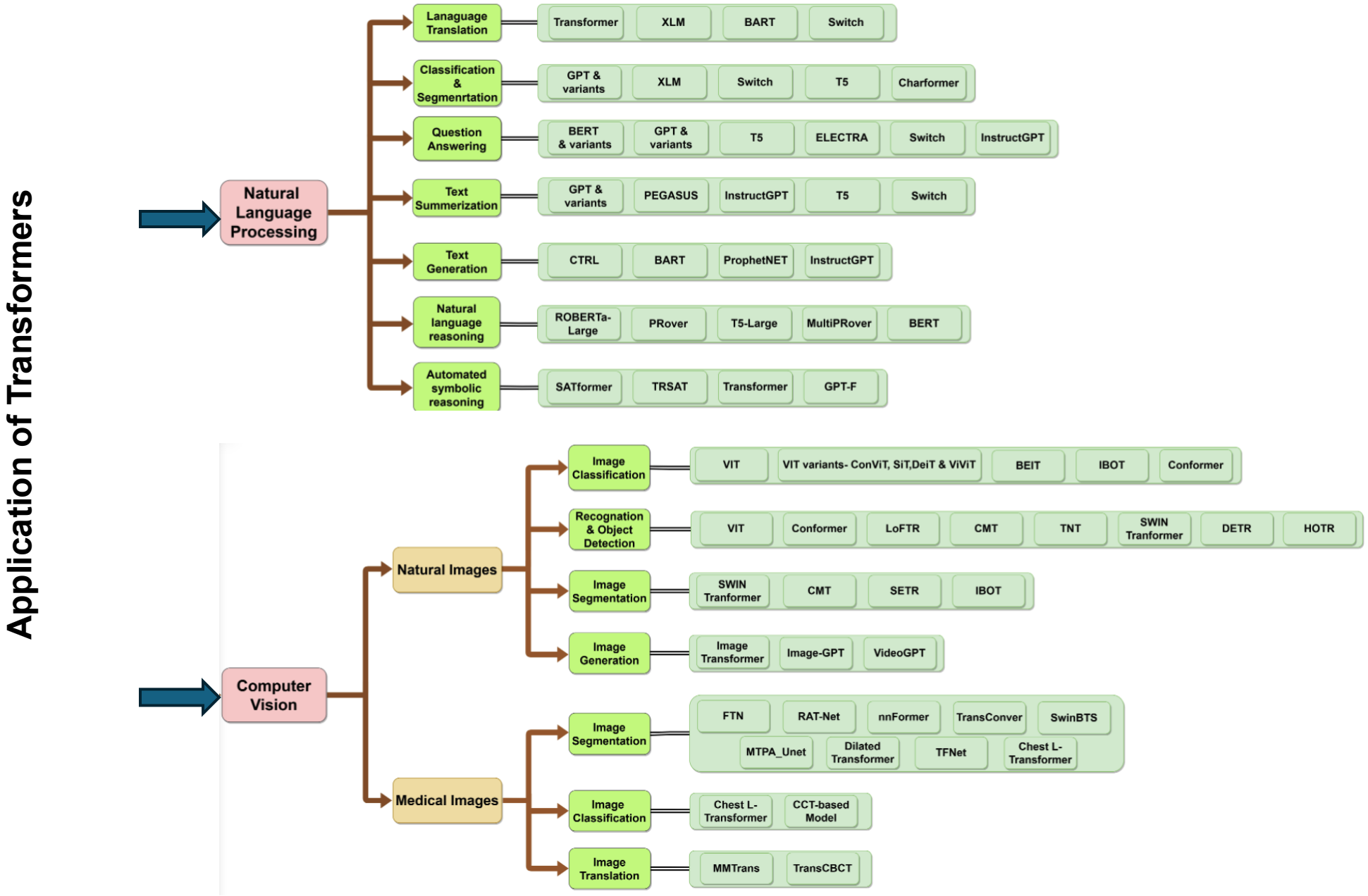
Tensor: 2x5x4

Matrix: 5x4

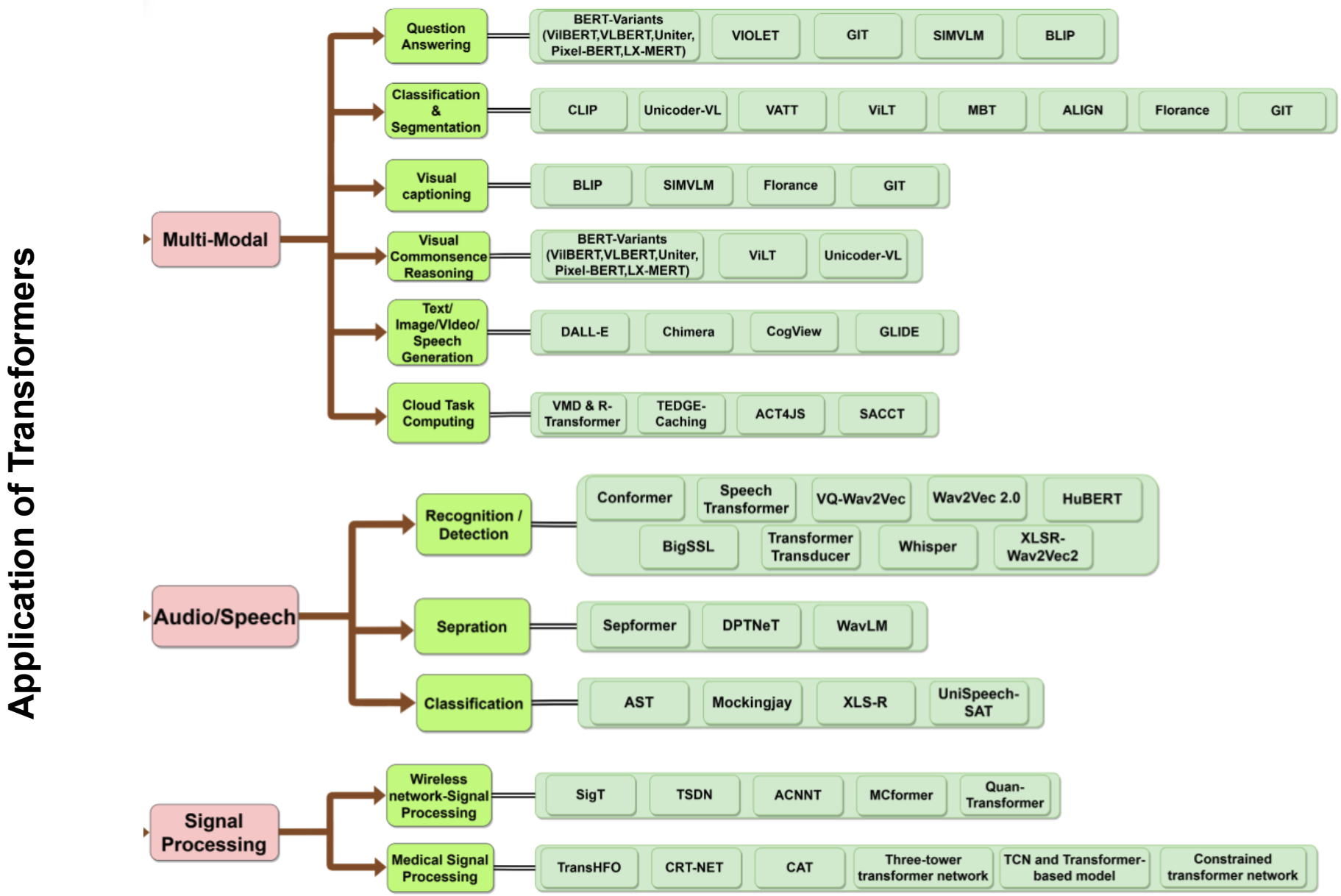
μ, ϑ

Batch Normalization

Applications



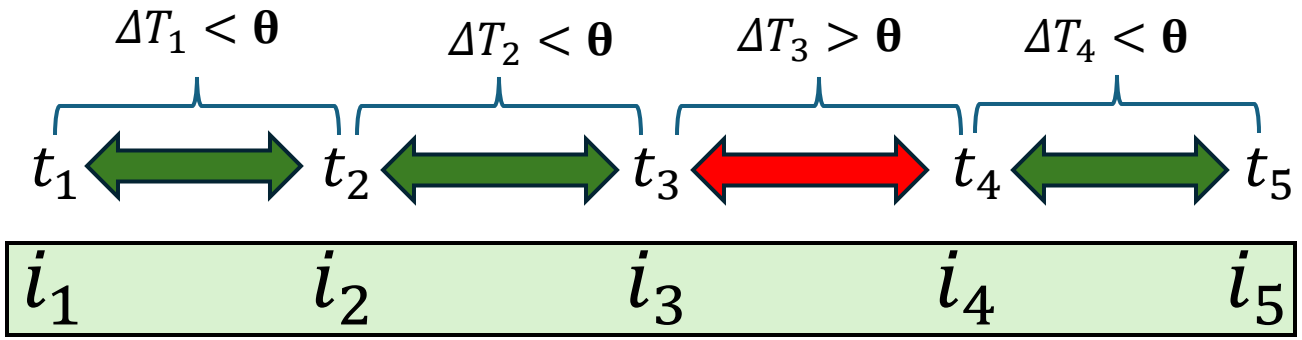
Applications…



Session-Based Recommendation Systems

Dataset

$user_id$	$item_id$	$time_stamp$
u_1	i_1	t_1
u_1	i_2	t_2
u_1	i_3	t_3
u_1	i_4	t_4
u_1	i_5	t_5

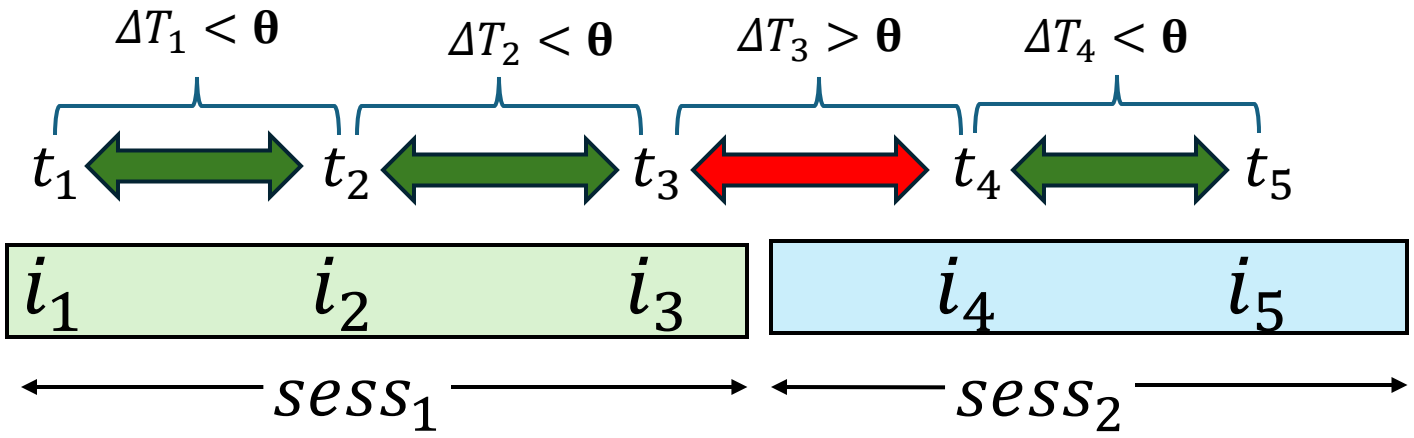


- A session continues to consider the interactions in same session if duration of inactivity between two interactions is smaller than threshold and is split otherwise.

