



THE **WEB** ACM  
CONFERENCE

**Tutorial**



THE UNIVERSITY  
OF QUEENSLAND  
AUSTRALIA

# Self-Supervised Learning in Recommender Systems

**Fundamentals and Advances**

Presenter: **Junliang Yu, Hongzhi Yin, Tong Chen**

**Responsible Big Data Intelligence Lab**

**The University of Queensland**

Tutorial website: <https://ssl-recsys.github.io/>

# ■ Presenters



**Junliang Yu**

- A third-year PhD student

School of Information Technology and  
Electrical Engineering, UQ

 [jl.yu@uq.edu.au](mailto:jl.yu@uq.edu.au)

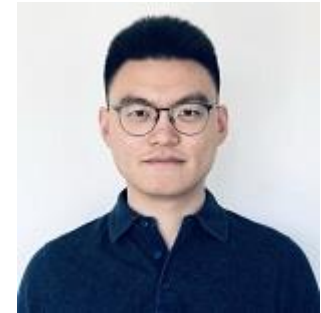


**Dr. Hongzhi Yin**

- Associate Professor

School of Information Technology and  
Electrical Engineering, UQ


 [h.yin1@uq.edu.au](mailto:h.yin1@uq.edu.au)



**Dr. Tong Chen**

- Lecturer

School of Information Technology and  
Electrical Engineering, UQ

 [tong.chen@uq.edu.au](mailto:tong.chen@uq.edu.au)

# CONTENTS

- **Background and History**
- **Taxonomy of Self-Supervised Recommendation (SSR)**
- **Data Augmentation Techniques**
- **SSR Methods**
  - Contrastive, Generative, Predictive, and Hybrid
- **A Library for SSR**
- **Limitations and Future Research**
- **Q&A**



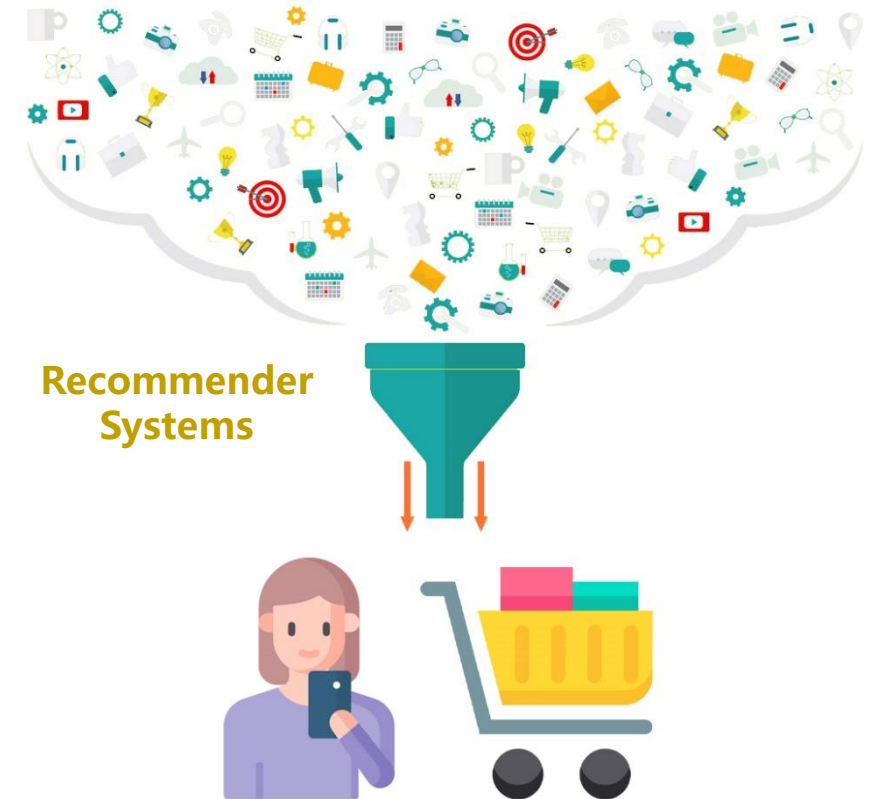
01

# Background and History

**Tutorial**

# ■ Recommender Systems

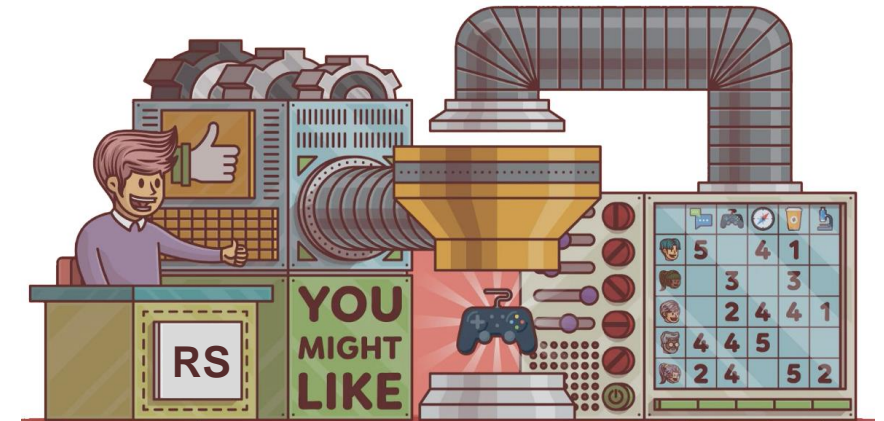
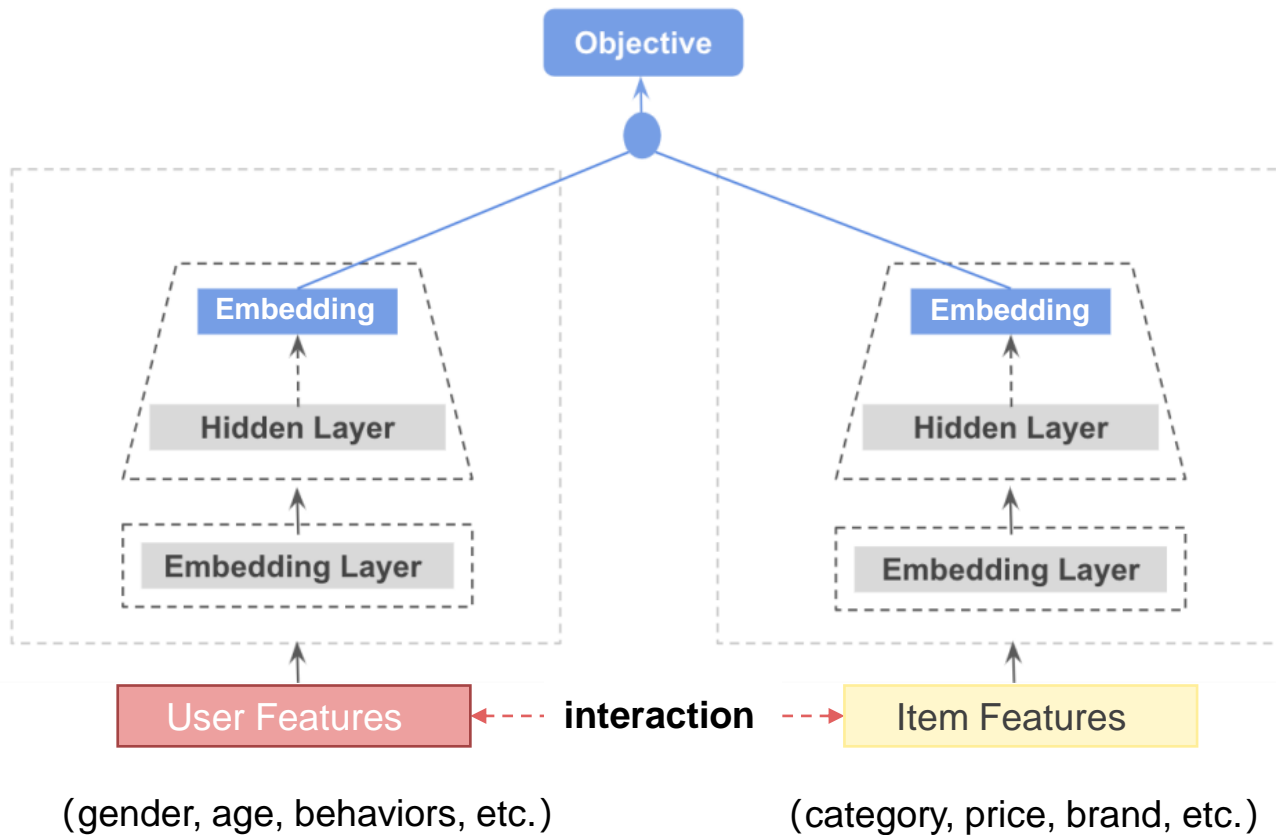
We are now living in an age of information explosion.



Recommender systems can discover users' interests and ease the way of decision-making.

# Recommender Systems

A general neural-architecture for recommendation.



Use the historical interaction data to compute item scores and recommend.

# ■ Problem of Data Sparsity

Most users generally only consume a tiny fraction of countless items.

									
	?		✓		✓			?	
		?	✓				✓		✓
	✓		?		✓		?	✓	
		✓				✓			?
	✓	✓		?					
		?		✓		?			✓

How to generate satisfactory recommendations with very sparse interactions?



# ■ Self-Supervised Learning (SSL)

## What is it?

Self-Supervised Learning (SSL) is a special type of representation learning that enables learning good data representation from unlabeled data.

## Why do we need it?

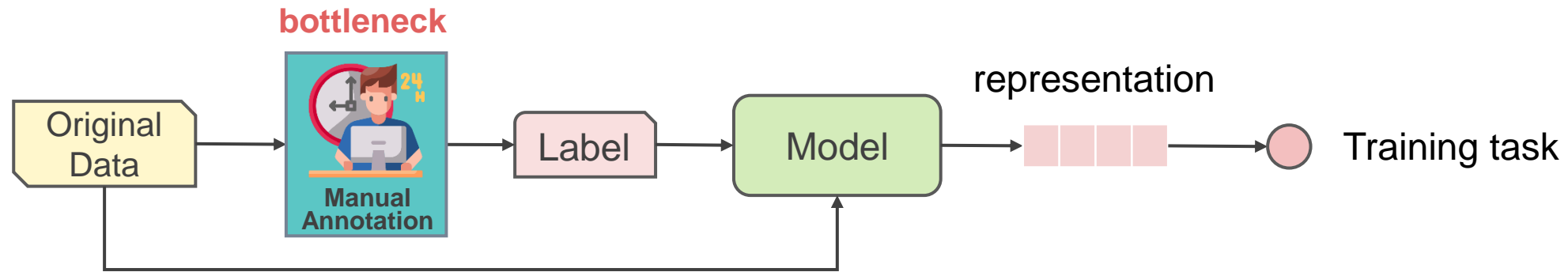
Data acquisition in recommender systems is costly since the data can only be generated by users themselves.

Learning good representation makes it easier to transfer useful information hiding in the unlabeled data to recommendation.

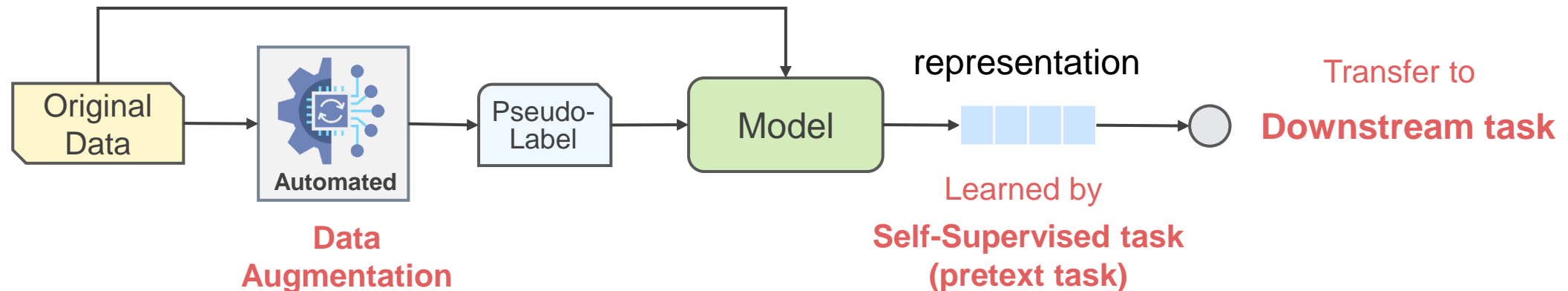


# ■ Self-Supervised Learning (SSL)

## Supervised learning workflow



## Self-Supervised learning workflow



# ■ Self-Supervised Recommendation (SSR)

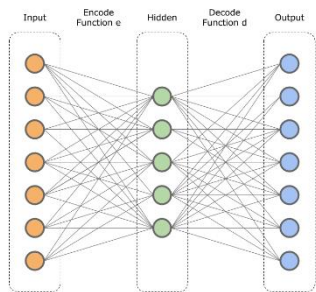
The principle of SSL is well-aligned with recommender systems' needs for more annotated data.

Self-supervised recommendation has following essential features<sup>1</sup>:

- Obtain more supervision signals by semi-automatically exploiting the raw data itself.
- Have a pretext task(s) to (pre-)train the recommendation model with the augmented data.
- Recommendation task is the unique primary task and the pretext task plays a secondary role to enhance recommendation.

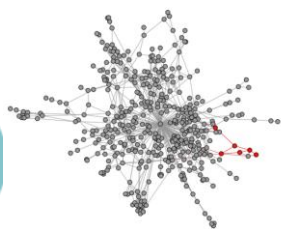
# Development of SSR

## Autoencoder-Based

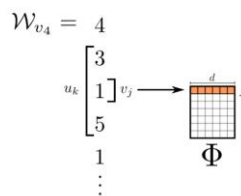


## Network-Embedding-Based

2017



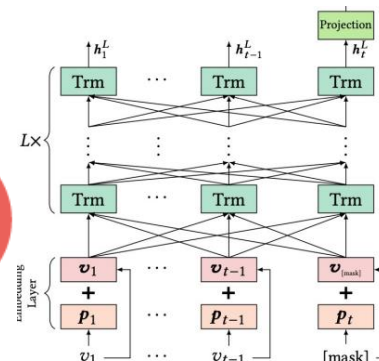
(a) Random walk generation.



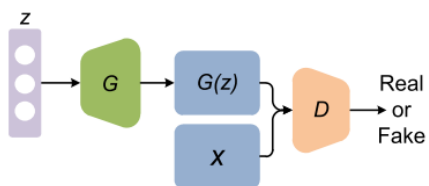
(b) Representation mapping.

