Note to other teachers and users of these slides: We would be delighted if you found our material useful for giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: http://cs224w.Stanford.edu

Stanford CS224W: Advanced Topics in Graph Neural Networks

CS224W: Machine Learning with Graphs
Charilaos Kanatsoulis
Jure Leskovec, Stanford University
http://cs224w.stanford.edu



Announcements

- **Project Milestone** due tomorrow (11/11)
- **Homework 3** due this Thursday 11/12
  - Recitation recording on Ed
- **Exam is coming up**
  - Wednesday 11/18 from 6-8PM
    - See Ed for logistics
    - If you're taking OAE/makeup exam, you should have received an email with details
- **Colab 5 released Thursday 11/12**
  - Due after break (12/3)

Response to high-frequency feedback

- **Practice Exam**

Practice Exam and solutions have been posted to Ed!

- **Exam Review Session**

We will hold an exam review session. Details will be posted on Ed soon.

- **Additional Office Hours**

We will host additional office hours to help with exam preparation!

Today's lecture

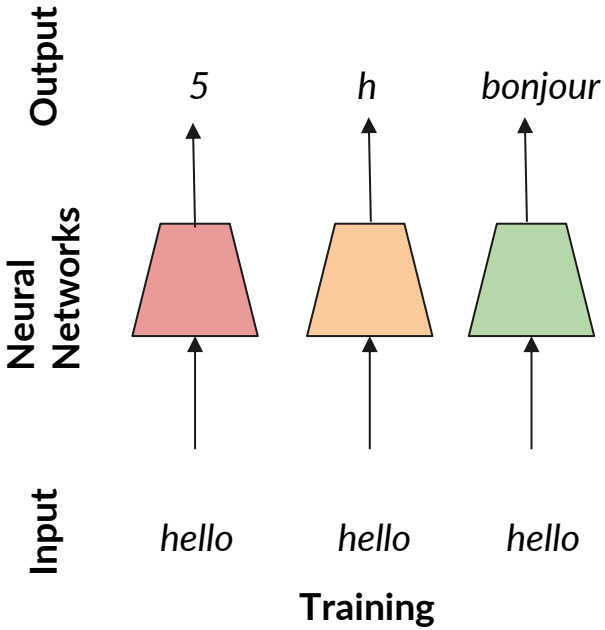
- **PRODIGY: Enabling In-context Learning Over Graphs**
  - Huang, Q., Ren, H., Chen, P., Krvzmanc, G., Zeng, D.D., Liang, P., & Leskovec, J. *PRODIGY: Enabling In-context Learning Over Graphs.* NeurIPS 2023

- **Relational Transformer**
  - Ranjan, R., Hudovernik, V., Znidar, M., Kanatsoulis, C. Relational Transformer: Toward Zero-Shot Foundation Models for Relational Data.

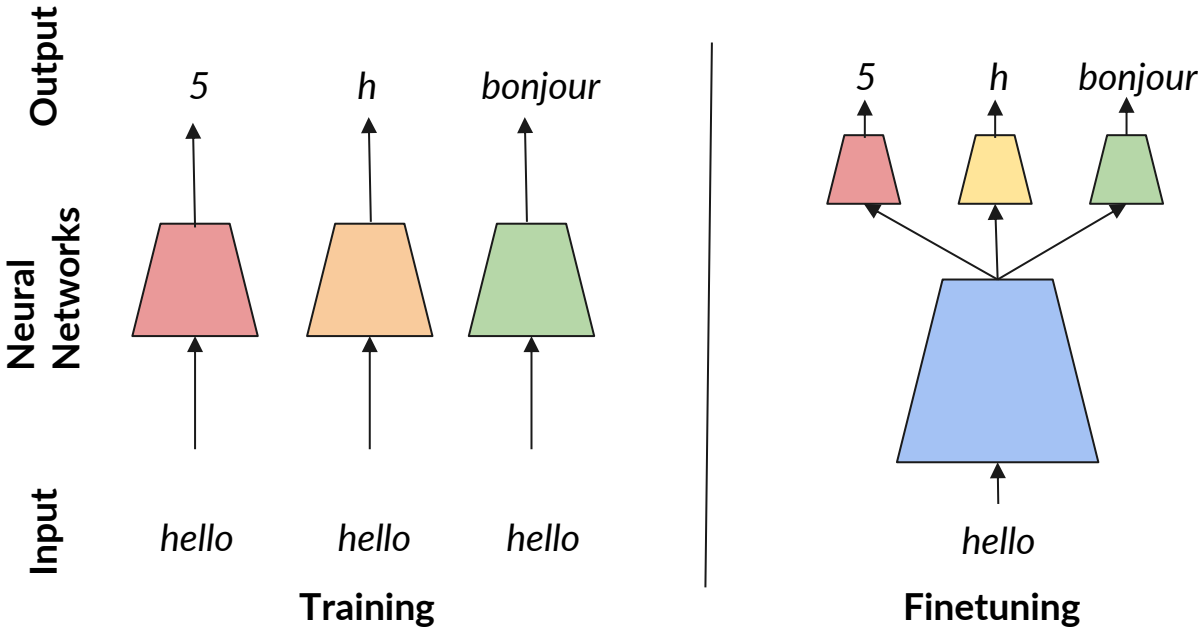
PRODIGY: Enabling In-context Learning Over Graphs

CS224W: Machine Learning with Graphs
Qian Huang, Stanford University

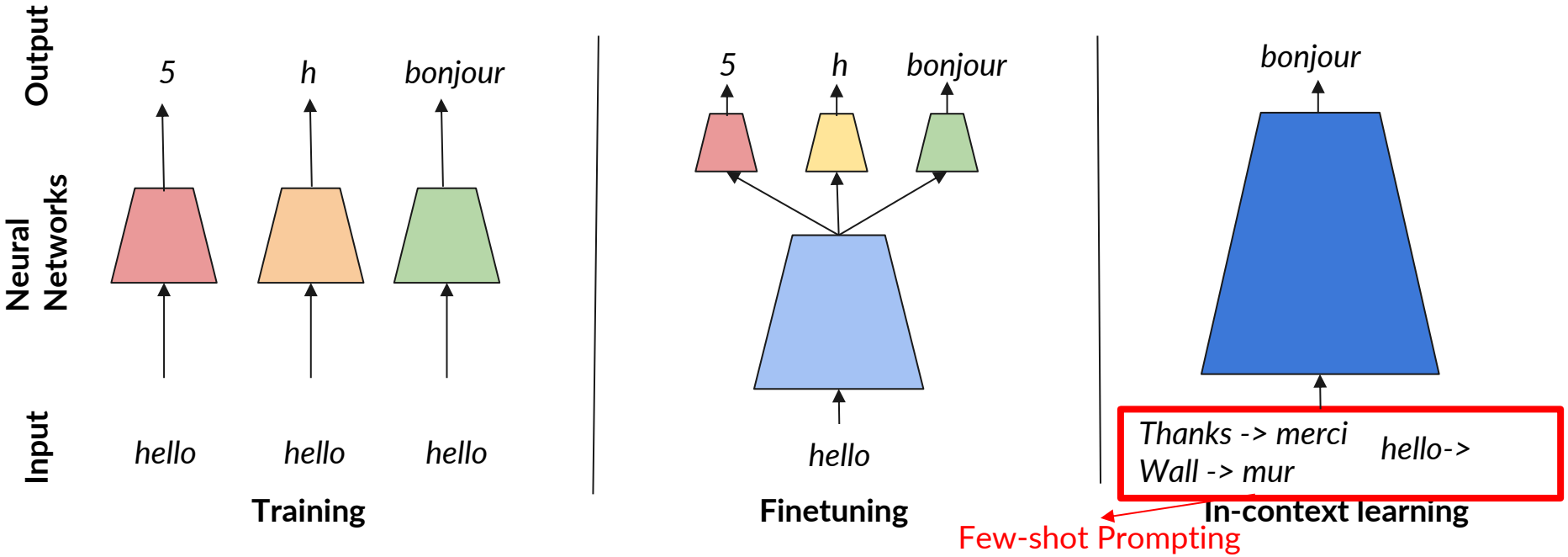
Huang, Q., Ren, H., Chen, P., Krvzmanc, G., Zeng, D.D., Liang, P., & Leskovec, J. (2023). PRODIGY: Enabling In-context Learning Over Graphs. NeurIPS 2023


Output
5
h
bonjour
Neural Networks
Input
hello
hello
hello
Training

Different Machine Learning Paradigms

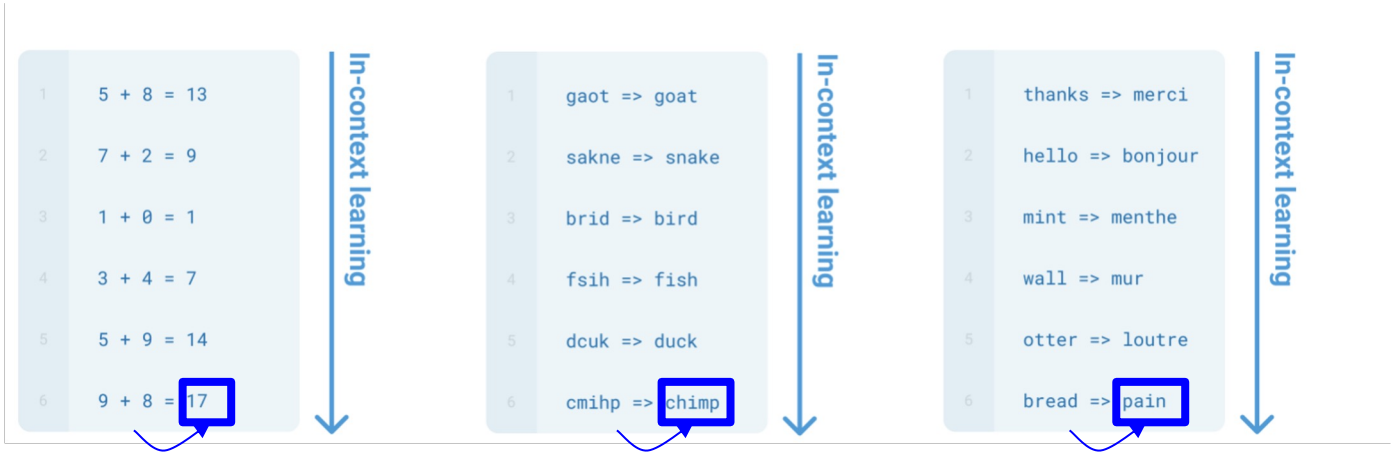


Different Machine Learning Paradigms



In-context Learning

Performing a new task by “learning” from the input context w/o gradient update. Very powerful — we only need one foundation model directly answering all tasks now!



Predicting sum

Unscrambling

translating

In-context Learning Over Graphs

Thanks -> merci
Wall -> mur

hello->?

Few-shot prompting over text

What is in-context learning over graphs?

Today's Plan

We formulate and enable **in-context learning over graphs**

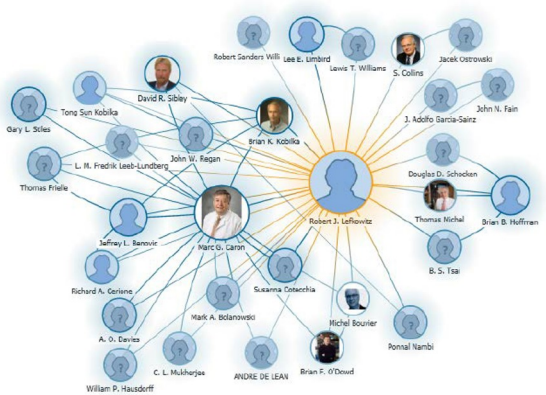
- **PRODIGY**: An in-context learner for graphs able to solve **novel tasks** on **novel graphs**.
  - **Prompt Graph representation:** represent the few-shot prompt for different graph tasks in the **same input format**, so that it can be consumed by one shared model
  - **Prompt Graph Inference:** in-context prediction GNN
  - **Pretraining:** generate diverse pretraining tasks in the format of PromptGraph
    - Neighbor Matching
    - MultiTask

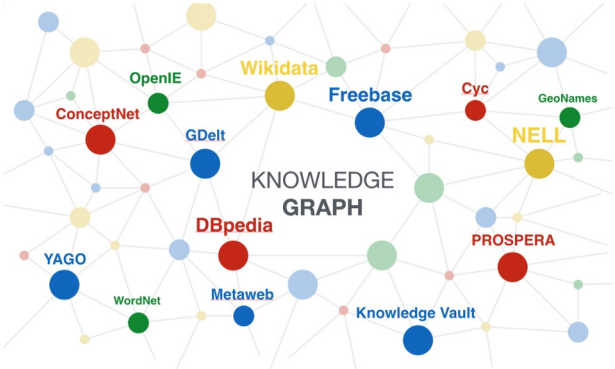
Formulation

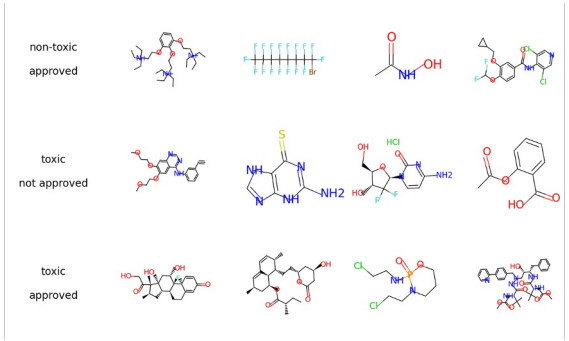
Graph Learning Tasks

What are the tasks on graphs?

- Node classification
- Link Prediction
- Graph classification







In-context Learning Over Graphs

Thanks -> merci
Wall -> mur

hello->?

Few-shot prompting over text

What is in-context learning over graphs?

In-context Learning Over Graphs: Link Prediction Example

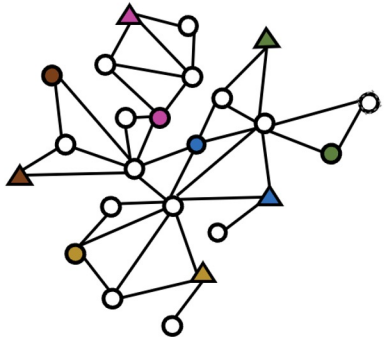
An in-context learner for graphs should be able to solve novel tasks on novel graphs without gradient updates.

Thanks -> merci
Wall -> mur

hello->?

different tasks

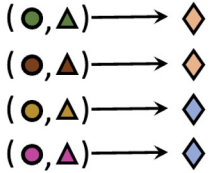
Graph $\mathcal{G}$:



different graphs

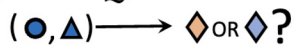
Prompt Examples $\mathcal{S}$:

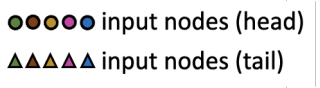
Input x Label y



different tasks

Queries $\mathcal{Q}$:





Few-shot prompting over text

Few-shot prompting over graph (for link classification)

But, how to achieve this? Two Challenges:

1. How to represent the few-shot prompt for **different graph tasks** in the **same input format**, so that it can be consumed by one shared model?

2. How to **pretrain** a model that can solve any task in this format?

But, how to achieve this? Two Challenges:

1. How to represent the few-shot prompt for **different graph tasks** in the **same input format**, so that it can be consumed by one shared model?

   => **Prompt Graph**: represent each few-shot prompt over graph as a meta hierarchical graph

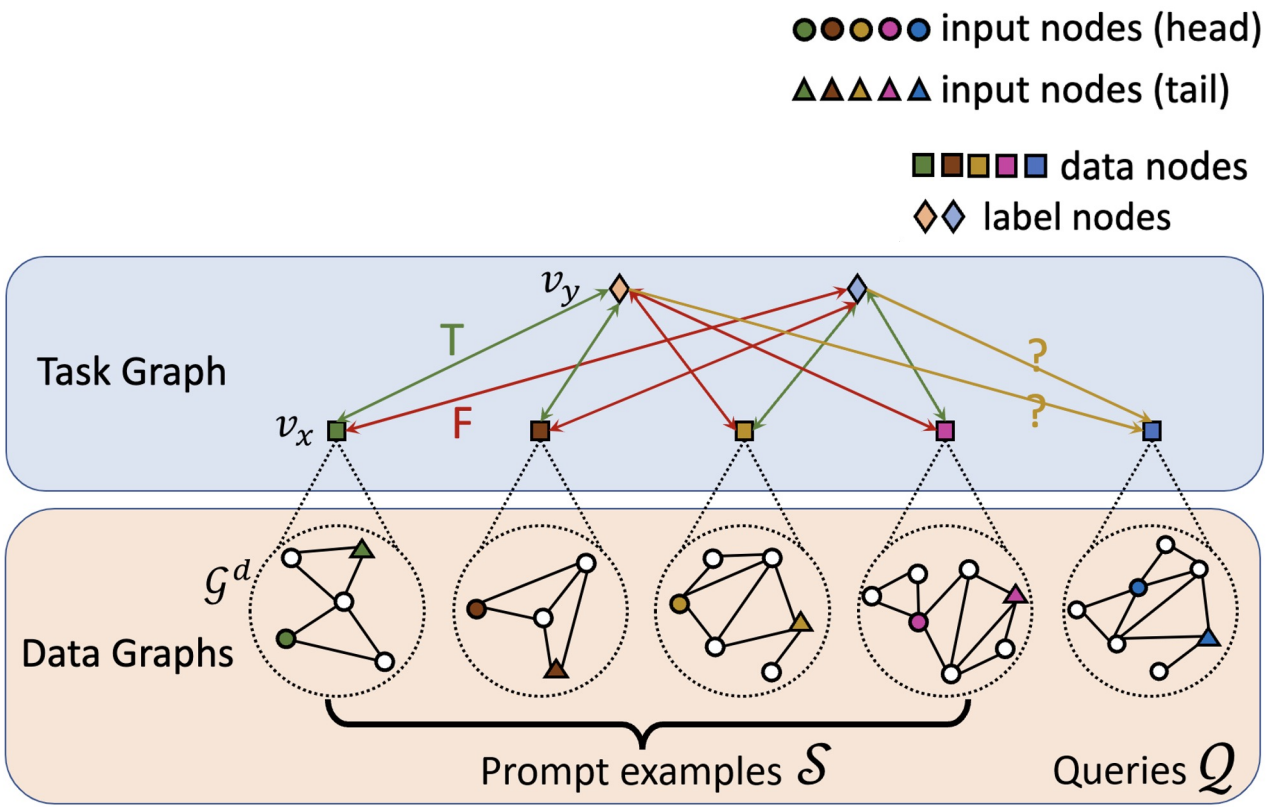
2. How to **pretrain** a model that can solve any task in this format?