Session-Based Recommendation Systems

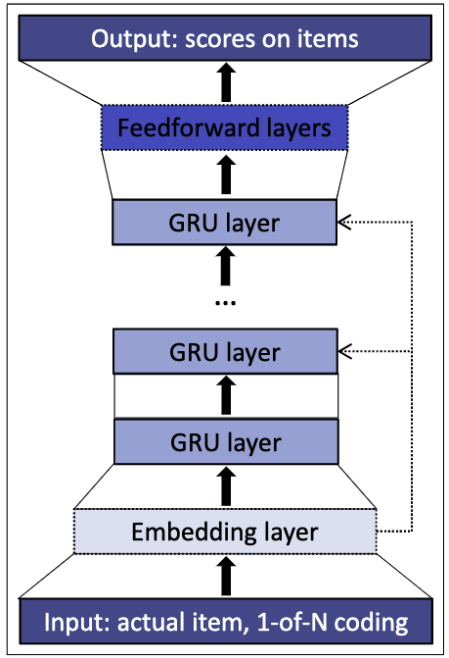
Dataset

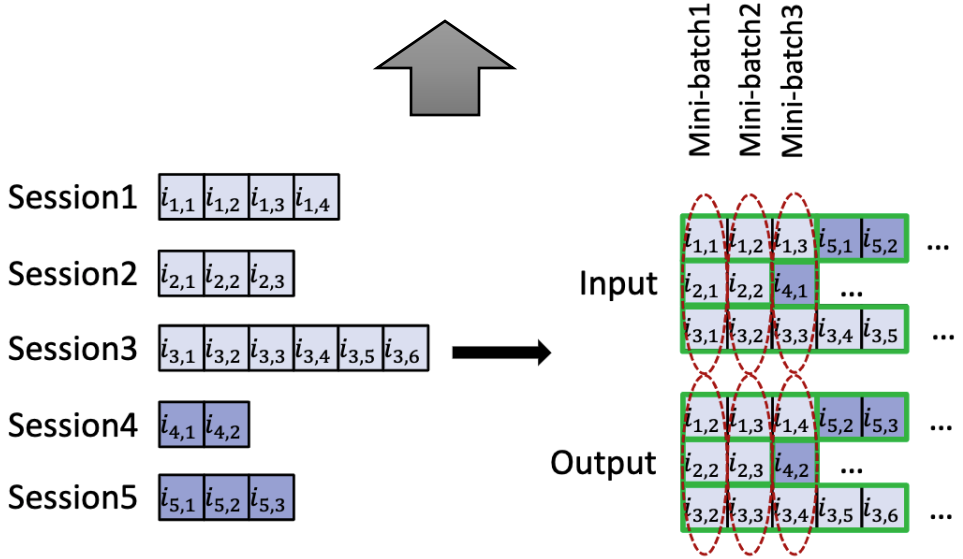
$user_id$	$item_id$	$time_stamp$
u_1	i_1	t_1
u_1	i_2	t_2
u_1	i_3	t_3
u_1	i_4	t_4
u_1	i_5	t_5

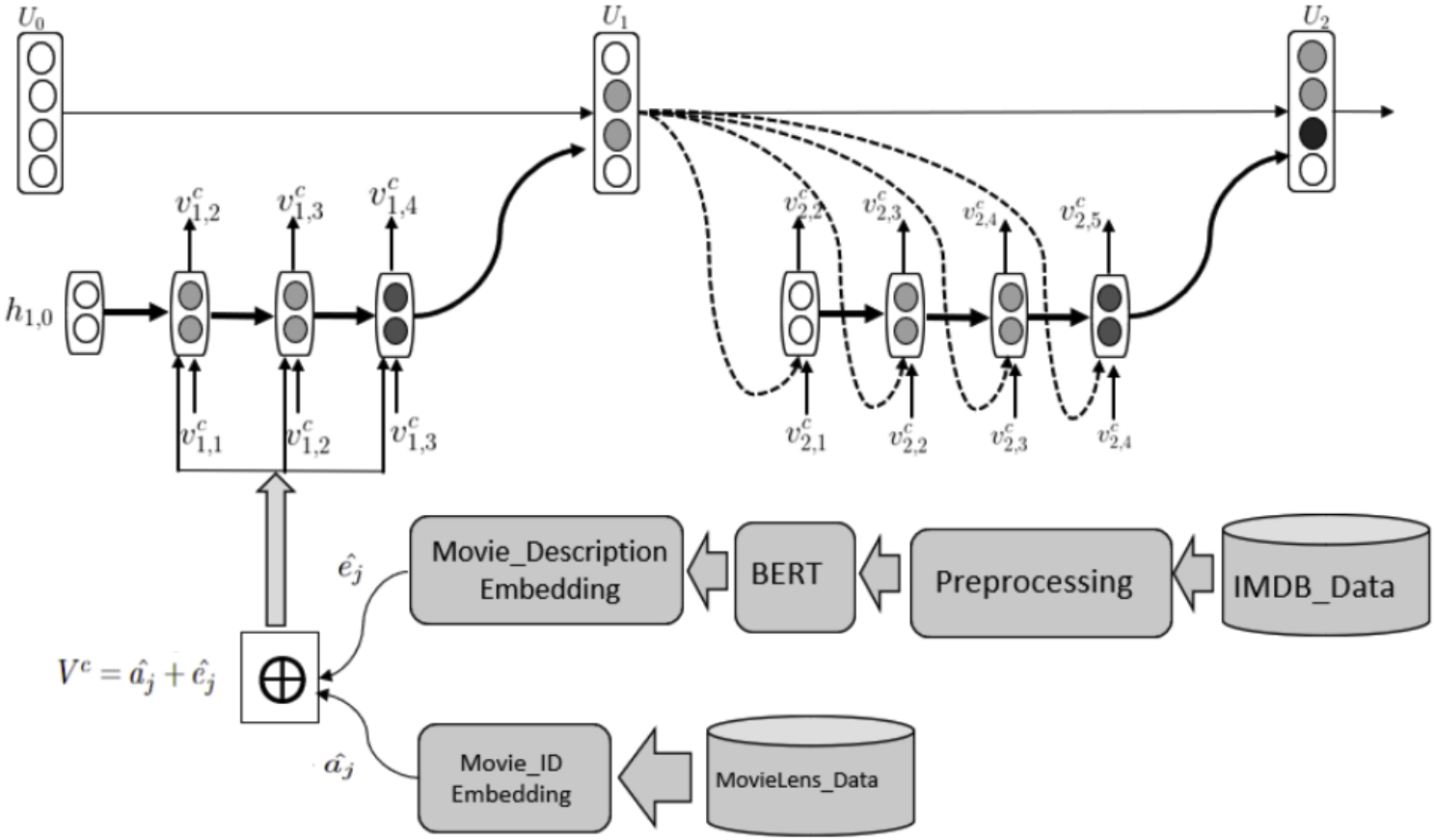


- A session continues to consider the interactions in same session if duration of inactivity between two interactions is smaller than threshold and is split otherwise.

Sequential Recommendations➔ Session Based Recommendation ➔ Next Item Prediction Task







SONY

AAAI-25 / IAAI-25 / EAAI-25
FEBRUARY 25 – MARCH 4, 2025 | PHILADELPHIA, USA

Ready for You When You are Back: Content-driven Session-based Recommendation for Continuity of Experience

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Time Driven Session:

"A case of Pleasure to Pain"

On Rats:

- An expectation of reward pumps the level of Dopamine which creates the feeling of pleasure [1].

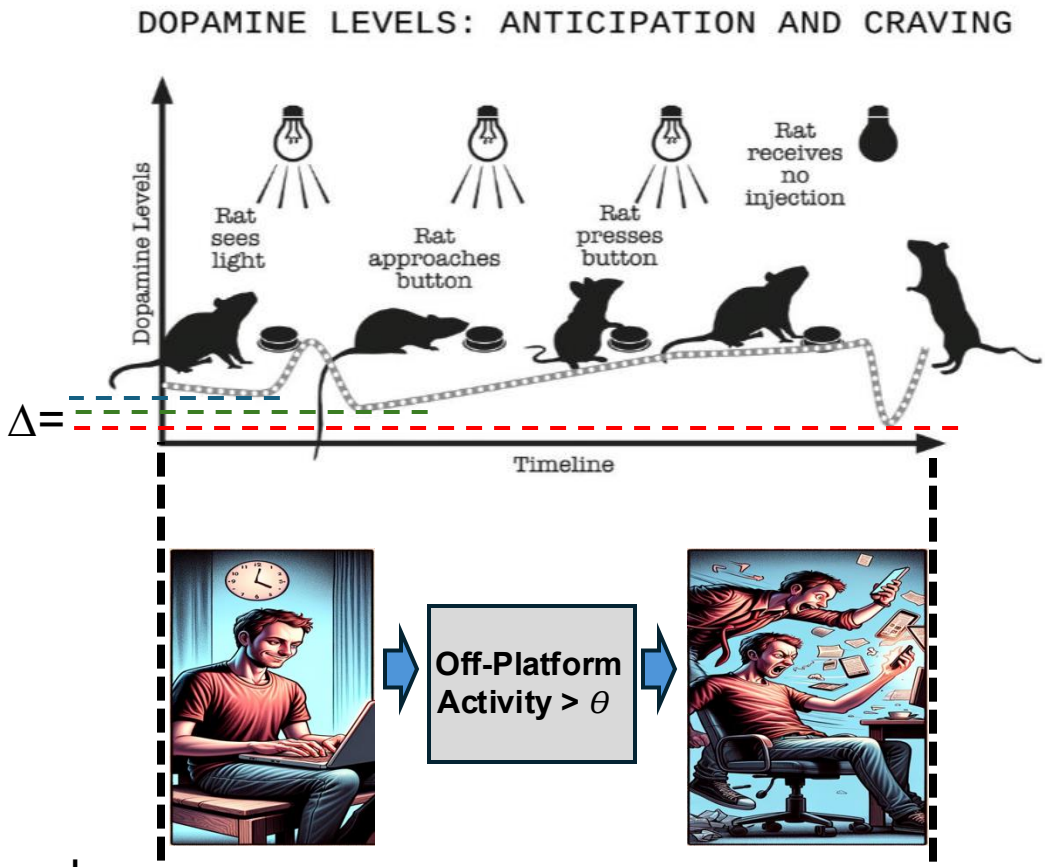
- In case expectations are not fulfilled, a sharp dip in Dopamine level is witnessed which causes more pain than the previous pleasure.

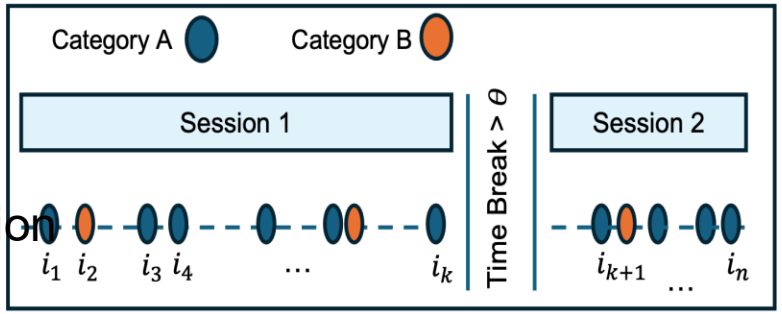
An expected reward that fails to materialize is worse than a reward that was never anticipated in the first place.

On Humans:

- A sequence of continuous activity on platform increases the level of Dopamine and consequently the level of satisfaction.

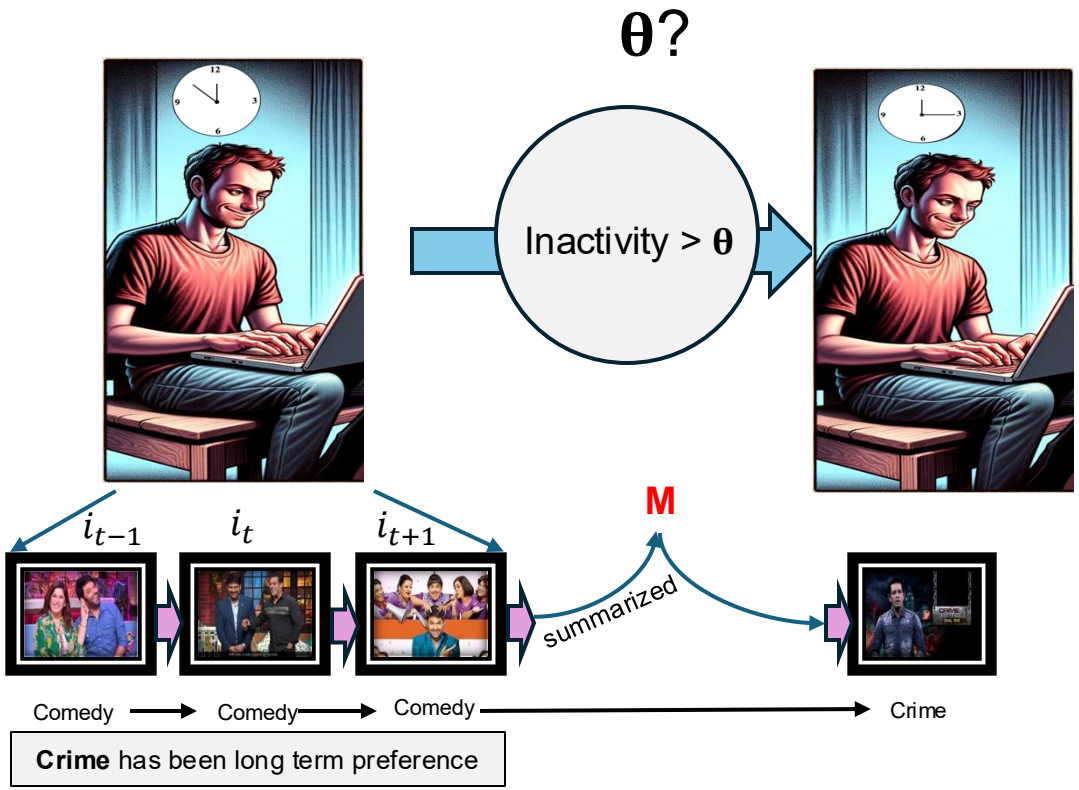
- In case, user gets inactive for a time > θ, when user come back s/he finds the sudden drop in experience because of changing the session which causes more unsatisfaction than the previous pleasure.

