AI607: Graph Mining and Social Network Analysis (Fall 2025)

[Term Project] I Saw, I Saved, I... Shopped?

Multi-Behavior Prediction in Online Shopping

Release: September 24, 2025
Progress Report: November 14, 2025, 11:59 PM
Final Report: December 5, 2025, 11:59 PM
Final Presentation: December 8 & 10, 2025

The ultimate goal of this project is to explore how multi-behavior signals can be used to better understand and predict user actions in online shopping. In this project, you will design, implement, and evaluate your own approach for predicting different types of user-item interactions and for ranking items according to users' likely behaviors. Also, you will (a) write a progress report, (b) write a final report, and (c) present your approach. While details of the following steps will be announced later, tentative schedules are as follows:

- Progress Report - November 14, 2025, 11:59 PM
- Final Report - December 5, 2025, 11:59 PM
- Presentation - December 8 & 10, 2025

This is a team project, and each team should consist of *two or three members*. You can find your teammates by all means (e.g., Classum), and one progress report should be submitted *per team*.

Your submission will be evaluated based on

- Presentation (final report & oral presentation) - 40%,
- Novelty of your proposed approach - 20%,
- Validity of your proposed approach - 20%,
- **Accuracy - 20%**.

Note that ***accuracy is not our only concern***. Instead of spending all your time optimizing the accuracy, we recommend spending more time on developing *novel and valid approaches* and making your *presentation clear and complete*.

1 Problem: Multi-Behavior Prediction in Online Shopping

(a) Coupang (b) Amazon (c) AliExpress

Figure 1: Examples of popular online shopping platforms.

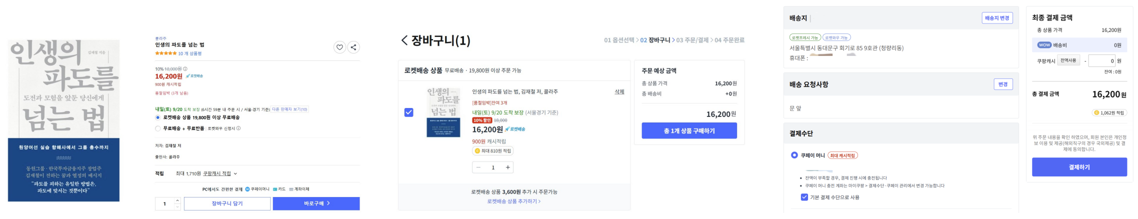
(a) Viewing an item (b) Saving an item (to the cart) (c) Purchasing an item

Figure 2: Illustration of different types of user behaviors in online shopping.

Online shopping platforms, e.g., Coupang, Amazon, and AliExpress (see Figure 1), record a wide variety of user interactions, ranging from simply viewing an item to adding it to the cart, and eventually purchasing it (see Figure 2). These actions represent different levels of user intent and together form a rich sequence of behavioral signals. Leveraging such multi-behavior data is critical for building effective recommender systems, as it enables platforms to anticipate not only what users are likely to buy, but also what they are likely to explore next.

From the perspective of data mining and machine learning, this problem goes beyond standard recommendation tasks: instead of focusing on a single behavior (e.g., predicting purchases), the goal is to jointly model and predict multiple types of user–item interactions. This requires handling imbalanced behavior distributions, capturing dependencies between different actions, and designing models that can generalize from sparse but informative signals (such as purchases) to abundant but weaker signals (such as views).

In this project, students will explore the challenge of *multi-behavior prediction* in online shopping. The ultimate goal is to design, implement, and evaluate methods that can (i) predict the type of interaction between a given user and item, and (ii) generate personalized ranked lists of items that a user is most likely to view, based on their observed behaviors.

2 Data Description

The dataset used in this project is derived from an online shopping platform and contains multi-behavior interactions between users and items. It is specifically designed for the two prediction tasks of this project.

2.1 Basic Statistics

- **Users:** 21,716 unique users

- **Items:** 7,977 unique items
- **Indexing:** Both user IDs and item IDs are represented by consecutive integers *starting from 1*.
- **Interaction types (labels):**
  - 0: *None* (No interaction)
  - 1: *Viewed* (clicked)
  - 2: Viewed and *saved* (added to cart)
  - 3: Viewed, saved, and *purchased*

2.2 Hierarchical Nature of Labels

The labels are hierarchical, meaning that a larger label number indicates a strictly more extensive interaction that subsumes all smaller labels. Specifically:

- Label 2 (*saved*) includes Label 1 (*viewed*) as a sub-interaction.
- Label 3 (*purchased*) includes both Label 1 (*viewed*) and Label 2 (*saved*) as sub-interactions.