   => **PGPretraining**: Pretrain a message passing model over self-supervised tasks in PromptGraph format with diverse underlying structures

Prompt Graph

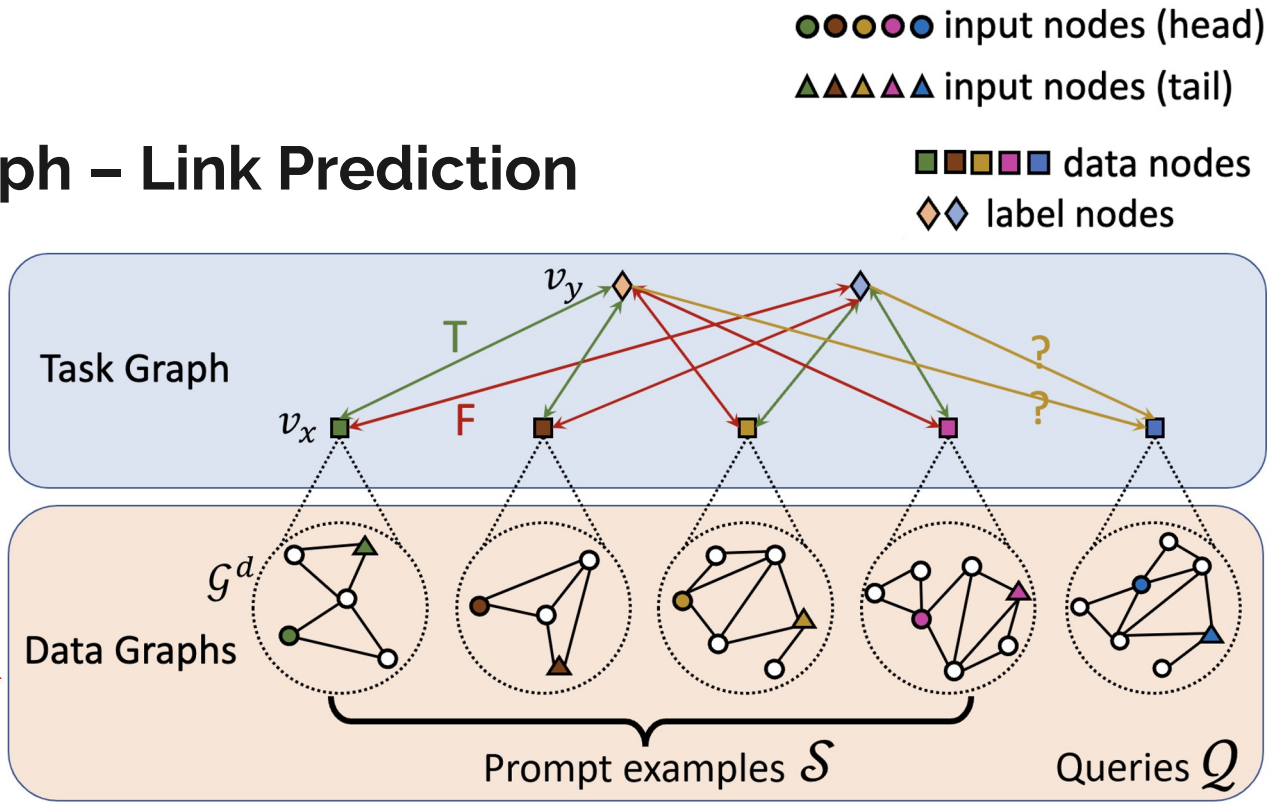
Prompt Graph

Prompt Graph is a unified representation of few-shot prompts over graph for diverse tasks



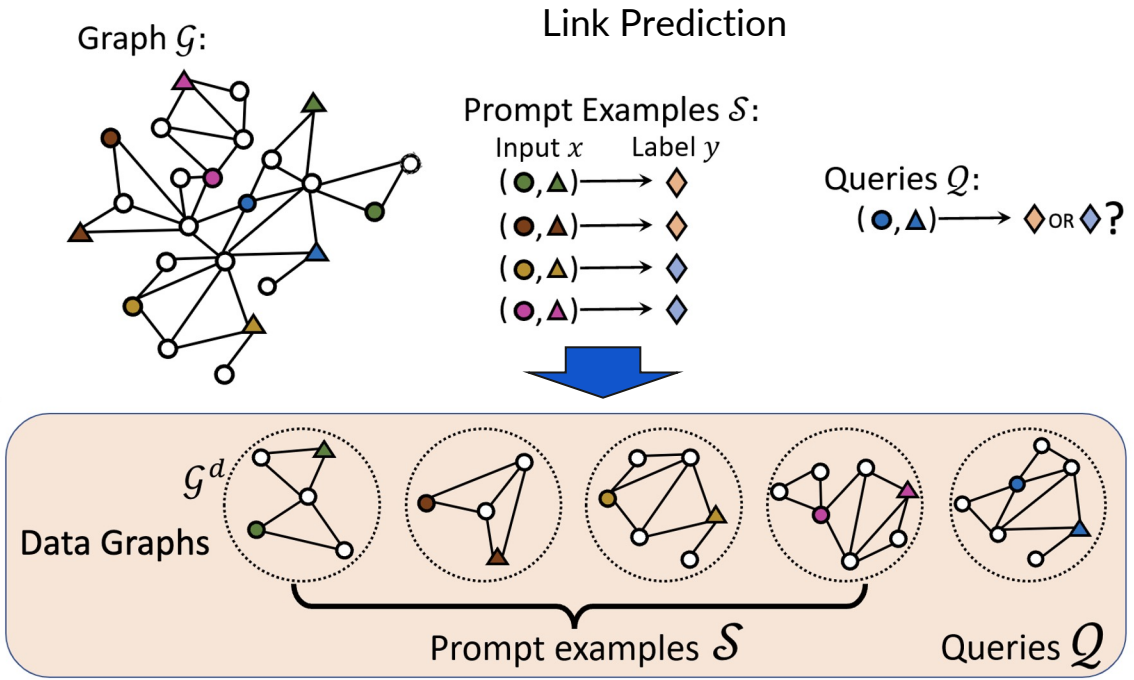
Step1: Data Graph – Link Prediction

Data Graph contextualizes each input x in the graph G (e.g. by **subgraph extraction**)



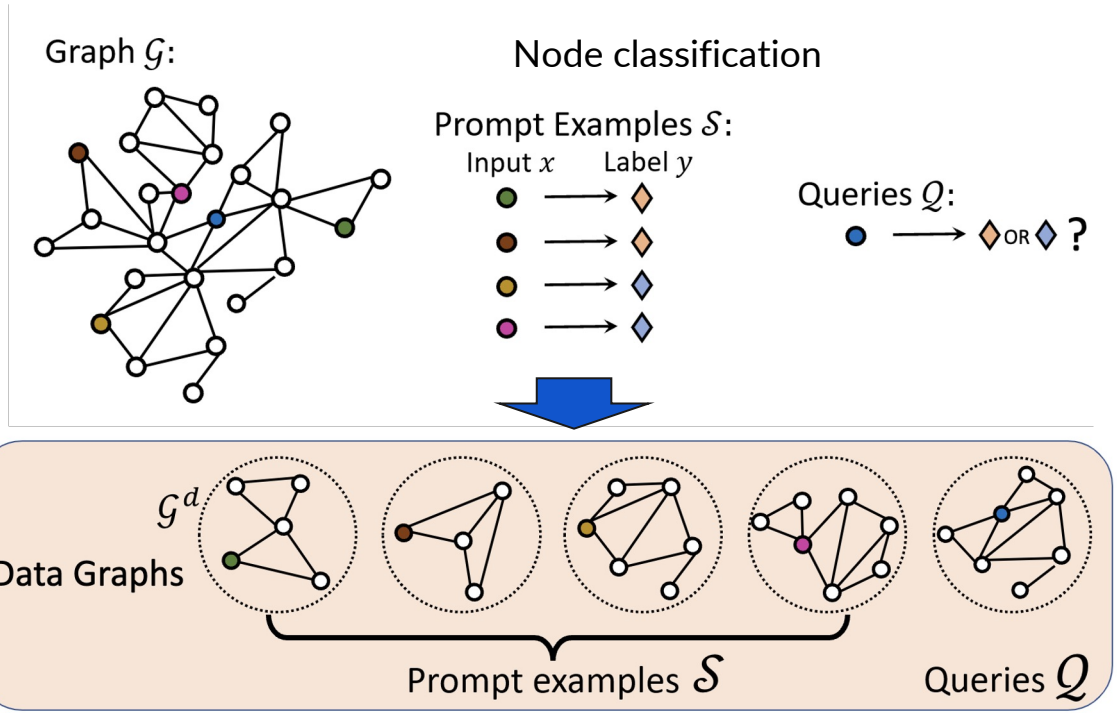
Step1: Data Graph – Link Prediction

Data Graph contextualizes each input x in the graph G (e.g. by **subgraph extraction)**



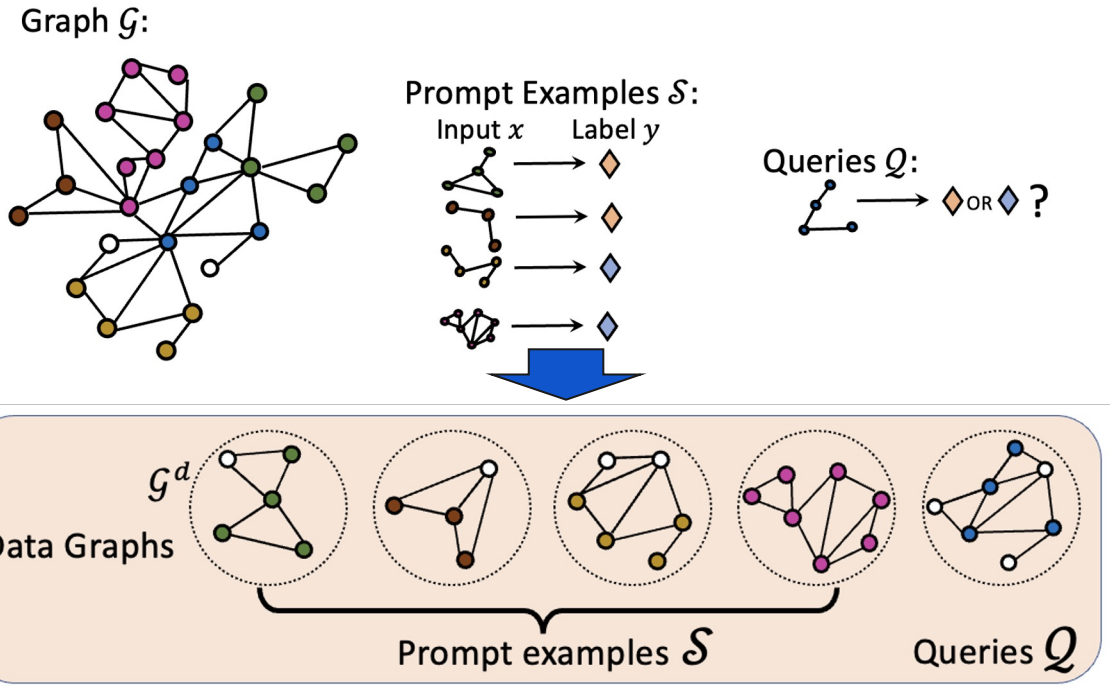
Step1: Data Graph – Node Classification

Data Graph contextualizes each input x in the graph G (e.g. by **subgraph extraction)**



Step1: Data Graph – Graph Classification

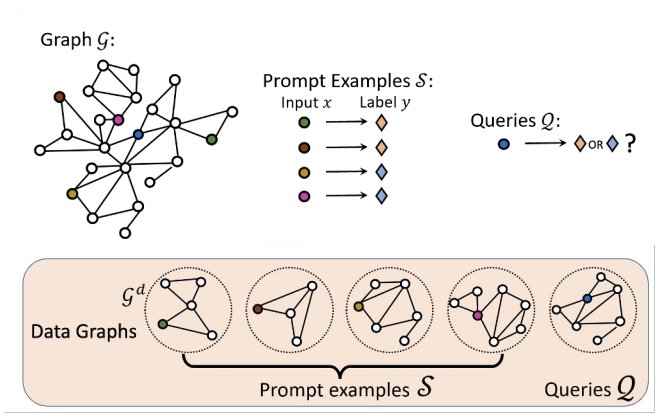
Data Graph contextualizes each input x in the graph G (e.g. by **subgraph extraction)**



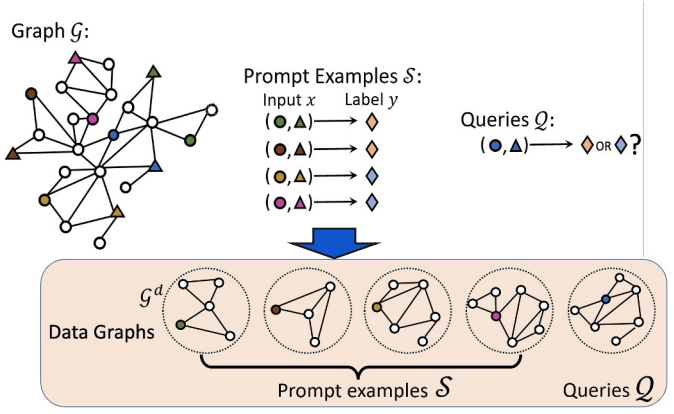
Step1: DataGraph Construction

DataGraph unifies the input format:

- Use different **input node set** for different classification over different levels (nodes vs edge vs graph)
- Use text feature to unify features over different datasets



Node Classification

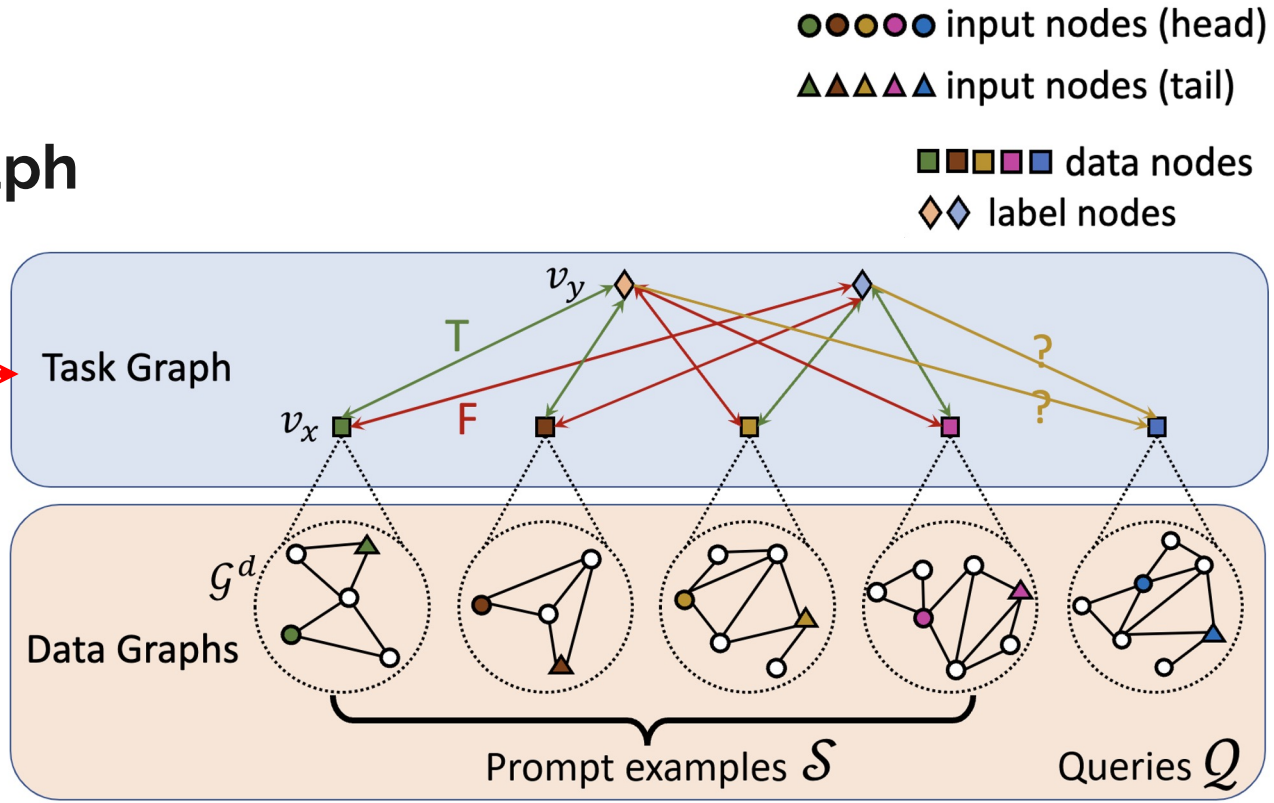


Edge Classification

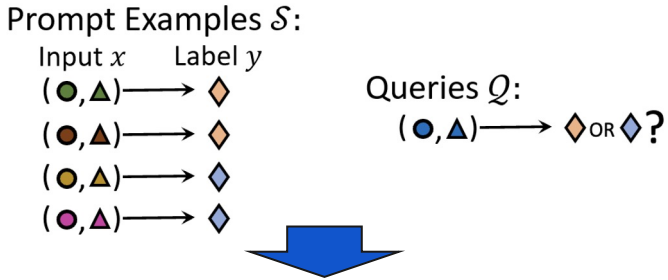
...

Step2: Task Graph

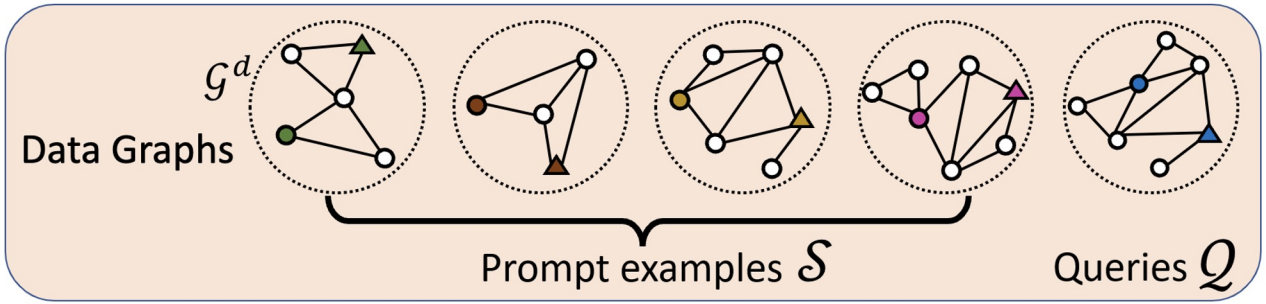
Task Graph interconnects inputs and labels across examples to form context for queries



Step2: Task Graph – Link Prediction

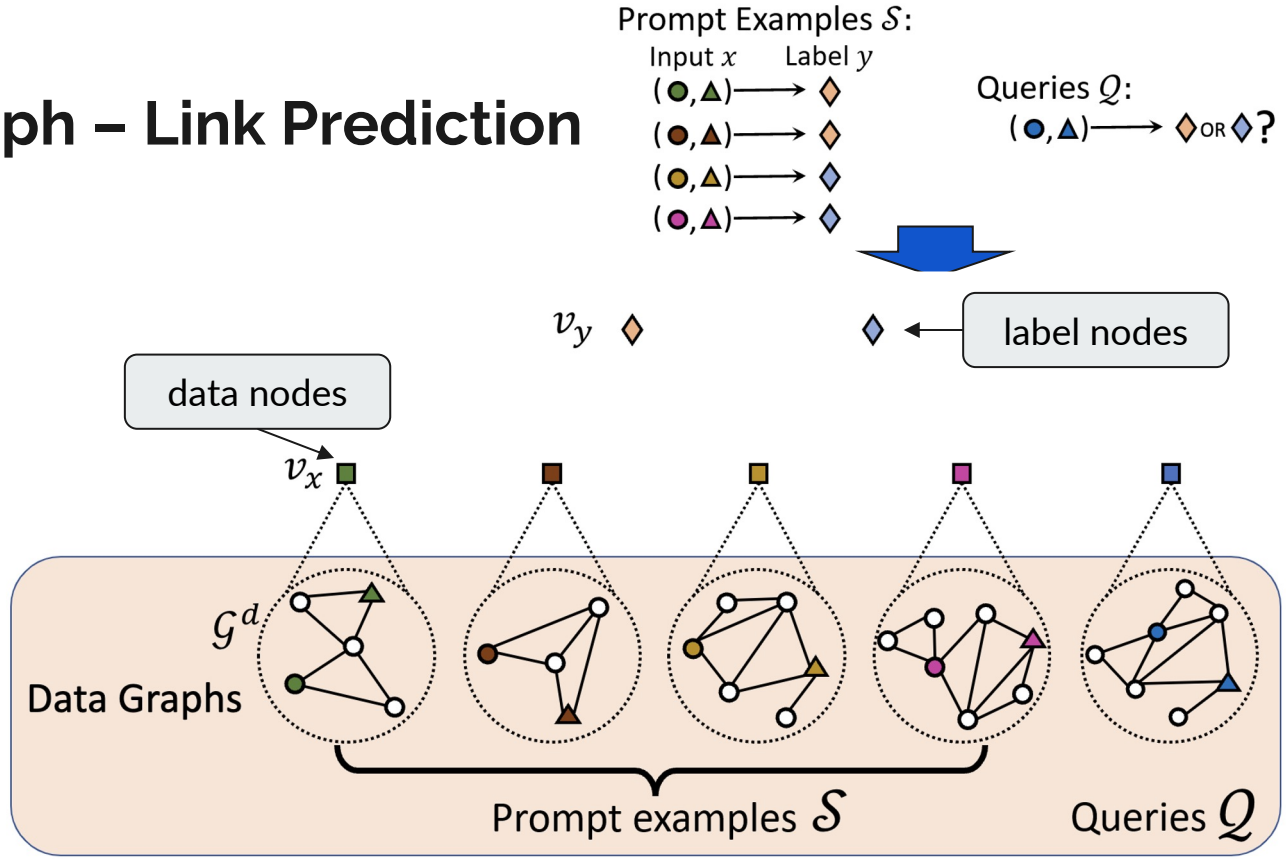


Task Graph interconnects inputs and labels across examples to form context for queries



Step2: Task Graph – Link Prediction

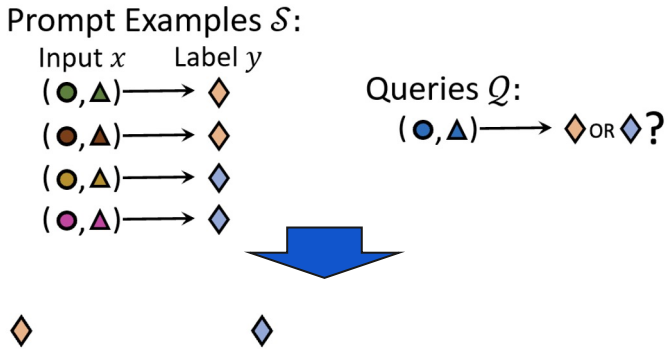
Task Graph interconnects inputs and labels across examples to form context for queries

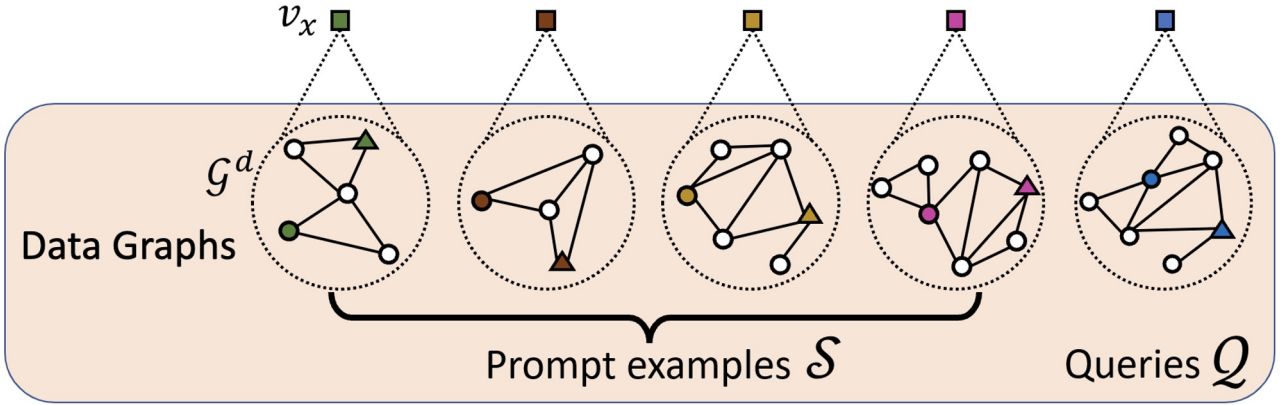


Step2: Task Graph – Link Prediction

Task Graph interconnects inputs and labels across examples to form context for queries

- **Prompt examples**: bidirectional edges between data nodes and all label nodes