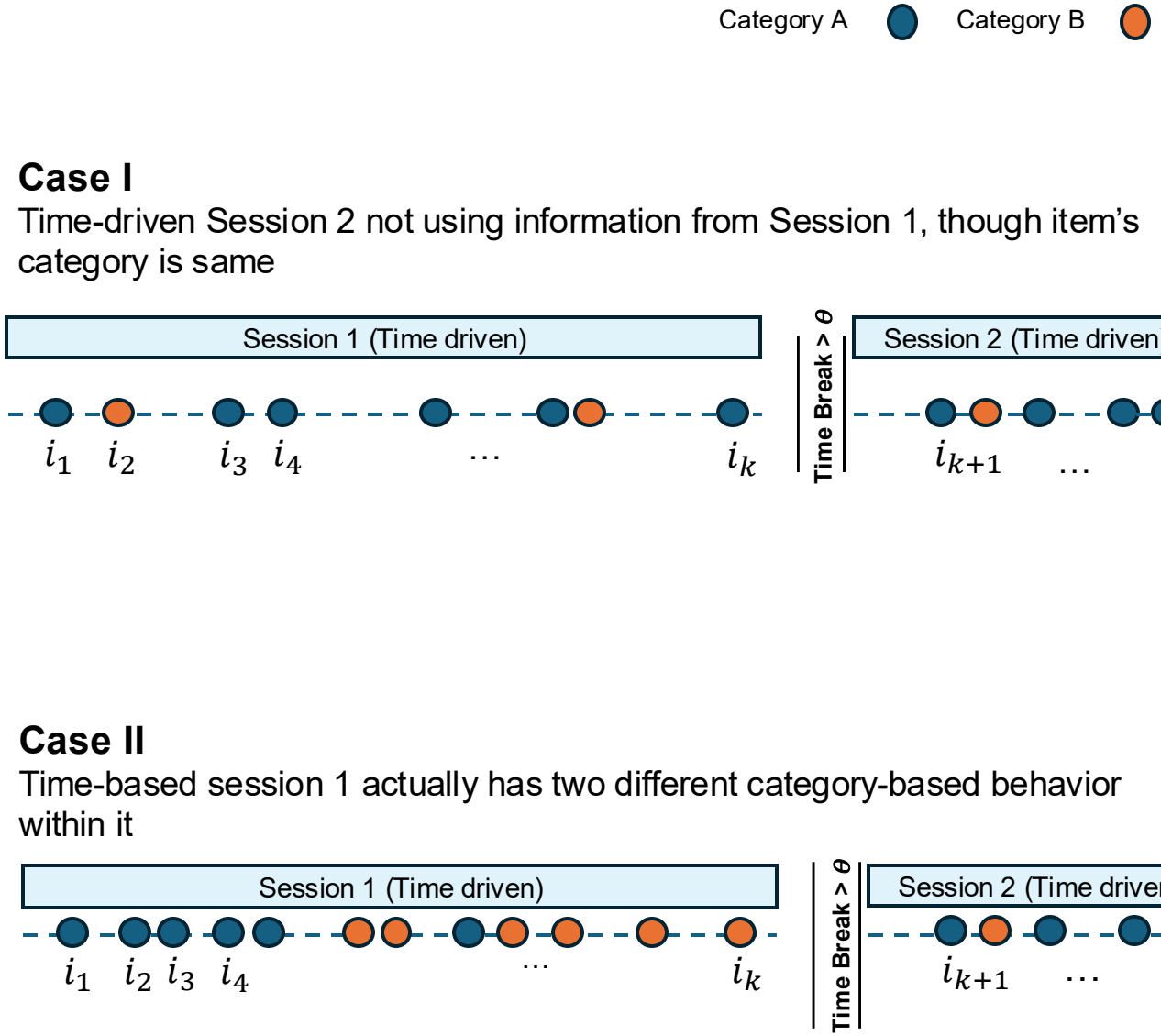




[1] Lembke, A. (2021). *Dopamine nation: Finding balance in the age of indulgence*. Penguin.

Research gap

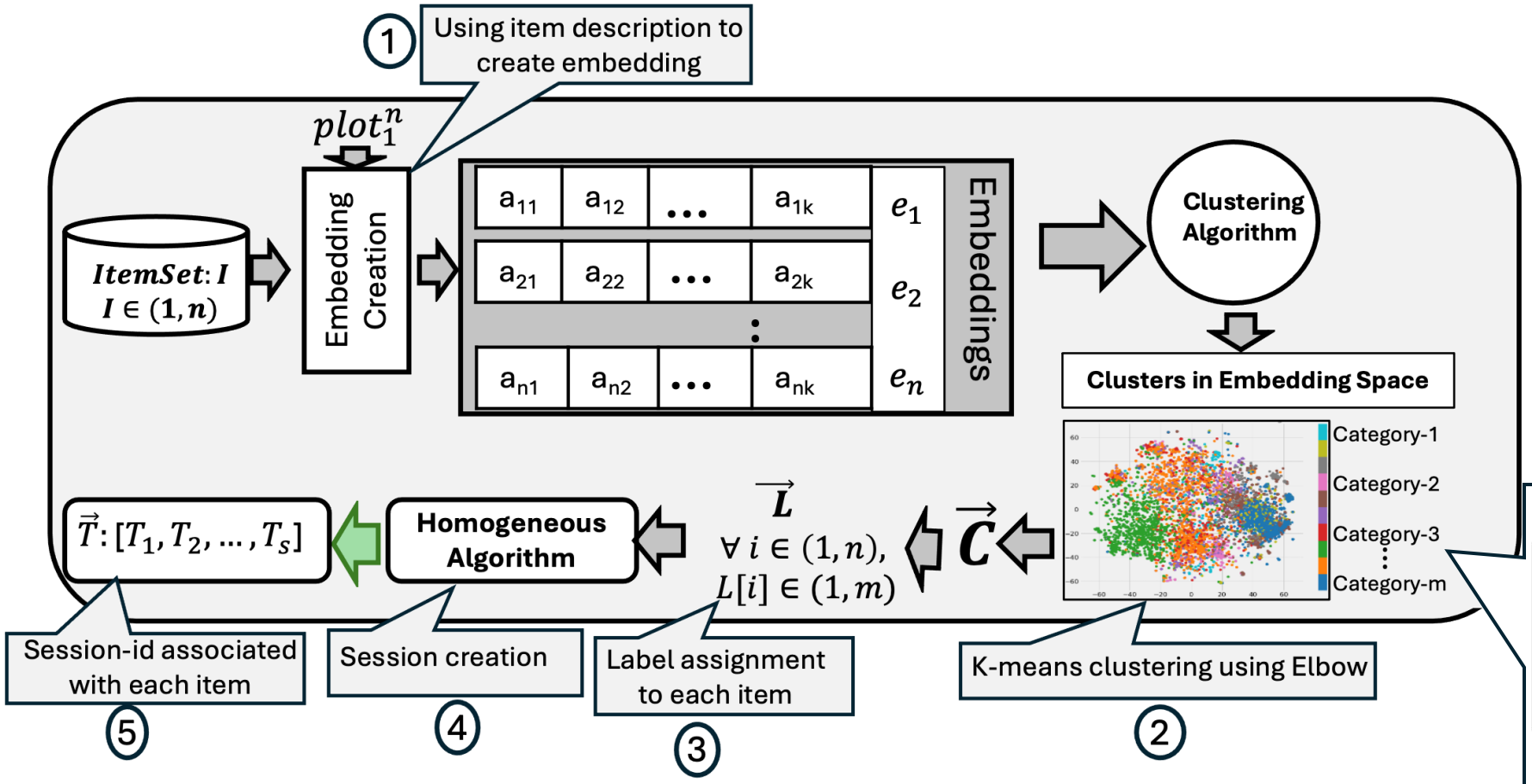




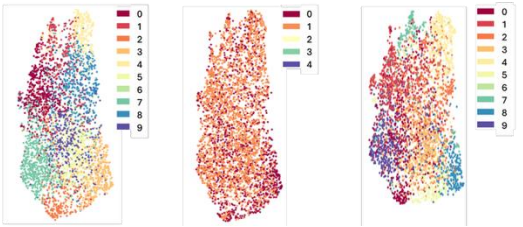
Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Proposed Solution

Requirement: User_id, Item_id, Item_description, Time_stamp

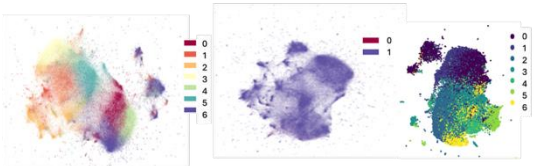


Movielens



Clustering	Silhouette Coefficient
K-Means	**0.02433**
DBSCAN	0.01745
Agglomerative	0.00573

Book



Clustering	Silhouette Coefficient
K-Means	0.03882379
DBSCAN	-0.2259855
Agglomerative	0.0268188

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

for each i $\in (0, len(\mathcal{I}))$ **do**
  $\Phi[\mathcal{I}[i]] \leftarrow \Phi[\mathcal{I}[i]] + 1$
  if $\Phi[item] > 2$ and $\mathcal{S} < 2$ **then**
    $\mathcal{M} = i$
  end if
  if $max(\Phi) > 2$ and $\mathcal{S} == 2$ **then**
    $\S.append(\mathcal{M})$
    if $\mathcal{M} + 1 < len(\mathcal{I} - 1)$ **then**
      $i = \mathcal{M}$
    end if
    $\mathcal{M} = 0$
    $\Phi = zeros(len(item_groups))$
  end if
  i=i+1
end for
for i in $(0, len(\S))$ **do**
  $\mathcal{H}.append(\mathcal{I}[\S[i] + 1 : \S[i + 1] + 1])$
end for

Working

Total categories: 5

Map with total 5 Categories

C1	C2	C3	C4	C5
1	0	0	0	0

$user_i$

Interaction
i_1: C_1

C1

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

for each i $\in (0, len(\mathcal{I}))$ **do**
  $\Phi[\mathcal{I}[i]] \leftarrow \Phi[\mathcal{I}[i]] + 1$
  if $\Phi[item] > 2$ and $\mathcal{S} < 2$ **then**
    $\mathcal{M} = i$
  end if
  if $max(\Phi) > 2$ and $\mathcal{S} == 2$ **then**
    $\S.append(\mathcal{M})$
    if $\mathcal{M} + 1 < len(\mathcal{I} - 1)$ **then**
      $i = \mathcal{M}$
    end if
    $\mathcal{M} = 0$
    $\Phi = zeros(len(item_groups))$
  end if
  i=i+1
end for
for i in $(0, len(\S))$ **do**
  $\mathcal{H}.append(\mathcal{I}[\S[i] + 1 : \S[i + 1] + 1])$
end for

Working…

Categories: 5

C1	C2	C3	C4	C5
1	1	0	0	0

$user_i$

Interaction
i_1: C_1
i_2: C_2

C1, C2

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

for each i $\in (0, len(\mathcal{I}))$ **do**
  $\Phi[\mathcal{I}[i]] \leftarrow \Phi[\mathcal{I}[i]] + 1$
  if $\Phi[item] > 2$ and $\mathcal{S} < 2$ **then**
    $\mathcal{M} = i$
  end if
  if $max(\Phi) > 2$ and $\mathcal{S} == 2$ **then**
    $\S.append(\mathcal{M})$
    if $\mathcal{M} + 1 < len(\mathcal{I} - 1)$ **then**
      $i = \mathcal{M}$
    end if
    $\mathcal{M} = 0$
    $\Phi = zeros(len(item_groups))$
  end if
  i=i+1
end for
for i in $(0, len(\S))$ **do**
  $\mathcal{H}.append(\mathcal{I}[\S[i] + 1 : \S[i + 1] + 1])$
end for

Working…

Categories: 5

C1	C2	C3	C4	C5
2	1	0	0	0

$user_i$

Interaction
i_1: C_1
i_2: C_2
i_3: C_1

C1, C2, C1

: Dominating category

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

for each i $\in (0, len(\mathcal{I}))$ **do**
  $\Phi[\mathcal{I}[i]] \leftarrow \Phi[\mathcal{I}[i]] + 1$
  if $\Phi[item] > 2$ and $\mathcal{S} < 2$ **then**
    $\mathcal{M} = i$
  end if
  if $max(\Phi) > 2$ and $\mathcal{S} == 2$ **then**
    $\S.append(\mathcal{M})$
    if $\mathcal{M} + 1 < len(\mathcal{I} - 1)$ **then**
      $i = \mathcal{M}$
    end if
    $\mathcal{M} = 0$
    $\Phi = zeros(len(item_groups))$
  end if
  i=i+1
end for
for i in $(0, len(\S))$ **do**
  $\mathcal{H}.append(\mathcal{I}[\S[i] + 1 : \S[i + 1] + 1])$
end for