This reflects the natural progression of user intent in online shopping: A user views an item before saving it, and saves it before purchasing it.

2.3 Task Partitioning

- Half of the users (10,858) are assigned to **Task 1** (interaction-level prediction).
- The other half (10,858) are assigned to **Task 2** (user-level prediction).
- The two sets of users are disjoint: a user belongs to exactly one task.
- The set of items is shared between the two tasks.

2.4 Training Coverage

For each task:

- All users *in that task* (i.e., half of the total users) appear at least once in the training set.
- All items appear at least once in the training set.

2.5 Cross-Task Usage

As mentioned above, the two tasks use disjoint user sets. However, you are allowed to use the data from one task to help with the other task.

3 Task Description

This project consists of two prediction tasks, each focusing on a different aspect of multi-behavior modeling in online shopping. The two tasks use disjoint sets of users but share the same pool of items.

3.1 Task 1: Interaction-Level Prediction

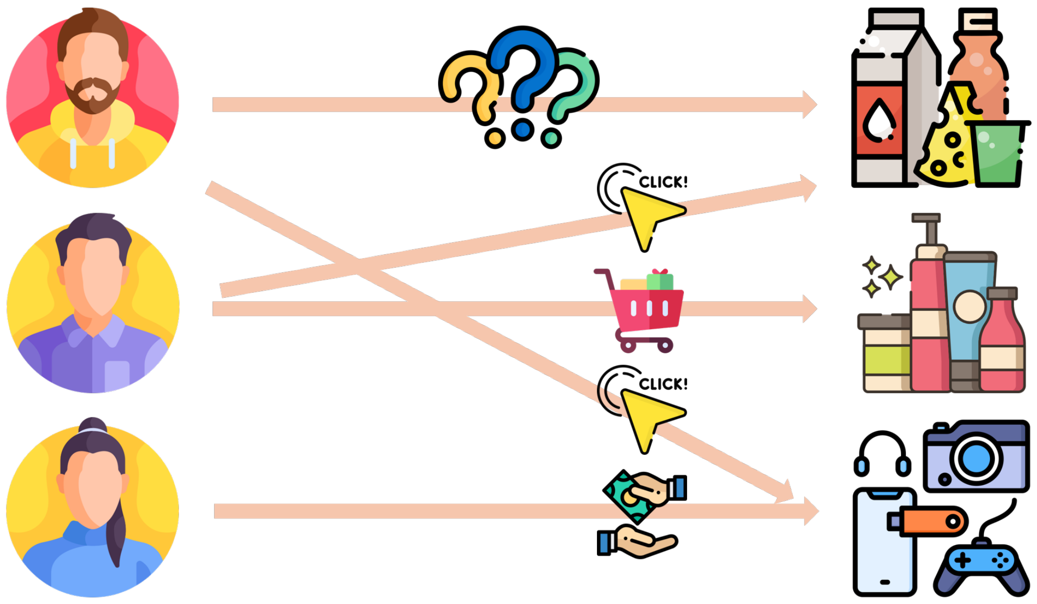

Figure 3: Illustration of Task 1. We are provided with *observed interactions* between users and items (e.g., viewing, saving, or purchasing). Given these observations, the goal is to predict the correct interaction type for unobserved user–item pairs from $\{0, 1, 2, 3\}$, representing *none* (0), *view* (1), *save* (2), and *purchase* (3).

3.1.1 Goal

You are provided with a set of *observed interactions* between users and items (e.g., viewing, saving, or purchasing). Based on these observations, the goal is: given a new (i.e., unobserved) user–item pair, predict its interaction type (label) from the set $\{0, 1, 2, 3\}$, representing *none* (0), *view* (1), *save* (2), and *purchase* (3). See Figure 3 for an illustration of task 1.