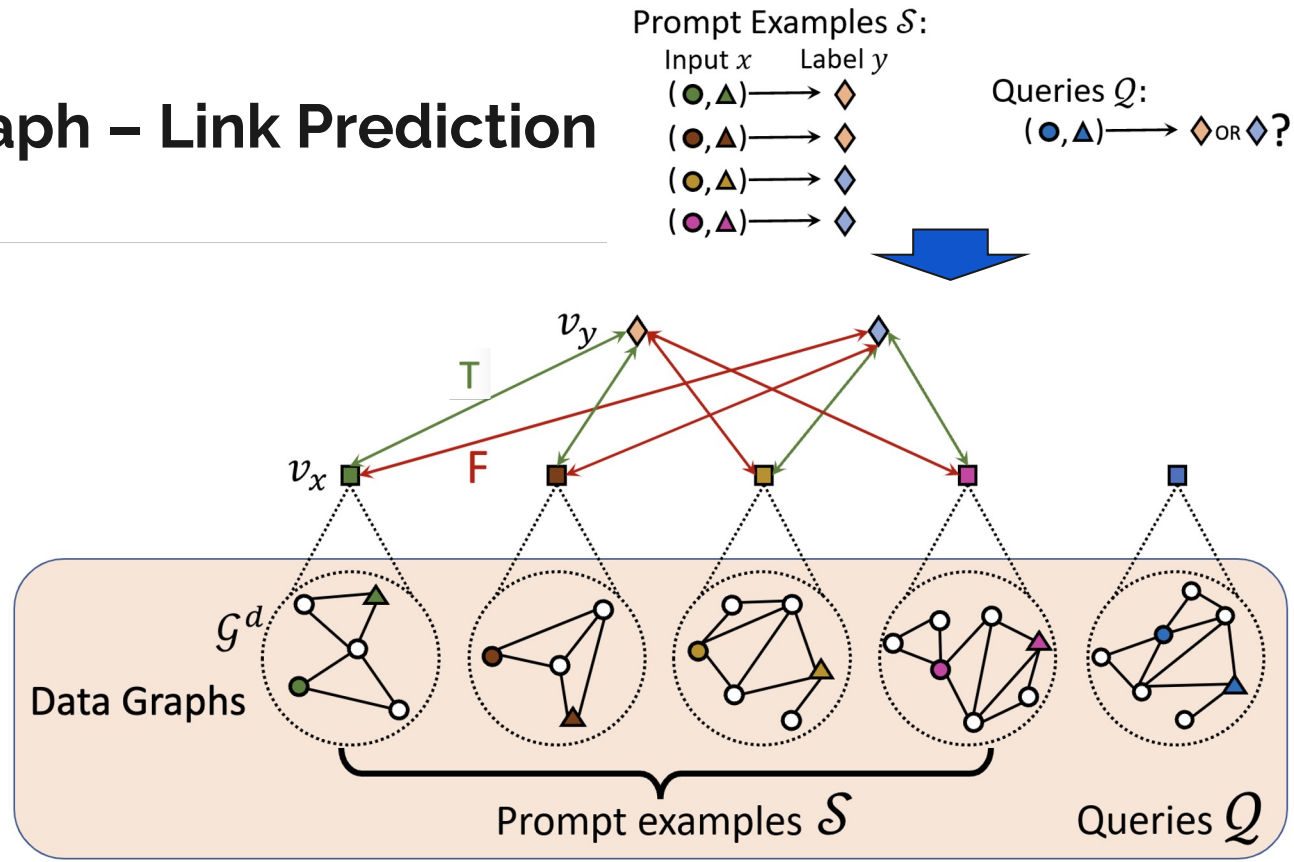




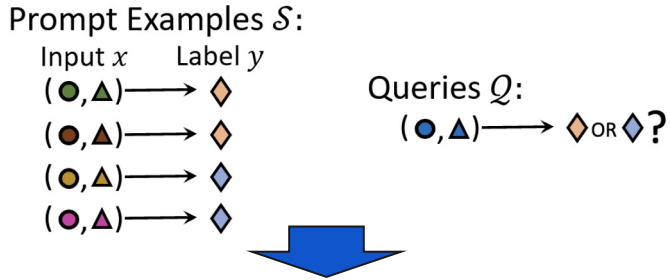
Step2: Task Graph – Link Prediction

Task Graph interconnects inputs and labels across examples to form context for queries

- **Prompt examples**: bidirectional edges between data nodes and all label nodes

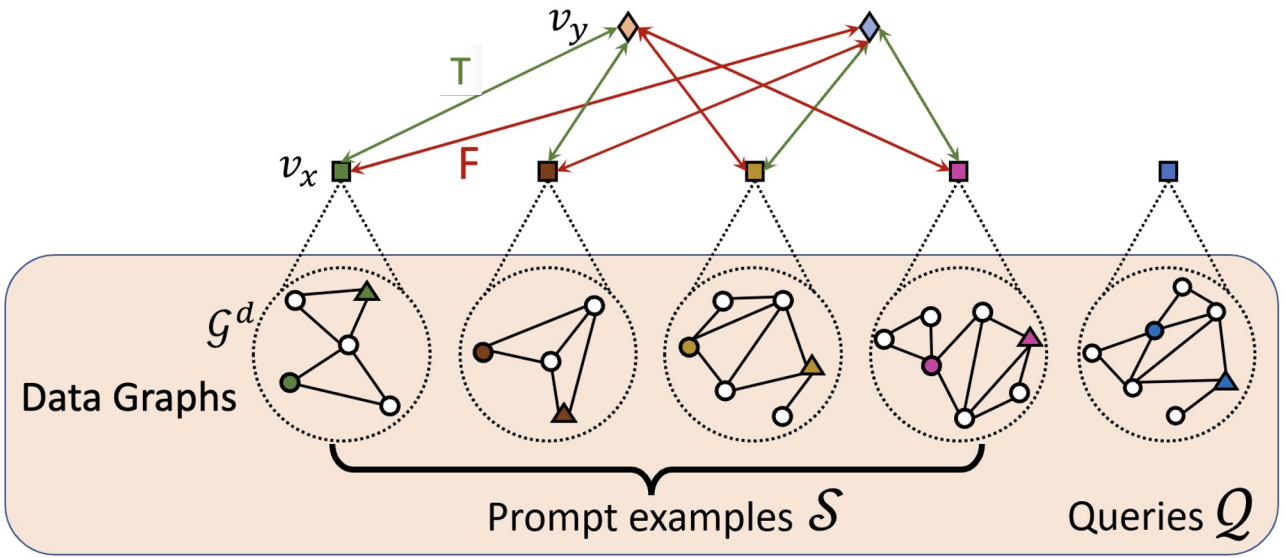


Step2: Task Graph – Link Prediction



Task Graph interconnects inputs and labels across examples to form context for queries

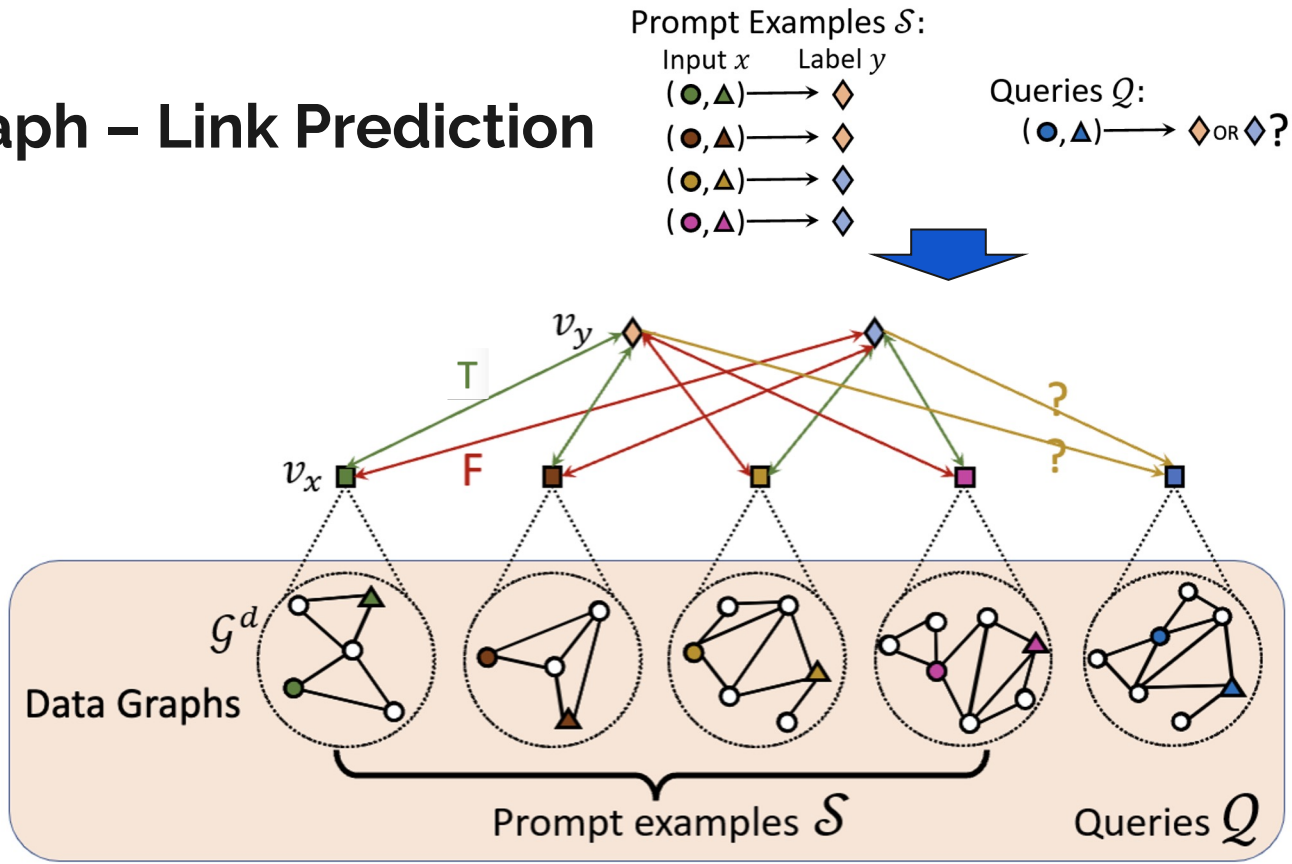
- **Prompt examples**: bidirectional edges between data nodes and all label nodes
- **Queries**: single directed edges from each label to each data node



Step2: Task Graph – Link Prediction

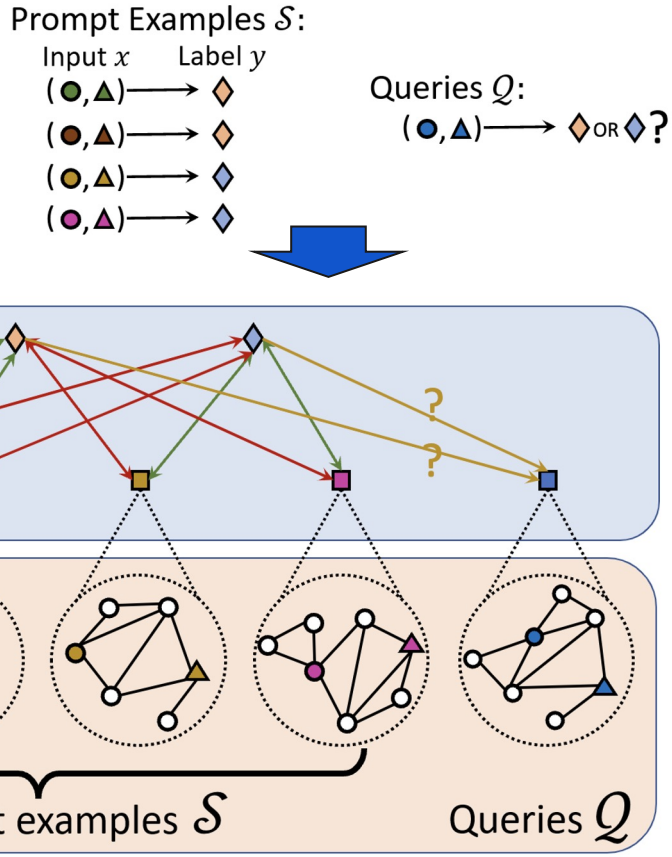
Task Graph interconnects inputs and labels across examples to form context for queries

- **Prompt examples**: bidirectional edges between data nodes and all label nodes
- **Queries**: single directed edges from each label to each data node

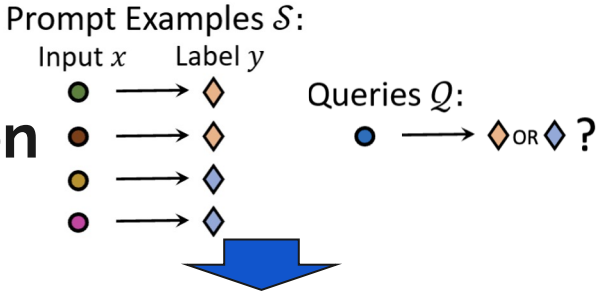


Step2: Task Graph – Link Prediction

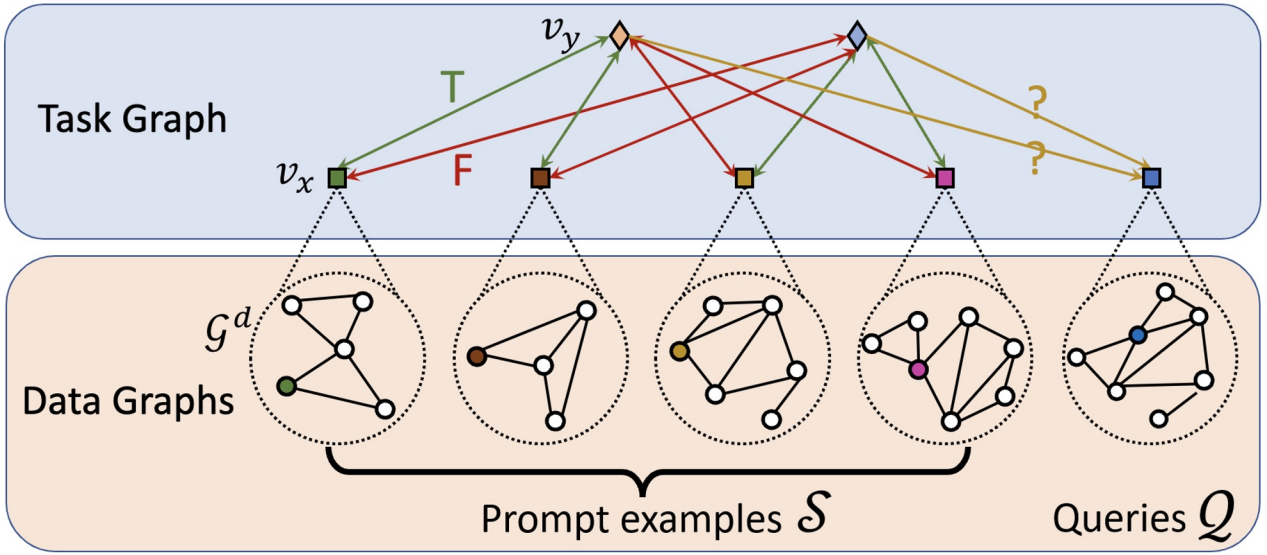
Task Graph interconnects inputs and labels across examples to form context for queries



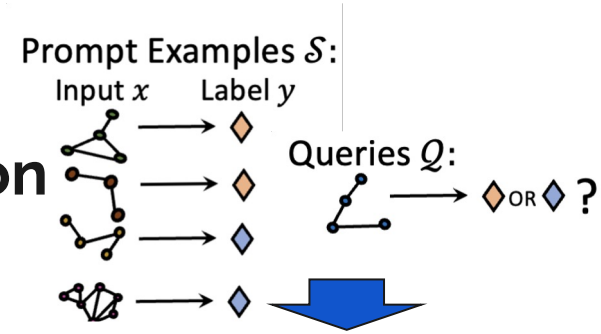
Step2: Task Graph – Node Classification



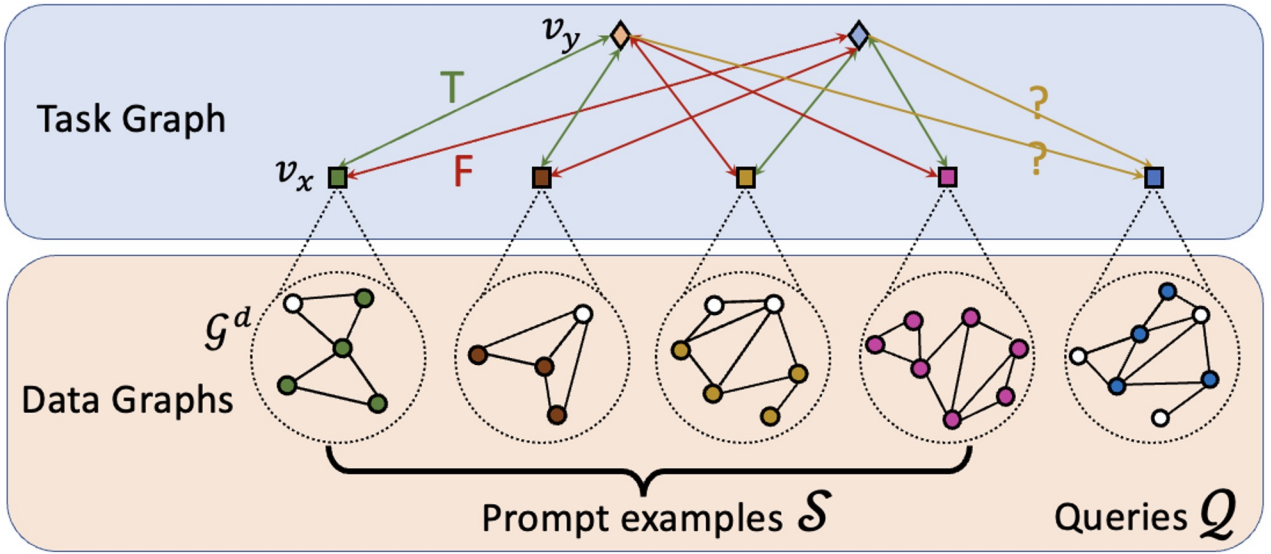
Task Graph interconnects inputs and labels across examples to form context for queries



Step2: Task Graph – Graph Classification



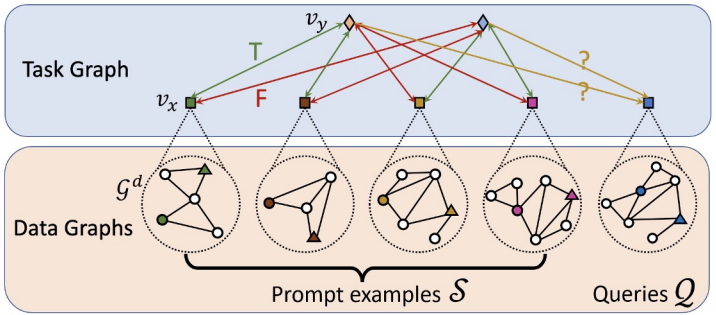
Task Graph interconnects inputs and labels across examples to form context for queries



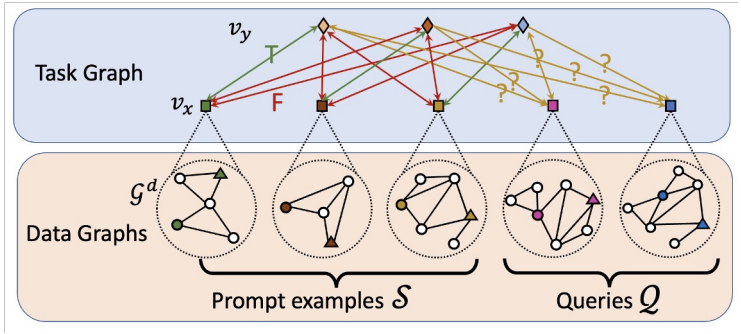
Flexibility of Task Graph

Task Graph unifies classification task format:

- Different number of **classes** are represented as different number of **label nodes**
- Different number of **prompt examples (i.e. shots)** and queries are represented as different number of **data nodes** as well as how they connect with label nodes



2-shots prompt for 2-class classification



1-shot prompt for 3-class classification

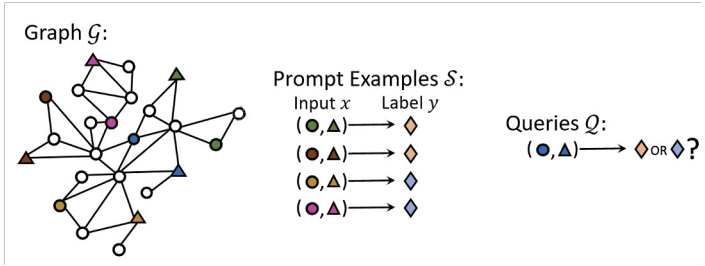
...

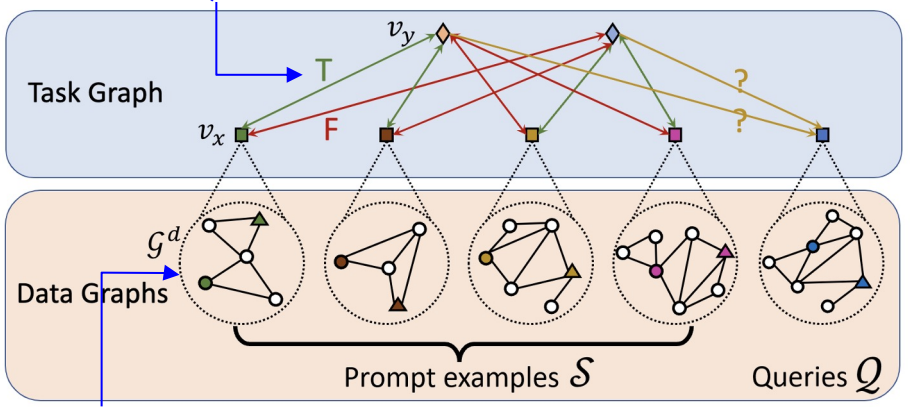
Prompt Graph for in-context learning

input nodes (head)
input nodes (tail)
data nodes
label nodes

How to use **PromptGraph** for in-context learning?