2019

## BERT-Based



## GAN-Based



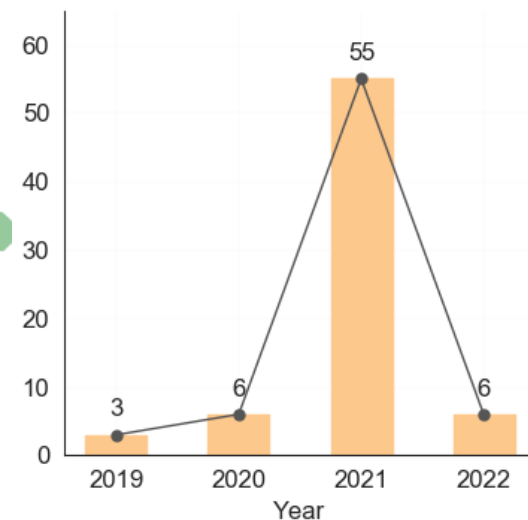
2018

**Self-Supervised Learning  
becomes an independent  
concept**

2020

Early prototypes and implementations

Article number (By 03/2022)



Becoming Diverse

02

# Taxonomy of Self-Supervised Recommendation

Tutorial

# Architecture of SSR

Unified Architecture: **Encoder** + **Projection Head**  
(GNNs, Transformer, etc.) (nonlinear/linear mapping, identical mapping, etc.)

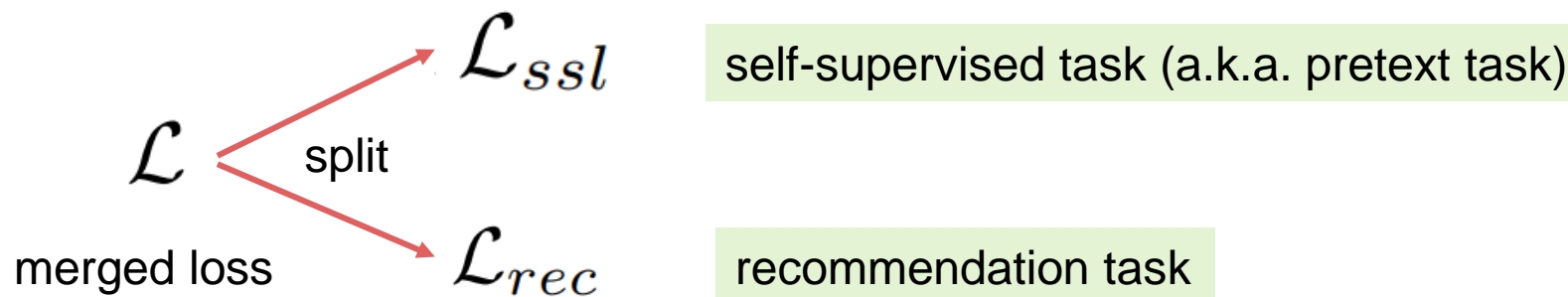
$$f_{\theta^*}, g_{\phi^*}, \mathbf{H}^* = \arg \min_{f_{\theta}, g_{\phi}} \mathcal{L} \left( g_{\phi} \left( f_{\theta} (\mathcal{D}, \tilde{\mathcal{D}}) \right) \right)$$

representation      projection-head      encoder      original data      augmented data

$$\tilde{\mathcal{D}} \sim \mathcal{T}(\mathcal{D})$$

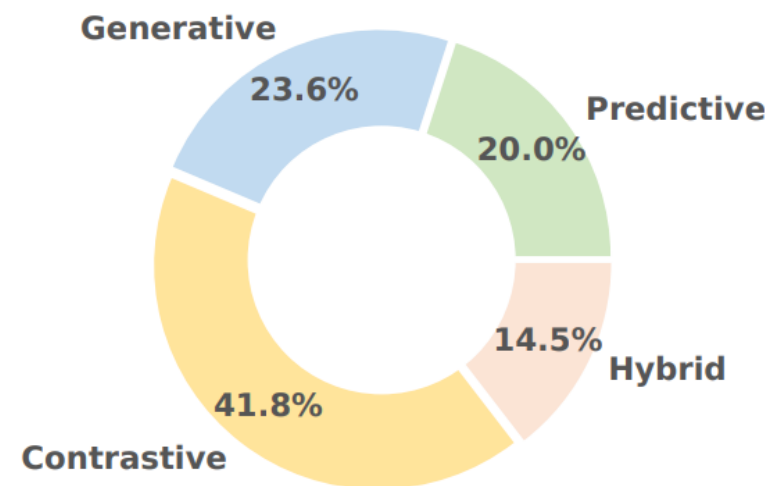
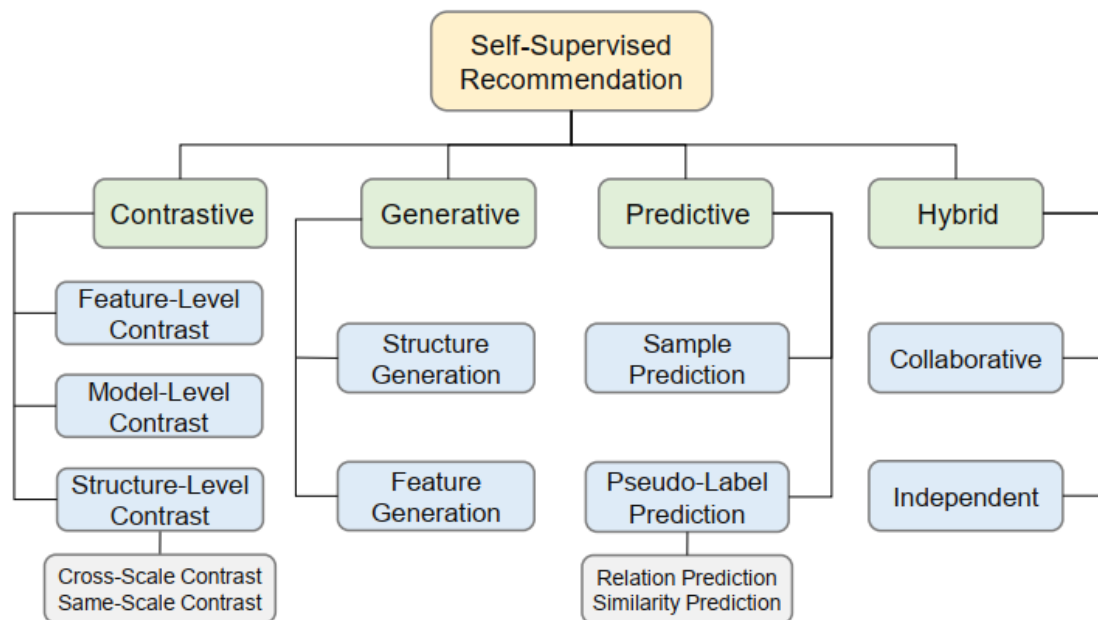
augmentation module

Multi-Task Learning:



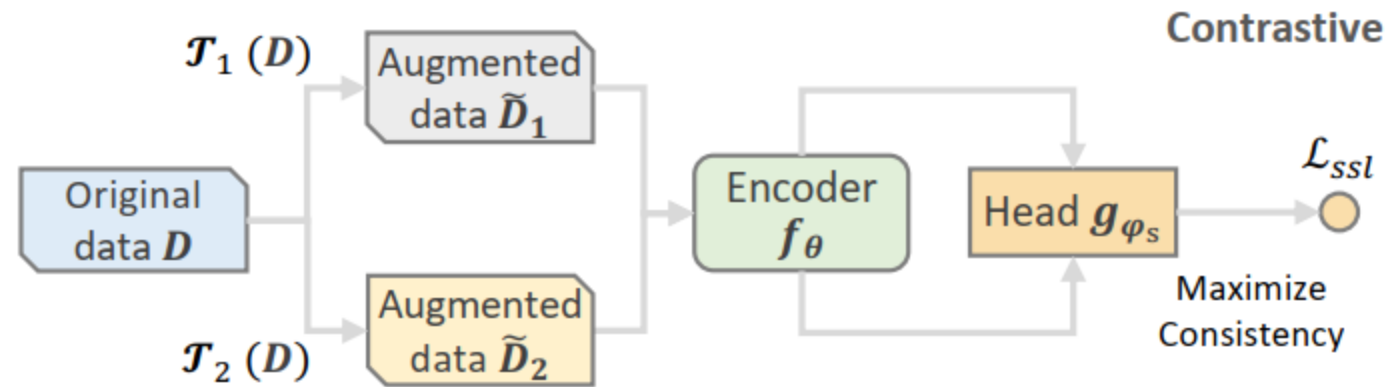
# ■ Taxonomy of SSR

According to the characteristics of pretext tasks, existing SSR methods can be divided into four categories: **contrastive**, **predictive**, **generative**, and **hybrid**.



# Taxonomy of SSR

## Contrastive Methods



To treat every instance (e.g., user/item/sequence) as a class, and then pull views of the same instance closer in the embedding space, and push views of different instances apart.

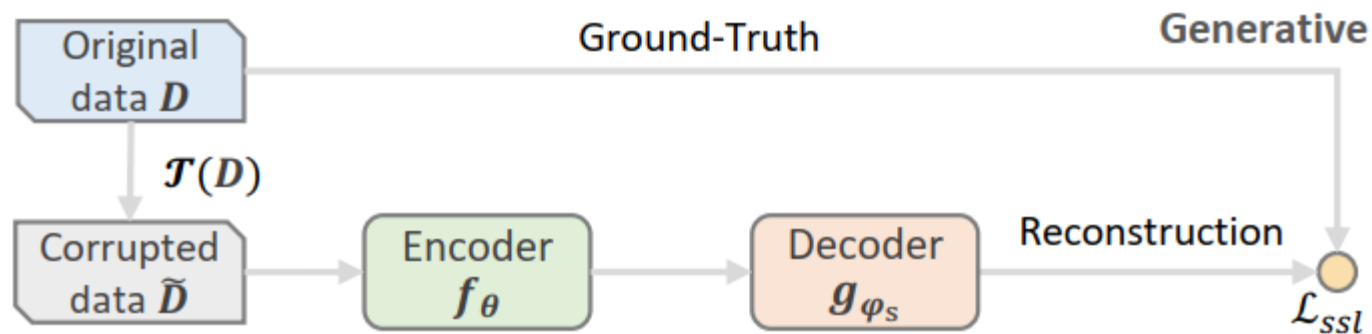
$$f_\theta^* = \arg \min_{f_\theta, g_{\phi_s}} \mathcal{L}_{ssl} \left( g_{\phi_s} \left( f_\theta(\tilde{D}_1) \right), f_\theta(\tilde{D}_2) \right)$$

Estimates the mutual information between views.

Views are created by imposing different transformations on the original data.

# ■ Taxonomy of SSR

## Generative Methods



To reconstruct the original user/item profile with its corrupted versions.

$$f_\theta^* = \arg \min_{f_\theta, g_{\phi_s}} \mathcal{L}_{ssl} \left( g_{\phi_s} (f_\theta(\tilde{\mathcal{D}})), \mathcal{D} \right)$$

Learn to predict a portion of the available data from the rest.

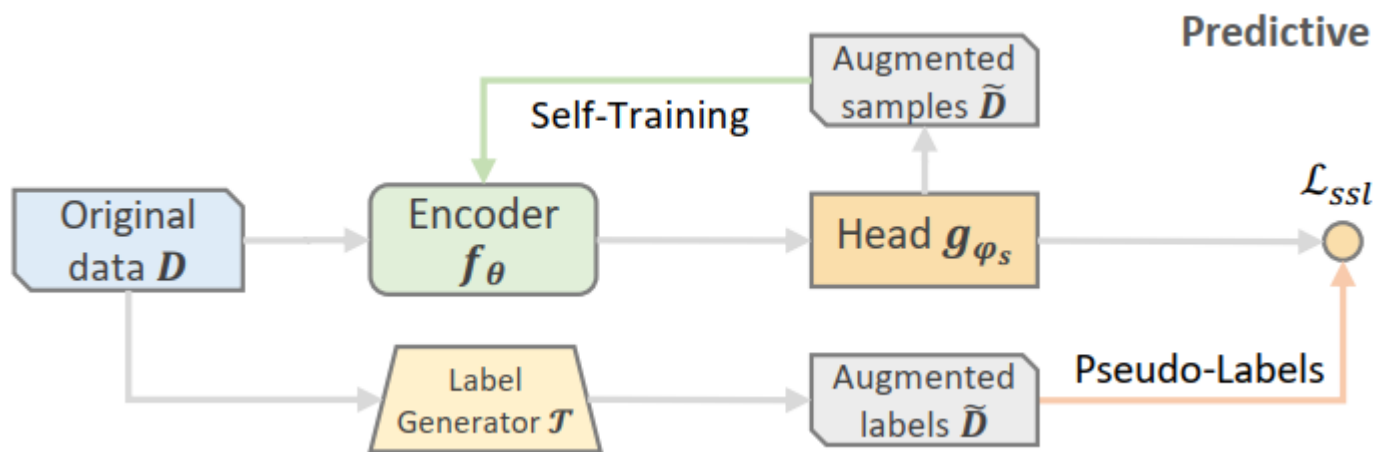
Corrupted data

**Example: BERT-based Models**



# ■ Taxonomy of SSR

## Predictive Methods



To generate new samples or labels from the original data in order to guide the pretext task.

$$f_\theta^* = \arg \min_{f_\theta, g_{\phi_s}} \mathcal{L}_{ssl}(g_{\phi_s}(f_\theta(\mathcal{D})), \tilde{\mathcal{D}})$$

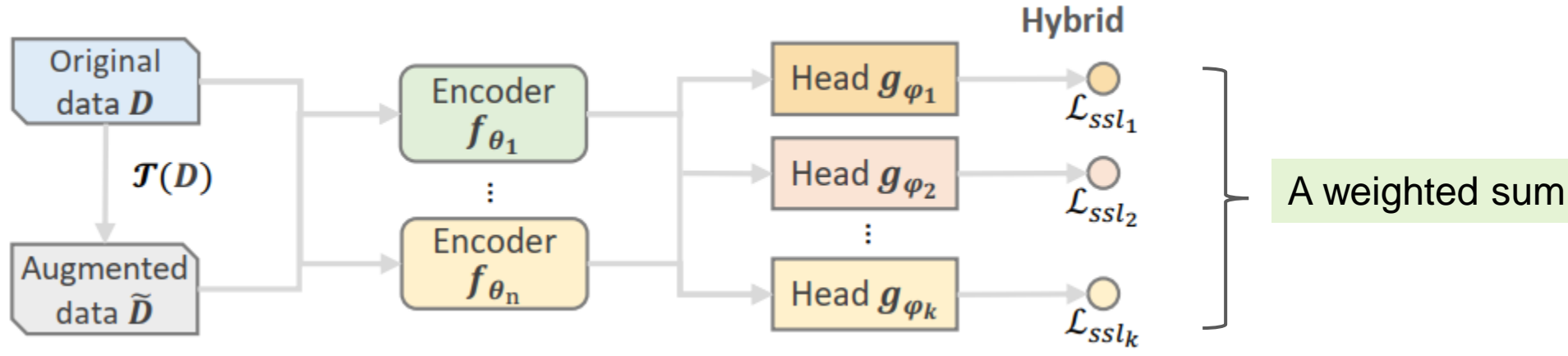
Predict the augmented pseudo-labels.