3.1.2 Files

- `task1_train.tsv`: Training set with 972,601 lines. Each line contains (1) user ID, (2) item ID, and (3) interaction type (label). Only positive labels (1, 2, or 3) appear in the training set. Below are three example rows from the file:

```
6234    3617    1   # user 6234 viewed (label 1) item 3617
9239    203     2   # user 9239 saved (label 2) item 203
3696    1127    3   # user 3696 purchased (label 3) item 1127
```

- `task1_val_queries.tsv`: Validation queries with 243,148 user–item pairs. Below are the first three rows from the file:

```
4630    327  # How did user 4630 interact with item 327?
10253   7268 # How did user 10253 interact with item 7268?
6451    7079 # How did user 6451 interact with item 7079?
```

- task1_val_answers.tsv: Validation answers with the same number of lines, but now each line contains (1) user ID, (2) item ID, and (3) interaction type (label). All possible labels (0, 1, 2, and 3) appear in the answers. Below are the first three rows from the file, which answer the three example lines above in task1_val_queries.tsv:

```
4630    327     2 # user 4630 saved (label 2) item 327
10253   7268    2 # user 10253 saved (label 2) item 7268
6451    7079    0 # user 6451 did not interact (label 0) item 7079
```

- task1_test_queries.tsv: Test queries with 243,154 user–item pairs. Labels are not provided; you need to predict them. Below are the first three rows from the file:

```
8064    5067 # How did user 8064 interact with item 5067?
5153    1186 # How did user 5153 interact with item 1186?
1844    2898 # How did user 1844 interact with item 2898?
```

- ***Note: All user–item pairs involving the users for task 1 that do not appear in any file have label 0.*** *That is, if a particular user–item pair involving a user for task 1 is not listed in the training, validation, or test files, it is guaranteed that there is no interaction (label 0) between them.*

3.1.3 Evaluation

The evaluation of Task 1 is based on the **Macro F1-score**, a standard metric for multi-class classification that is particularly well-suited for scenarios with class imbalance. This metric assesses a model's performance on each class individually and then averages their scores, ensuring that the performance on rare classes is weighted equally to that of frequent classes. The calculation proceeds as follows:

1. For each interaction label (i.e., class) $c \in \{0, 1, 2, 3\}$, we first compute its precision and recall:

$$\text{Precision}_c = \frac{\text{True Positives}_c}{\text{True Positives}_c + \text{False Positives}_c}$$
$$\text{Recall}_c = \frac{\text{True Positives}_c}{\text{True Positives}_c + \text{False Negatives}_c}$$

2. Next, the F1-score for each class is calculated as the harmonic mean of its precision and recall:

$$F_1\text{-score}_c = 2 \times \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

   If both precision and recall for a class are zero, its F1-score is defined as 0.

3. Finally, the overall Macro F1-score is the arithmetic mean of the F1-scores across all classes:

$$\text{Macro F1} = \frac{1}{4}\sum_{c=0}^{3} F_1\text{-score}_c$$

The final score will be this Macro F1-score, which ranges from 0 (worst performance) to 1 (perfect performance). As an implementation tip, you can implement this score using the `sklearn.metrics.f1_score` function from the scikit-learn library, with the parameter set to `average='macro'`.[1]

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

3.1.4 Performance of Simple Baselines

Here, for your reference, we report the performance of two simple baseline methods on the *validation set*:

- **Always 0**: Always predict label 0 (*none*) for every user–item pair. Final Score = 0.167.
- **Random (Empirical Distribution)**: Predict each label by sampling from $\{0, 1, 2, 3\}$ according to their empirical frequencies in the observed data (i.e., the training set). Final Score = 0.210 ± 0.0009 (mean $\pm$ standard deviation over five independent trials).

3.1.5 Report and Submission

In your report, you need to include the evaluation accuracy of your predictions on the validation set. **For the final submission**, you need to submit `task1_test_answers.tsv`, which should contain 243,154 lines. Each line should contain (1) user ID, (2) item ID, and (3) predicted label, separated by tabs, which answers the query in the corresponding line in `task1_test_queries.tsv`. That is, the format should match that of the file of validation answers `task1_val_answers.tsv`.