Reflects what is the task using examples (commonalities and differences)





Captures all relevant information about the input

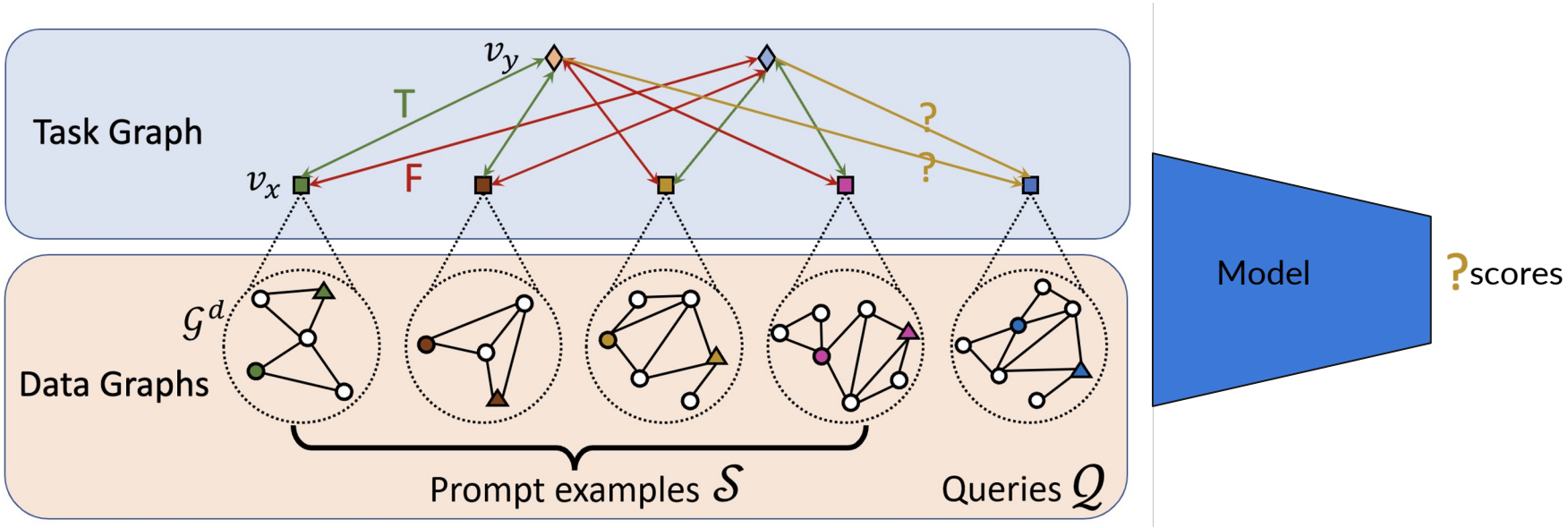
In-context learning over graph

inductive **link prediction** over hierarchical graph

PrompGraph Inference

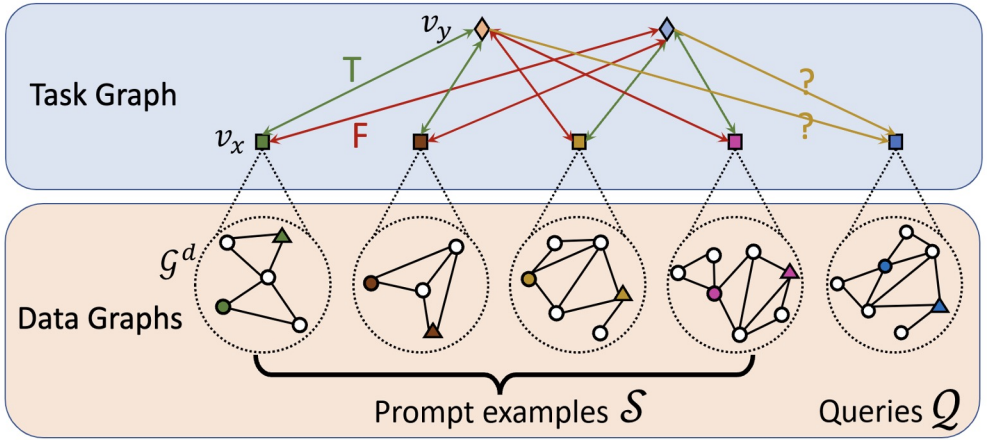
In context prediction GNN

In context prediction GNN



In context prediction GNN

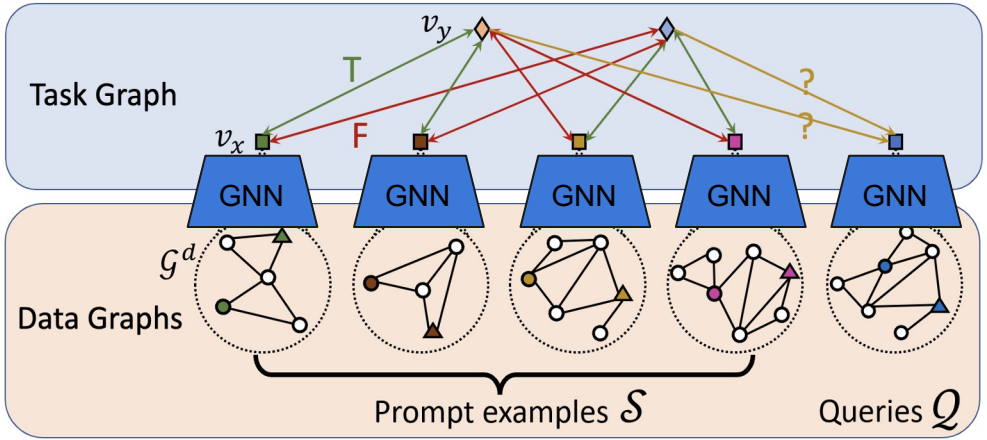
Hierarchical Message passing over PromptGraph



In context prediction GNN

Hierarchical Message passing over PromptGraph

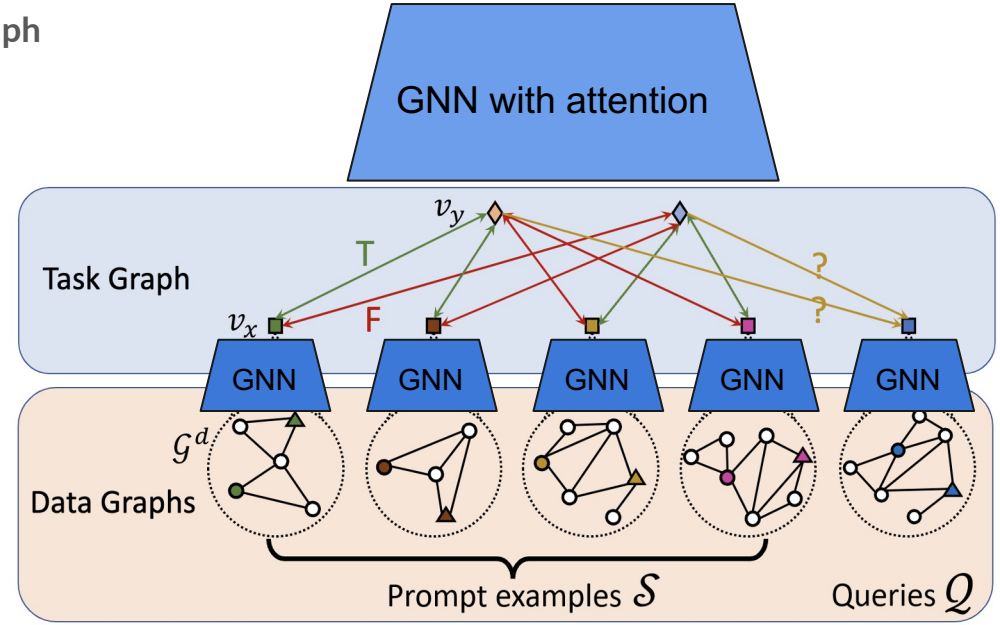
- Data Graph Encoder



In context prediction GNN

Hierarchical Message passing over PromptGraph

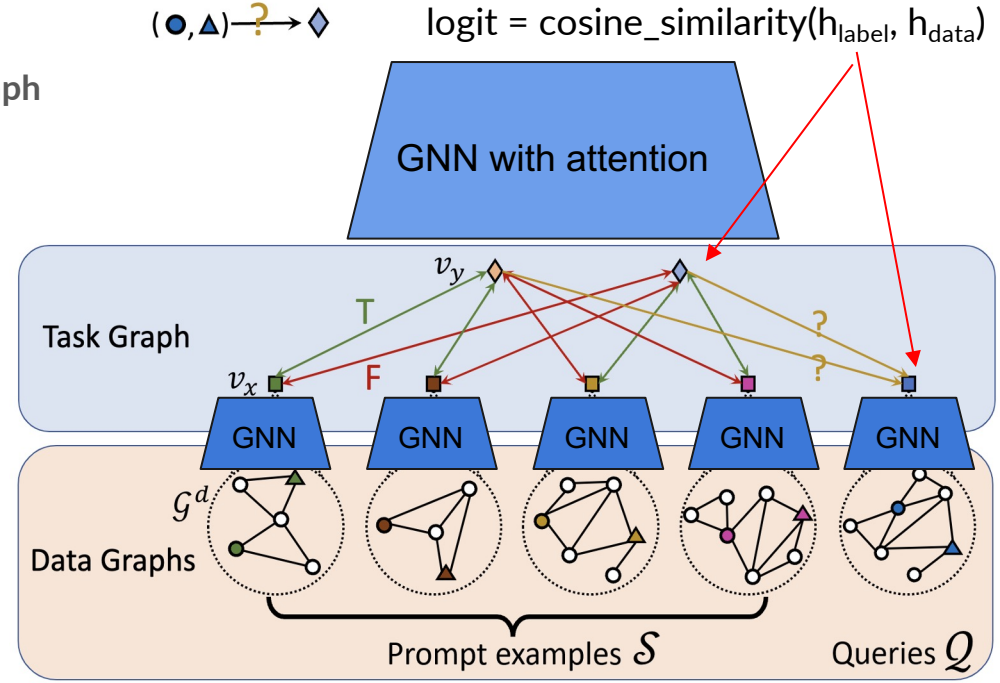
- Data Graph Encoder
- Message Passing over Task Graph



In context prediction GNN

Hierarchical Message passing over PromptGraph

- Data Graph Encoder
- Message Passing over Task Graph
- Compute Logits



PRODIGY Pretraining

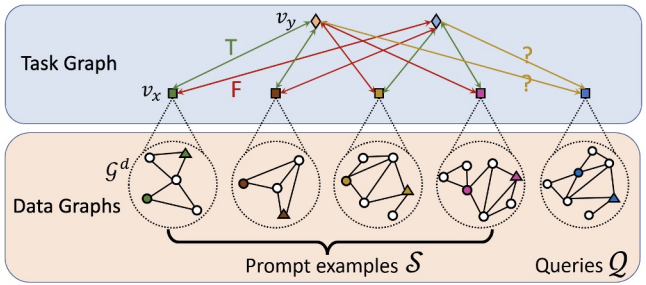
In-context Pretraining

We want to generate pretraining tasks in the format of **PromptGraph** and pretrain our model over them!

Pretraining

Data/Graph

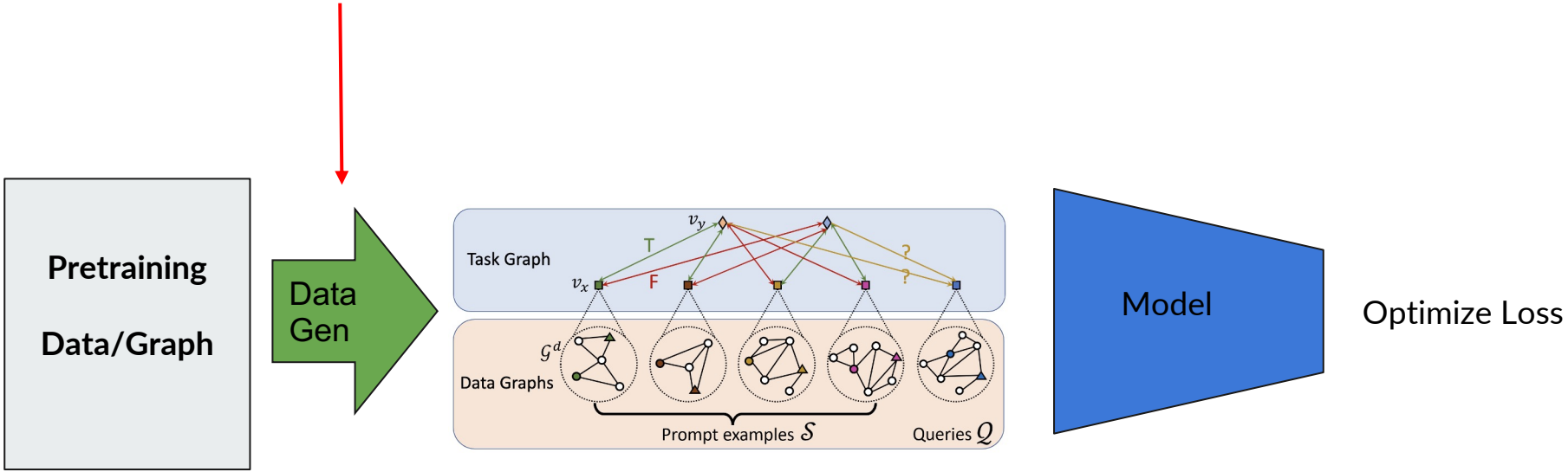
Data Gen



Model

Optimize Loss

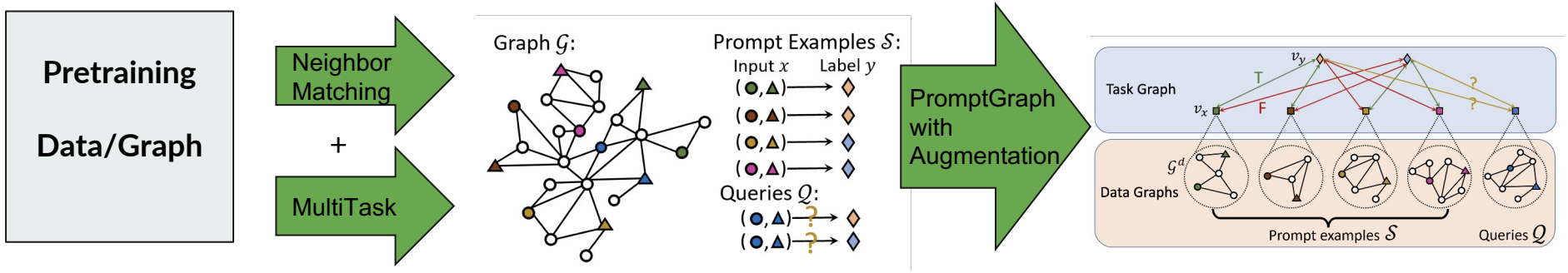
Pretraining Data Generation



Pretraining Data Generation

Two stages:

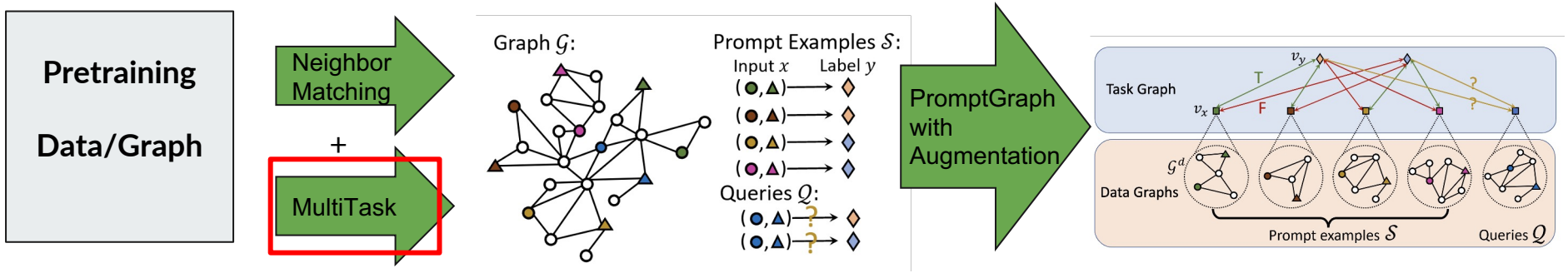
few-shot prompt generation **convert to PromptGraph**



Pretraining Data Generation

Two stages:

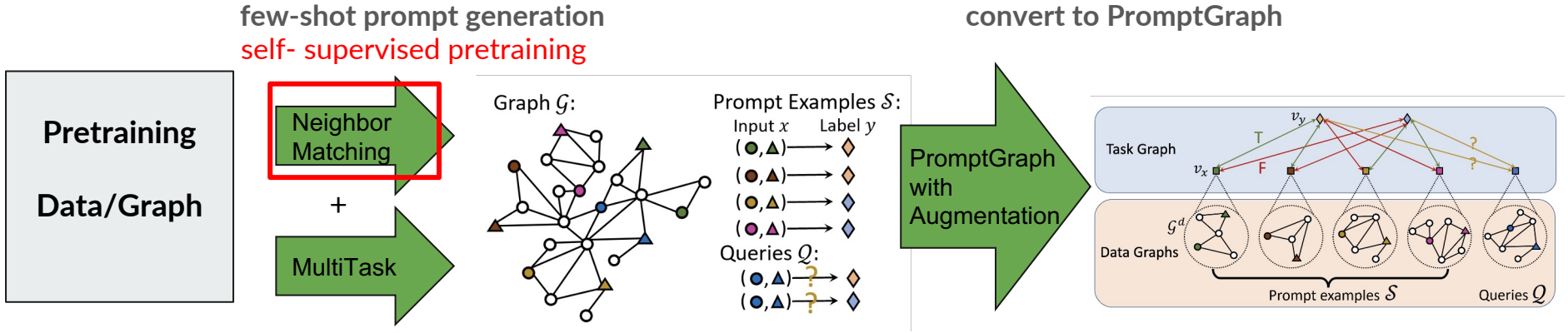
few-shot prompt generation **convert to PromptGraph**



Simply select multiple tasks for pretraining
-> Supervised pretraining

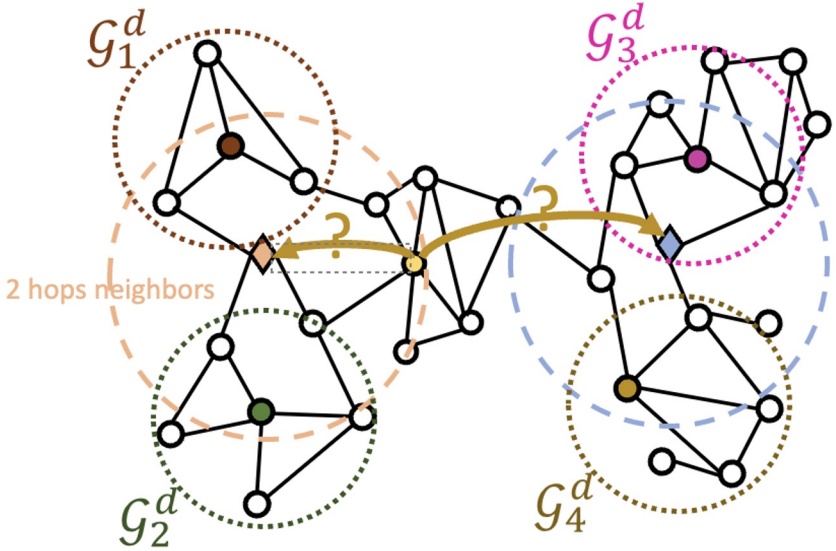
Pretraining Data Generation

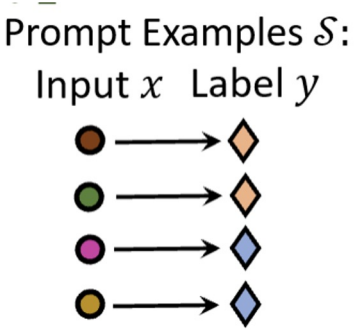
Two stages:

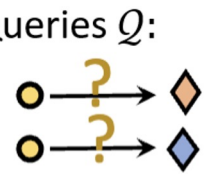


Self-supervised Task Example: Neighbor Matching

Idea: the task is to classify which neighborhood a node is in, where each neighborhood is defined by other nodes in it.