Augmented pseudo-labels

# Taxonomy of SSR

## Hybrid Methods

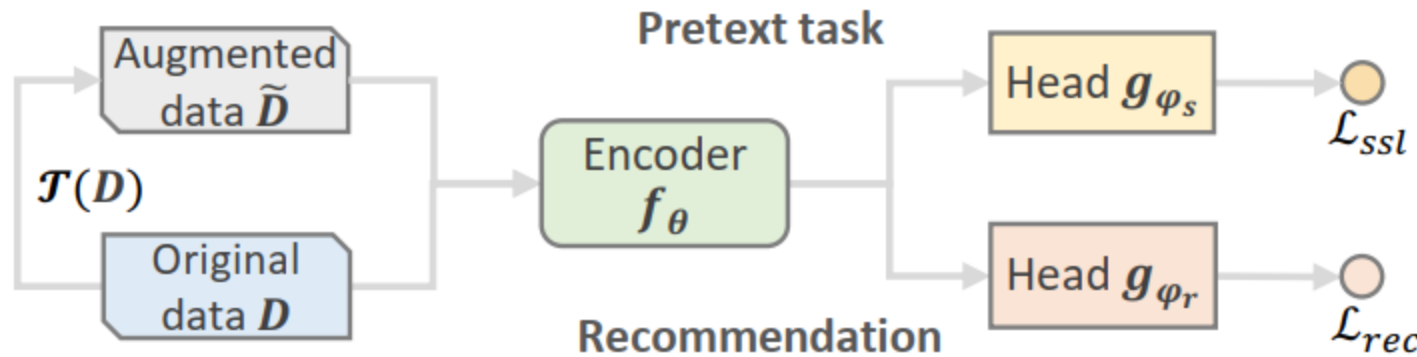
More than one encoder and projection head are needed.



To obtain comprehensive self-supervision by combining different pretext tasks and integrating them into one recommendation model.

# Typical Training Schemes

**Joint Learning (JL)** mostly used in contrastive methods.



The pretext task and the recommendation task are jointly optimized with a shared encoder.

$$f_{\theta^*}, g_{\phi_r^*}, \mathbf{H}^* = \arg \min_{f_\theta, g_\phi} [\mathcal{L}_{rec}(g_{\phi_r}(f_\theta(\mathcal{D}))) + \alpha \mathcal{L}_{ssl}(g_{\phi_s}(f_\theta(\tilde{\mathcal{D}})))]$$

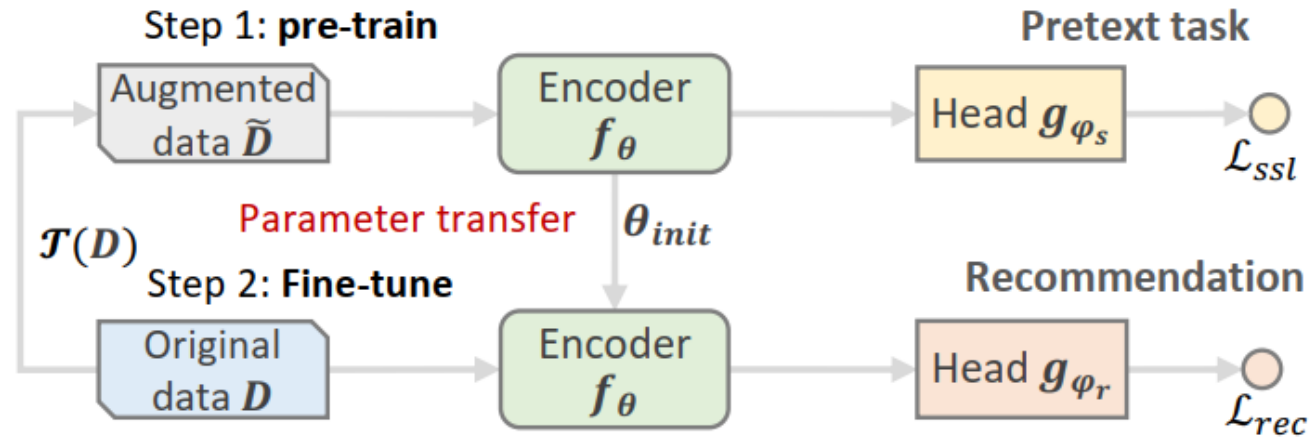
Recommendation: primary task

control the magnitude of self-supervision

SSL: auxiliary task, not concerned

# Typical Training Schemes

**Pre-training and Fine-tuning (PF)** used in most of generative methods and a few contrastive methods.



The encoder is firstly pretrained with pretext tasks on the augmented data for a good initialization of the encoder's parameters. It is then fine-tuned on the original data for recommendation

$$f_{\theta_{init}} = \arg \min_{f_\theta, g_{\phi_s}} \mathcal{L}_{ssl}(g_{\phi_s}(f_\theta(\tilde{\mathcal{D}})), \mathcal{D})$$
$$f_{\theta^*}, g_{\phi_r^*}, \mathbf{H}^* = \arg \min_{f_{\theta_{init}}, g_{\phi_r}} \mathcal{L}_{rec}(g_{\phi_r}(f_\theta(\mathcal{D})))$$

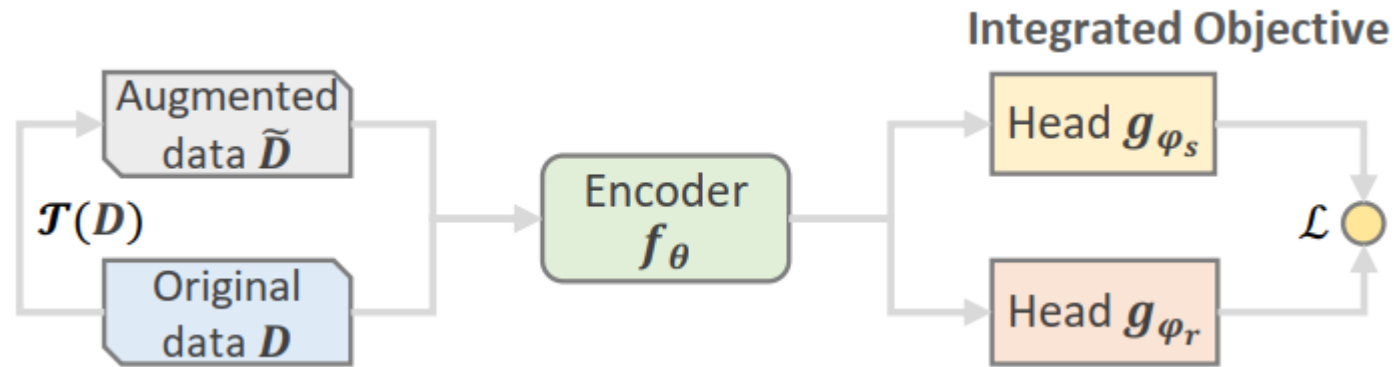
Pre-train the encoder

Fine-tune the encoder for recommendation

# Typical Training Schemes

## Integrated Learning (IL)

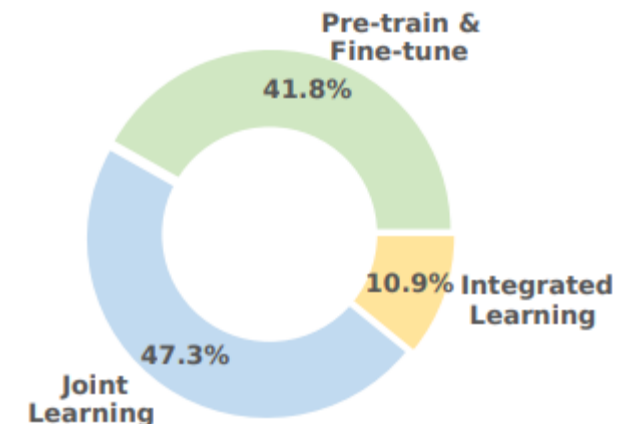
mainly used by the pseudo-labels-based predictive methods and a few contrastive methods.



The pretext task and the recommendation task are well-aligned under this setting and they are unified into an integrated objective.

$$f_{\theta^*}, g_{\phi_r^*}, \mathbf{H}^* = \arg \min_{f_\theta, g_\phi} \mathcal{L} \left( g_{\phi_r}(f_\theta(D)), g_{\phi_s}(f_\theta(\tilde{D})) \right)$$

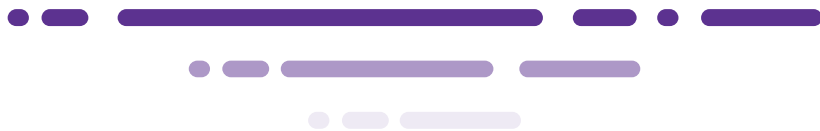
Integrated loss



03



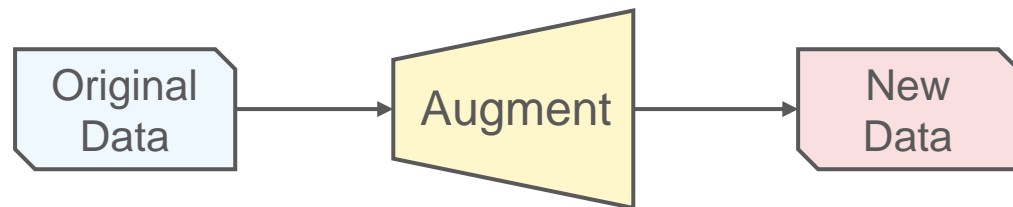
# Data Augmentation Techniques



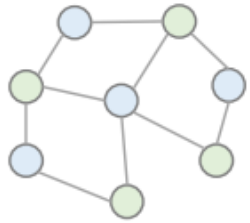
# Tutorial

# ■ Data Augmentation

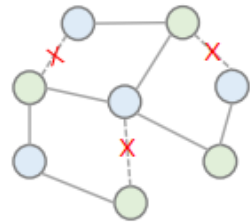
- Data augmentation increases the amount of training data by creating plausible variations of existing data.
- It plays a pivotal role in learning quality and generalizable representations.
- In SSR, the most commonly used data augmentation approaches can be divided into three categories: **graph-based**, **sequence-based**, and **feature-based**.



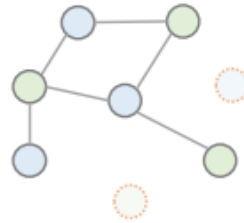
# ■ Graph-Based Augmentation



Original Graph



Edge Dropout



Node Dropout

- **Edge/Node Dropout:** With the probability  $\rho$ , each edge/node would be removed from the graph.

Idea behind

Only partial connections/nodes are informative to learn quality node representations.

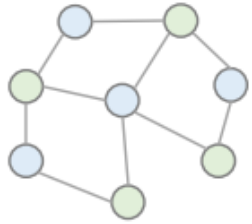
Formulation

$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{E-dropout}}(\mathcal{G}) = (\mathcal{V}, \mathbf{m} \odot \mathcal{E}) \quad \mathbf{m} \in (0, 1)^{|\mathcal{E}|}$$

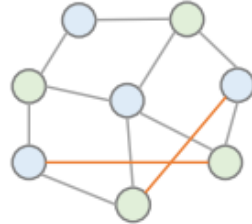
$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{N-dropout}}(\mathcal{G}) = (\mathcal{V} \odot \mathbf{m}, \mathcal{E} \odot \mathbf{m}') \quad \mathbf{m} \in (0, 1)^{|\mathcal{V}|}$$



# ■ Graph-Based Augmentation



Original Graph



Diffusion

- **Graph Diffusion:** Opposite to the dropout-based method, the diffusion-based augmentation adds edges into the graph to create views.

## Idea behind

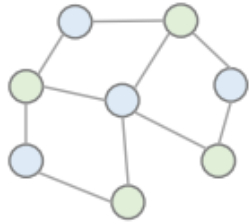
The missing user behaviors include unknown positive preferences which can be represented with weighted user-item edges.

## Formulation

$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{diffusion}}(\mathcal{G}) = (\mathcal{V}, \mathcal{E} + \tilde{\mathcal{E}})$$

Discover the possible edges by calculating the similarities of the user and item representations and retain the edges with top- $K$  similarities.

# ■ Graph-Based Augmentation



Original Graph



Subgraph Sampling

- **Subgraph Sampling:** samples a portion of nodes and edges by rules to form subgraphs.

Idea behind

The sampled graph reflects the local connectivity and semantics.

Formulation

$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{sampling}}(\mathcal{G}) = (\mathcal{Z} \in \mathcal{V}, \mathbf{A}[\mathcal{Z}, \mathcal{Z}]) \quad \mathcal{Z} \text{ the sampled node set}$$

A lot of approaches can be used to induce subgraphs like meta-path guided random walks and the ego-network sampling.

# Sequence-Based Augmentation



- **Item Masking:** randomly masks a portion of items and replaces them with special tokens [mask].

Idea behind

A user's intention is relatively stable during a period of time. Though part of items are masked, the primary intent is still retained in the rest.

Formulation

$$\tilde{S} = \mathcal{T}_{\text{masking}}(S) = [\tilde{i}_1, \tilde{i}_2, \dots, \tilde{i}_k], \tilde{i}_t = \begin{cases} i_t, & t \notin \mathcal{M} \\ [\text{mask}], & t \in \mathcal{M} \end{cases}$$

$\mathcal{M}$  denotes the indices of the masked items.

# Sequence-Based Augmentation



- **Item Cropping:** a sub-sequence with a certain length is randomly cropped.

## Idea behind

The selected sub-sequence is expected to endow the model with the ability to learn generalized representations without the comprehensive user profile.

## Formulation

$$\tilde{S} = \mathcal{T}_{\text{cropping}}(S) = [\tilde{i}_c, \tilde{i}_{c+1}, \dots, \tilde{i}_{c+L_c-1}]$$

where  $L_c = \lfloor \eta * |S| \rfloor$  and  $\eta \in (0, 1)$

# Sequence-Based Augmentation



- **Item Shuffling:** shuffle a continuous sub-sequence to create sequence augmentations.

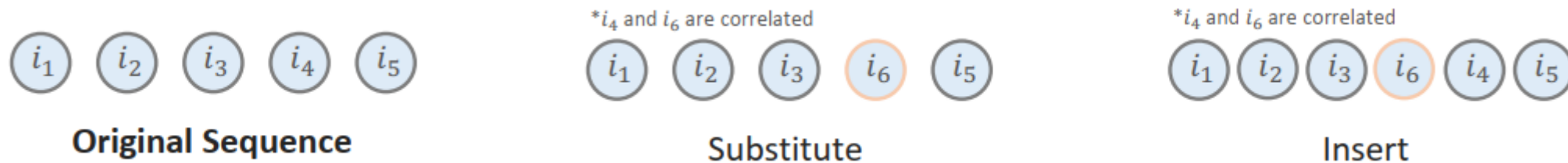
Idea behind

The item order in a sequence is not strict, and different item orders may actually correspond to the same user intent.