Working…

Categories: 5

C1	C2	C3	C4	C5
2	1	1	0	0

$user_i$

Interaction
i_1: C_1
i_2: C_2
i_3: C_1
i_4: C_3

C1, C2, C1, C3

: Dominating category

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

for each i $\in (0, len(\mathcal{I}))$ **do**
  $\Phi[\mathcal{I}[i]] \leftarrow \Phi[\mathcal{I}[i]] + 1$
  if $\Phi[item] > 2$ and $\mathcal{S} < 2$ **then**
    $\mathcal{M} = i$
  end if
  if $max(\Phi) > 2$ and $\mathcal{S} == 2$ **then**
    $\S.append(\mathcal{M})$
    if $\mathcal{M} + 1 < len(\mathcal{I} - 1)$ **then**
      $i = \mathcal{M}$
    end if
    $\mathcal{M} = 0$
    $\Phi = zeros(len(item_groups))$
  end if
  i=i+1
end for
for i in $(0, len(\S))$ **do**
  $\mathcal{H}.append(\mathcal{I}[\S[i] + 1 : \S[i + 1] + 1])$
end for

Working…

Categories: 5

C1	C2	C3	C4	C5
2	1	1	1	0

$user_i$

Interaction
i_1: C_1
i_2: C_2
i_3: C_1
i_4: C_3
i_5: C_4

C1, C2, C1, C3, C4

: Dominating category

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

for each i $\in (0, len(\mathcal{I}))$ **do**
  $\Phi[\mathcal{I}[i]] \leftarrow \Phi[\mathcal{I}[i]] + 1$
  if $\Phi[item] > 2$ and $\mathcal{S} < 2$ **then**
    $\mathcal{M} = i$
  end if
  if $max(\Phi) > 2$ and $\mathcal{S} == 2$ **then**
    $\S.append(\mathcal{M})$
    if $\mathcal{M} + 1 < len(\mathcal{I} - 1)$ **then**
      $i = \mathcal{M}$
    end if
    $\mathcal{M} = 0$
    $\Phi = zeros(len(item_groups))$
  end if
  i=i+1
end for
for i in $(0, len(\S))$ **do**
  $\mathcal{H}.append(\mathcal{I}[\S[i] + 1 : \S[i + 1] + 1])$
end for

Working…

Categories: 5

C1	C2	C3	C4	C5
2	1	1	1	1

$user_i$

Interaction
i_1: C_1
i_2: C_2
i_3: C_1
i_4: C_3
i_5: C_4
i_6: C_5

C1, C2, C1, C3, C4, C5

: Dominating category

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

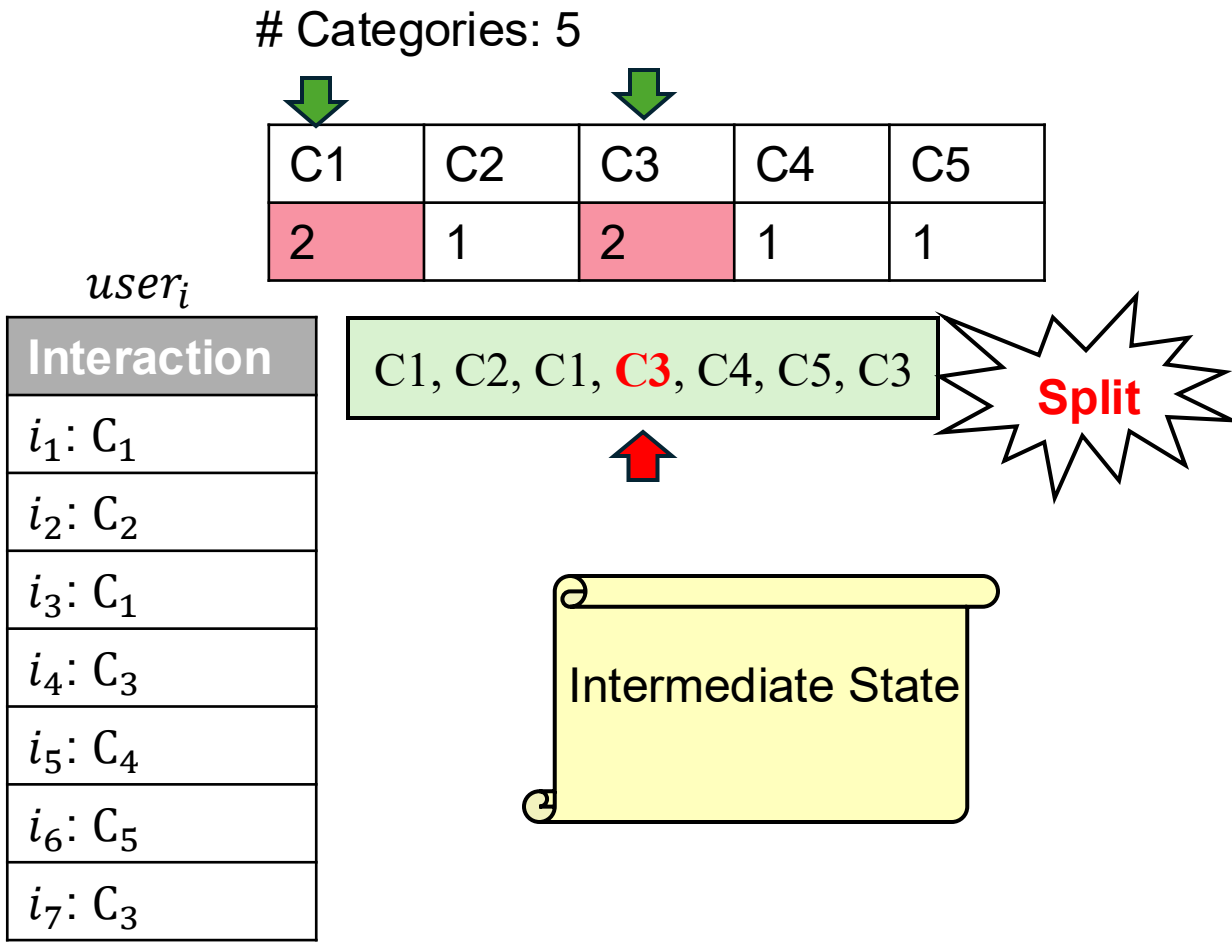
Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

```
for each i ∈ (0, len(I)) do
    Φ[I[i]] ← Φ[I[i]] + 1
    if Φ[item] > 2 and S < 2 then
        M = i
    end if
    if max(Φ) > 2 and S == 2 then
        §.append(M)
        if M + 1 < len(I − 1) then
            i = M
        end if
        M = 0
        Φ = zeros(len(item_groups))
    end if
    i=i+1
end for
for i in (0, len(§)) do
    H.append(I[§[i] + 1 : §[i + 1] + 1])
end for
```

Working…



: Dominating category

Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Homogeneous Session Creation Algo.

Algorithm 1: *Homogeneous Content Window Creation*

Input: $\mathcal{I}$ **Output**: $\vec{\mathcal{H}}$

```
for each i ∈ (0, len(I)) do
    Φ[I[i]] ← Φ[I[i]] + 1
    if Φ[item] > 2 and S < 2 then
        M = i
    end if
    if max(Φ) > 2 and S == 2 then
        §.append(M)
        if M + 1 < len(I − 1) then
            i = M
        end if
        M = 0
        Φ = zeros(len(item_groups))
    end if
    i=i+1
end for
for i in (0, len(§)) do
    H.append(I[§[i] + 1 : §[i + 1] + 1])
end for
```

Working…

Categories: 5

C1	C2	C3	C4	C5
0	0	2	1	1

$user_i$

Interaction
i_1: C_1
i_2: C_2
i_3: C_1
i_4: C_3
i_5: C_4
i_6: C_5
i_7: C_3

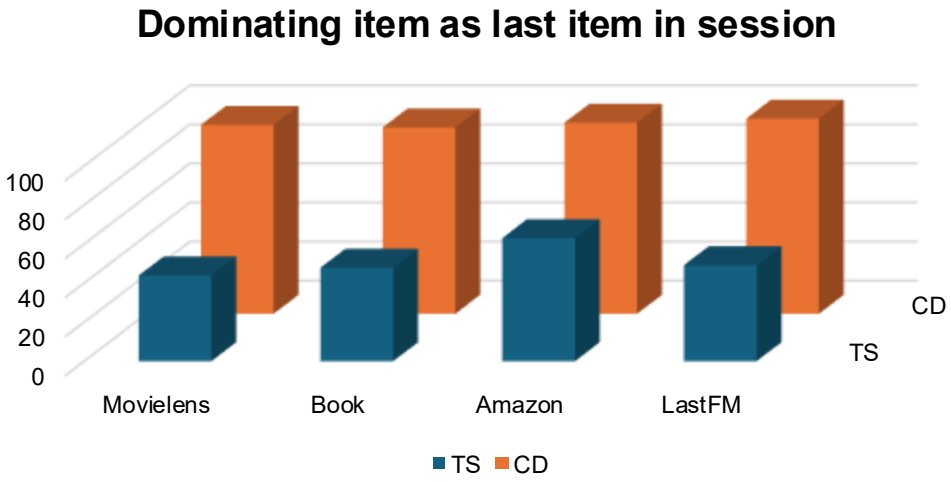
C1, C2, C1	C3, C4, C5, C3
$sess_1$	$sess_2$

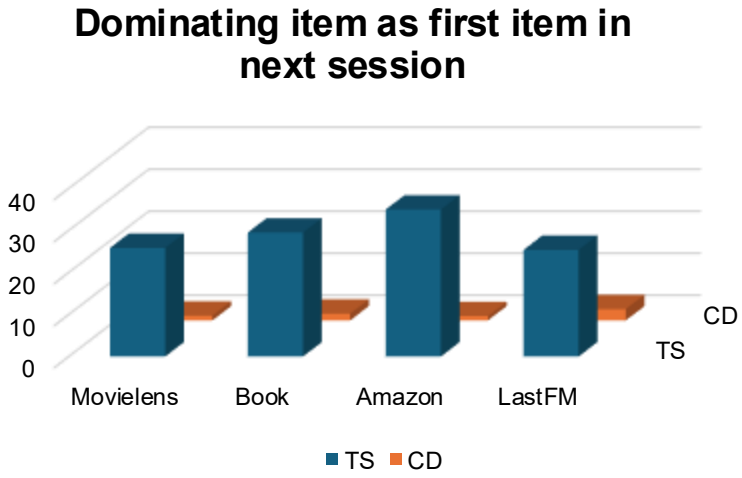
: Dominating category

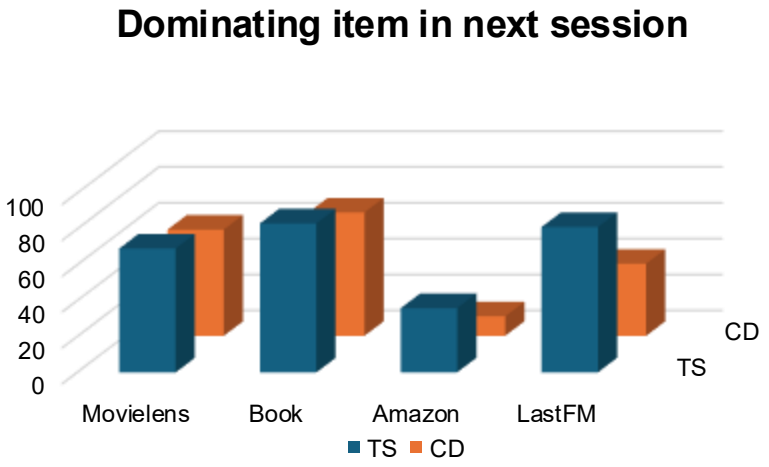
Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Dominating Item in the Datasets

$d \in \mathcal{S}$ (%)	$d \in \mathcal{S}_{N[-1]}$		$d \in \mathcal{S}_{(N+1)[1]}$		$d \in \mathcal{S}_{(N+1)}$	
	TS	CD	TS	CD	TS	CD
Movielens	43.88	96.69	25.85	1.14	69.12	59.30
Book	47.87	95.40	29.51	1.61	83.16	68.86
Amazon	62.98	97.77	34.99	1.04	35.89	10.93
LastFM	49.13	99.85	25.37	2.70	81.11	40.24



