A Pre-training Framework for Relational Data with Information-theoretic Principles

Quang Truong, Zhikai Chen, Mingxuan Ju, Tong Zhao, Neil Shah, Jiliang Tang

NEURAL INFORMATION PROCESSING SYSTEMS

The Problem of Pre-training on Relational Databases (RDBs): Task Heterogeneity

A single RDB supports many possible downstream tasks (e.g., predicting customer churn, item sales, user lifetime value). Tasks are defined by complex, task-specific factors:

- Relational schema graphs (joins across tables).
- Temporal dependencies (e.g., "next-window" properties).
- SQL-defined label logic (e.g., COUNT, AVG).

Standard SSL methods often fail:

- Methods like Masked Autoencoders (MAE) or Contrastive Learning (CL) learn from input data alone.
- They ignore next-window dynamics, which are critical for predictive tasks.

Task Formalization: A Set Function

How are downstream tasks (labels) generated?

- We observe that any predictive task is entity-centric (e.g., predicting for a specific customer).
- Labels are based on rows in other tables, describing dynamics of the entity.

For the sake of simplicity, assume we only consider the k–hop neighbors with the same entity-type, given an entity v and the timestamp t, its label $y_v^{(t)}$ is the output of a set function l defined over the features of its set of k–hop neighbors $\mathbf{X}_v^{(t+1)}$ in the next timestamp $t+1$:

$$y_v^{(t)} = l(\mathbf{X}_v^{(t+1)}), \quad \mathbf{X}_v^{(t+1)} = \{ f_v(u) \mid u \in \mathcal{N}_k^{(t+1)}(v) \}$$

Task examples:

❑ Task 1: Predict # of reviews for a customer.

- l = COUNT
- $\mathbf{X}_v^{(t+1)}$ = Set of review of the customer in the next window. (customer → review)

❑ Task 2: Predict average price of products reviewed by a customer.

- l = AVERAGE
- $\mathbf{X}_v^{(t+1)}$ = Set of products from a 2-hop join in the next window. (customer → review → product)

Key Insight: All tasks depend on this set of next-window values $\mathbf{X}_v^{(t+1)}$. Therefore, a good pre-training objective should learn to represent this set.

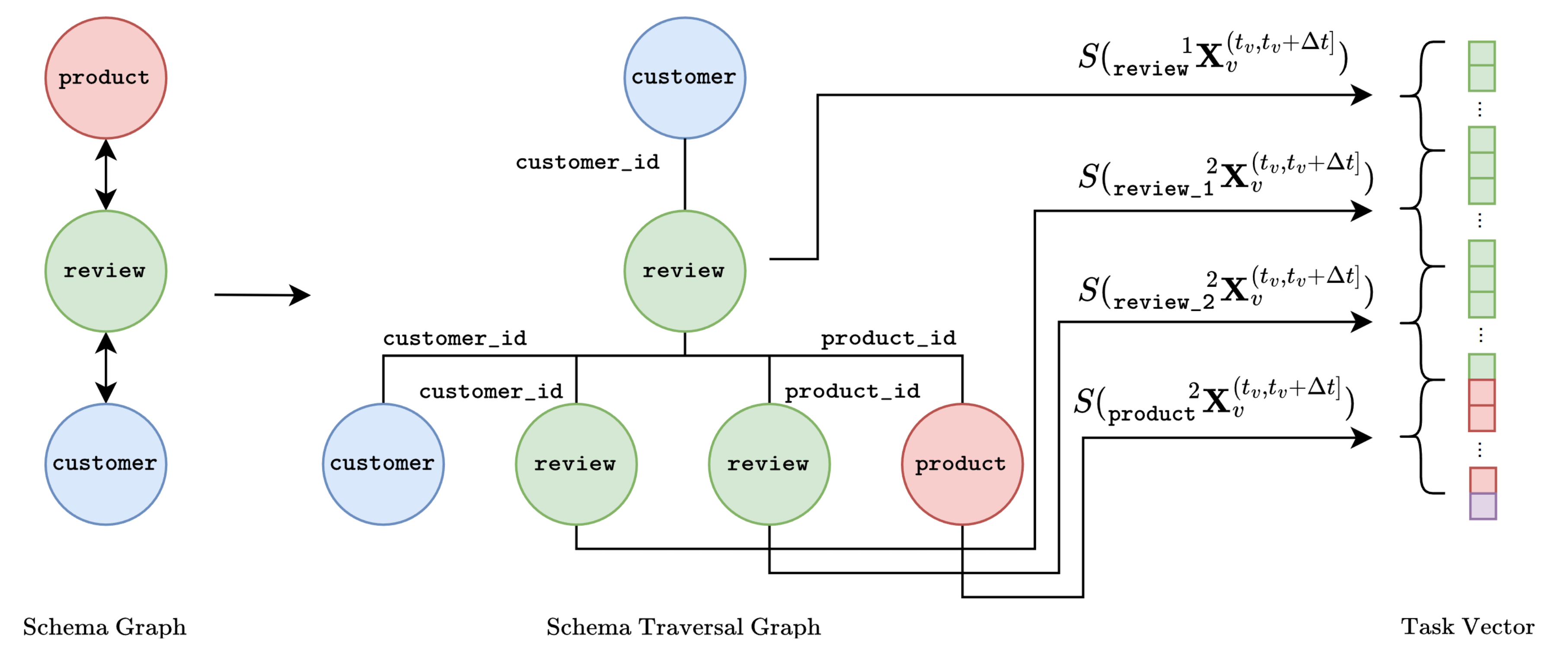
Task Vector Estimation (TVE)