Formulation

$$\tilde{S} = \mathcal{T}_{\text{reordering}}(S) = [i_1, \dots, \tilde{i}_r, \tilde{i}_{r+1}, \dots, \tilde{i}_{r+L_r-1}, \dots, i_k]$$

# Sequence-Based Augmentation



- **Item Substitution/Insertion:** substitute/insert items in(to) short sequences with highly correlated items

## Idea behind

Random item cropping and masking could exaggerate the data sparsity issue in short sequences. Substituting/Inserting highly-correlated items injects less corruptions.

## Formulation

$$\tilde{S} = \mathcal{T}_{\text{substitution}}(S) = [\tilde{i}_1, \tilde{i}_2, \dots, \tilde{i}_k], \quad \tilde{i}_t = \begin{cases} i_t, & t \notin \mathcal{Z} \\ \text{item correlated to } i_t, & t \in \mathcal{Z} \end{cases}$$

$$\tilde{S} = \mathcal{T}_{\text{insert}}(S) = [i_1, \dots, \tilde{i}_{id_1}, i_{id_1}, \dots, \tilde{i}_{id_l}, i_{id_l}, \dots, i_k]$$

# ■ Feature-Based Augmentation

- **Feature Dropout:** similar to the item masking and edge dropout, which randomly masks/drops a small portion of features.

Idea behind

The whole profile/information can be inferred by a portion of features.

Formulation

$\tilde{\mathbf{X}} = \mathcal{T}_{\text{F-dropout}}(\mathbf{X}) = \mathbf{X} \odot \mathbf{M}$  The matrix  $\mathbf{M}$  is generated by Bernoulli distribution.

- **Feature Shuffling:** switches rows and columns in the feature matrix.

Idea behind

By randomly changing the contextual information, the feature matrix is corrupted to yield augmentations.

Formulation

$\tilde{\mathbf{X}} = \mathcal{T}_{\text{shuffling}}(\mathbf{X}) = \mathbf{P}_r \mathbf{X} \mathbf{P}_c$   $\mathbf{P}_r$  and  $\mathbf{P}_c$  are permutation matrices

# ■ Feature-Based Augmentation

- **Feature Mixing:** mix the original features with features from other users/items or previous versions to synthesize informative negative/positive examples .

Idea behind

Mixing related features can generate hard examples.

Formulation

$$\tilde{\mathbf{x}}_i = \mathcal{T}_{\text{mixing}}(\mathbf{x}_i) = \alpha \mathbf{x}_i + (1 - \alpha) \mathbf{x}'_j, \quad \alpha \in [0, 1]$$

- **Feature Clustering:** the augmented prototype representations can be learned via clustering algorithms.

Idea behind

Assume that there are prototypes in the feature/representation space and user/item representations should be closer to their assigned prototypes.

Formulation

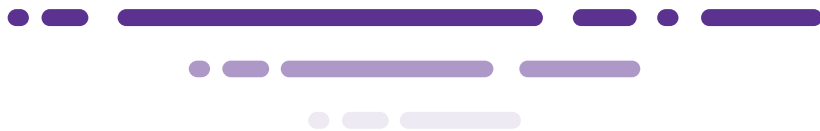
$$\tilde{\mathbf{C}} = \mathcal{T}_{\text{clustering}}(\mathbf{X}) = \text{EM}(\mathbf{X}, \mathcal{C}), \quad \text{where } \mathcal{C} \text{ is the presupposed clusters (prototypes) and } \tilde{\mathbf{C}} \text{ is the augmented prototype representations.}$$



04

## SSR Methods

- Contrastive, Generative, Predictive, and Hybrid

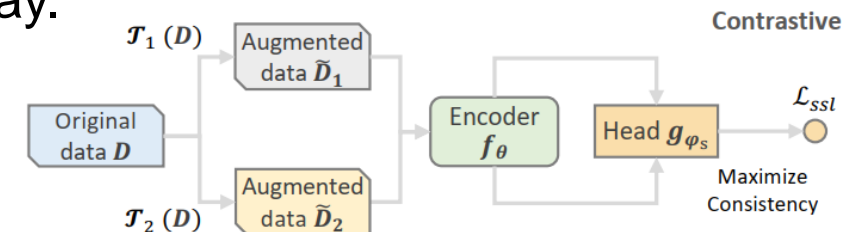
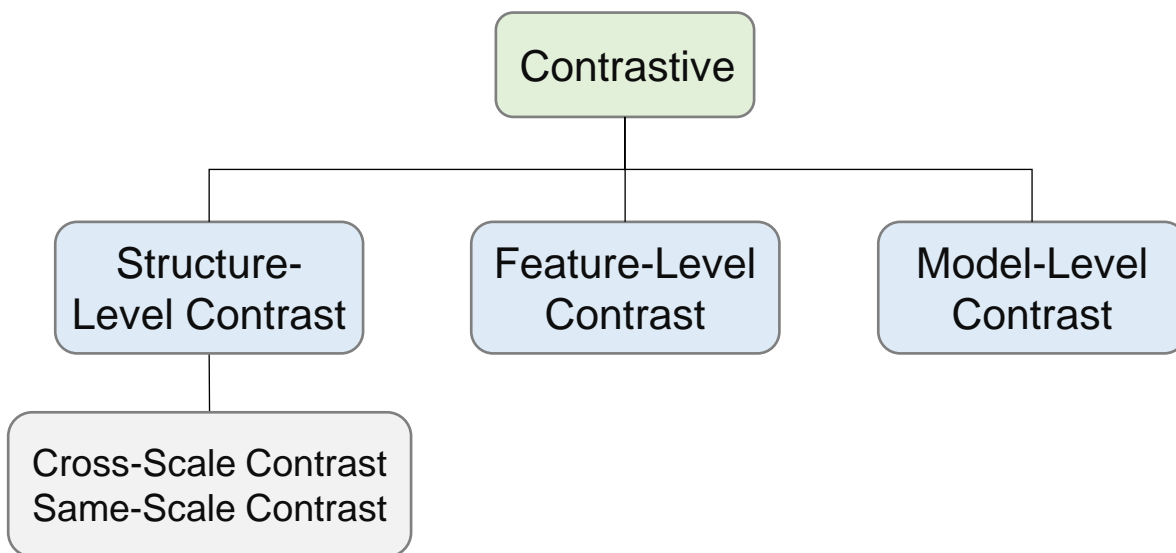


# Tutorial

# Contrastive SSR Methods

## Contrastive Methods

Pull similar samples closer and Push dissimilar samples away.



$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{ssl} \left( g_{\phi_s} \left( f_{\theta}(\tilde{D}_1), f_{\theta}(\tilde{D}_2) \right) \right)$$

According to where the self-supervision signals come from, we divide them into three categories: **structure-level contrast**, **feature-level contrast**, and **model-level contrast**.

# ■ Structure-Level Contrast

## Local-Local



$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{\mathcal{MI}}(g_{\phi_s}(\tilde{\mathbf{h}}_i, \tilde{\mathbf{h}}_j))$$

$\tilde{\mathbf{h}}_i$  and  $\tilde{\mathbf{h}}_j$  are node representations

## Same-Scale

## Global-Global



$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{\mathcal{MI}}(g_{\phi_s}(\text{Agg}(f_{\theta}(\tilde{S}_i)), \text{Agg}(f_{\theta}(\tilde{S}_j))))$$

$\tilde{S}_i$  and  $\tilde{S}_j$  are two sequence augmentations

## Local-Global



$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{\mathcal{MI}}(g_{\phi_s}(\tilde{\mathbf{h}}, \mathcal{R}(f_{\theta}(\tilde{\mathcal{G}}, \tilde{\mathbf{A}}))))$$

$\mathcal{R}$  is the readout function that generates global-level graph representation.

## Cross-Scale

## Local-Context



$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{\mathcal{MI}}(g_{\phi_s}(\mathbf{h}_i, \mathcal{R}(f_{\theta}(\mathcal{C}_j))))$$

$\mathcal{C}_j$  denotes the context of node (sequence)  $j$ .

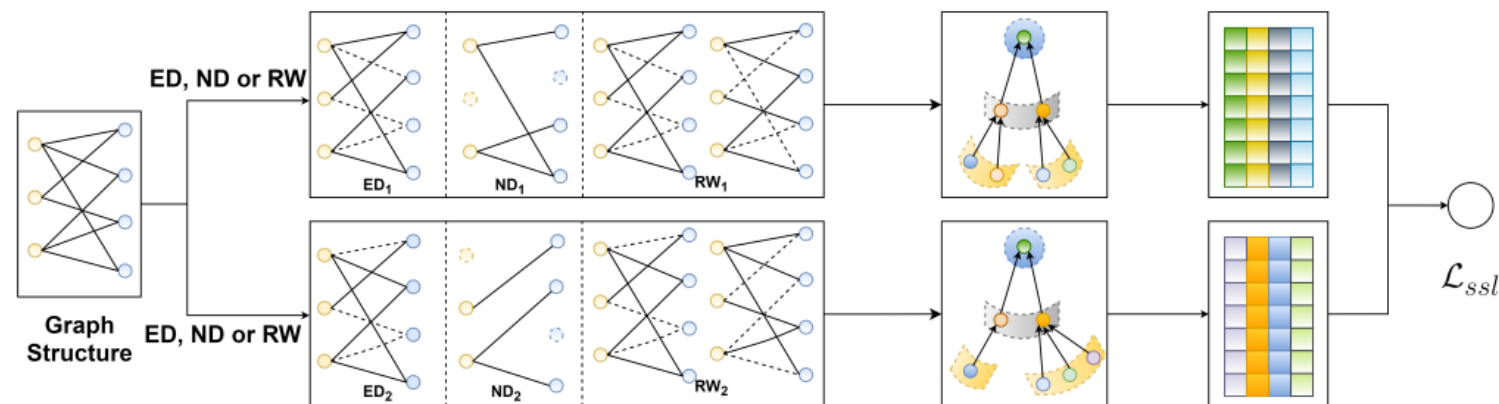
# Structure-Level Contrast

## Local-Local Contrast: SGL (Wu et al. 2021)



### BPR Loss

$$\mathcal{L}_{main} = \sum_{(u,i,j) \in O} -\log \sigma(\hat{y}_{ui} - \hat{y}_{uj})$$



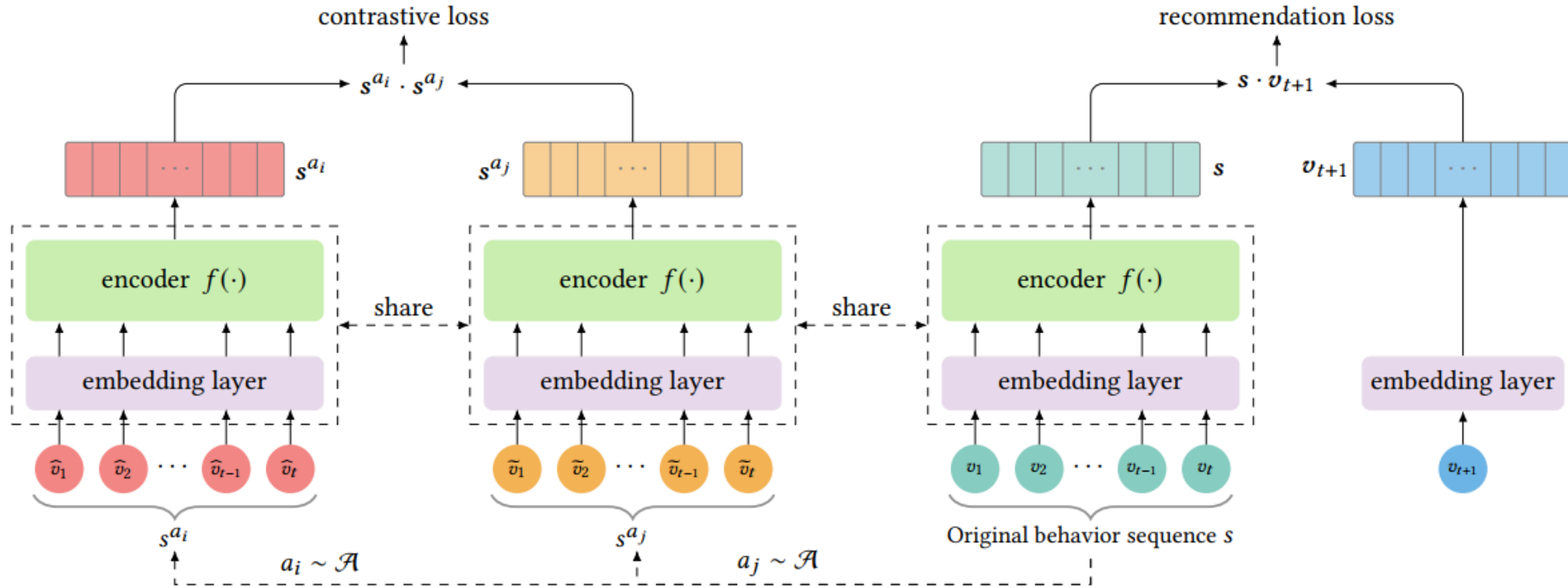
### InfoNCE Loss

$$\mathcal{L}_{ssl}^{user} = \sum_{u \in \mathcal{U}} -\log \frac{\exp(s(\mathbf{z}'_u, \mathbf{z}''_u)/\tau)}{\sum_{v \in \mathcal{U}} \exp(s(\mathbf{z}'_u, \mathbf{z}''_v)/\tau)}$$

$$\mathcal{L} = \mathcal{L}_{main} + \lambda_1 \mathcal{L}_{ssl} + \lambda_2 \|\Theta\|_2^2$$