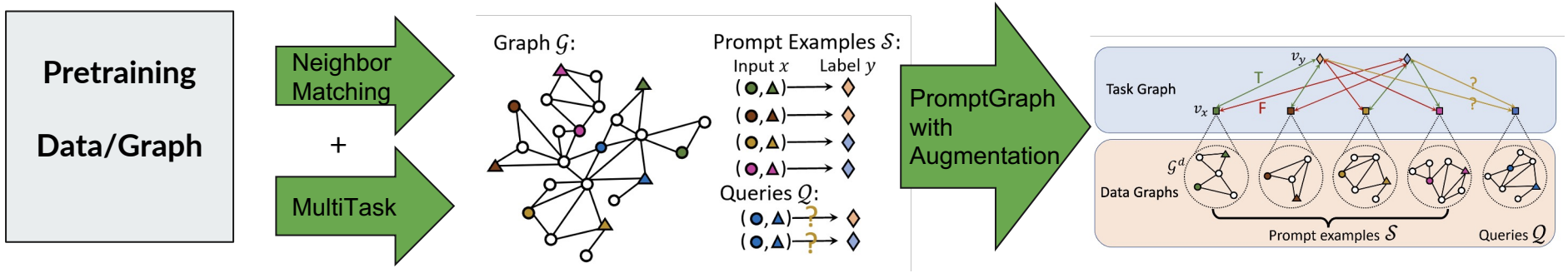




Pretraining Data Generation

Two stages:

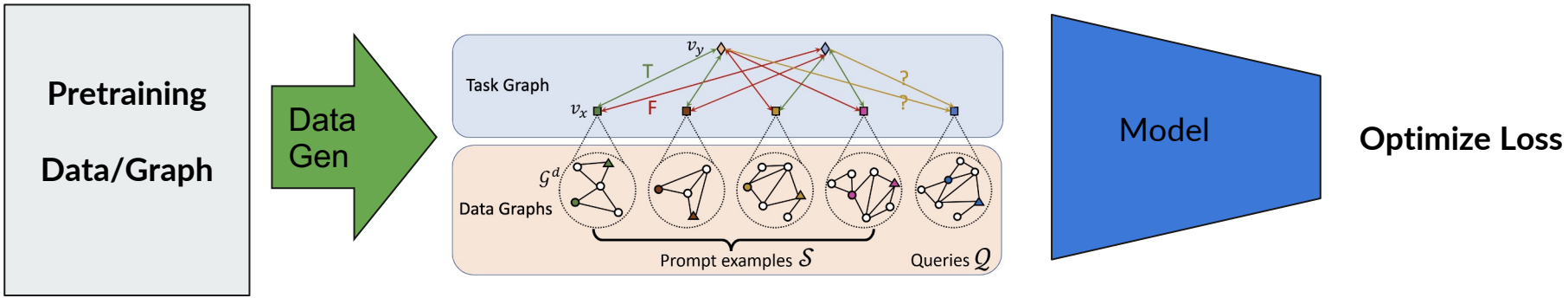
few-shot prompt generation **convert to PromptGraph**



Mixing different data generation pipeline gives more diverse pretraining data and better results!

In-context Pretraining Objectives

- **Classification loss** over generated tasks
- Attribute prediction loss: reconstruct corrupted features during DataGraph augmentation



Experimental Setup

Evaluation and Datasets

- We use 2 datasets for pretraining:
  - **MAG240M**, a large scale citation network
  - **Wiki**, a knowledge graph constructed from Wikipedia (Xiong et al.,2018b).
- After pretraining, we evaluate the in-context learning performance on diverse classification tasks over downstream graphs respectively
  - **Paper category prediction:**
    - **arXiv**
  - **KG Link Prediction:**
    - **FB15K-237, NELL, ConceptNet**

Experimental Setup

Baselines and our methods

- We consider three baselines:
  - **NoPretrain**, which uses a randomly-initialized model with the same architecture as our pretrained models
  - **Contrastive, w**hich employs a standard contrastive learning method with the same augmentation as above and uses a hard-coded nearest neighbor algorithm to adapt to our in-context learning setting
  - **Finetune**, which trains an additional linear classification head on top of the graph encoder pretrained with contrastive learning
- We consider three variants of out methods with different pretraining data genereration
  - **PG-NM**, which uses Neighbor Matching task for pretraining
  - **PG-MT,** which employs Multi-task pretraining
  - **PG-ALL,** which combines the previous two methods

How do we handle graphs with different features?

Here we use **text embedding** provided by pretrained language model (e.g. Bert).

In-context Learning Results

Paper category classification: pretrain on MAG -> in-context learning on Arxiv

Classes	NoPretrain	Contrastive	PG-NM	PG-MT	PRODIGY	Finetune
3	33.16 ± 0.30	65.08 ± 0.34	72.50 ± 0.35	65.64 ± 0.33	**73.09 ± 0.36**	65.42 ± 5.53
5	18.33 ± 0.21	51.63 ± 0.29	61.21 ± 0.28	51.97 ± 0.27	**61.52 ± 0.28**	53.49 ± 4.61
10	9.19 ± 0.11	36.78 ± 0.19	46.12 ± 0.19	37.23 ± 0.20	**46.74 ± 0.20**	30.22 ± 3.77
20	4.72 ± 0.06	25.18 ± 0.11	33.71 ± 0.12	25.91 ± 0.12	**34.41 ± 0.12**	17.68 ± 1.15
40	2.62 ± 0.02	17.02 ± 0.07	23.69 ± 0.06	17.19 ± 0.08	**25.13 ± 0.07**	8.04 ± 3.00

In-context Learning Results

Paper category classification: pretrain on MAG -> in-context learning on Arxiv

Classes	NoPretrain	Contrastive	PG-NM	PG-MT	PRODIGY	Finetune
3	33.16 ± 0.30	65.08 ± 0.34	72.50 ± 0.35	65.64 ± 0.33	**73.09 ± 0.36**	65.42 ± 5.53
5	18.33 ± 0.21	51.63 ± 0.29	61.21 ± 0.28	51.97 ± 0.27	**61.52 ± 0.28**	53.49 ± 4.61
10	9.19 ± 0.11	36.78 ± 0.19	46.12 ± 0.19	37.23 ± 0.20	**46.74 ± 0.20**	30.22 ± 3.77
20	4.72 ± 0.06	25.18 ± 0.11	33.71 ± 0.12	25.91 ± 0.12	**34.41 ± 0.12**	17.68 ± 1.15
40	2.62 ± 0.02	17.02 ± 0.07	23.69 ± 0.06	17.19 ± 0.08	**25.13 ± 0.07**	8.04 ± 3.00

Result1: PRODIGY improves performance in all cases, up to **48%** over contrastive and **3X** over Finetune.

In-context Learning Results

Paper category classification: pretrain on MAG -> in-context learning on Arxiv

Classes	NoPretrain	Contrastive	PG-NM	PG-MT	PRODIGY	Finetune
3	33.16 ± 0.30	65.08 ± 0.34	72.50 ± 0.35	65.64 ± 0.33	**73.09 ± 0.36**	65.42 ± 5.53
5	18.33 ± 0.21	51.63 ± 0.29	61.21 ± 0.28	51.97 ± 0.27	**61.52 ± 0.28**	53.49 ± 4.61
10	9.19 ± 0.11	36.78 ± 0.19	46.12 ± 0.19	37.23 ± 0.20	**46.74 ± 0.20**	30.22 ± 3.77
20	4.72 ± 0.06	25.18 ± 0.11	33.71 ± 0.12	25.91 ± 0.12	**34.41 ± 0.12**	17.68 ± 1.15
40	2.62 ± 0.02	17.02 ± 0.07	23.69 ± 0.06	17.19 ± 0.08	**25.13 ± 0.07**	8.04 ± 3.00

Result2: PRODIGY can induce strong in-context learning with very different self-supervised pretraining tasks.

Pretraining has no idea about classifying paper categories!

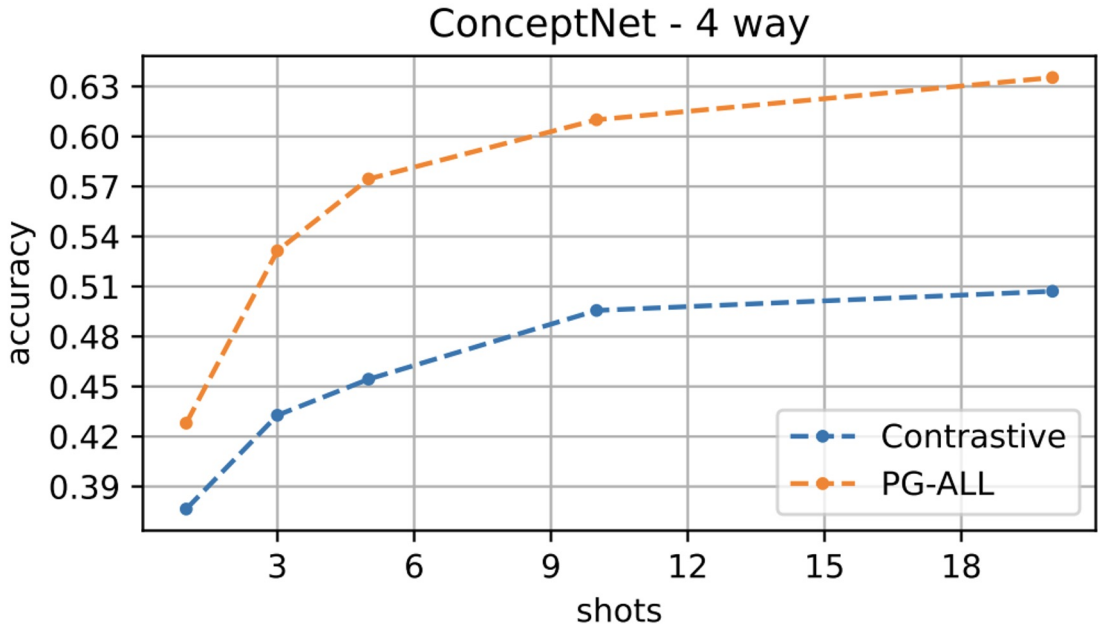
In-context Learning Results

Paper category classification: pretrain on MAG -> in-context learning on Arxiv

Classes	NoPretrain	Contrastive	PG-NM	PG-MT	PRODIGY	Finetune
3	33.16 ± 0.30	65.08 ± 0.34	72.50 ± 0.35	65.64 ± 0.33	**73.09 ± 0.36**	65.42 ± 5.53
5	18.33 ± 0.21	51.63 ± 0.29	61.21 ± 0.28	51.97 ± 0.27	**61.52 ± 0.28**	53.49 ± 4.61
10	9.19 ± 0.11	36.78 ± 0.19	46.12 ± 0.19	37.23 ± 0.20	**46.74 ± 0.20**	30.22 ± 3.77
20	4.72 ± 0.06	25.18 ± 0.11	33.71 ± 0.12	25.91 ± 0.12	**34.41 ± 0.12**	17.68 ± 1.15
40	2.62 ± 0.02	17.02 ± 0.07	23.69 ± 0.06	17.19 ± 0.08	**25.13 ± 0.07**	8.04 ± 3.00

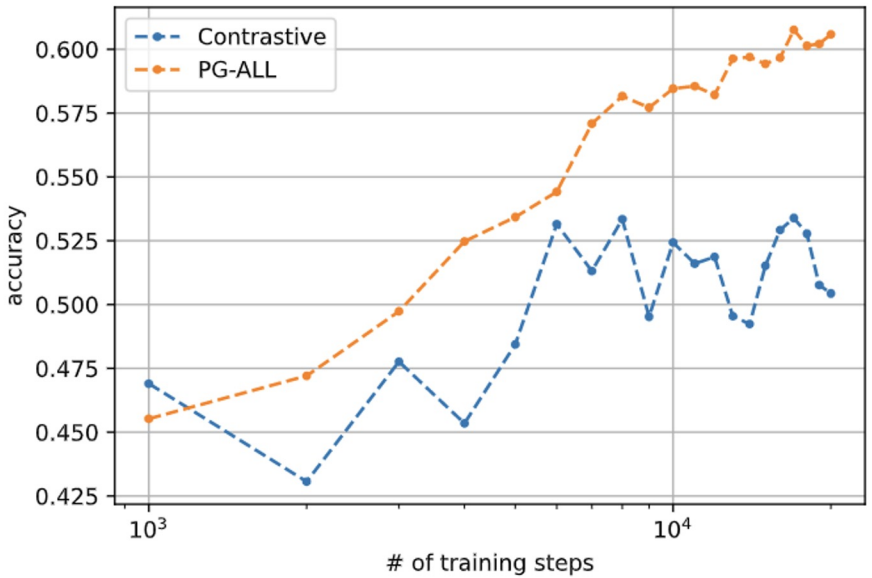
Result3: Mixing different data generation pipeline gives more diverse pretraining data and better results!

Performance Scaling with More Shots



Pretraining with only 3 shots => model can learn from context from much more than 3 shots!

Data Scaling



Hard and diverse pretraining tasks generation pipeline allows the model to keep scaling with more training data

Conclusion

PRODIGY enables **in-context learning over graphs** with a novel in-context task representation, Prompt Graph, and corresponding pretraining architecture and objectives.

- **Prompt Graph** provides a unified representation of few-shot prompts over graph for diverse tasks
- **Pretrain** a hierarchical message passing model over Prompt Graph enables in-context learning over unseen tasks on unseen graphs
- **Hard and diverse pretraining tasks** allow the model to keep scaling with more training data

Relational Transformer: Toward Zero-Shot Foundation Models for Relational Data, Ranjan, Hudovernik, Znidar, C.K. et al. 2025

Designing Transformers for Relational Data:

Towards Foundation Models for Forecasting over Relational Databases

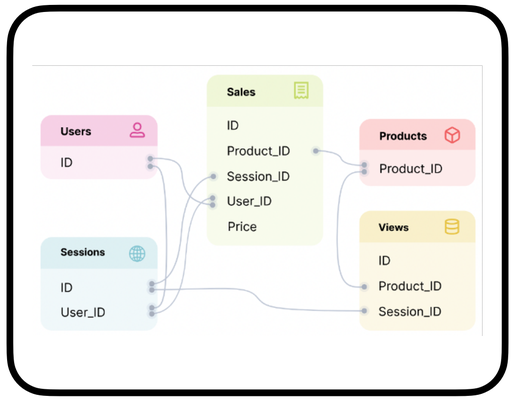
Charilaos Kanatsoulis
Stanford University



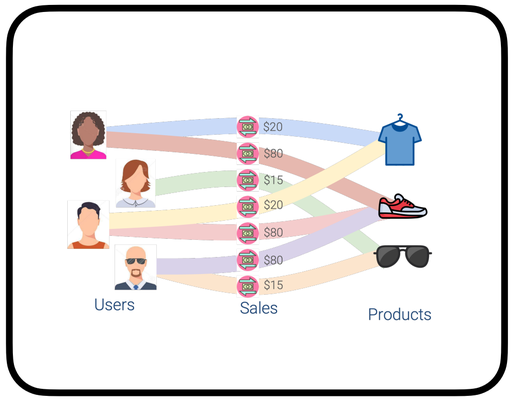


Relational Deep Learning (RDL) [Fey et al. 2023]

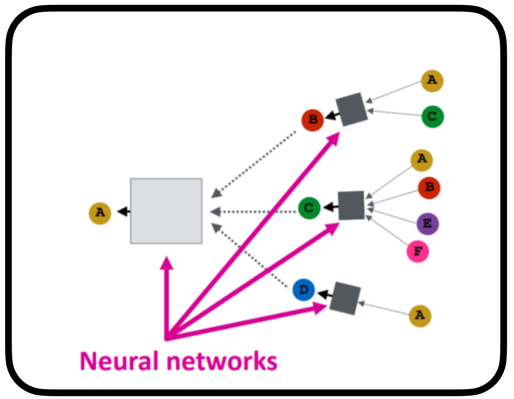
Relational DB



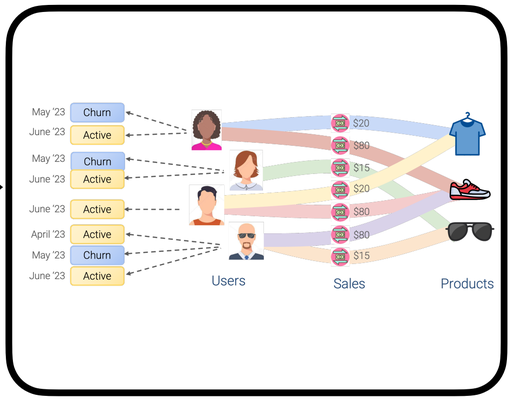
Graph Problem



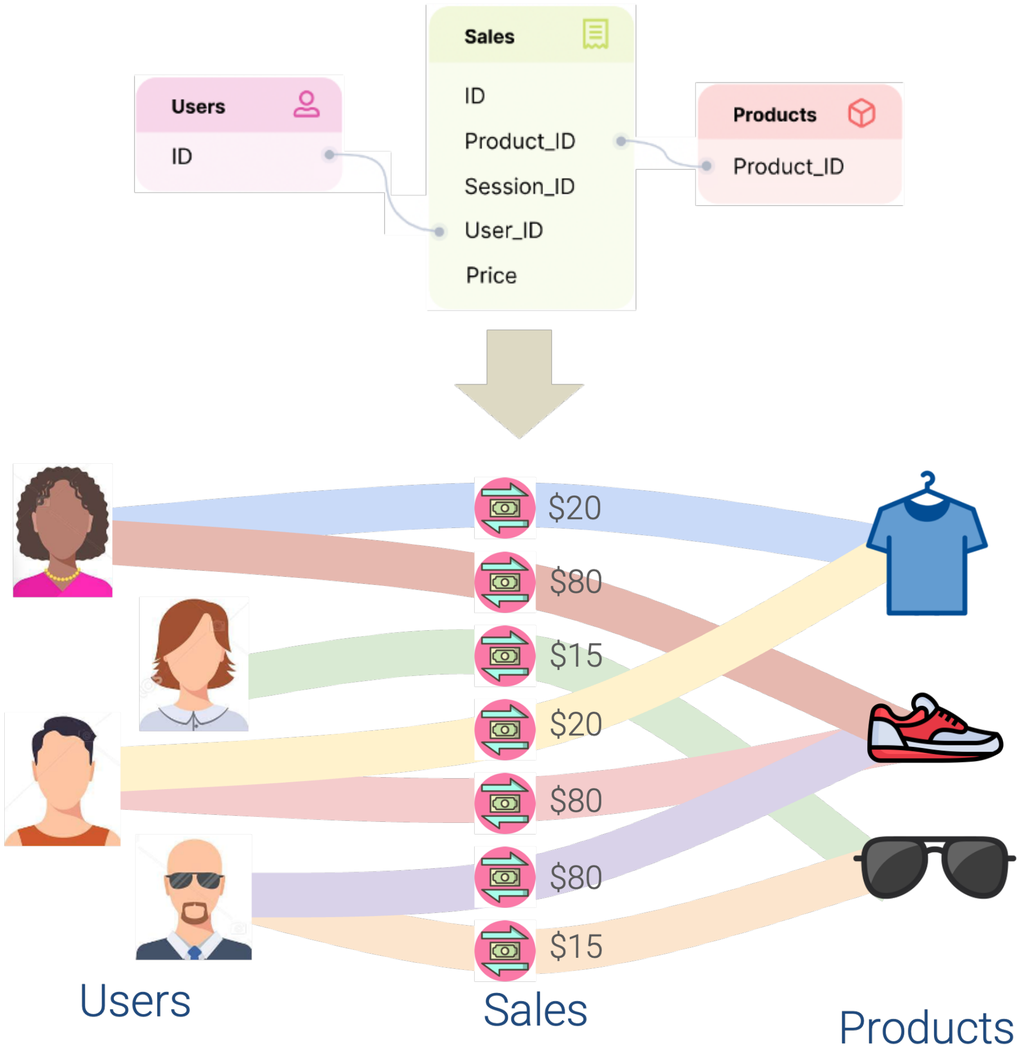
Graph ML



Solutions



Key Insight: Relational Databases as Entity-Level Graphs

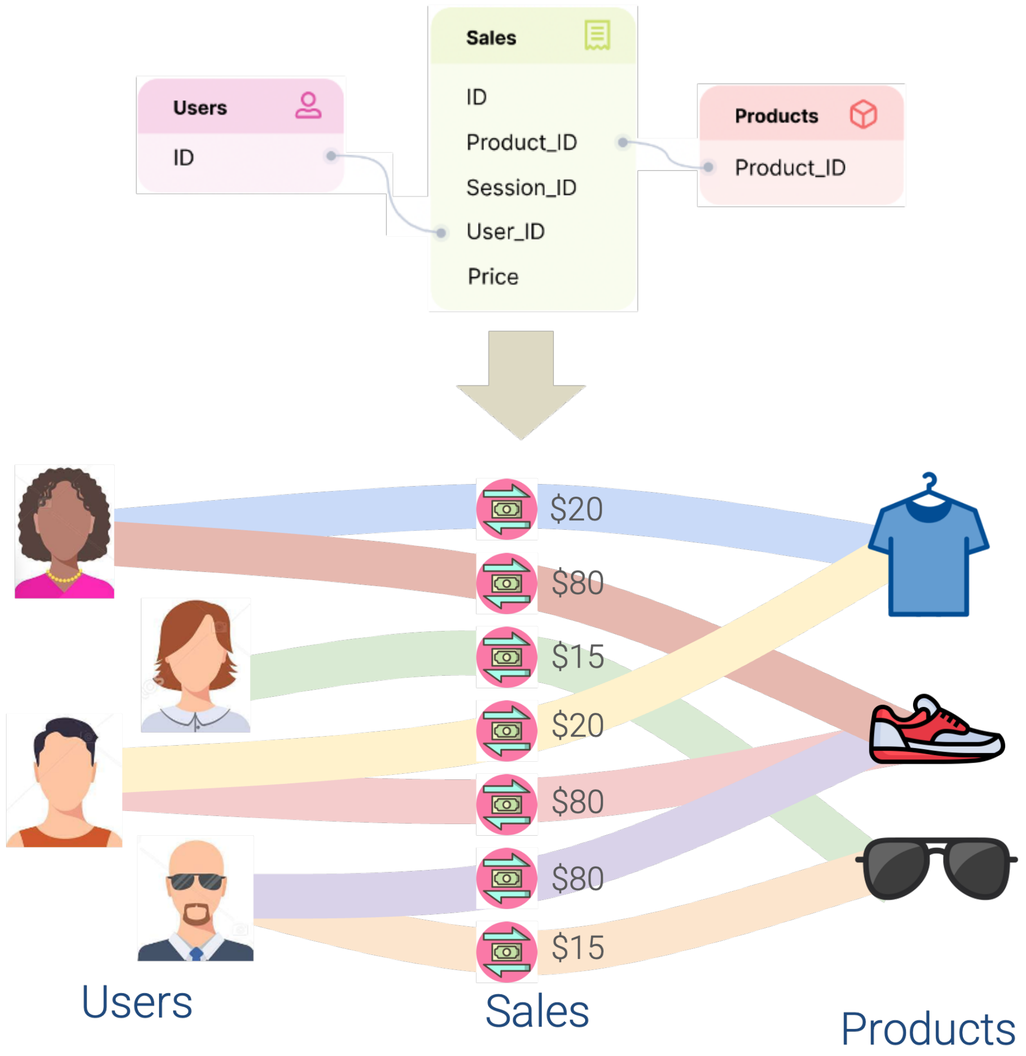


Every Entity (row) is a Node

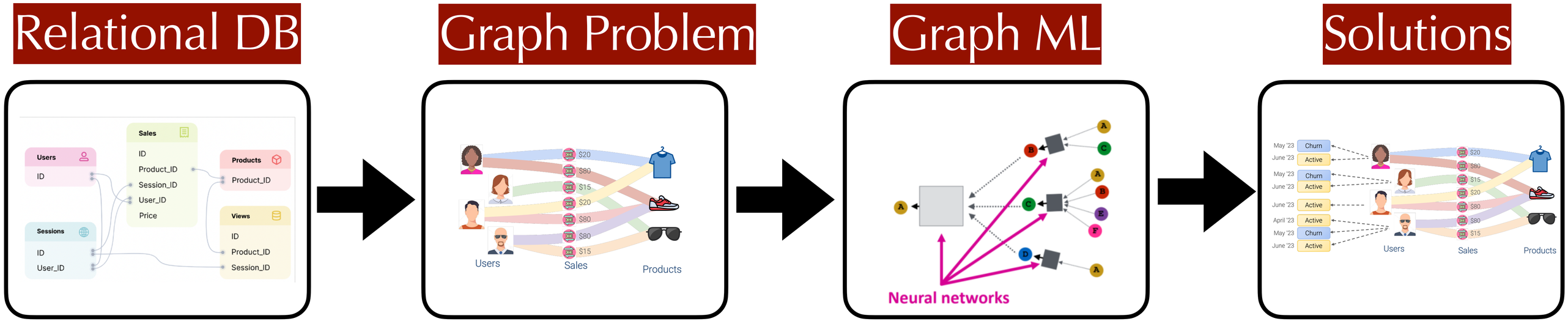
Edges defined by foreign key primary key links