3.2 Task 2: User-Level Prediction

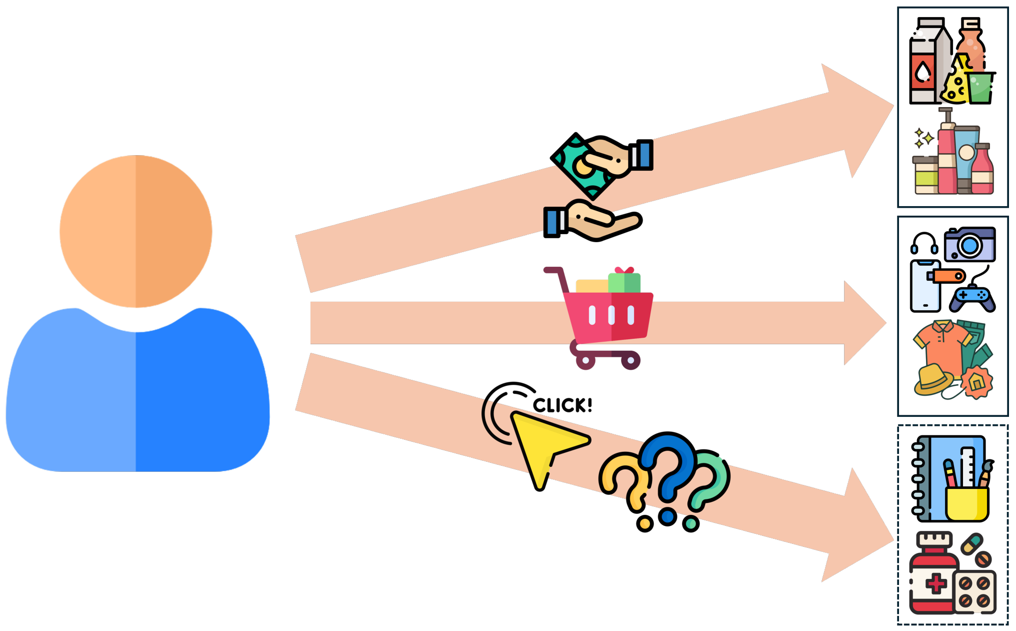

Figure 4: Illustration of Task 2. For each user, given their history of stronger interactions, i.e., labels 2 (saved) and 3 (purchased), the goal is to predict the items they most likely *viewed* (label 1).

3.2.1 Goal

For each user, given their history of stronger interactions (labels 2 and 3), predict the items they most likely *viewed* (label 1) among the remaining items. Specifically, generate a ranked list of the top-k (k is consistently 50 for each user in this task) items per user, where the most likely viewed items should be listed first. See Figure 4 for an illustration of task 2.

3.2.2 Files

- `task2_train.tsv`: Training set with 319,905 lines, containing all user–item pairs with labels 2 and 3.
- `task2_val_queries.tsv`: Validation queries with 3,752 user IDs.
- `task2_val_answers.tsv`: Validation answers with 374,905 lines, containing all label-1 pairs for the queried users. Each user has at least 50 such label-1 items.
- `task2_test_queries.tsv`: Test queries with 3,752 user IDs (disjoint from validation users). Each queried user is guaranteed to have at least 50 label-1 interactions.

3.2.3 Evaluation

The evaluation of Task 2 is based on a **weighted ranking scheme** that takes into account both the correctness of predicted items and their positions in the ranked list. Specifically, each correctly predicted item contributes a score that decreases with its rank position, reflecting the intuition that higher-ranked predictions are more important and should be rewarded more.

The scheme is designed with the following principles:

- **Higher ranks are rewarded more.** Correct hits appearing earlier in the top-50 list contribute more to the score than those placed lower. For example, a correct hit at rank 1 is worth more than a correct hit at rank 20.
- **All hits receive some credit.** Any ground-truth viewed item appearing anywhere in the top-50 list contributes positively to the score, even if ranked late.

Formally, let $\mathcal{U}$ denote the set of all users in the evaluation set. For each user $u \in \mathcal{U}$, let G_u be the set of ground-truth viewed items and let

$$R_u = [r_{u,1}, r_{u,2}, \ldots, r_{u,50}]$$

denote the ranked list of 50 predicted items. The user-level score is defined as

$$S(u) \ = \ \sum_{j=1}^{50} w_j \cdot \mathbf{1}[r_{u,j} \in G_u] \in \left[0, \sum_{j=1}^{50} w_j\right],$$

where $\mathbf{1}[\cdot]$ is the indicator function and w_j is a position-dependent weight that decreases with j. We use the weighted $w_j = 1/\log_2(j+1)$ borrowed from the Discounted Cumulative Gain (DCG) measure.

The normalized score for user u is

$$\tilde{S}(u) \ = \ \frac{S(u)}{\max(S(u))} \in [0, 1],$$

where $\max(S(u))$ denotes the maximum achievable score for user u. Since we guarantee that each user has at least 50 label-1 items, $\max(S(u)) = \sum_{j=1}^{50} w_j$ for all users.