Core Idea: Instead of reconstructing the input (like MAE) or contrasting views (like CL), the model is pre-trained to estimate a "Task Vector".

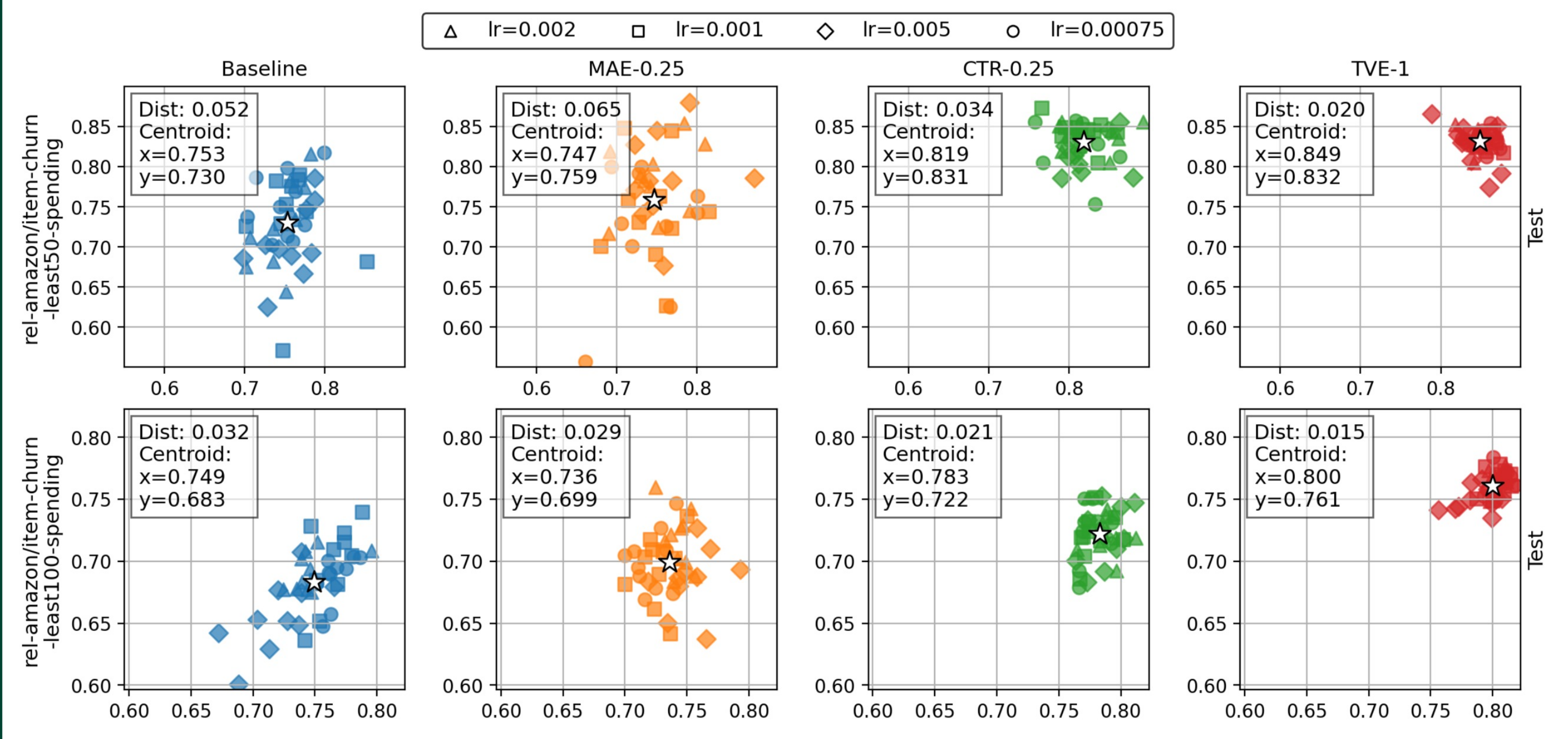
What is the Task Vector?