Predictive Tasks are now Graph ML problems

- Node-level
  - Churn
  - Life-time value
- Link-level
  - Product affinity
  - Recommendation
- Graph-level
  - Fraud detection

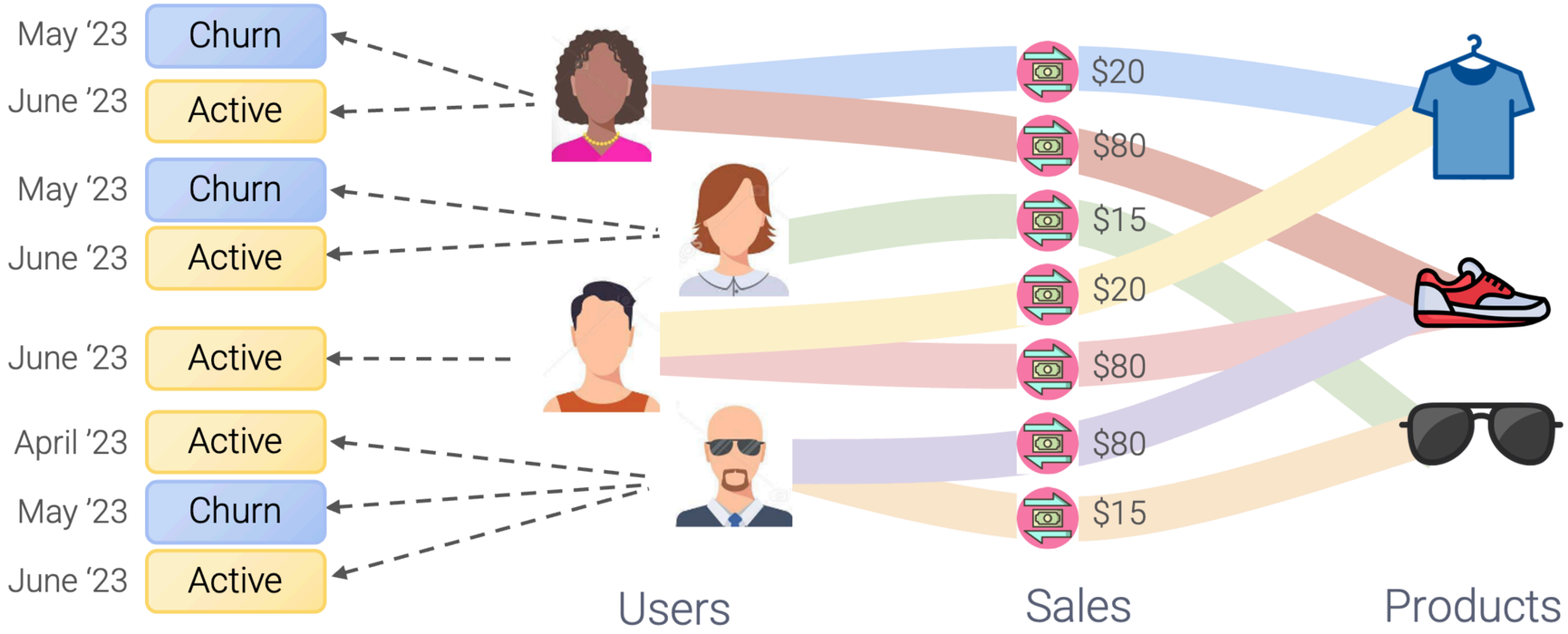


RDL architectures



RDL-GNN [Robinson et al. 2024], RelGNN [Cheng, C.K., Leskovec 2025], Relational Graph Transformer (RelGT) [Dwivedi, Jaladi, Shen, Lopez, C.K., et al. 2025], RelLLM [Wu, et al. 2025]

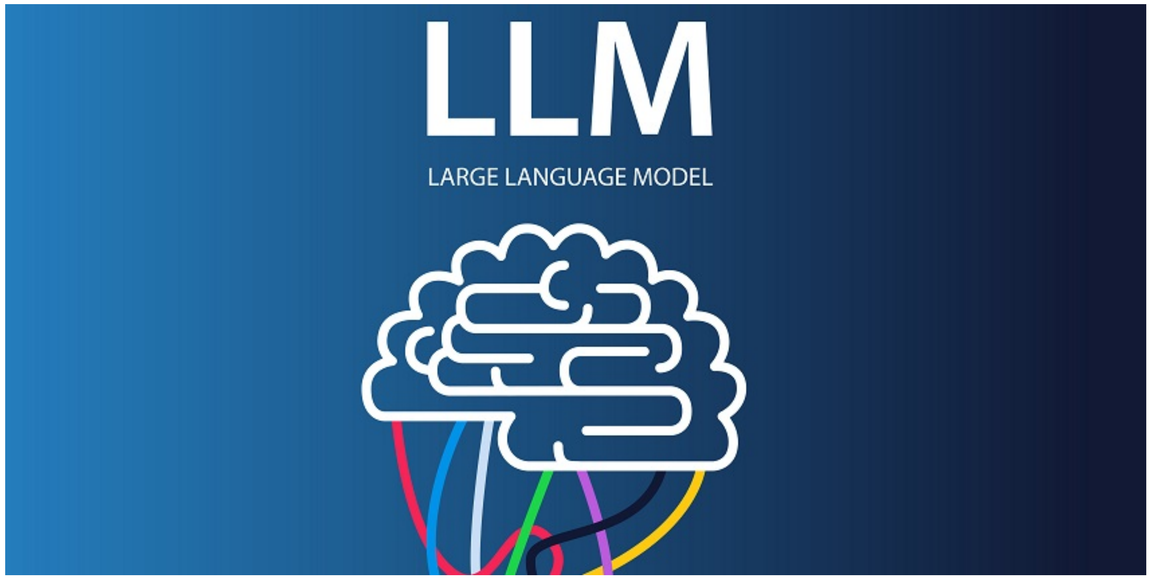
Disadvantages of end-to-end RDL

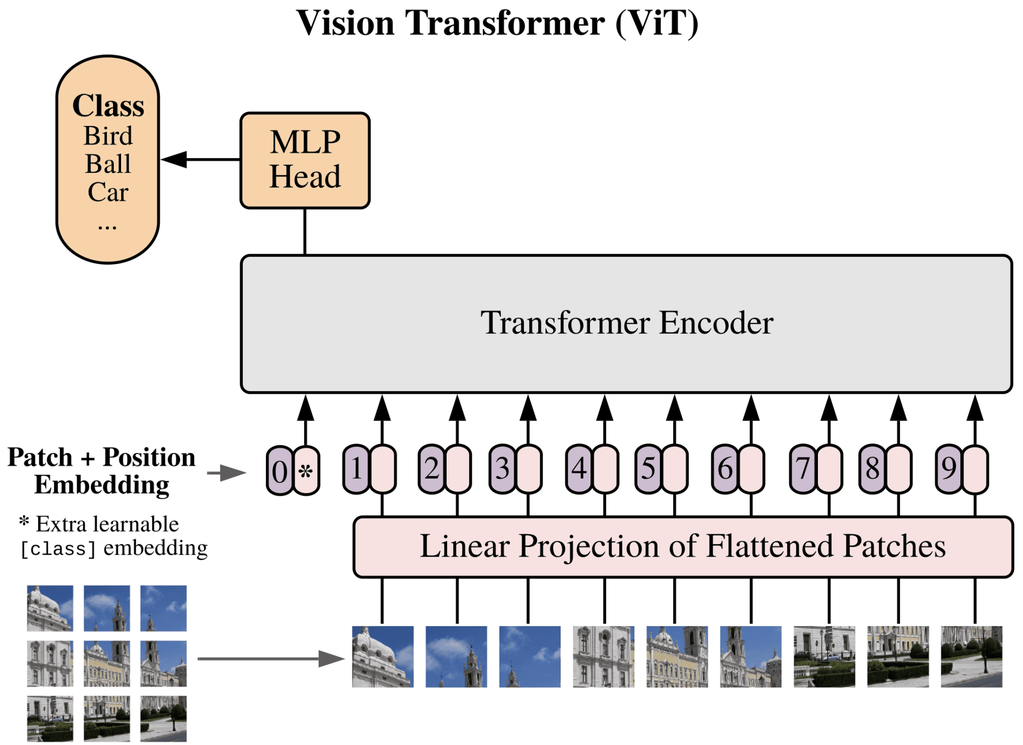


1. Schema- and task- specific
2. Limited transfer: cannot generalize to new databases or tasks.

Foundation Model for Relational Databases

Goal: Bring the success of foundation models (LLMs, ViT) to relational data.





The Promise of the Relational Foundation Model

- Zero-shot forecasting → new Databases, unseen tasks.

- Few-shot / fine-tuning with limited supervision.

Broader Impact: Democratize AI for structured data

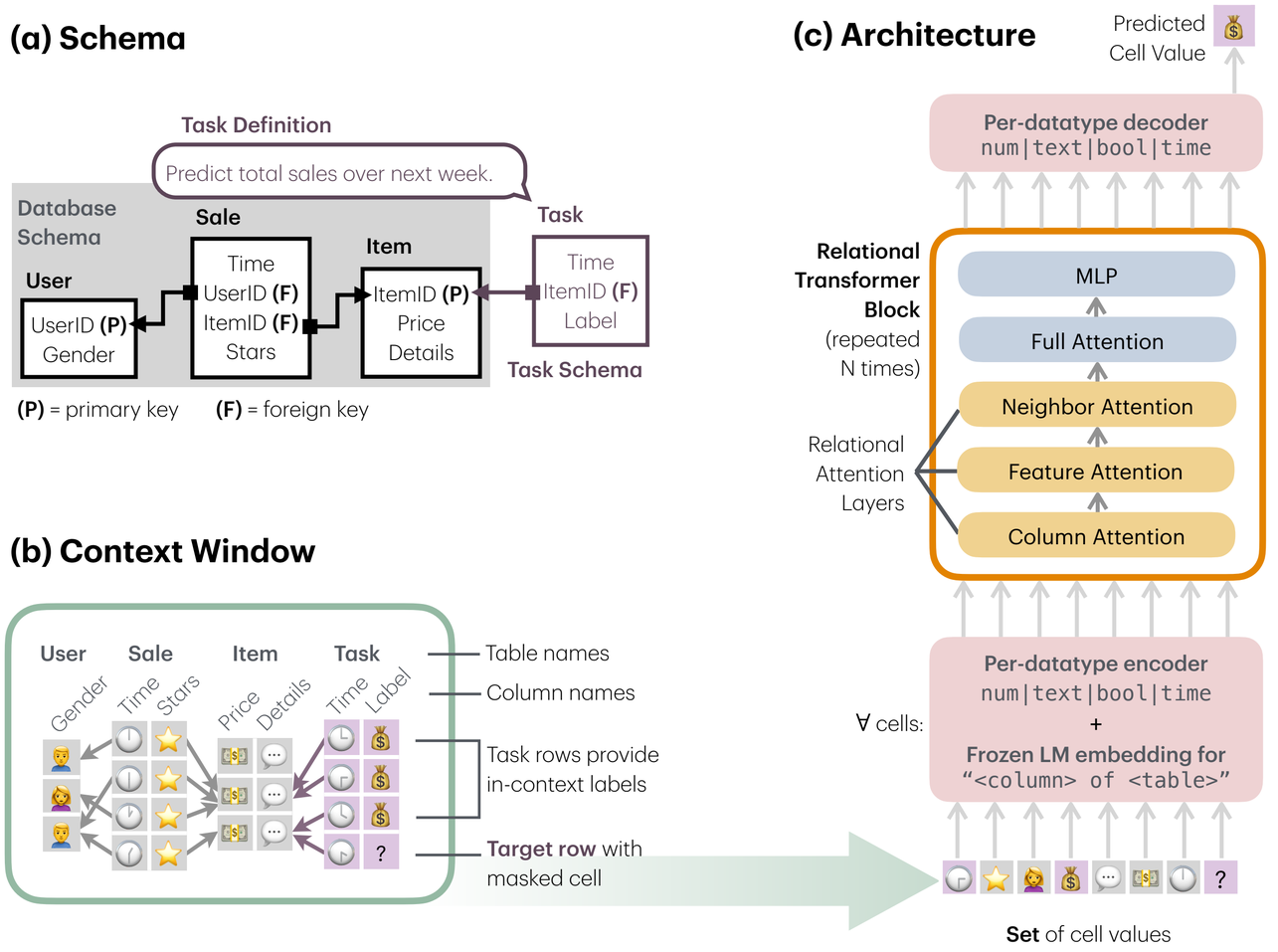
(Non-experts gain accessible predictive tools, Experts gain strong pretrained initialization)

Relational Transformer (RT)

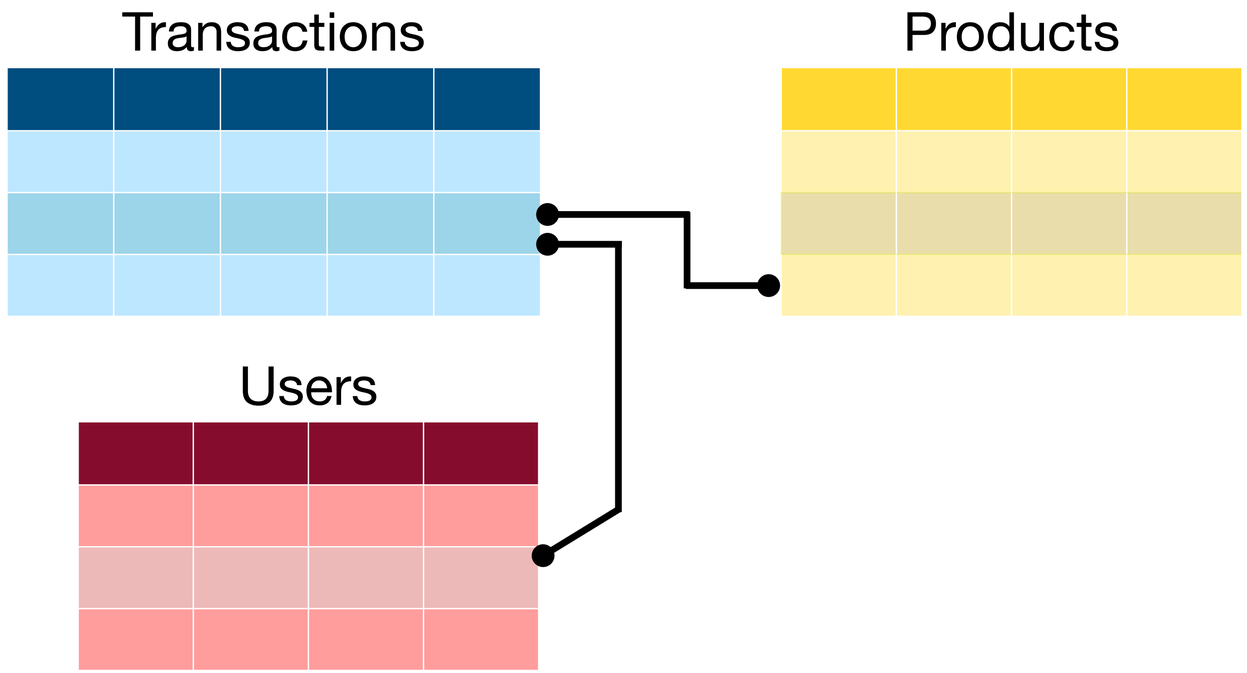
Key idea: Treat each database cell as a token and model its relationships through **Relational Attention**.

Core innovations:

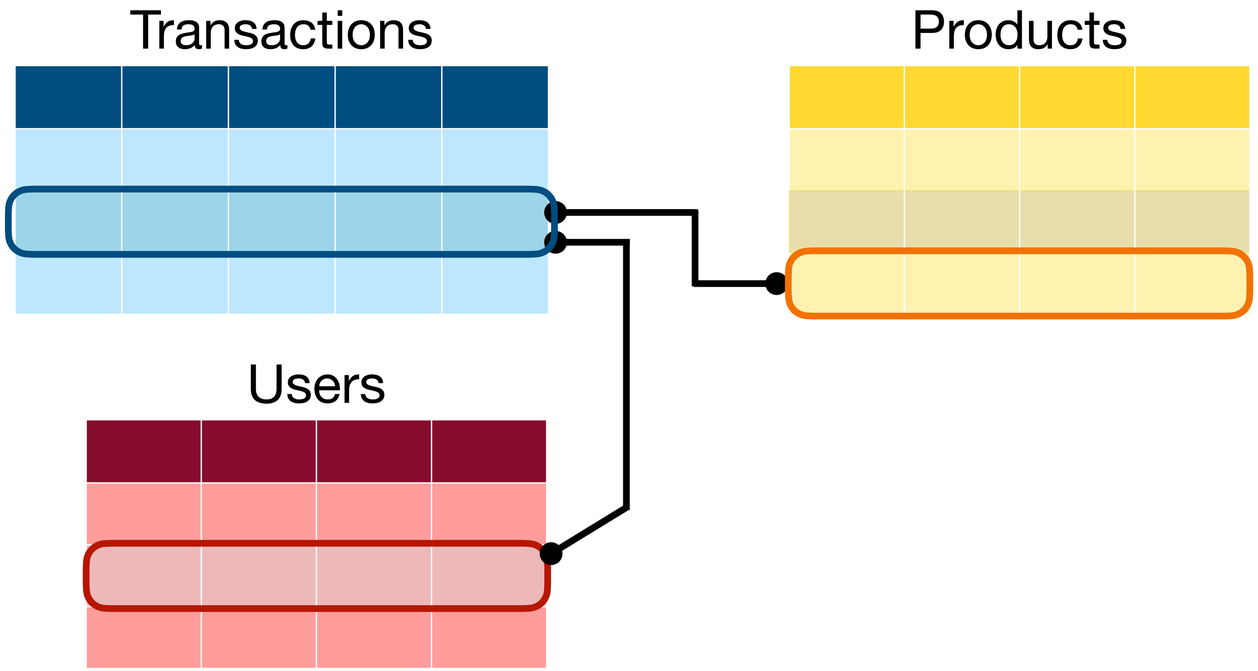
1. Cell-level tokenization
2. Task-table integration
3. Relational Attention



Input Representation in RT



Input Representation in RT



Prior work, e.g., RDL-GNN, RelGNN, RelGT, RelLLM, **tokenize at the row-level**.