# Structure-Level Contrast

## Global-Global Contrast: CL4SRec (Xie et al. 2021)

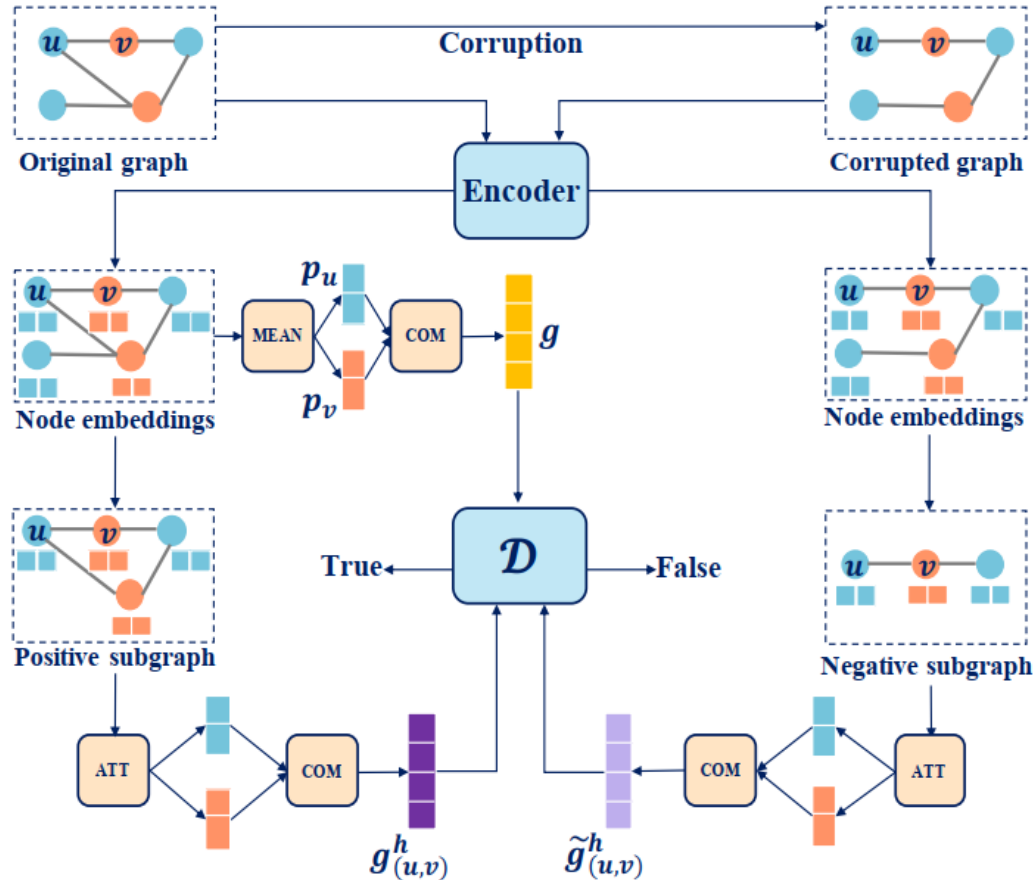


$$\mathcal{L}_{cl}(s_u^{a_i}, s_u^{a_j}) = -\log \frac{\exp(\text{sim}(s_u^{a_i}, s_u^{a_j}))}{\exp(\text{sim}(s_u^{a_i}, s_u^{a_j})) + \sum_{s^- \in S^-} \exp(\text{sim}(s_u^{a_i}, s^-))}$$

$$\mathcal{L}_{main}(s_{u,t}) = -\log \frac{\exp(s_{u,t}^\top v_{t+1}^+)}{\exp(s_{u,t}^\top v_{t+1}^+) + \sum_{v_{t+1}^- \in \mathcal{V}^-} \exp(s_{u,t}^\top v_{t+1}^-)}$$

# Structure-Level Contrast

## Local-Global Contrast: BiGI (Cao et al. 2021)



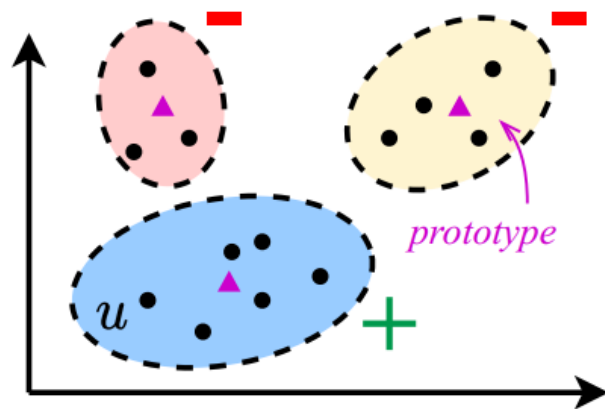
## CL Loss

$$\mathcal{L}_m = -\frac{1}{|E| + |\tilde{E}|} \left( \sum_{i=1}^{|E|} \mathbb{E}_G [\log \mathcal{D}(g^h_{(u,v)_i}, g)] + \sum_{i=1}^{|\tilde{E}|} \mathbb{E}_{\tilde{G}} [\log (1 - \mathcal{D}(\tilde{g}^h_{(u,v)_i}, g)) \right].$$

$$\mathcal{D}(g^h_{(u,v)_i}, g) = \sigma((g^h_{(u,v)_i})^T W_b g).$$

# ■ Structure-Level Contrast

**Local-Context Contrast:** NCL (Lin et al. 2022)



CL Loss

$$\mathcal{L}_P^U = \sum_{u \in \mathcal{U}} -\log \frac{\exp(\mathbf{e}_u \cdot \mathbf{c}_i / \tau)}{\sum_{\mathbf{c}_j \in C} \exp(\mathbf{e}_u \cdot \mathbf{c}_j / \tau)}$$

Similar users/items tend to fall in neighboring embedding space, and the prototypes are the center of clusters that represent a group of semantic neighbors.

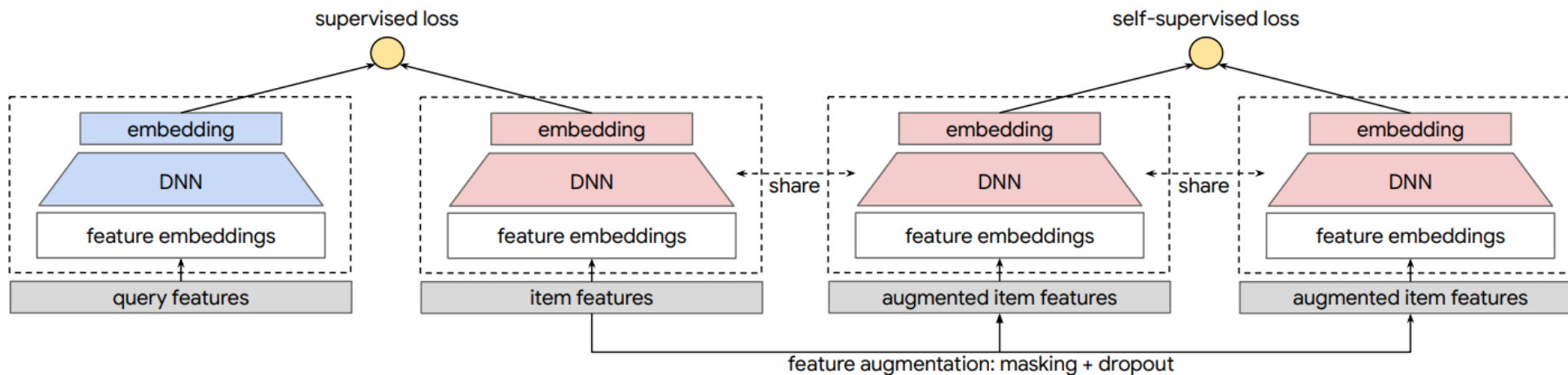
# ■ Feature-Level Contrast

## Formulation

$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{\mathcal{MI}}(g_{\phi_s}(f_{\theta}(\tilde{\mathbf{x}}_i), f_{\theta}(\tilde{\mathbf{x}}_j)))$$

$\tilde{\mathbf{x}}_i$  and  $\tilde{\mathbf{x}}_j$  are feature-level augmentations which can be obtained by modifying the input feature

## SSL4ItemRec (Yao et al. 2022)





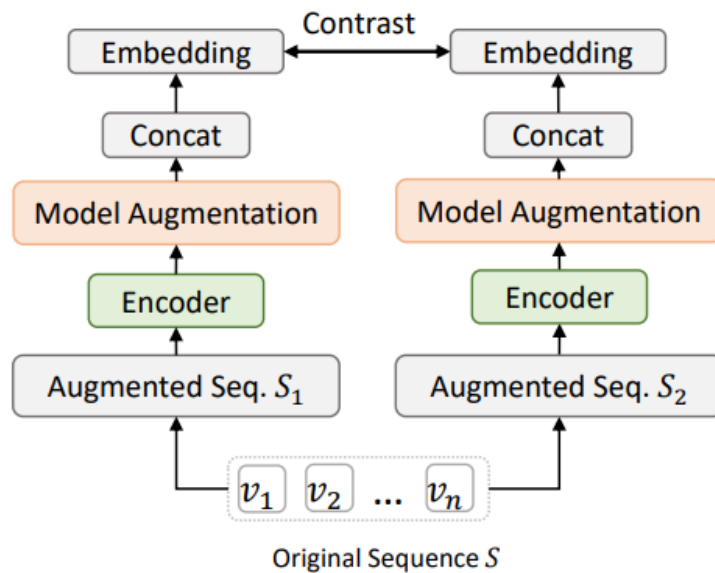
# Model-Level Contrast

## Formulation

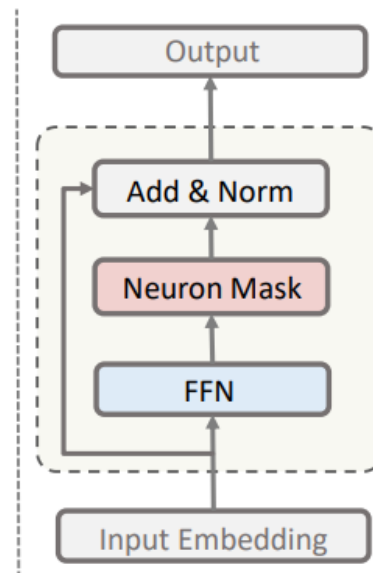
$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{\mathcal{MI}}(g_{\phi_s}(f_{\theta'}(\mathcal{D}), f_{\theta''}(\mathcal{D})))$$

$f_{\theta'}$  and  $f_{\theta''}$  are perturbed versions of  $f_{\theta}$ .

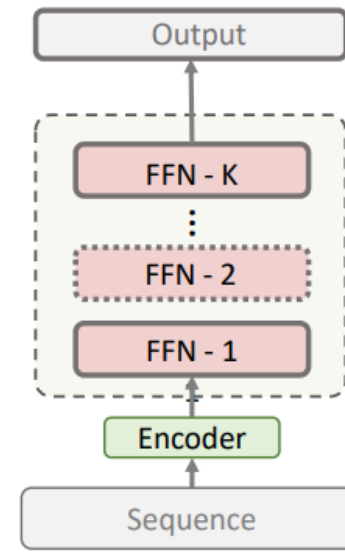
## SRMA (Liu et al. 2022)



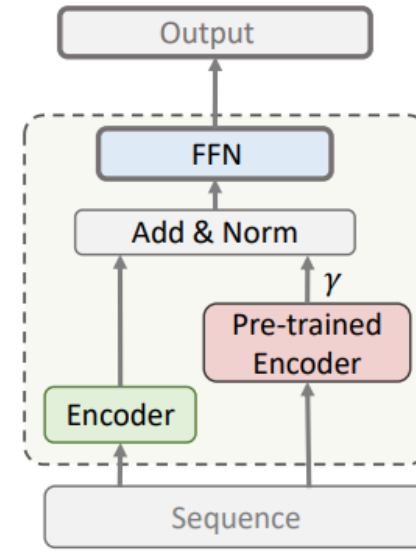
(a) Contrastive Self-supervised Learning



(b) Neuron Masking



(c) Layer Dropping

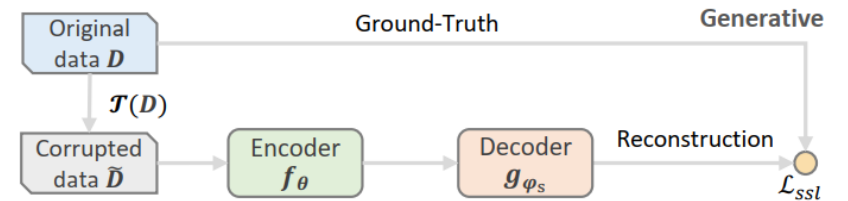
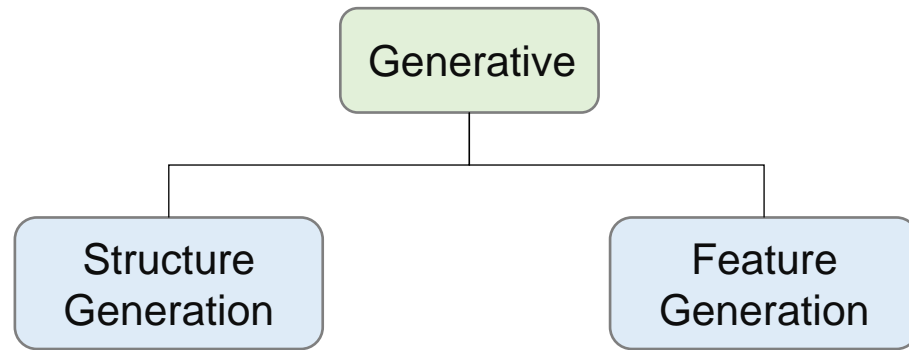


(d) Encoder Complementing

# ■ Generative SSR Methods

## Generative Methods

Reconstructing the original input with its corrupted version.



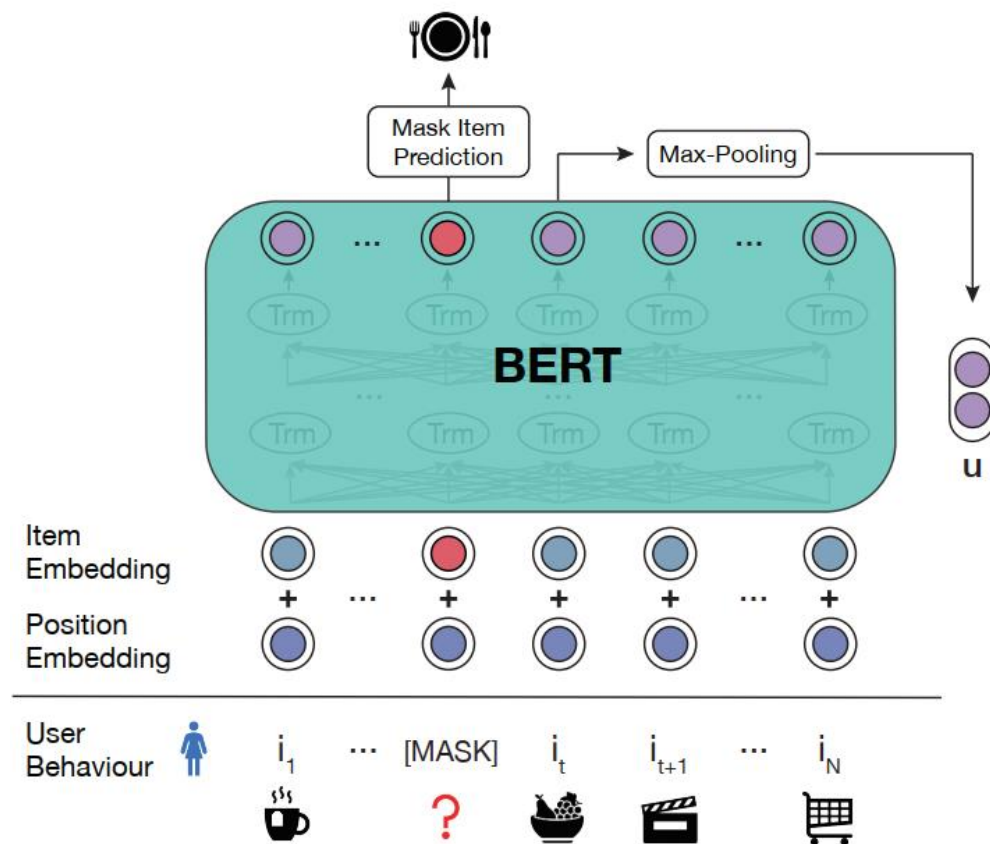
$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{ssl} \left( g_{\phi_s} (f_{\theta}(\tilde{\mathcal{D}})), \mathcal{D} \right)$$

According to the reconstruction objectives, generative SSR methods are divided into two categories: **Structure Generation** and **Feature Generation**.