- It is a surrogate objective constructed to represent the set of all potential downstream tasks.
- It's built by aggregating statistics from the next-window of data.
- It explicitly models the three key factors:
  1. Schema Graphs: By traversing joinable tables.
  2. Temporal Dynamics: By aggregating next-window values.
  3. Task Heterogeneity: By using a rich set of simple SQL aggregations (e.g., COUNT, MEAN, SUM).

Construction of Task Vector

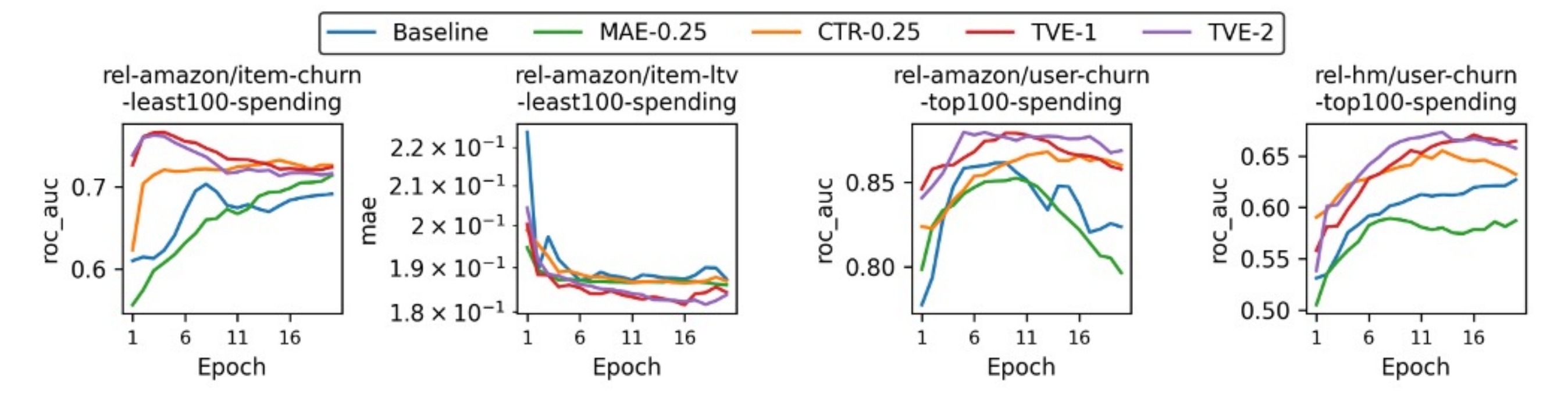

product
review
customer
customer_id
customer_id
product_id
customer_id
product_id
customer
review
review
product
$S({}_{\text{review}}{}^{1}\mathbf{X}_v^{(t_v,t_v+\Delta t]})$
$S({}_{\text{review\_1}}{}^{2}\mathbf{X}_v^{(t_v,t_v+\Delta t]})$
$S({}_{\text{review\_2}}{}^{2}\mathbf{X}_v^{(t_v,t_v+\Delta t]})$
$S({}_{\text{product}}{}^{2}\mathbf{X}_v^{(t_v,t_v+\Delta t]})$
Schema Graph
Schema Traversal Graph
Task Vector