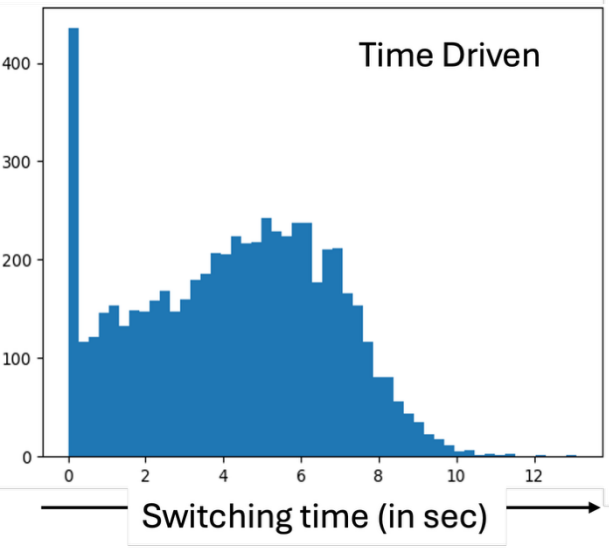


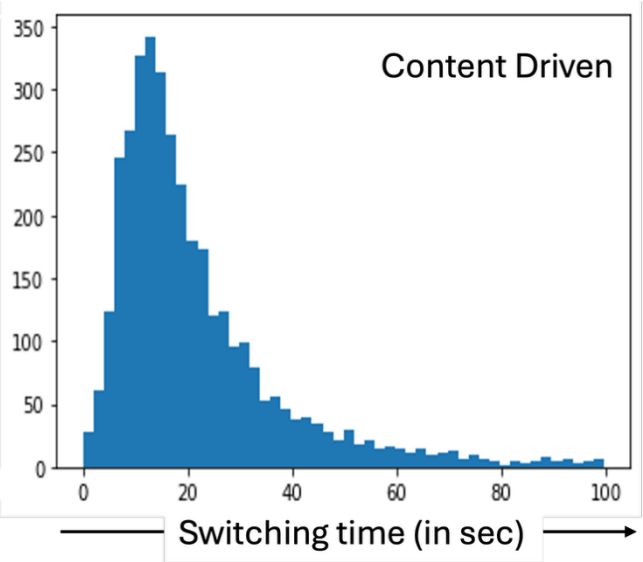


Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

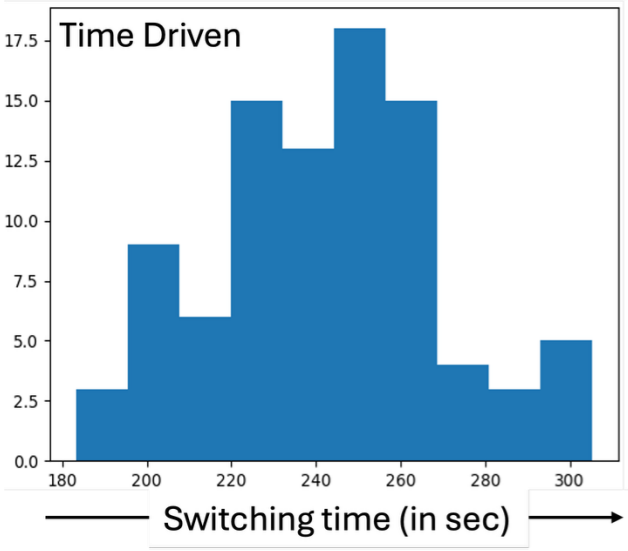
Switching Time Analysis in Both Sessions

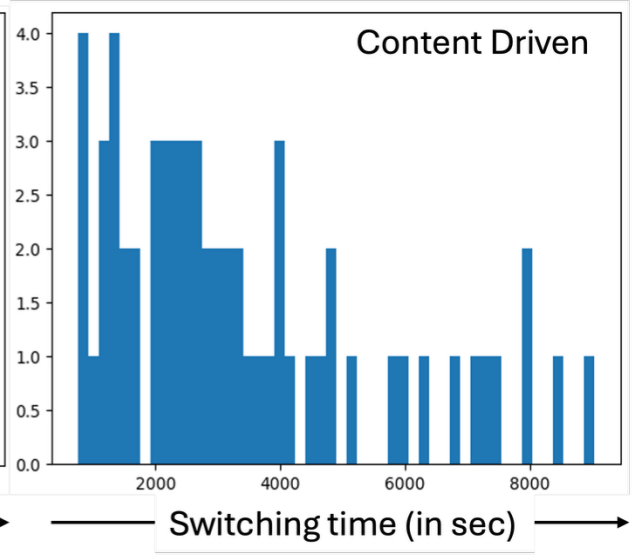
Movielens



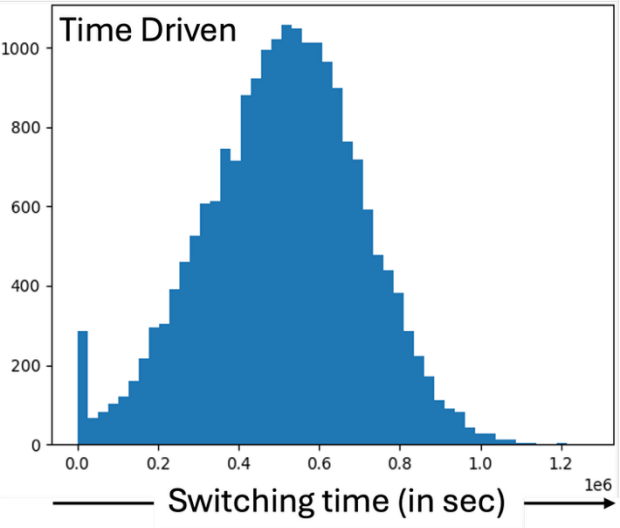


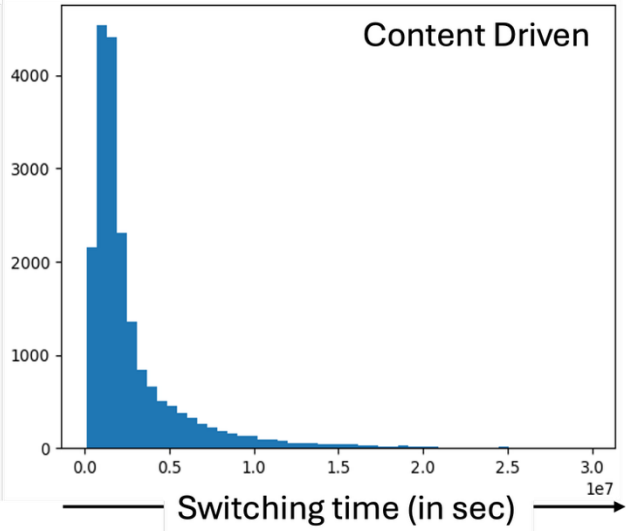
Music



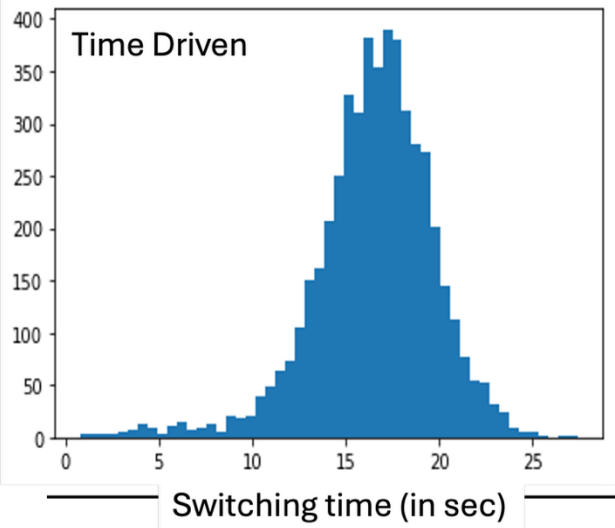


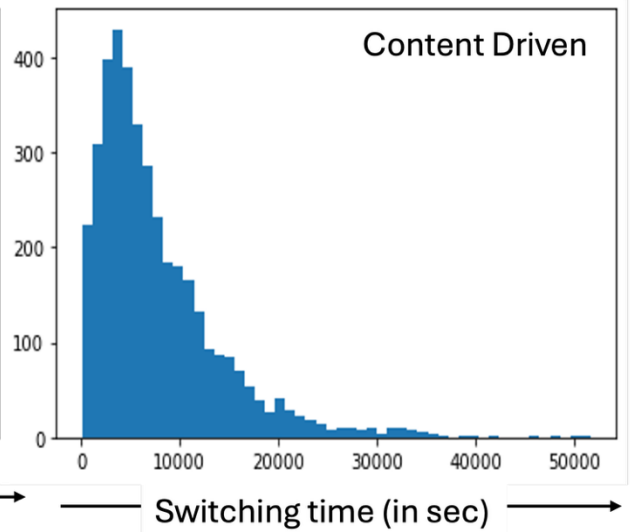
Books



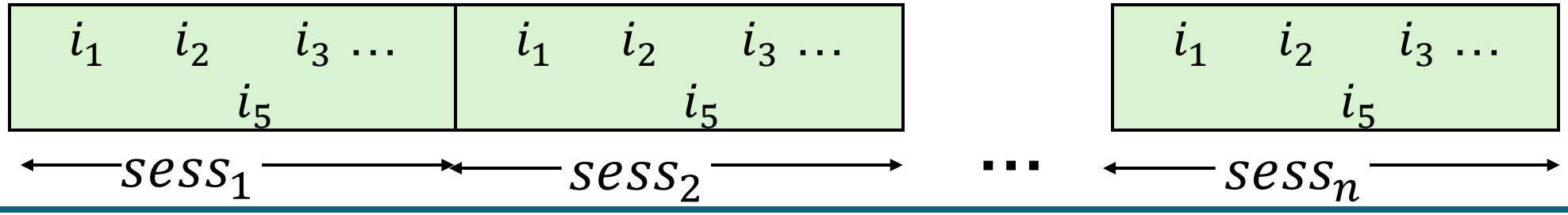


Amazon

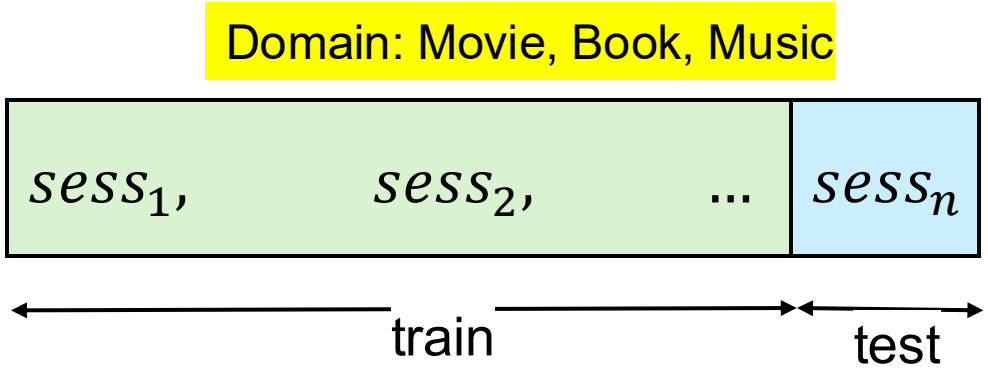




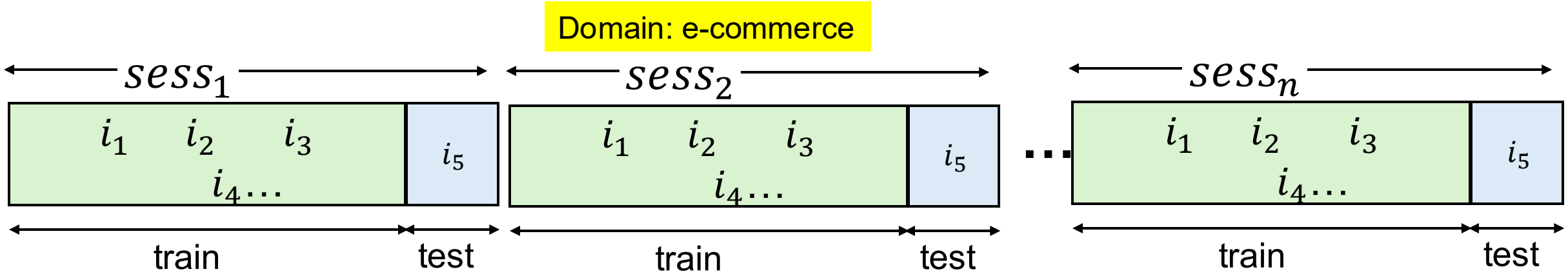
Evaluation



Evaluation:1



Evaluation:2



Singh, B., & Pedanekar, N. (2025, April). Ready for You When You Are Back: Content-Driven Session-Based Recommendation for Continuity of Experience. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 12, pp. 12533-12540).

Cold Start Problems in Recommendation System

Cold Start Scenario

Cold start is the situation where the system lacks sufficient interaction history (ratings, clicks, purchases, etc.) to build reliable user or item representations for recommendation.

User Cold Start

The **User Cold Start** problem occurs when a **new user** joins the system, but the recommender has **little or no interaction history** (ratings, clicks, views, etc.) for that user.

Item Cold Start

The **Item Cold Start** problem occurs when a **new item** is added to the system, but **no user has interacted** with it yet.

Meta Learning:

A mechanism of Learning to Learn...!!!

The **meta-learning-based cold-start problem** refers to the challenge of **quickly adapting** a recommender system to new users or new items that have **few or no historical interactions**, by leveraging **meta-learned knowledge** from previously seen users or items.