# Structure Generation

## Sequence Reconstruction

### UPRec (Xiao et al. 2022)



Pre-trained with the masked-item-prediction method.

$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{ssl} \left( g_{\phi_s} \left( f_{\theta}(\tilde{S}) \right), S \right)$$

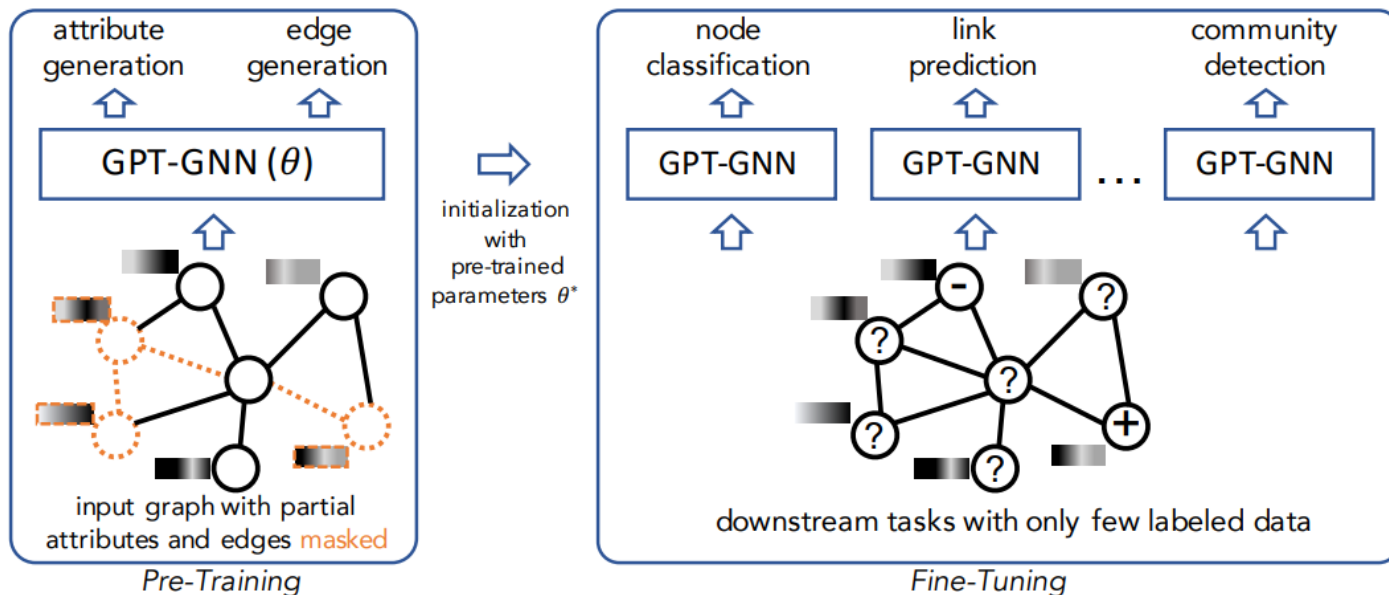
$$\mathcal{L} = \frac{1}{|S_u^m|} \sum_{i_m \in S_u^m} -\log P \left( i_m = i_m^* \mid \tilde{S}_u \right)$$

Fine-tuned by predicting the last item of each sequence.

# Structure Generation

## Graph Reconstruction

### GPT-GNN (Hu et al. 2020)

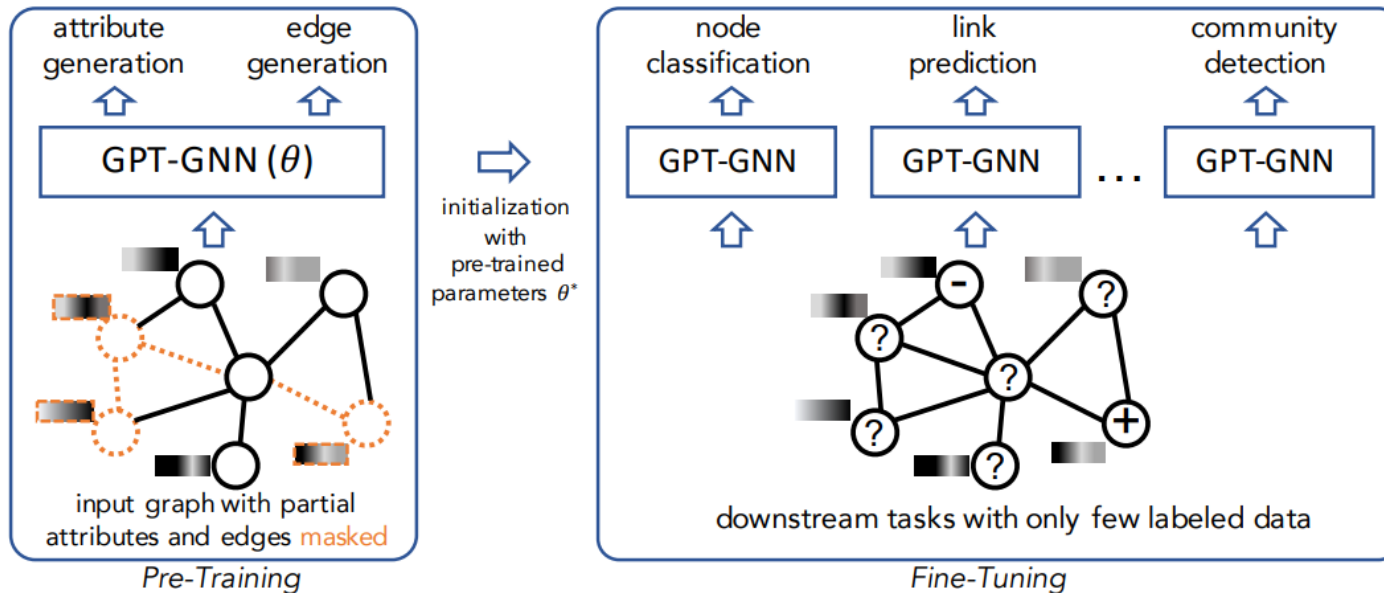


A GNN is pre-trained with the self-supervised generative task — structure generations. Second, the pretrained model and its parameters are then used to initialize models for downstream tasks on the input graph or graphs of the same domain such as recommendation.

$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \mathcal{L}_{ssl} \left( g_{\phi_s} \left( f_{\theta}(\tilde{\mathcal{G}}) \right), \mathbf{A} \right)$$

# Feature Generation

## GPT-GNN (Hu et al. 2020)

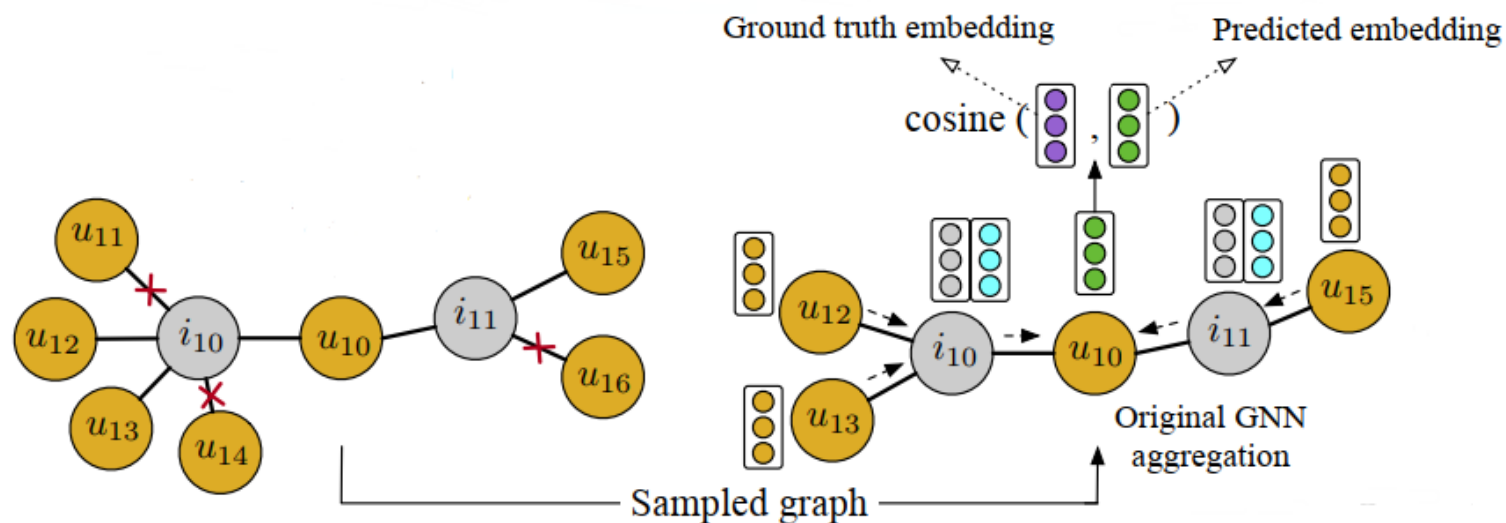


A GNN is pre-trained with the self-supervised generative task — feature generations. Second, the pretrained model and its parameters are then used to initialize models for downstream tasks on the input graph or graphs of the same domain such as recommendation.

$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \left\| g_{\phi_s} \left( f_{\theta}(\tilde{\mathcal{D}}) \right) - \mathbf{X} \right\|^2$$

# Feature Generation

## PT-GNN (Hao et al. 2021)

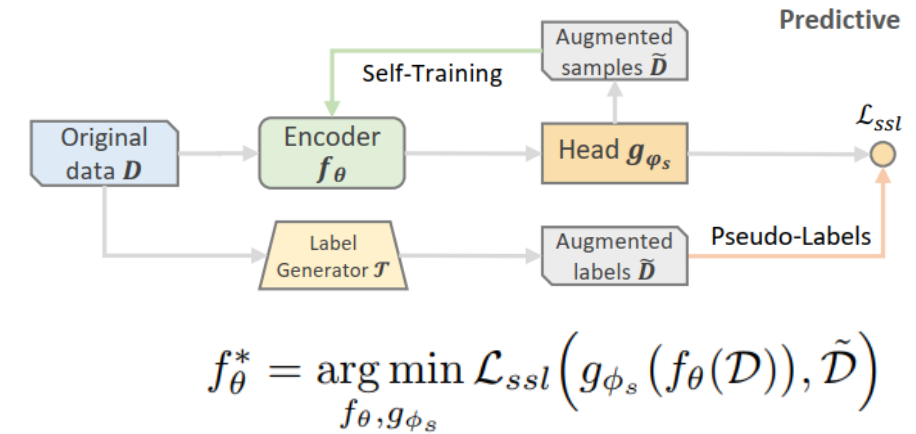
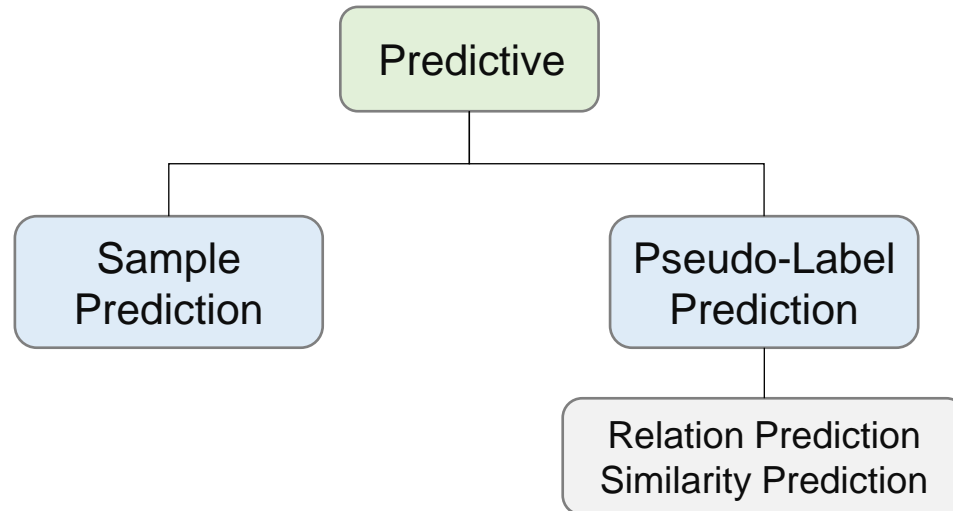


$$f_{\theta}^* = \arg \min_{f_{\theta}, g_{\phi_s}} \left\| g_{\phi_s} \left( f_{\theta}(\tilde{D}) \right) - \mathbf{X} \right\|^2$$

# Predictive SSR Methods

## Predictive Methods

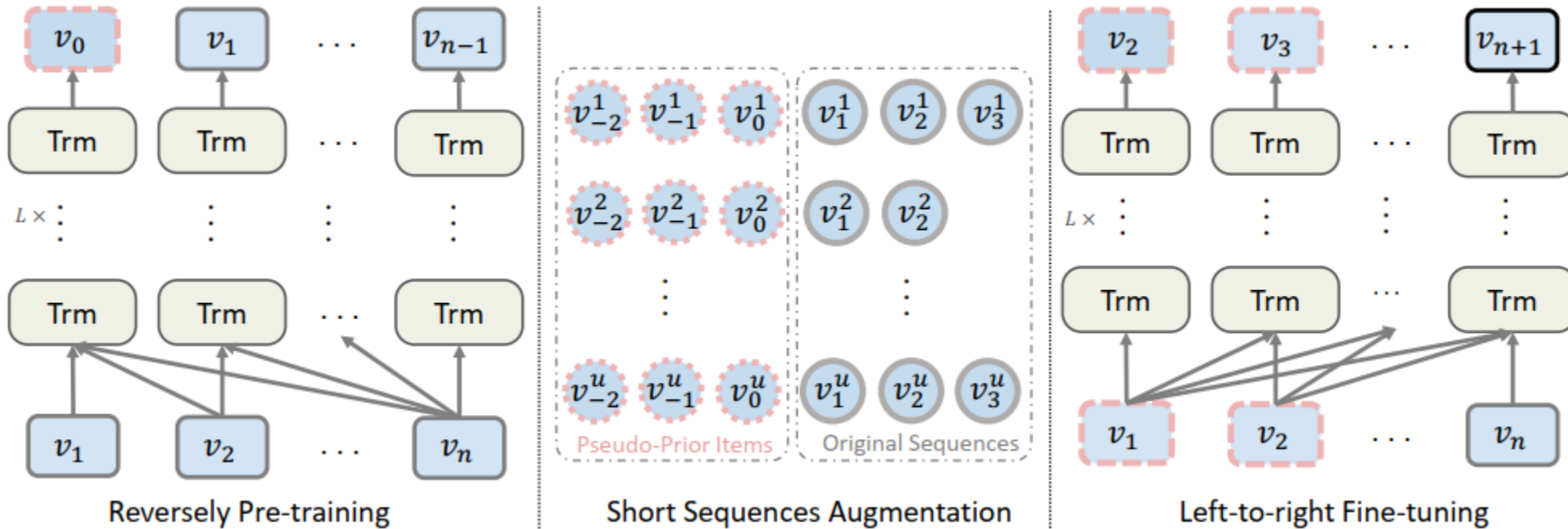
Deal with the self-generated supervisory signals obtained from the complete original data.