Performance Sensitivity Analysis



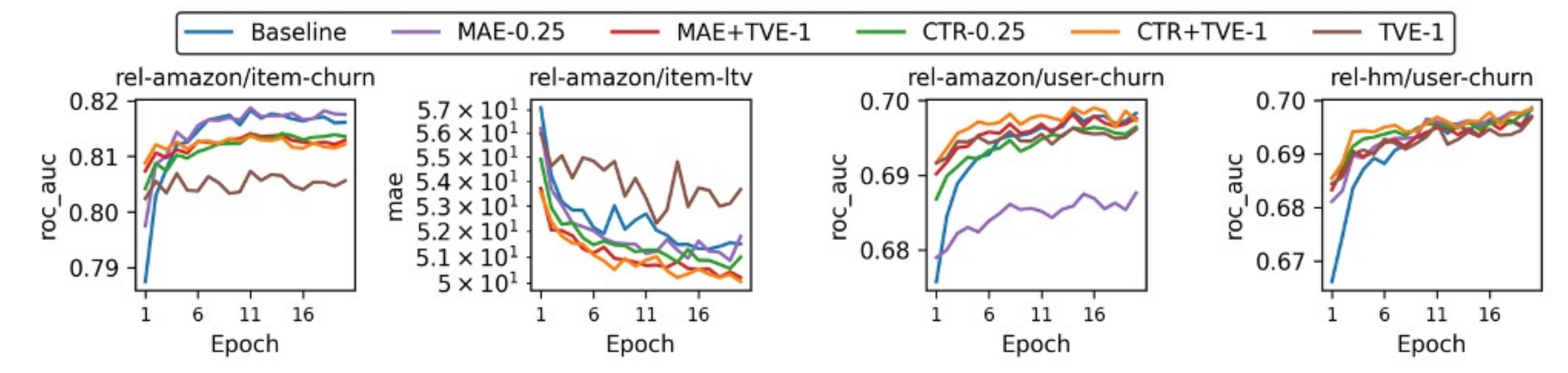
Data-Limited Settings

- TVE consistently and significantly outperforms baselines in low-data regimes.



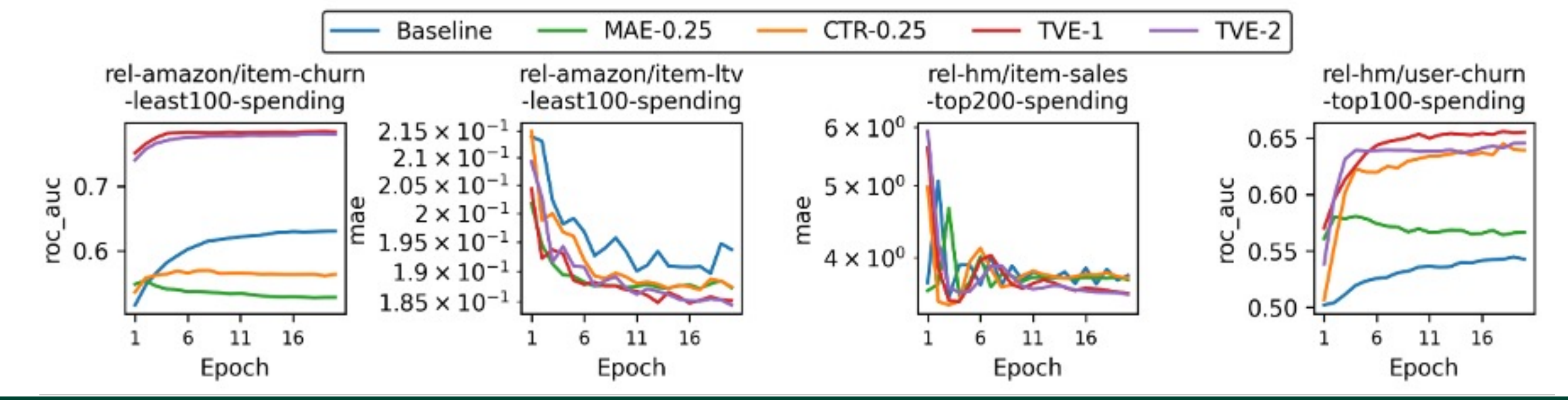
Data-Sufficient Settings

- Gains are marginal when using TVE alone (plentiful fine-tuning data can recover temporal patterns).
- TVE is complementary. A combined loss (e.g., MAE + TVE) outperforms single-objective models and more reliable.



Linear Probing

- By freezing the pre-trained backbone, TVE's representations are shown to be more stable and transferable.



Conclusion

Main Contributions:

- Task Vector Estimation (TVE): A new pre-training framework that explicitly models task heterogeneity, schema, and temporal dynamics.
- Theoretical Justification: An information-theoretic proof showing that task-aware representations (like TVE's) provably retain more downstream-relevant information.

Key Takeaway:

For predictive modeling on relational data, we shouldn't just pre-train on the *input data we have*. We must pre-train using a signal that *represents the future tasks we want to solve*.