Leads to schema-specific architectures

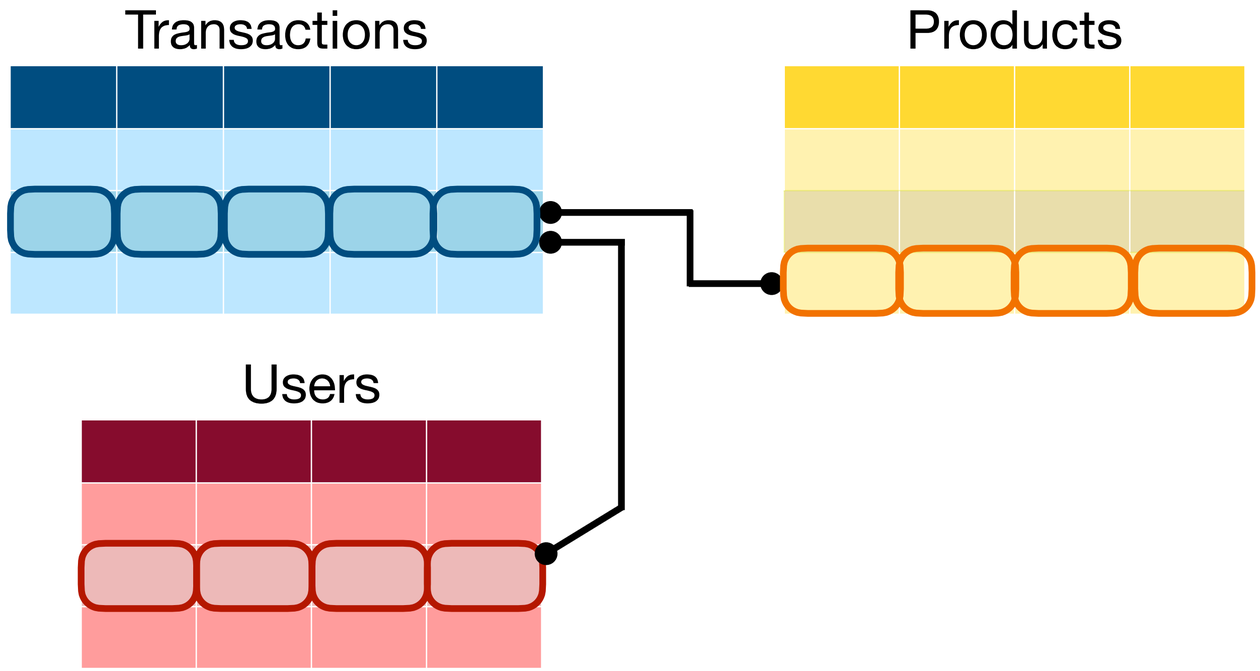
Input Representation in RT

Cell-level tokenization:

Each token = (value, column, table)

Table is a MiniLMv2 description of the table

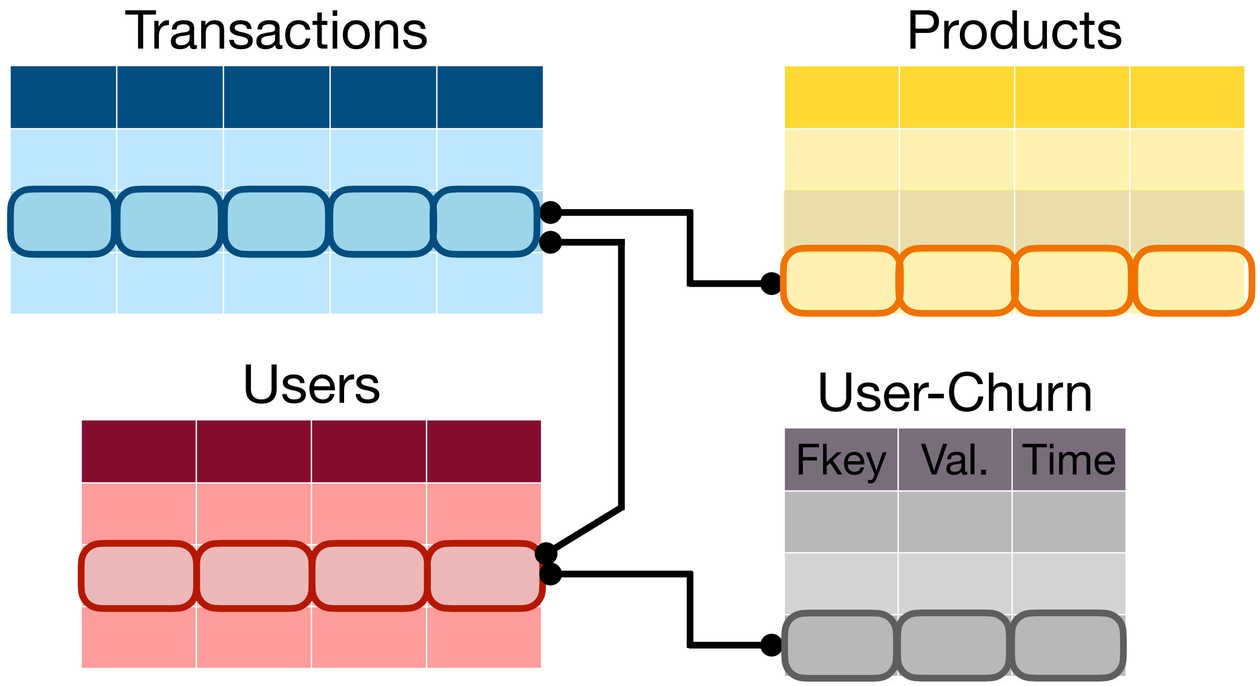
Column is a MiniLMv2 text description of the table



Benefits

- Unifies relational databases of any schema into a consistent input space.
- Leverages cell-to-cell dependencies.

Task Label Integration



Benefits

- Cast any node-level task as masked token prediction (MTP).
- Adds autoregressive labels for task tokens.

Context sampling

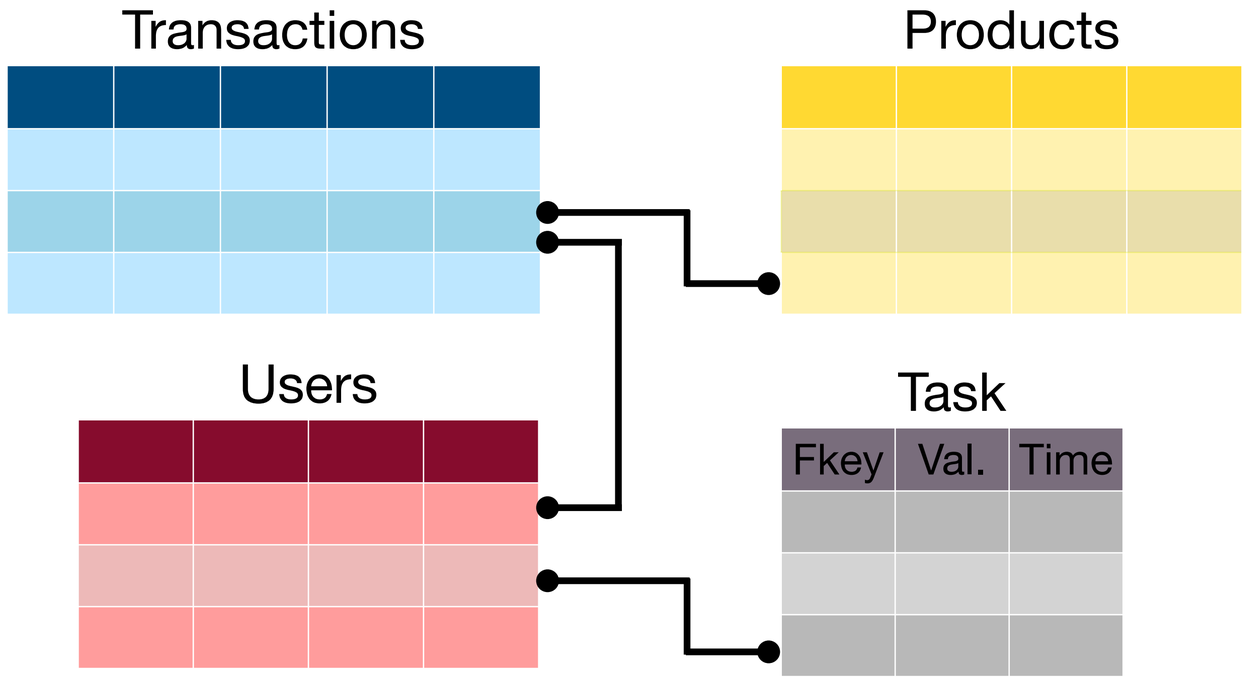
Sampling Rules

1.Sample **seed node** and hide value

2.Always **sample parent** nodes (F →P keys)

3.Run **Breadth-First-Search** on children (P →F keys)