The final evaluation metric is the average normalized score across all users:

$$\text{Final Score} \;=\; \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \tilde{S}(u) \;\in [0, 1].$$

3.2.4 Performance of Simple Baselines

Here, for your reference, we report the performance of two simple baseline methods on the *validation set*:

- **Greedy (Most Popular Items)**: For each item, we count the number of users with any interactions (labels 1, 2, or 3) in the training set. For each query user, we then recommend the most popular items that are not already in their strong-interaction history (labels 2 or 3), ranking them in descending order of popularity. In case of ties in popularity, items are ordered by ascending item ID (small IDs are listed first). Final Score = 0.118.

- **Random (Popularity-Weighted Sampling)**: Based on the same observed popularity distribution, for each query user we sample items *without replacement*, weighted by item popularity, to form the top-50 list. The sampled items are then ordered by descending popularity (ties broken by ascending item ID). Final Score = 0.054 ± 0.0509 (mean $\pm$ standard deviation over five independent trials).

3.2.5 Report and Submission

In your report, you need to include the evaluation accuracy of your predictions on the validation set. **For the final submission**, you need to submit `task2_test_answers.tsv` with 3,752 lines. Each line must contain (1) user ID and (2) the IDs of 50 items (so each line should contain 51 numbers in total), separated by tabs, ranked in descending order of predicted confidence (i.e., most likely viewed items are listed first). Note that this format **differs from** that of `task2_val_answers.tsv`.

4 Notes

- To ensure fairness, we may run your submitted code on an alternative query set if your answers appear suspiciously similar to those of another group.

- You may encounter some subtleties during your implementation. You can come up with your own design and/or contact the TAs — Fanchen Bu (boqvezen97@kaist.ac.kr) and Yeonje Choi (yeonje-choi@kaist.ac.kr) — for discussion. If you receive ideas or inspiration from others (including TAs, external resources, or peers), please explicitly acknowledge them in the **readme.txt** file when you submit your assignment.

- Unlike the other assignments, for this project, you are free to use any programming language and any external library.

5 Project Submission Guidelines

This project requires three separate submissions: a progress report, a final report, and a video presentation. Each must be uploaded to its own entry on KLMS by its respective deadline. The specific requirements for each submission are detailed below.

5.1 Progress Report Submission (By November 14, 2025, 11:59 PM)

You are required to submit a progress report, written using the attached template, to KLMS by November 14, 2025, 11:59 PM. The file should be named according to the following rule:

```
report-[studentID1_studentID2_...].pdf
```

For example, if your team members have the student IDs `20189000`, `20199000`, and `20209000`, then the report filename should be:

```
report-20189000_20199000_20209000.pdf
```

Further details regarding the report format and content will be announced soon.

5.2 Final Report Submission (By December 5, 2025, 11:59 PM)

You are required to submit your final report as a single compressed archive to KLMS by December 5, 2025, 11:59 PM. The file should be named according to the following rule:

```
project-[studentID1_studentID2_...].tar.gz
```

For example, if your team members have the student IDs `20189000`, `20199000`, and `20209000`, then the archive filename should be:

```
project-20189000_20199000_20209000.tar.gz
```

The archive must contain the following files:

- **final_report.pdf**: Written using the attached LaTeX template. This report must include the accuracy of your predictions on the validation sets for both tasks.
- **test_prediction.tar.gz**: A compressed archive containing the answers for the test set for both tasks: `task1_test_answers.tsv` and `task2_test_answers.tsv`.
- **readme.txt**: A text file listing the names of any individuals from whom you received help, along with a brief description of the assistance. This includes support from friends, classmates, lab members, TAs, etc. You may also use this file to provide comments that could help us grade your project more fairly, as well as any feedback or additional ideas you wish to share.
- **code.tar.gz**: A compressed archive containing your full implementation.

Please ensure that no additional files are included in the submitted archive.

5.3 Video Presentation Submission (By December 7, 2025, 11:59 PM)

You are required to submit your video presentation as a single compressed archive to KLMS by December 7, 2025, 11:59 PM. The submitted submissions will be presented in class on December 8 & 10, 2025. The file should be named according to the following rule:

```
presentation-[studentID1_studentID2_...].tar.gz
```

For example, if your team members have the student IDs `20189000`, `20199000`, and `20209000`, then the archive filename should be:

```
presentation-20189000_20199000_20209000.tar.gz
```

The archive must contain the following files:

- **slides.pdf**: The slides used for your final presentation.
- **video.mp4**: A recorded presentation video not exceeding 5 minutes in length.

Please ensure that no additional files are included in the submitted archive.