Meta-Learning Training Phase

During **meta-training**, MAML (or any similar meta-learning method) learns:

- A **set of meta-parameters** θ^* —a *universal initialization* for your model.
- These parameters are optimized so that the model can **adapt quickly** to any new user (task) with **very few interactions**.

In meta-training, we simulate this process across many users:

- For each user (task) $\mathcal{T}_i$:
  - You split data into **Support Set** S_i)for adaptation) and **Query Set** (for evaluation).
- The meta-objective encourages the model to learn an initialization θ^*that minimizes query losses *after* inner adaptation on support data.
- So after training, we have this:

$$\boldsymbol{\theta}^* = \textit{meta-learned initialization parameters.}$$

What Happens at Inference (Meta-Test) Time:

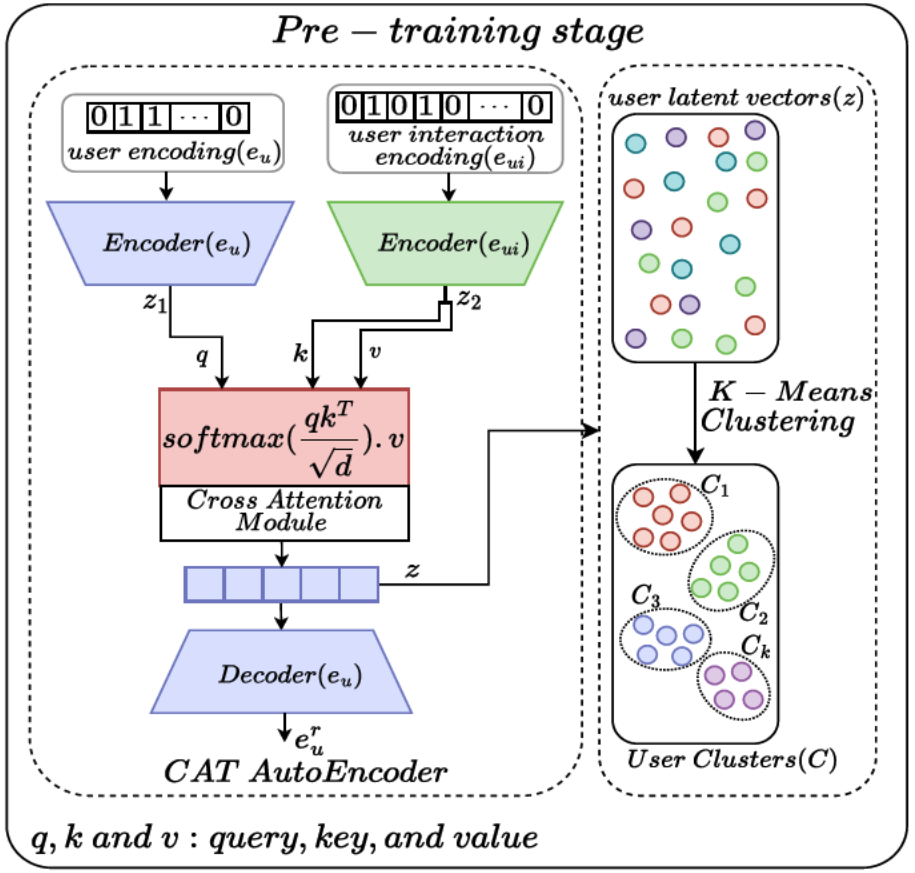
Now a new, unseen user Tnew arrives — this is your cold-start user.

The model has *never seen* this user before, but it knows how to **adapt fast**.
Here's what happens:

Stage	Description
Step 1: Observe a few interactions	The new user provides a few ratings or clicks — this forms the **support set** S_{new} .Example: User likes 3 movies.
Step 2: Initialize model	Load the **meta-learned parameters** θ^*.
Step 3: Adaptation (inner loop)	Perform 1–5 gradient descent steps using the **support set** S_{new} :$\theta'_{new} = \theta^* - \alpha \nabla_\theta \mathcal{L}_{S_{new}}(\theta^*)$ This "fine-tunes" the model to the user's preference.
Step 4: Prediction on new items	Use the *adapted parameters* θ'_{new} to predict ratings or ranking for **unseen items** — this is the actual recommendation step.

Dynamic Task-adaptive Meta Optimization for Cold-Start Recommendation:

Pre-training Stage:



Pointers:

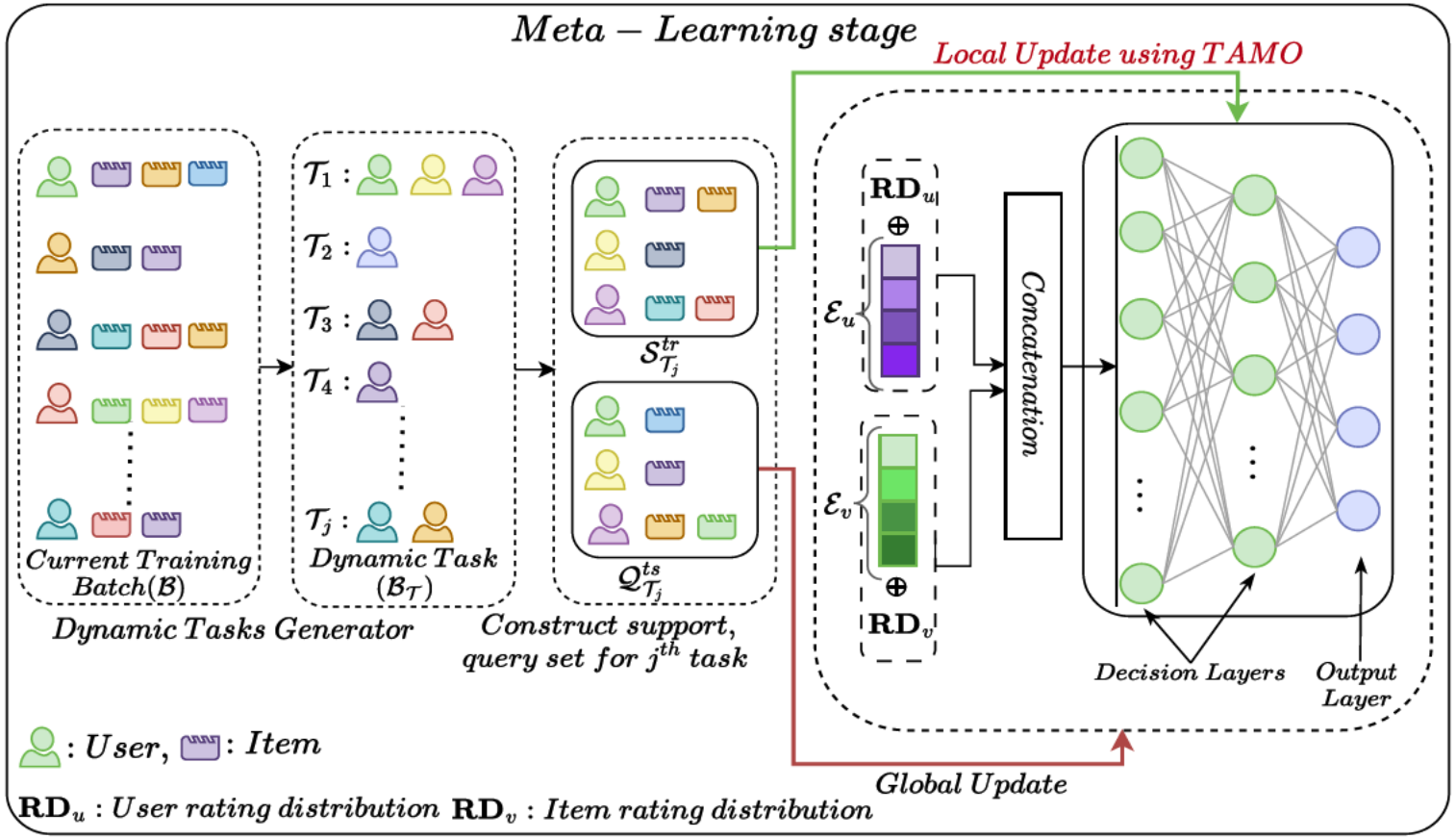
- The CAT autoencoder captures high-order dependencies between user profiles and interaction histories.

- To achieve this, for a user u, we define user encoding (e_u) which is the concatenation of categorical one-hot vector of all user profile attributes, and user interaction encoding, e_{ui}, which is the one-hot vector of the user's interaction history.

$$e_u = [c_1 || \dots || c_p]; \; e_{ui} = [i_1, i_2, \dots, i_m]$$

$$L_u = \sum_{u \in U_w} ||e_u - e_u^r||^2 + \lambda ||w_u||^2$$

Meta Learning Stage:



- **Pointers**

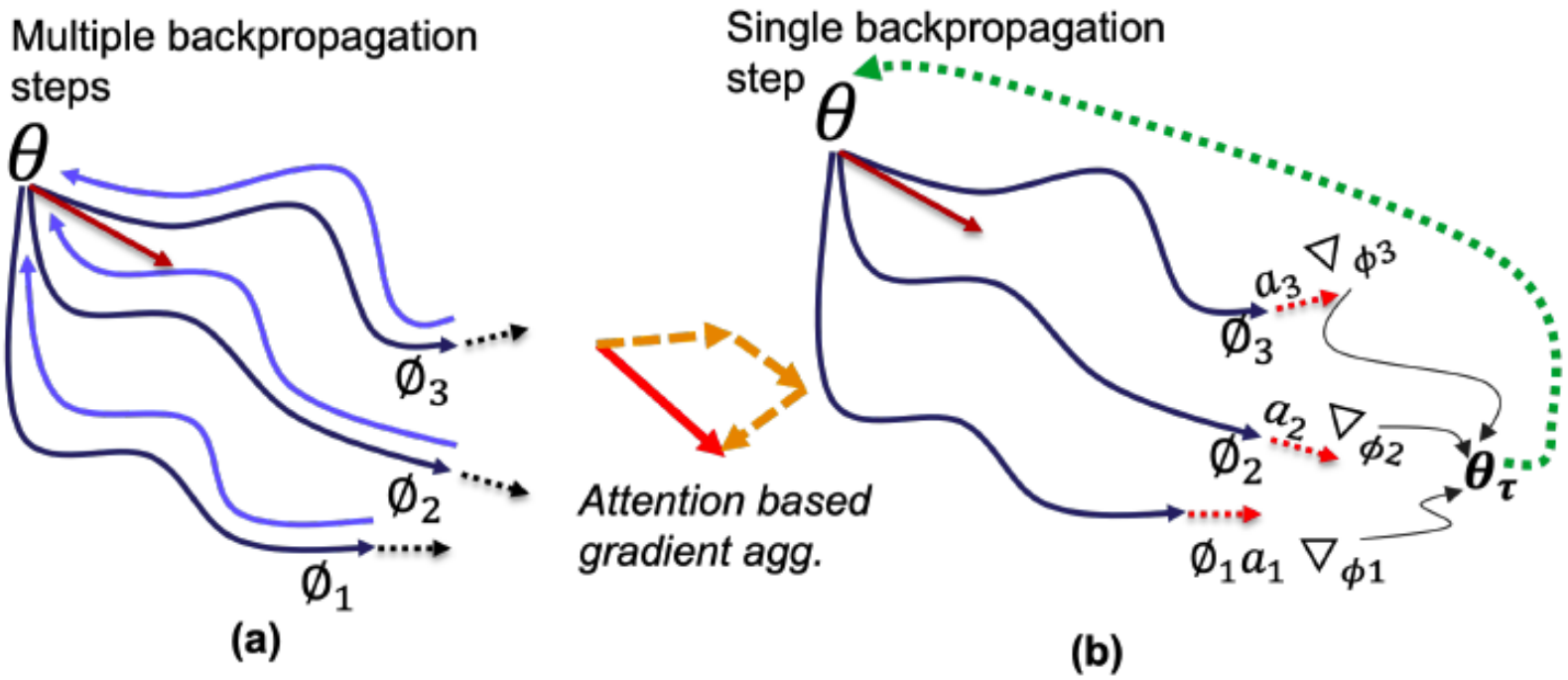
$$\varepsilon_u = [E_{u1}||, \dots, ||E_{uP}]^T; \varepsilon_v = [E_{v1}||, \dots, ||E_{vQ}]^T$$

$$\varepsilon_u^f = [\varepsilon_u || RD_u], \qquad \varepsilon_v^f = [\varepsilon_v || RD_v]$$

$$x_k = ReLU(w_k^T x_{k-1} + b_k), \forall k \in \{1, \dots, N\}$$

$$\widehat{y_{uv}} = \sigma(w_0^T x_N + b_0)$$

Dynamic Task Generator:

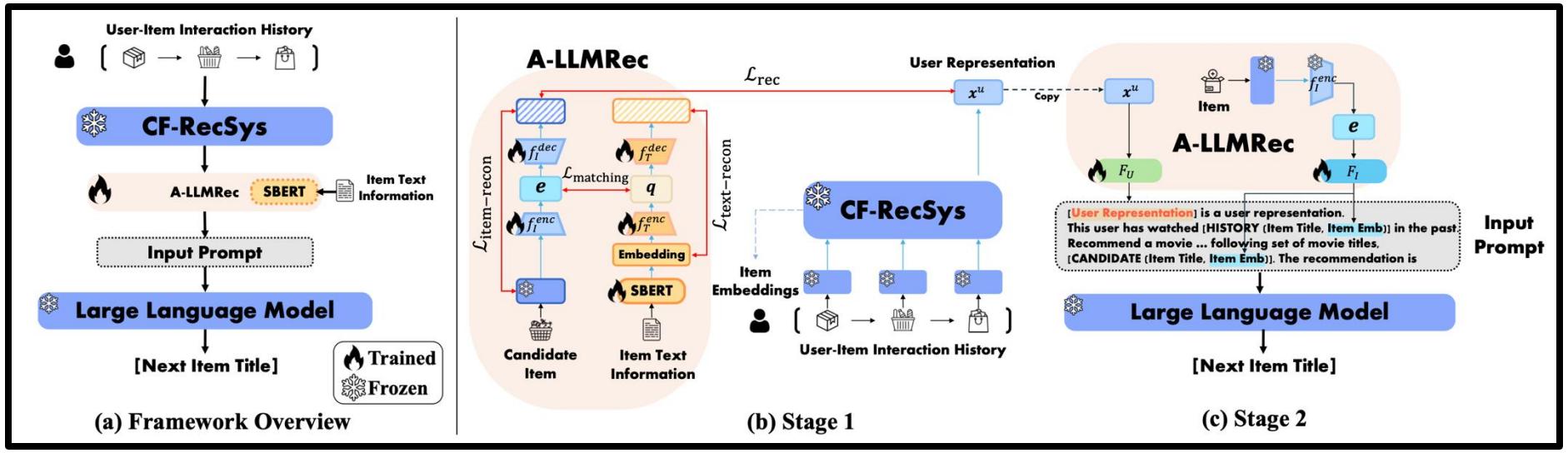


$$\boxed{\theta_{T_i} = \theta - \alpha \sum_{i \in t} a_i \nabla_{\emptyset_i} L_i(f_\theta)}$$

LLMs for Recommendation System

Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System

Sein Lim et al.
KDD-24



Stage-2

$$\max_{\theta} \sum_{\mathcal{S}^u \in \mathcal{S}} \sum_{k=1}^{|y^u|} log(P_{\theta,\Theta}(y_k^u | p^u, y_{<k}^u))$$

Stage-1

a). Alignment between Collaborative and Textual Knowledge

$$\mathcal{L}_{\text{matching}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} [MSE(\mathbf{e}_i, \mathbf{q}_i)] \right] = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} [MSE(f_I^{enc}(\mathbf{E}_i), f_T^{enc}(\mathbf{Q}_i))] \right]$$

b). Avoiding Over-smoothed Representation

$$\mathcal{L}_{\text{item-recon}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{E}_i, f_I^{dec}(f_I^{enc}((\mathbf{E}_i))) \right] \right]$$

$$\mathcal{L}_{\text{text-recon}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{Q}_i, f_T^{dec}(f_T^{enc}((\mathbf{Q}_i))) \right] \right]$$

c). Recommendation Loss

$$\mathcal{L}_{\text{rec}} = -\sum_{\mathcal{S}^u \in \mathcal{S}} \left[log(\sigma(s(\mathbf{x}^u_{|\mathcal{S}^u|-1}, f_I^{dec}(f_I^{enc}(\mathbf{E}_{i^u_{|\mathcal{S}^u|}}))))) + log(1 - \sigma(s(\mathbf{x}^u_{|\mathcal{S}^u|-1}, f_I^{dec}(f_I^{enc}(\mathbf{E}_{i^{u,-}_{|\mathcal{S}^u|}}))))) \right]$$

$$\mathcal{L}_{\text{stage-1}} = \mathcal{L}_{\text{matching}} + \alpha \mathcal{L}_{\text{item-recon}} + \beta \mathcal{L}_{\text{text-recon}} + \mathcal{L}_{\text{rec}}$$

PROPOSED METHOD: A-LLMREC

- **Alignment between Collaborative and Textual Knowledge (Stage-1):**

To align the item embeddings from a frozen CF-RecSys with their associated text information to capture both collaborative and textual knowledge. item encoder f_I^{enc} and text encoder f_T^{enc} , each containing a 1-layer Multi-Layer Perceptron (MLP), to align the item embeddings from a frozen CF-RecSys with the text embeddings from SBERT.

$$\mathcal{L}_{\text{matching}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{e}_i, \mathbf{q}_i) \right] \right]$$
$$= \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(f_I^{enc}(\mathbf{E}_i), f_T^{enc}(\mathbf{Q}_i)) \right] \right]$$

- **Avoiding Over-smoothed Representation**

$$\mathcal{L}_{\text{item-recon}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{E}_i, f_I^{dec}(f_I^{enc}((\mathbf{E}_i))) \right] \right]$$
$$\mathcal{L}_{\text{text-recon}} = \mathop{\mathbb{E}}_{\mathcal{S}^u \in \mathcal{S}} \left[\mathop{\mathbb{E}}_{i \in \mathcal{S}^u} \left[MSE(\mathbf{Q}_i, f_T^{dec}(f_T^{enc}((\mathbf{Q}_i))) \right] \right]$$

Kim, S., Kang, H., Choi, S., Kim, D., Yang, M., & Park, C. (2024, August). Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 1395-1406).

- **Recommendation Loss:**

$$\mathcal{L}_{\text{rec}} = -\sum_{\mathcal{S}^u \in \mathcal{S}} \Big[log(\sigma(s(\mathbf{x}^u_{|\mathcal{S}^u|-1}, f_I^{dec}(f_I^{enc}(\mathbf{E}_{i^u_{|\mathcal{S}^u|}}))))) + log(1 - \sigma(s(\mathbf{x}^u_{|\mathcal{S}^u|-1}, f_I^{dec}(f_I^{enc}(\mathbf{E}_{i^{u,-}_{|\mathcal{S}^u|}}))))) \Big]$$

→ where $\mathbf{x}^u_{|\mathcal{S}^u|-1} = \text{CF-RecSys}(\mathcal{S}^u_{1:|\mathcal{S}^u|-1}) \in \mathbb{R}^d$ is the user representation extracted from the collaborative filtering recommender

→ $e_i = f_I^{enc}(E_i)$ is the joint collaborative-text embedding of item i, which will be passed to LLM a input.

- **Final Loss of Stage-1:**

$$\mathcal{L}_{\text{stage-1}} = \mathcal{L}_{\text{matching}} + \alpha \mathcal{L}_{\text{item-recon}} + \beta \mathcal{L}_{\text{text-recon}} + \mathcal{L}_{\text{rec}}$$

Kim, S., Kang, H., Choi, S., Kim, D., Yang, M., & Park, C. (2024, August). Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 1395-1406).

Alignment between Joint Collaborative-Text Embedding and LLM (Stage-2)

LLM Input:	[User Representation] is a user representation. This user has watched **[HISTORY (Item Titles, Item Emb)]** in the past. Recommend a movie for this user to watch next from the following set of movie titles, **[CANDIDATE (Item Titles, Item Emb)].** The recommendation is
LLM Output:	**[Next Item Title]**

Goal in Stage-2 is to align these joint embeddings with the token space of the LLM and design a prompt that allows the LLM to solve the recommendation task by leveraging the learned collaborative knowledge

- **Projecting collaborative knowledge onto the token space of LLM**

$$\boldsymbol{O_u} = \boldsymbol{F_U}(\boldsymbol{x^u}), \boldsymbol{O_i} = \boldsymbol{F_I}(\boldsymbol{e_i})$$

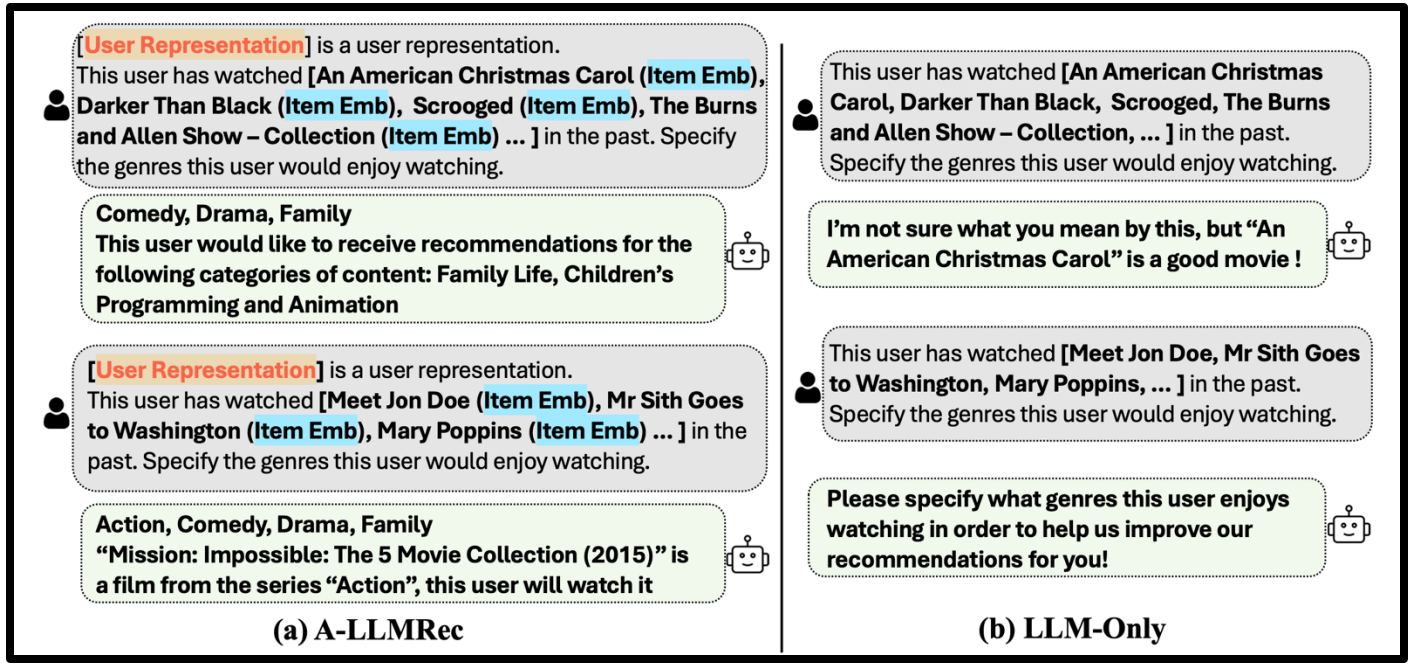
$$F_U\colon R^d \rightarrow R^{d^{token}}$$
$$F_I\colon R^{d'} \rightarrow R^{d^{token}}$$

- **Prompt Design for Integrating Collaborative Knowledge**

$$\max_{\theta} \sum_{\mathcal{S}^u \in \mathcal{S}} \sum_{k=1}^{|y^u|} log(P_{\theta,\Theta}(y_k^u | p^u, y_{<k}^u))$$

where θ denotes the learnable parameters of F_U and F_I, Θ is the frozen parameters of LLM, p^u and y^u are the input prompt and the next item title of user u, respectively. y_k^u is the k-th token of y^u and $y_{<k}^u$ represents the tokens before y_k^u. Note that we only use the last item of each user sequence to train Equation 8 for efficiency.

Results



A-LLMRec v.s. LLM-Only on the favorite genre prediction task (Movies and TV dataset used).

Table 1: Overall model performance (Hit@1) over various datasets. The best performance is denoted in bold.