4.Only sample from **previous timestamps**



Context sampling

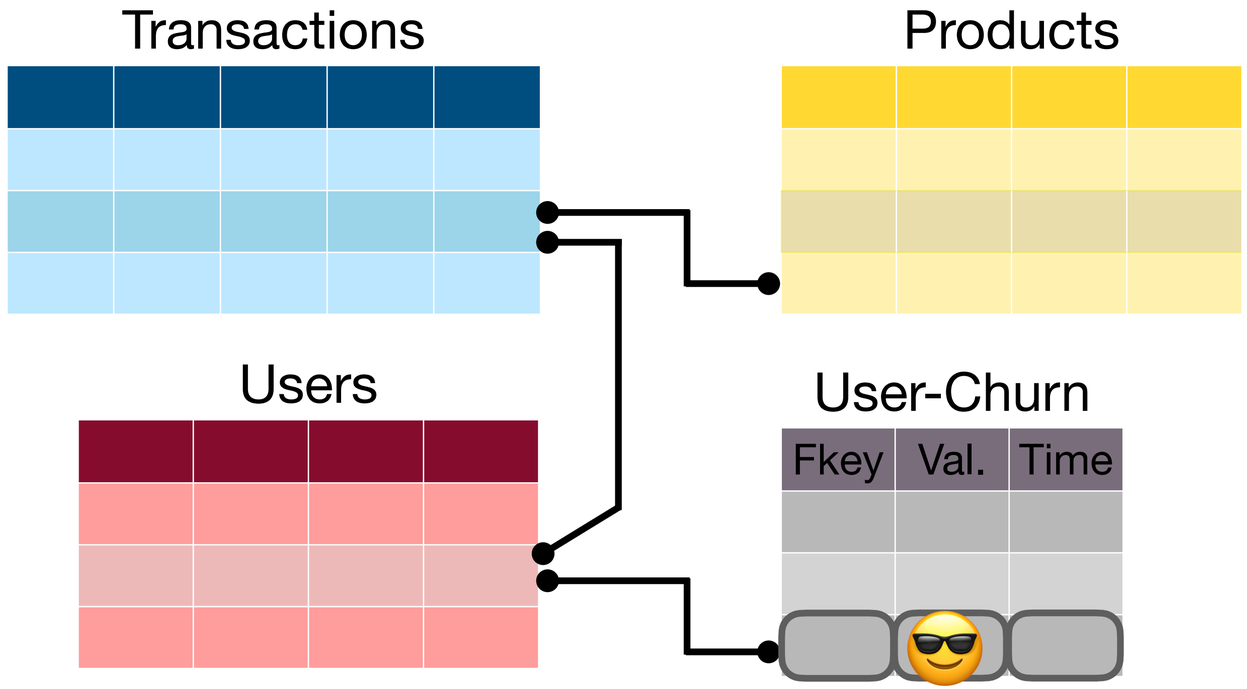
Sampling Rules

1. Sample **seed node** and hide value

2. Always **include parent** nodes (F →P keys)

3. Subsample child rows (P →F keys)

4. Only sample from **previous timestamps**



Context sampling

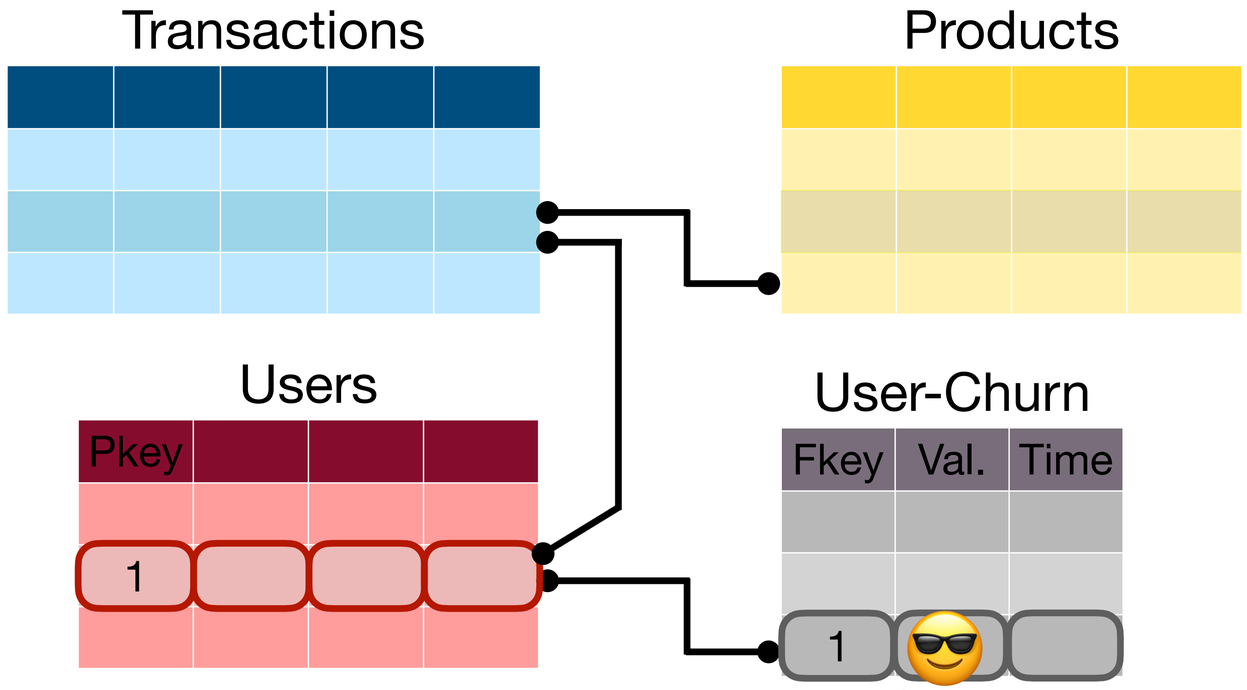
Sampling Rules

1. Sample **seed node** and hide value

2. Always **include parent** nodes (F →P keys)

3. Subsample child rows (P →F keys)

4. Only sample from **previous timestamps**



Context sampling

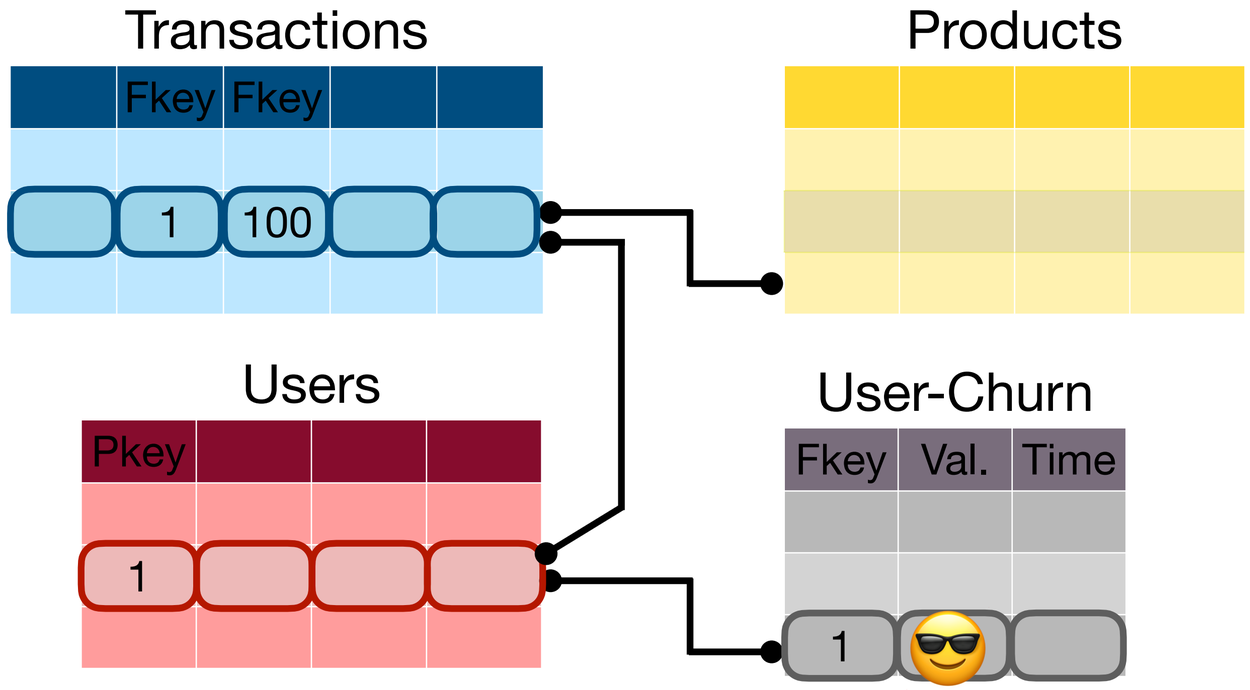
Sampling Rules

1. Sample **seed node** and hide value

2. Always **include parent** nodes (F →P keys)

3. Subsample child rows (P →F keys)

4. Only sample from **previous timestamps**



Context sampling

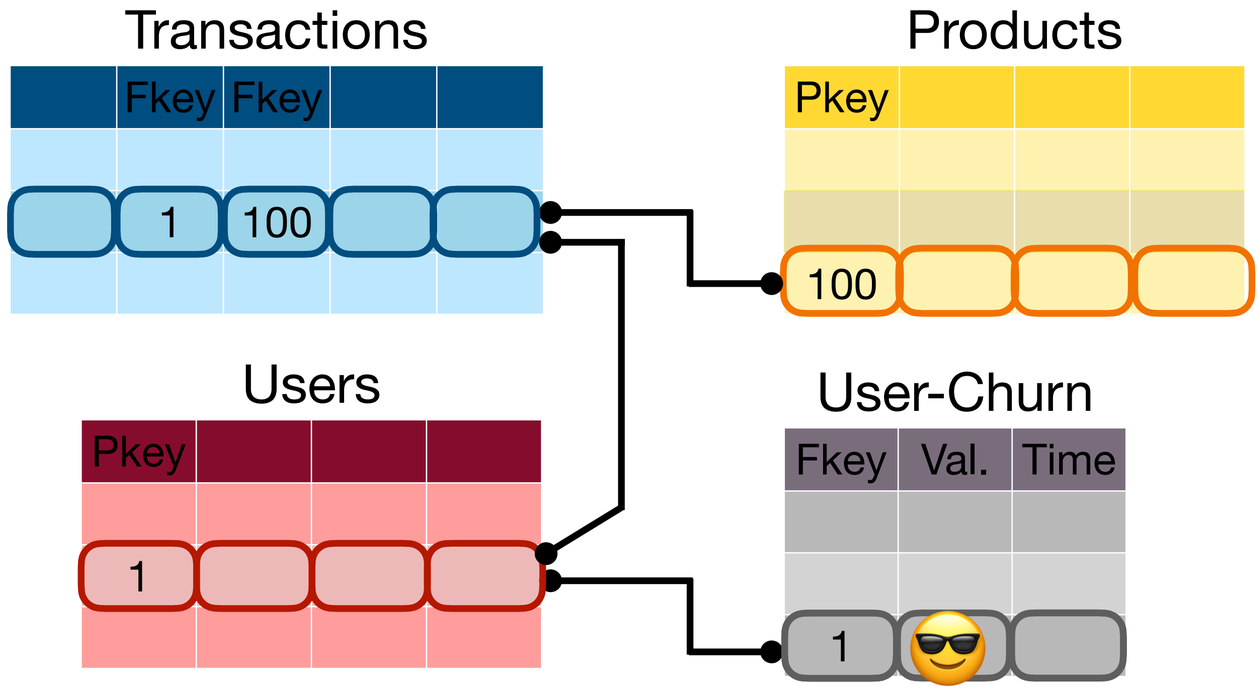
Sampling Rules

1. Sample **seed node** and hide value

2. Always **include parent** nodes (F →P keys)

3. Subsample child rows (P →F keys)

4. Only sample from **previous timestamps**



Context sampling

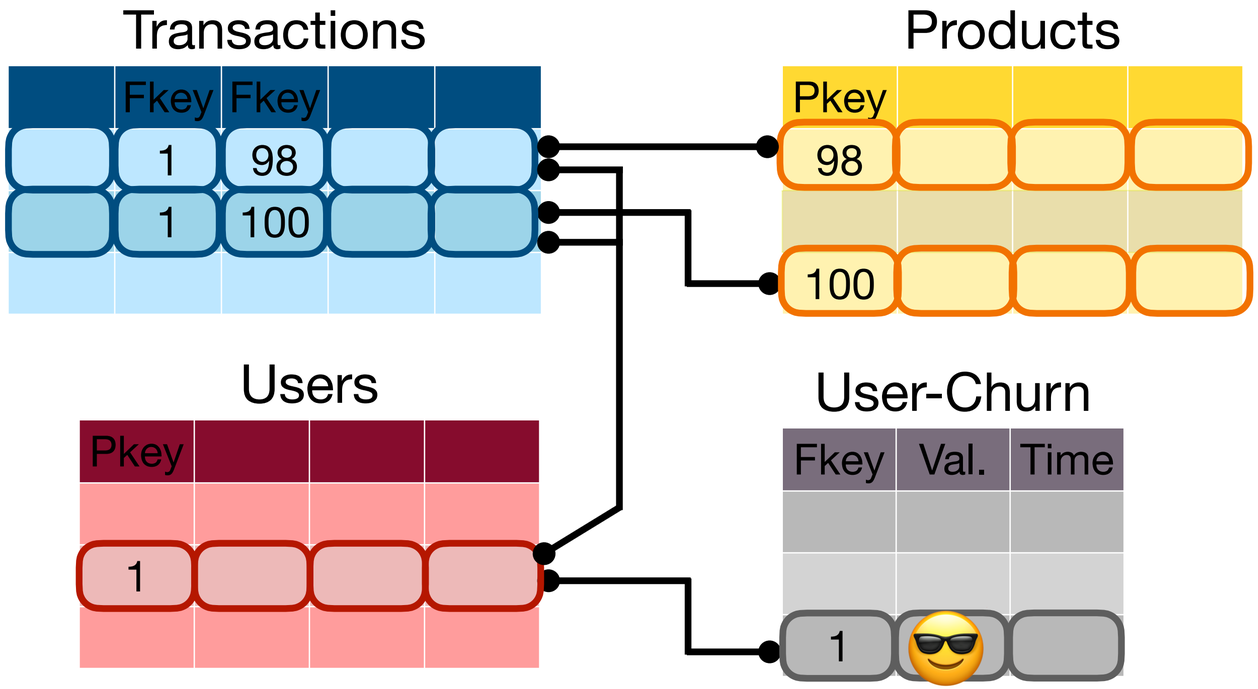
Sampling Rules

1. Sample **seed node** and hide value

2. Always **include parent** nodes (F →P keys)

3. Subsample child rows (P →F keys)

4. Only sample from **previous timestamps**



Context sampling

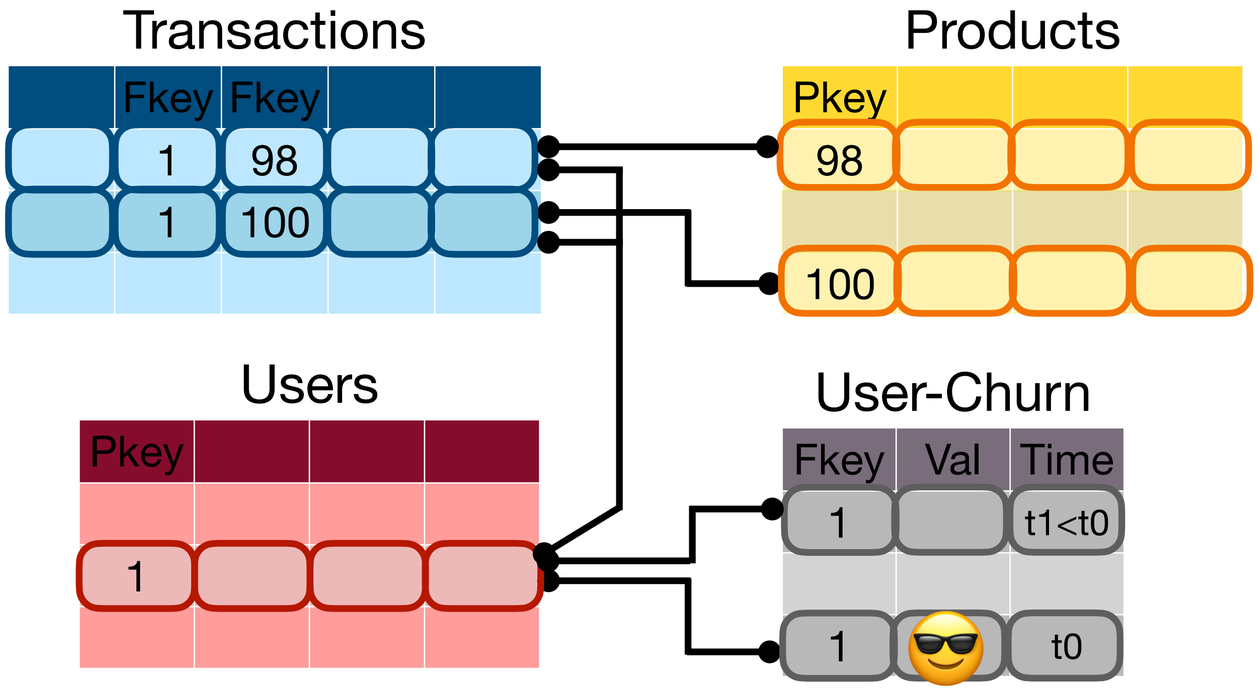
Sampling Rules

1.Sample **seed node** and hide value

2.Always **include parent** nodes (F →P keys)

3.Subsample child rows (P →F keys)

4.Only sample from **previous timestamps**



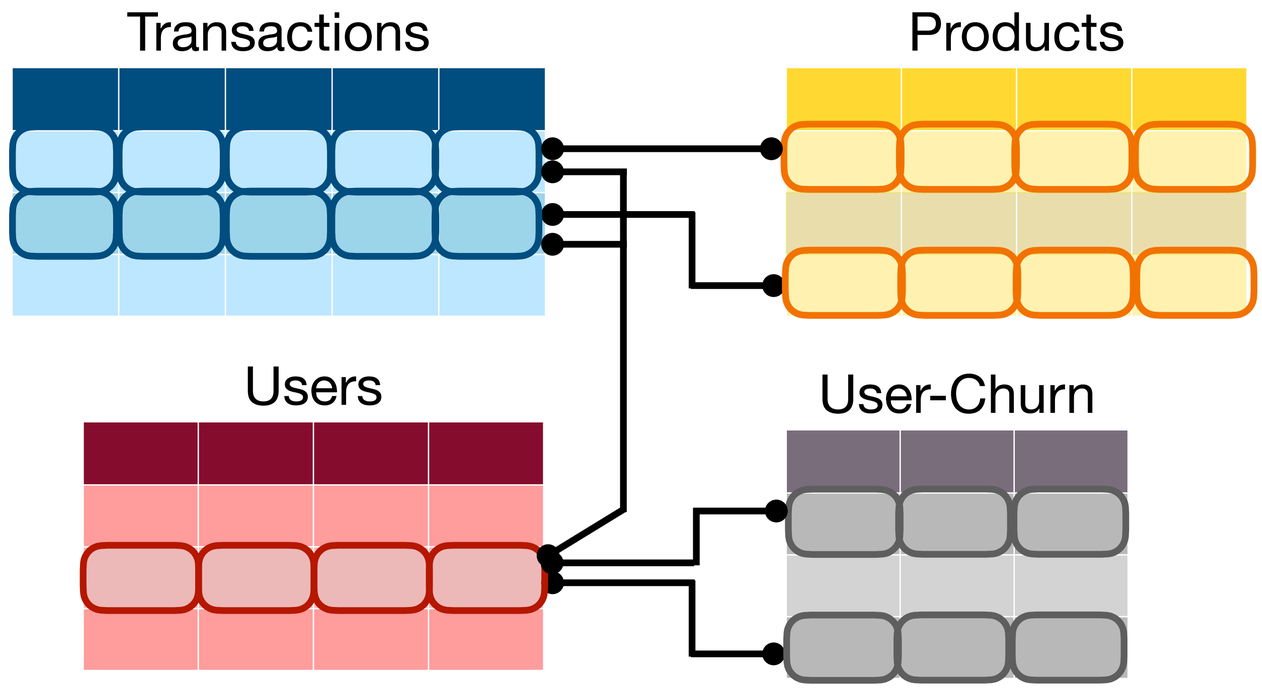
- Avoid temporal leakage

Relational Attention

1. Column Attention

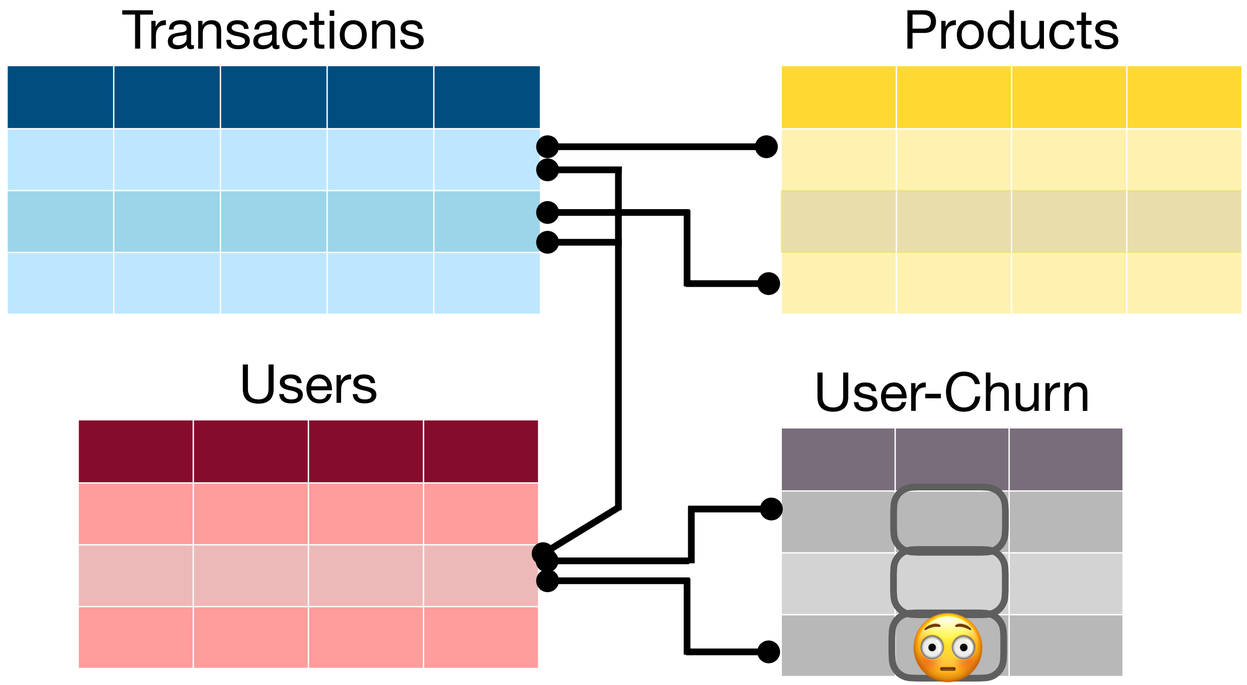
2. Feature Attention

3. Neighbor Attention



Relational Attention

1. Column Attention
2. Feature Attention
3. Neighbor Attention

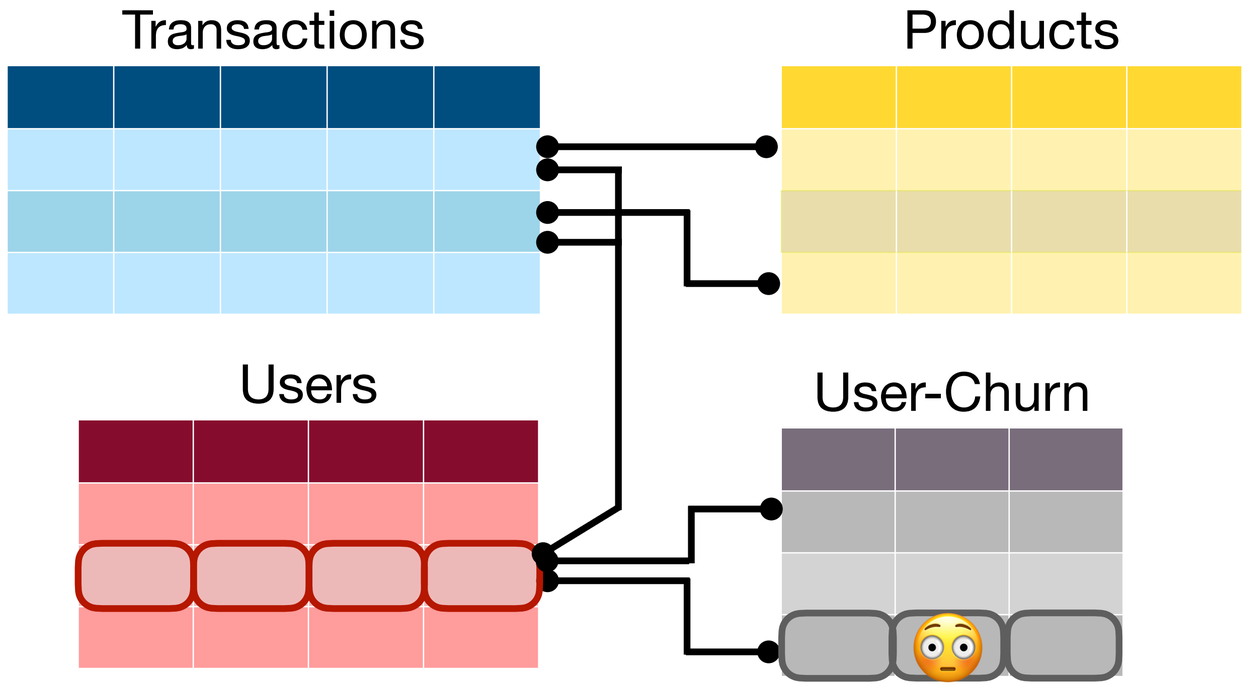


Relational Attention

1.Column Attention

2.Feature Attention

3.Neighbor Attention

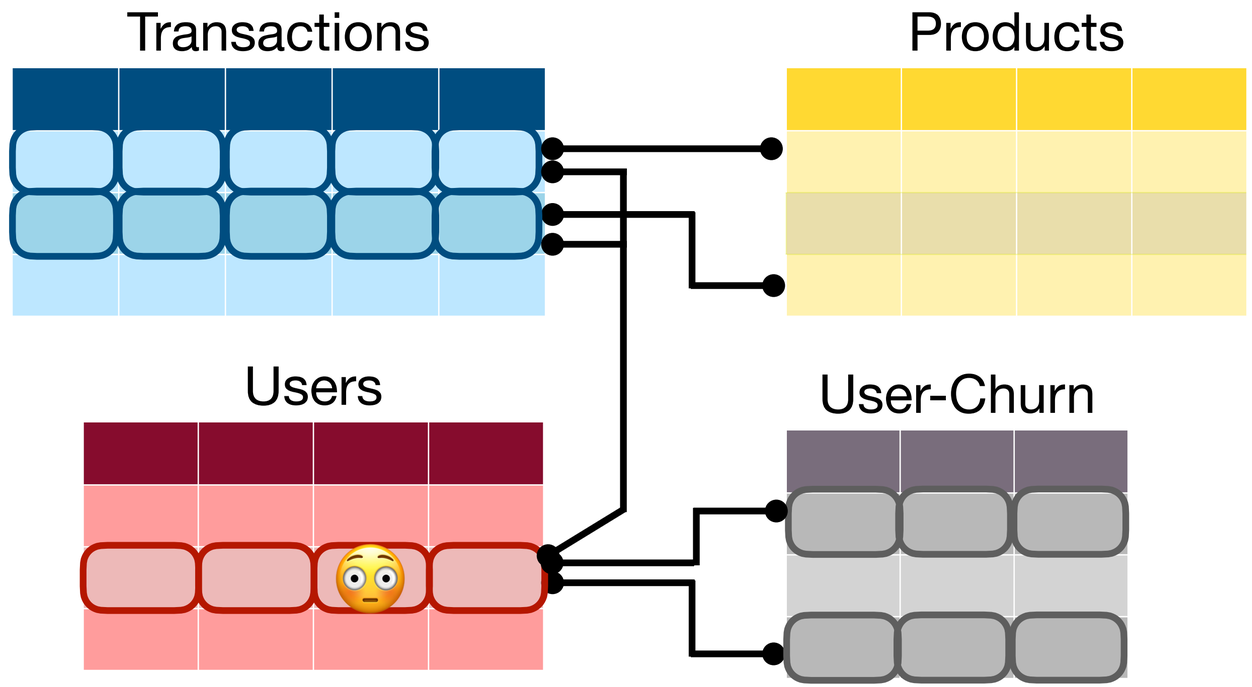


Relational Attention

1.Column Attention

2.Feature Attention

3.Neighbor Attention



Relbench Datasets

7 Diverse Datasets



E-Commerce

- rel-amazon
- rel-avito
- rel-hm



Social

- rel-event
- rel-stack



Sports

- rel-f1



Medical

- rel-trial

Experimental Setup

Tasks: Binary classification and regression

Pretraining: **Leave-one-database-out** pretraining

Test on unseen schema

7 Diverse Datasets



E-Commerce

- rel-amazon
- rel-avito
- rel-hm



Social

- rel-event
- rel-stack



Sports

- rel-f1



Medical

- rel-trial

Experimental Setup

Tasks: Binary classification and regression

Pretraining: **Leave-one-database-out** pretraining

Test on unseen schema

7 Diverse Datasets



E-Commerce

- rel-amazon
- rel-avito
- rel-hm



Social

- rel-event
- rel-stack



Sports

- rel-f1



Medical

- rel-trial

Baselines

1. Griffin [Wang et al. 2025]: Use a universal feature encoder followed by a GNN.
2. Gemma LLMs [Wydmuch]: Prompt Gemma3 with text-serialized database subgraph.
3. RDL-GNN [Robinson et al. 2024]: Schema specific GNN baseline.

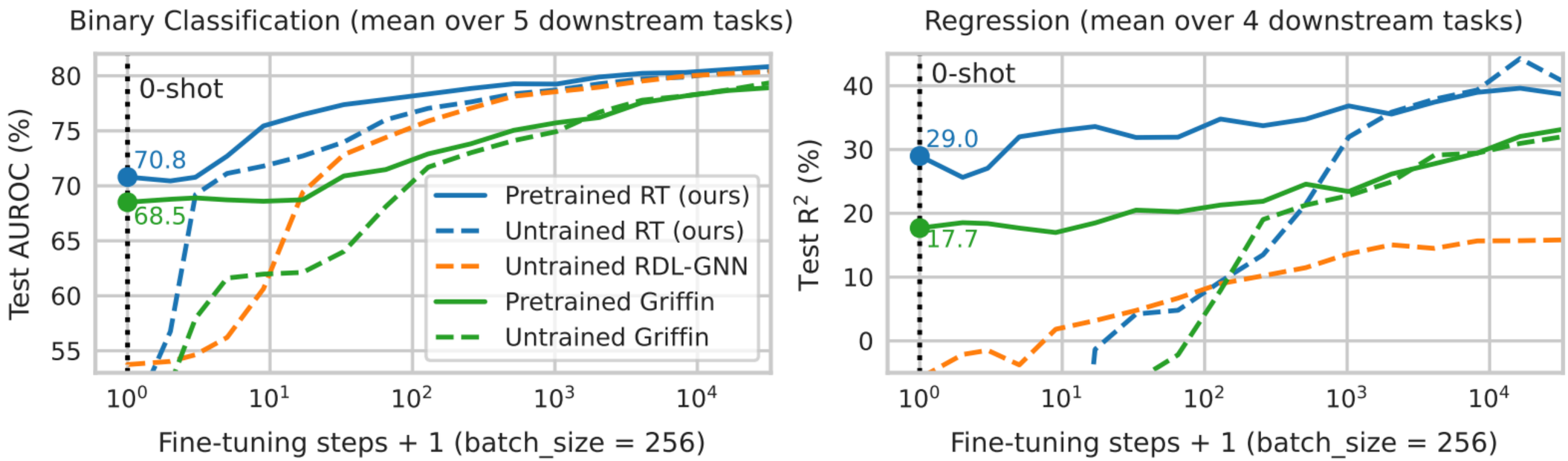
RT Pretraining

- Backbone: 12 layers, 22M parameters
- Pre-training task: Masked Token Prediction on RelBench
- Training duration: 50k steps 8xA100
- Linear decoder
- Objective: Huber for regression, BCE for classification

Results

Finetuning Tests

Test-set learning curves up to 32k fine-tuning steps



Zero-shot Transfer Results

Target dataset ∈ pretraining? →		Maybe			No			Yes	
Dataset ↓	Task ↓	Gemma	Gemma	Gemma	Entity Mean	Griffin	RT (ours)	Griffin	RT (ours)
	Parameter count →	4B	12B	27B	0	22M	22M	22M	22M
rel-amazon	item-churn	62.1	55.0	42.1	73.0	69.0	70.2	71.9	**74.3**
rel-amazon	user-churn	58.1	54.7	50.5	64.4	62.3	63.9	64.1	**65.2**
rel-avito	user-clicks	54.5	59.5	59.8	44.7	45.9	59.5	45.9	**60.8**
rel-avito	user-visits	60.1	57.9	**62.7**	60.7	60.7	61.8	62.2	62.6
rel-f1	driver-dnf	56.2	54.6	75.8	75.4	57.7	**82.0**	57.7	**82.0**
rel-f1	driver-top3	84.6	90.5	**91.4**	85.0	82.5	89.1	81.8	89.3
rel-hm	user-churn	59.8	47.1	48.7	**64.4**	60.2	62.8	60.4	63.1
rel-stack	user-badge	79.1	79.8	80.0	66.2	73.5	80.0	82.3	**83.6**
rel-stack	user-engage	65.9	67.8	78.0	83.5	77.5	77.1	**89.4**	87.8
rel-trial	study-out	52.6	57.4	57.2	50.0	51.0	54.5	57.2	**60.1**
	Mean AUROC →	63.3	62.4	64.6	66.7	64.0	70.1	67.3	**72.9**

Strong **zero-shot regression** ability too

RT is the only neural method to outperform EntityMean (non-neural)

Table 2: Zero-shot test R^2 (%) for 8 regression tasks. Higher is better. Global mean baseline is 0.0. Setup is same as Table 1. LLM baselines are poor (App. G.2).

Target dataset ∈ pretraining? →		No			Yes	
Dataset ↓	Task ↓	Entity Mean	Griffin	RT (ours)	Griffin	RT (ours)
rel-amazon	item-ltv	**54.2**	20.1	33.2	20.1	35.4
rel-amazon	user-ltv	19.9	20.6	36.4	24.4	**39.7**
rel-avito	ad-ctr	3.4	2.4	4.5	2.4	**7.7**
rel-f1	driver-pos	38.2	−0.7	54.7	4.6	**58.4**
rel-hm	item-sales	1.8	2.7	14.0	2.5	**30.4**
rel-stack	post-votes	**43.7**	27.4	32.4	27.1	32.7
rel-trial	site-succ	−6.4	1.4	**5.2**	2.6	3.5
rel-trial	study-adv	−0.5	−2.5	2.1	−2.5	**3.4**
	Mean R^2 →	19.3	8.9	22.8	10.1	**26.4**

RT has **1000x** fewer params

and needs **100x** smaller context

to give **13%** better zero-shot AUROC

than Gemma3-27B

Discussions

- RT as a **powerful model** capable of robust zero-shot transfer in relational settings, while enabling rapid fine-tuning when supervision is available.

- Strong **empirical evidence** that relational databases, despite spanning diverse applications, share transferable patterns.

- A step toward foundation models for structured enterprise data.

Limitations

- No link prediction / recommendation tasks.
- Is trained with limited data.
- Potential improvements in cell encoders & positional biases.