According to what the predictive pretext task predicts, predictive methods can be categorized into two branches: **Sample Prediction** and **Pseudo-Labels Prediction**.

# Sample Prediction

## ASReP (Liu et al. 2021)



Firstly pre-train a transformer in a reverse direction to predict prior items.

Use this transformer to generate fabricated historical items at the beginning of short sequences

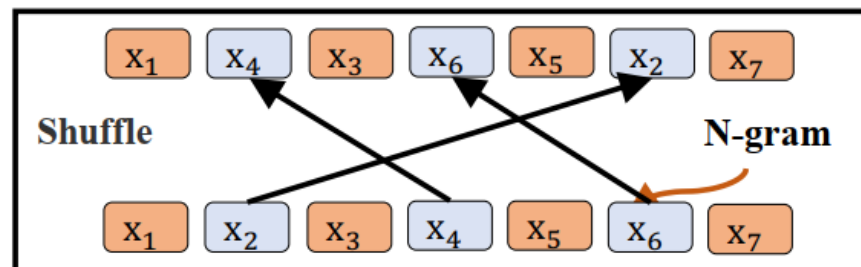
Fine-tune the transformer using these augmented sequences from the time order to predict the next item.



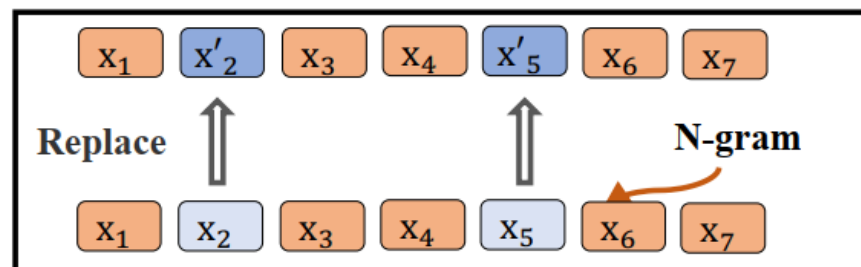
# Pseudo-Label Prediction

## Relation Prediction

### SSI (Song et al. 2021)



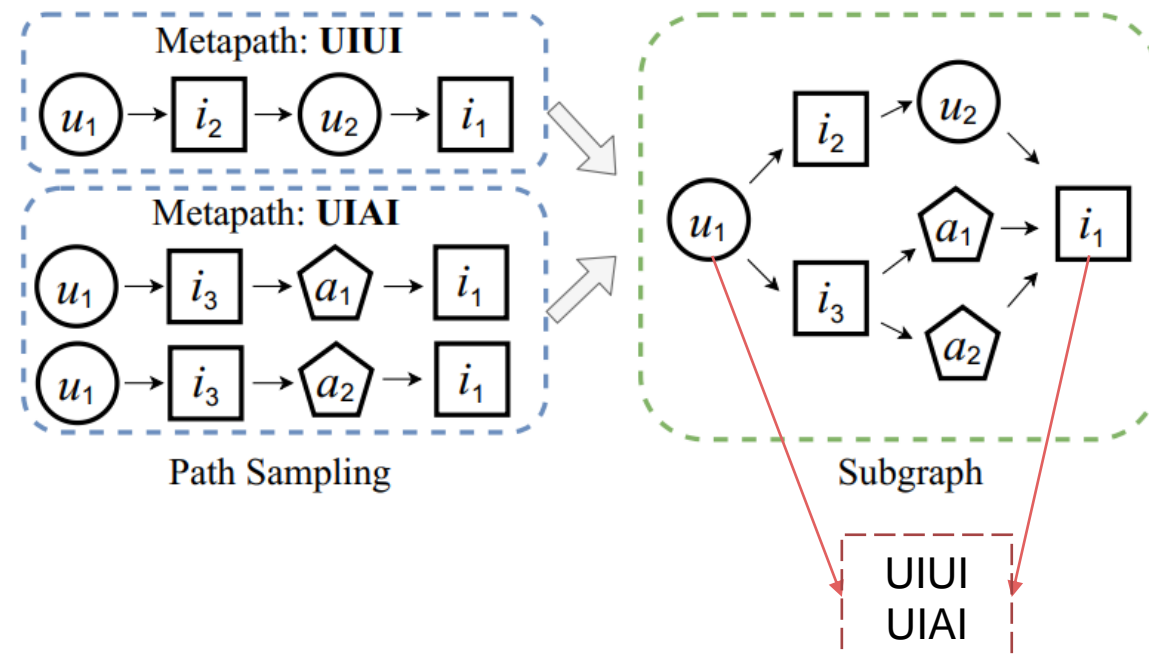
(a) Temporal Consistency Task



(b) Persona Consistency Task

Shuffle/replace a portion of items in a given sequence, and then predicts if the modified sequence is in the original order/from the same user.

### CHEST (Wang et al. 2021)

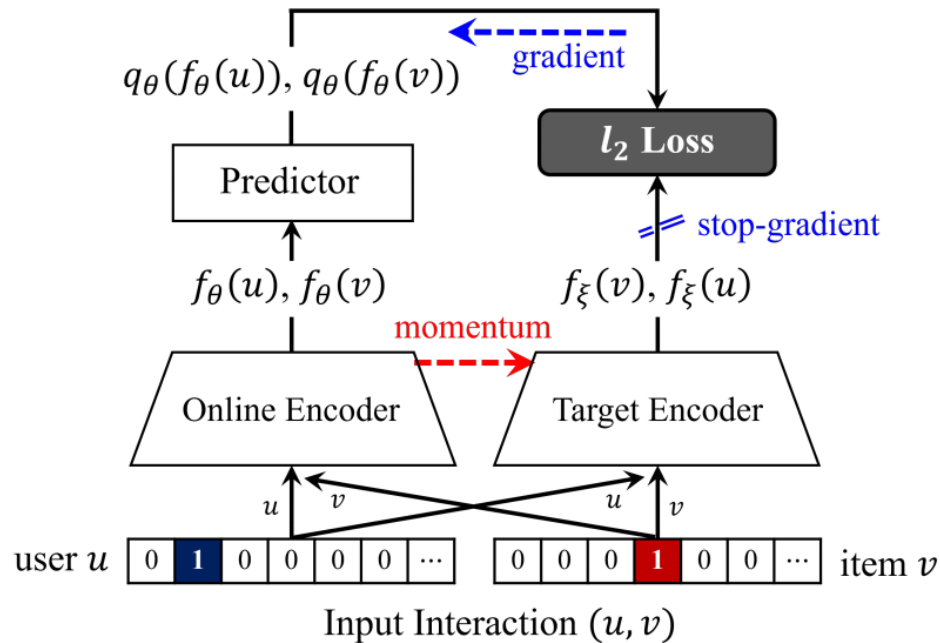


Predict if there exists a path instance of a specific meta-path between a user-item pair.

# Pseudo-Label Prediction

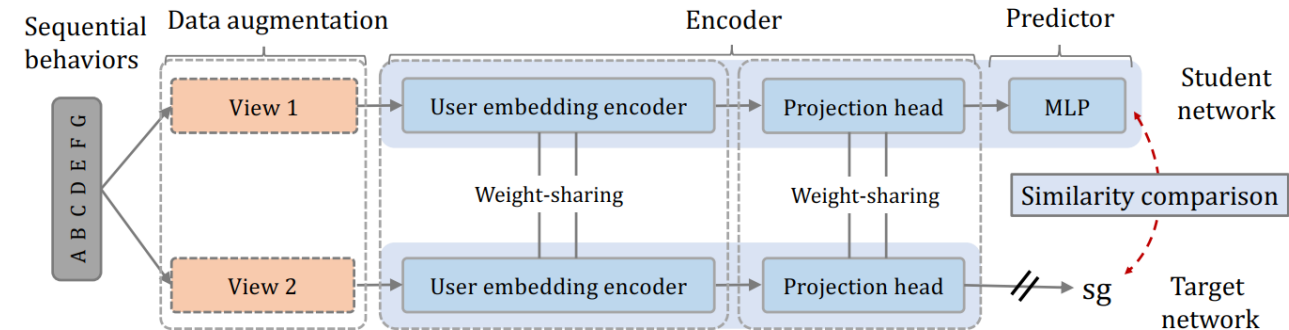
## Similarity Prediction

### BUIR (Lee et al. 2021)



The online network is fed with the user representation and is trained to predict the item representation output by the target network and vice versa.

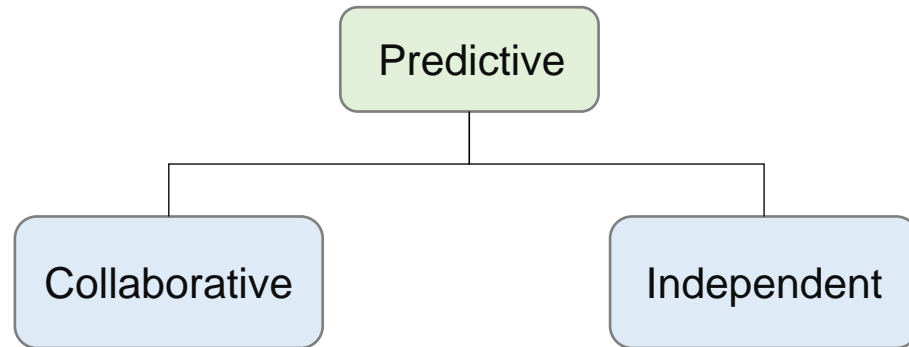
### CLUE (Cheng et al. 2021)



# Hybrid SSR Methods

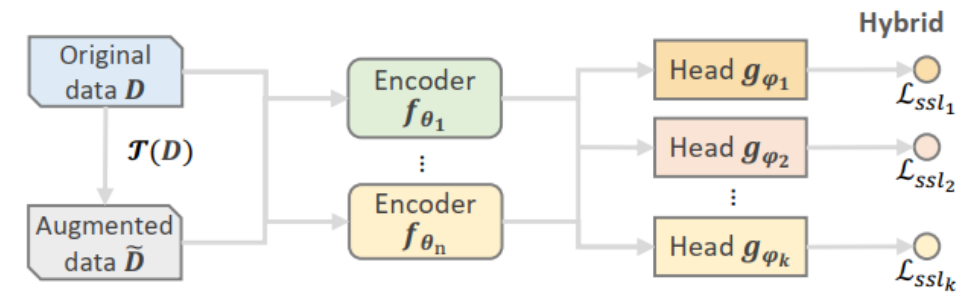
## Hybrid Methods

Assemble multiple types of pretext tasks to take advantage of different self-supervision signals.



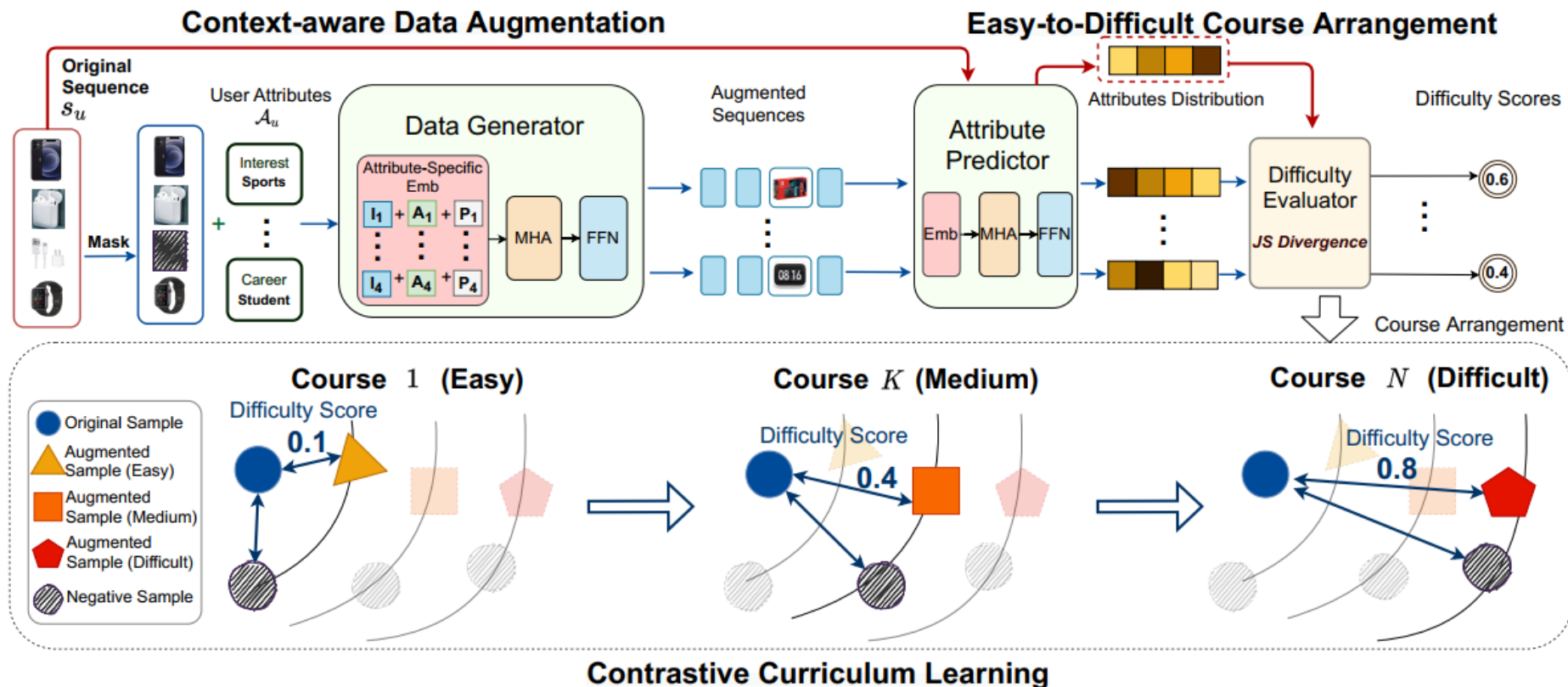
**Collaborative:** Different pretext tasks collaborate in a way to serve one primary pretext task.