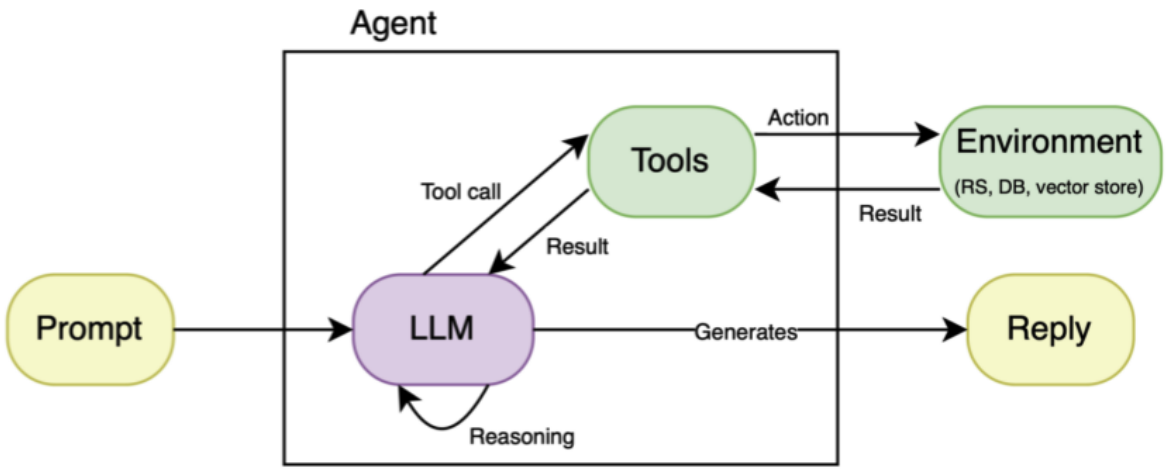
	Collaborative filtering				Modality-aware			LLM-based			
	NCF	NextItNet	GRU4Rec	SASRec	MoRec	CTRL	RECFORMER	LLM-Only	TALLRec	MLP-LLM	A-LLMRec
Movies and TV	0.4273	0.5855	0.5215	0.6154	0.4130	0.3467	0.4865	0.0121	0.2345	0.5838	**0.6237**
Video Games	0.3159	0.4305	0.4026	**0.5402**	0.4894	0.2354	0.4925	0.0168	0.4403	0.4788	0.5282
Beauty	0.2957	0.4231	0.4131	0.5298	0.4997	0.3963	0.4878	0.0120	0.5542	0.5548	**0.5809**
Toys	0.1849	0.1415	0.1673	0.2359	0.1728	0.1344	0.2871	0.0141	0.0710	0.3225	**0.3336**

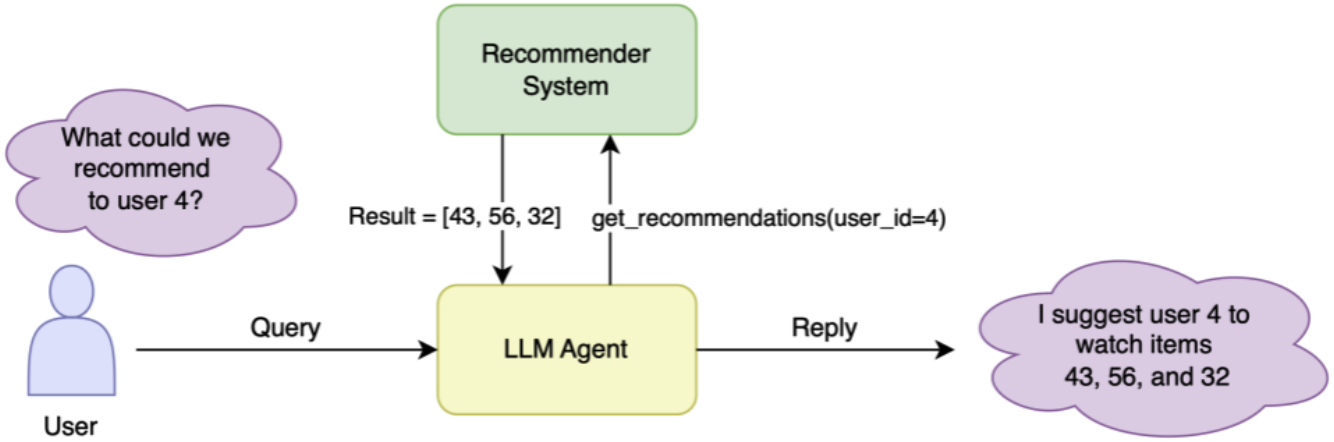
Kim, S., Kang, H., Choi, S., Kim, D., Yang, M., & Park, C. (2024, August). Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 1395-1406).

Large Language Model-based Recommendation System Agents

Carraro, T., Singh, B., & Pedanekar, N. (2025, September). Large Language Model-based Recommendation System Agents. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems* (pp. 1340-1342).

Recommendation System Agent





- An Agent is an LLM that can access external tools to enlarge its context/memory.
- These tools are Python code
- Possible Tools:
  - Recommendation engine
  - MySQL database
  - Vector Store Database

Carraro, T., Singh, B., & Pedanekar, N. (2025, September). Large Language Model-based Recommendation System Agents. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems* (pp. 1340-1342).

System Prompt

- **General rules to instruct the model about how to call tools**
  The default number of recommended items is 5
  The recommendation tool always needs a user ID
- **Tool call planning rules** -> the LLM must explain the tool call plan before execution
- **Examples of tool calls**
  Query: Recommend some horror movies to user 3
  Tool calls: item_filter -> recommendation -> item_metadata_retrieval
- **Explanation rules** -> determine how recommendations should be explained

Available Tools

Tool name	Input	Output
Recommendation	User ID, k, item IDs	Ranking of k items
Vector store search	Query	Top 10 matching items
Item metadata retrieval *	Item IDs, item feature labels	Requested item metadata
User metadata retrieval *	User ID, user feature labels	Requested user metadata
Like percentage computation	Item IDs	Percentage of users that like the given items
Item filter	Item feature filters	IDs of items satisfying the given filters
User history retrieval	User ID	20 most recent items the user interacted with
Get popular items	Popularity type, k, item IDs	Return the k most popular items

***Item features:** title, genres, actors, directors, producers, description, storyline, release date, IMDb rating, duration, country
***User features:** gender, age group

Carraro, T., Singh, B., & Pedanekar, N. (2025, September). Large Language Model-based Recommendation System Agents. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems* (pp. 1340-1342).

Standard recommendation and explanation

This use case shows how the agent behaves when standard recommendations are requested. In this case, we ask recommendations for user 8. The agent understands it has to call:

1. The recommendation tool (i.e., get_top_k_recommendations_tool) to get the top 5 (default value) recommended items for user 8.

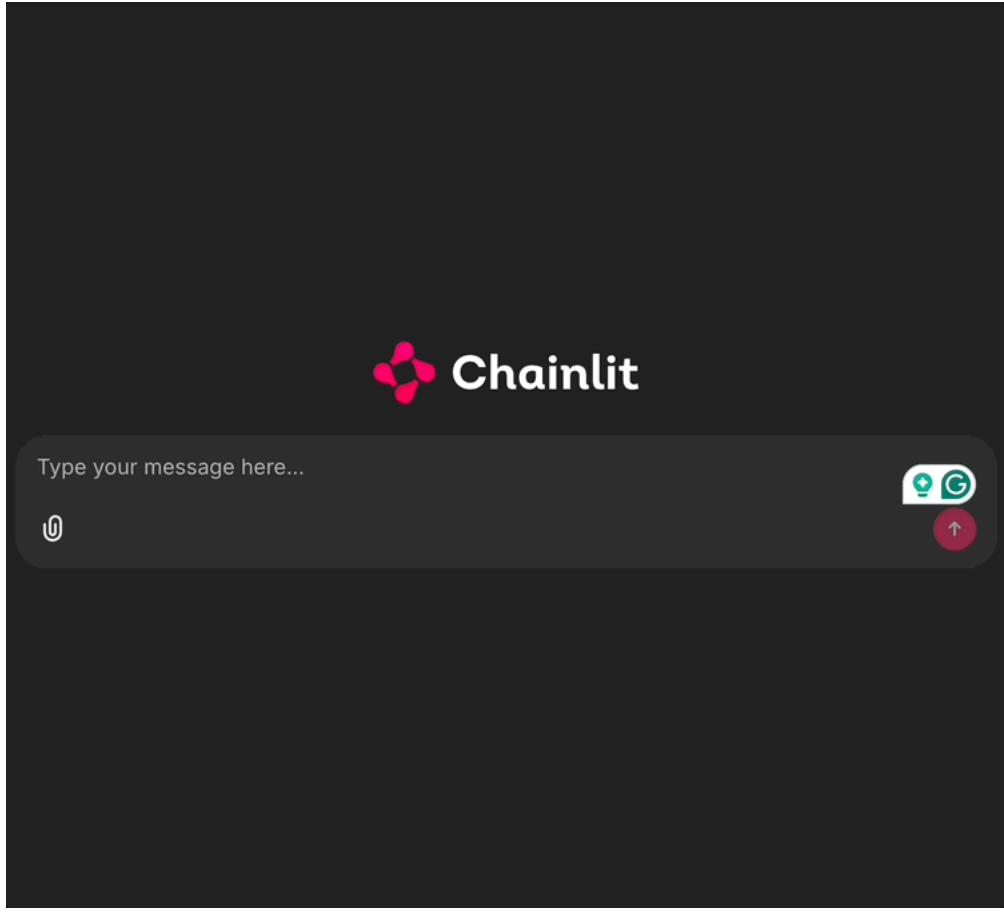
2. The item metadata retrieval tool (i.e, get_item_metadata_tool) to get useful information to display regarding these items.

After the recommendation, the agent asks the user whether he/she would like to get an explanation. Since the user positively replies, the agent proceeds with the explanation. The process consists of:

1. Calling the historical interactions retrieval tool (i.e., get_interacted_items_tool) to get the most recent 20 interactions of user 8;

2. Calling the item metadata retrieval tool (i.e., get_item_metadata_tool) to get content features of these interacted items;

3. Comparing content features of interacted and recommended items to provide a content-based and personalized explanation to the user.



Carraro, T., Singh, B., & Pedanekar, N. (2025, September). Large Language Model-based Recommendation System Agents. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems* (pp. 1340-1342).

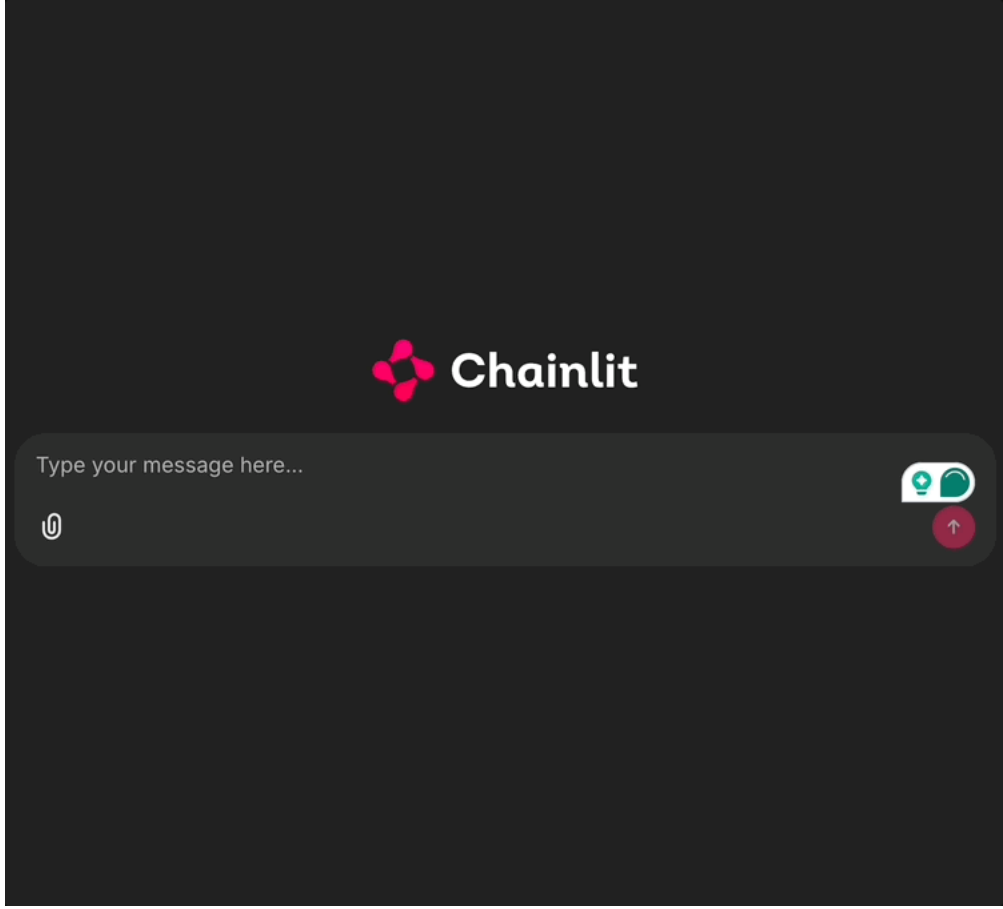
Constrained recommendation (single filter)

This use case shows how the agent behaves when constrained recommendations are requested. In this case, we ask for horror movie recommendations for user 9. The agent understands it has to call:

1. The item filtering tool (i.e., item_filter_tool) to get the item IDs of horror movies in the dataset;

2. The recommendation tool (i.e., get_top_k_recommendations_tool) to get the top 5 horror movie recommendations for user 9. In this step, the previously retrieved item IDs are given to the recommendation tool to restrict ranking computation;

3. The item metadata retrieval tool (i.e., get_item_metadata_tool) to get useful information to display the recommended movies.



Carraro, T., Singh, B., & Pedanekar, N. (2025, September). Large Language Model-based Recommendation System Agents. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems* (pp. 1340-1342).

Thank You

Email: brijraj.08@gmail.com