**Independent:** There are no correlations between different pretext tasks and they work independently.



# Collaborative Methods

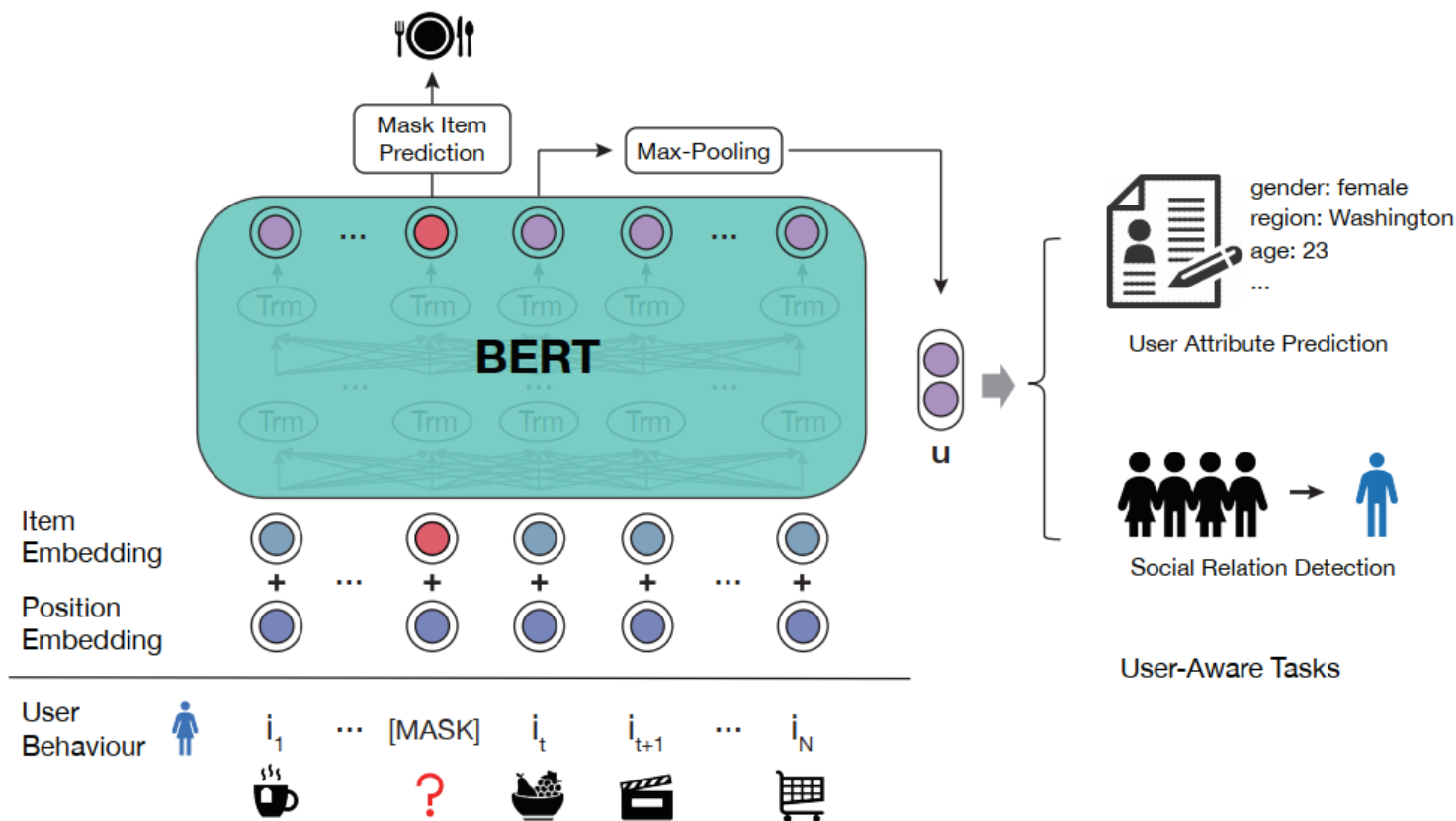
## CCL (Bian et al. 2021)



The generative task serves the contrastive task.

# Independent Methods

## UPRec (Xiao et al. 2022)



One Generative Task  
Two Predictive Tasks

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{MIP}} + \lambda_2 \mathcal{L}_{\text{UAP}} + \lambda_3 \mathcal{L}_{\text{SRD}}$$

# ■ Pros and Cons

## Contrastive

**Pros** 😊

Flexible to design augmentations and pretext tasks.

**Cons** 😞

Often compromised by low-quality augmentations.

## Generative

**Pros** 😊

Can follow the successful experience for training masked language models

**Cons** 😞

Confronted with heavy computation when building general-purpose models.

## Predictive

**Pros** 😊

Acquire samples and pseudo-labels in more dynamic and flexible ways.

**Cons** 😞

Collect the pseudo-labels based on heuristics, without assessing how relevant these labels and predictive tasks are to recommendation.

## Hybrid

**Pros** 😊

Can get enhanced and comprehensive self-supervision.

**Cons** 😞

Confronted with the problem of coordinating multiple pretext tasks.

05

## SELFRec

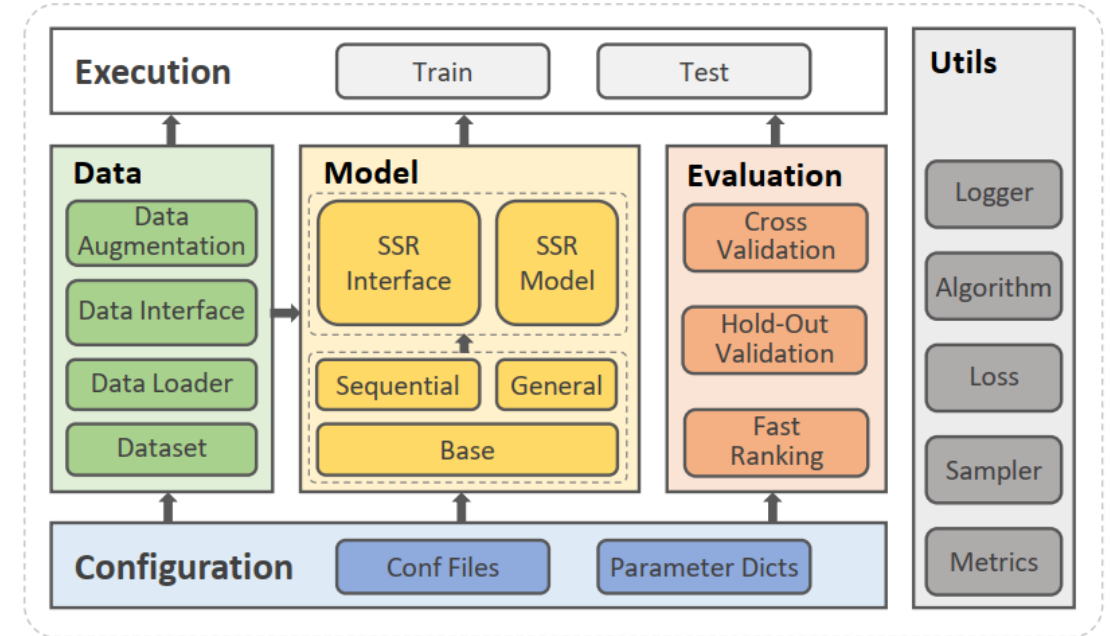
- A Library for self-supervised recommendation



# Tutorial

# SELFRec

- SELFRec incorporates multiple high-quality datasets and metrics.
- There are more than 10+ state-of-the-art SSR methods implemented in SELFRec for fair empirical comparison.
- SELFRec is developed with Python 3.7+, Tensorflow 1.14+ and PyTorch 1.7+.
- SELFRec provides a set of simple and high-level interfaces, by which new SSR models can be easily added in a plug-and-play fashion.
- SELFRec decouples the model design from other procedures.



**SELFREC**  
Framework for Self-Supervised Recommendation

<https://github.com/Coder-Yu/SELFRec>



06

Limitations and Future  
Research

Tutorial

# ■ Limitations and Future Research

## • Theory for Augmentation Selection

- Cannot seamlessly transplant augmentation approaches designed for other fields to recommendation
- Most augmentation approaches are based on heuristics
- Cumbersome trial-and-error work is needed to search for useful augmentations

A solid recommendation-specific theoretical foundation for the augmentation selection is urgently needed.

### Example



What is the criterion for augmentation selection in recommendation?

In vision tasks, views should not share too much information (left) or too little information (right), but should find an optimal mix (the “sweet spot”, middle) that maximizes the downstream performance.

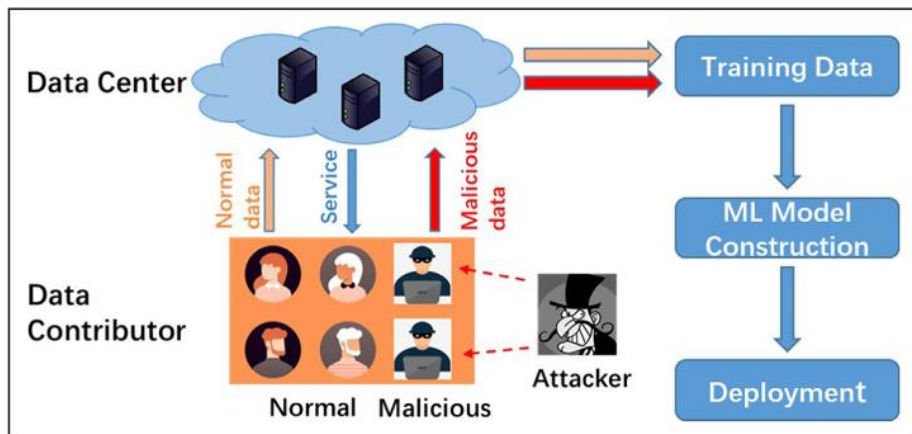
# ■ Limitations and Future Research

- **Attacking and Defending in Pre-trained Recommendation Models**

- Recommender systems are vulnerable to the data poisoning attack
- It remains unknown if the recommendation models pretrained in a self-supervised way are robust to such attacks

Developing new-type attacks and defending self-supervised pretrained recommender systems against these attacks will be an interesting future research direction.

## Example



Are pre-trained recommendation models robust to data poisoning attacks?

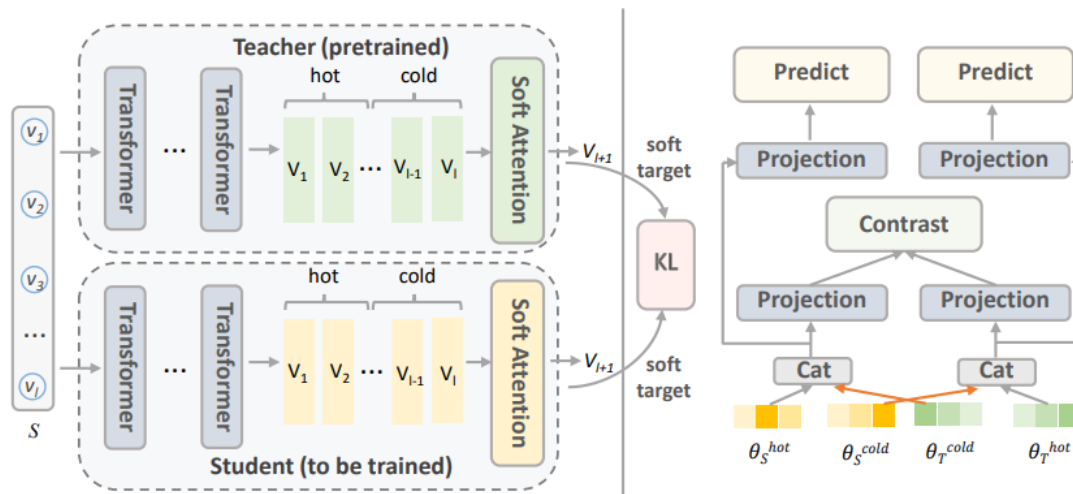
# ■ Limitations and Future Research

## • On-Device Self-Supervised Recommendation

- On-device recommender systems are compromised by the highly compressed model size and limited labeled data.
- Combined with the technique of knowledge distillation, SSL may largely compensate for the accuracy degradation of on-device recommendation models.

On-device self-supervised recommendation is still under-explored, and it deserves further study.

### Example



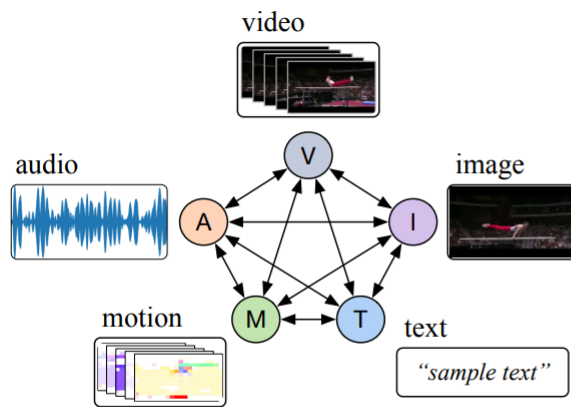
How to empower on-device recommendation with SSL?

# ■ Limitations and Future Research

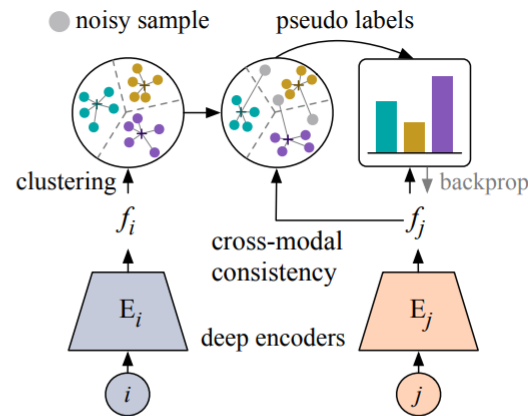
## • Towards General-Purpose Pre-Training

- Data in recommender systems is multi-modal and scenarios are rather diverse
- Explore general-purpose recommendation models which are pre-trained with the multi-modal SSL on large-scale data.
- Enable models to adapt to multiple downstream recommendation tasks with the cheap finetuning.

### Example



(a) Cross-modal supervision.



(b) Modality  $i \rightarrow$  modality  $j$ .

How to develop a once and for all pretraining technique for diverse modalities and downstream tasks?

## Self-Supervised Learning for Recommender Systems: A Survey

Junliang Yu, Hongzhi Yin\*, Xin Xia, Tong Chen, Jundong Li, and Zi Huang

**Abstract**—Neural architecture-based recommender systems have achieved tremendous success in recent years. However, when dealing with highly sparse data, they still fall short of expectation. Self-supervised learning (SSL), as an emerging technique to learn with unlabeled data, recently has drawn considerable attention in many fields. There is also a growing body of research proceeding towards applying SSL to recommendation for mitigating the data sparsity issue. In this survey, a timely and systematical review of the research efforts on self-supervised recommendation (SSR) is presented. Specifically, we put forward an exclusive definition of SSR, on top of which we build a comprehensive taxonomy to divide existing SSR methods into four categories: contrastive, generative, predictive, and hybrid. For each category, the narrative unfolds along its concept and formulation, the involved methods, and its pros and cons. Meanwhile, to facilitate the development and evaluation of SSR models, we release an open-source library SELFRec, which incorporates multiple benchmark datasets and evaluation metrics, and has implemented a number of state-of-the-art SSR models for empirical comparison. Finally, we shed light on the limitations in the current research and outline the future research directions.

**Index Terms**—Recommendation, Self-Supervised Learning, Contrastive Learning, Pre-Training, Data Augmentation.



<https://arxiv.org/abs/2203.15876>





THE UNIVERSITY  
OF QUEENSLAND  
AUSTRALIA

# Thanks Q&A

