Amazon Co-Purchase and Review Networks for Next-Gen Recommendations

Yeseul Han
York University
Toronto, Canada
hany72@my.yorku.ca

Erfan YousefMoumji
York University
Toronto, Canada
eym98@yorku.ca

Xi Lu
York University
Toronto, Canada
lucy0505@my.yorku.ca

Keywords

Amazon 5-core, co-purchase graph, link prediction, time-aware evaluation, leakage-free split, cold-start, two-hop blocking, Adamic–Adar, Common Neighbors, Jaccard, Preferential Attachment, Average Precision, AUC, Precision@K, suspicion filtering, burstiness, near-duplicate detection, rating–time drift, GraphSAGE, LightGCN

1 Abstract

This project investigates **time-aware link prediction** on Amazon 5-core co-purchase networks and evaluates how well classical graph heuristics can forecast future product relationships under strict, leakage-free temporal constraints. We construct **weighted, undirected item–item graphs** for three categories—*Electronics*, *Home & Kitchen*, and *All Beauty*—and apply a chronological train/test split using the 80th percentile of edge timestamps. All graph features and scoring functions are computed solely from data with timestamps $\leq t$, replicating a realistic recommendation setting where only past information is available for prediction.

We evaluate four widely used link-prediction heuristics—**Common Neighbors**, **Jaccard**, **Adamic–Adar**, and **Preferential Attachment**—and report AUC, Average Precision, and Precision@K under both **warm-start** (nodes seen at training time) and **cold-start** (newly introduced nodes) conditions. In warm-start settings, Adamic–Adar and Common Neighbors consistently achieve the strongest performance, reflecting the predictive value of shared-neighborhood structure in co-purchase graphs. However, performance drops sharply in cold-start evaluation across all categories, underscoring the fundamental difficulty of predicting edges for products that lack any prior connectivity.

We further include a **runtime analysis** covering graph construction, candidate generation, and baseline scoring, as well as a **synthetic Barabási–Albert experiment** demonstrating that these heuristics perform strongly on graphs with clear structural patterns. This contrast highlights that cold-start failure on Amazon stems primarily from **graph sparsity and new-item emergence**, rather than limitations of the heuristics themselves. Overall, the project provides a fully reproducible, time-aware evaluation framework for co-purchase link prediction and clarifies key structural and temporal challenges faced by recommendation systems in large-scale e-commerce environments.

2 Introduction

Recommender systems rely on understanding how products relate to one another, and one of the most informative signals comes from *co-purchase behavior*. Products frequently reviewed or purchased by overlapping users tend to become connected over time, forming a dynamic item–item graph. A key problem in this setting is *link prediction*: forecasting which product pairs will form connections in the future based on the current structure of the graph.

Many prior studies evaluate link prediction on static snapshots, unintentionally mixing information across time and allowing models to leverage "future" edges when predicting the past. This introduces *temporal leakage*, artificially inflating metrics and producing unrealistic conclusions about model performance. To address this limitation, we adopt a **strictly time-aware evaluation protocol** for Amazon 5-core co-purchase networks. Using per-edge timestamps, we apply an 80th-percentile chronological cutoff and compute all features, neighborhoods, and scores using only information available at or before this point. This setup reflects the constraints of real-world recommender systems, where predictions must rely exclusively on past behavior.

A major challenge revealed by such temporally honest evaluation is the **cold-start problem**. Newly introduced products appear only after the cutoff and therefore have no edges or neighborhood information at training time. Classical heuristics—such as Common Neighbors, Jaccard, Adamic–Adar, and Preferential Attachment—depend heavily on structural evidence and thus struggle when encountering previously unseen items. Quantifying how severely performance degrades in these cases is essential for understanding the practical limits of topology-only models.

In this project, we construct item–item co-purchase graphs for three Amazon categories—*Electronics*, *Home & Kitchen*, and *All Beauty*—apply a leakage-free temporal split, generate two-hop candidate sets for efficient evaluation, and benchmark four classical link-prediction heuristics under both **warm-start** and **cold-start** scenarios. To contextualize the results, we additionally include a **runtime analysis** covering graph construction, candidate generation, and baseline scoring, as well as a **synthetic Barabási–Albert experiment** showing that these heuristics perform strongly on graphs with clear structural patterns.

Overall, our goal is to provide a **reliable, reproducible, and temporally faithful framework** for studying link prediction in co-purchase networks, characterize the differences between warm-start and cold-start performance, and highlight the structural and temporal constraints that shape recommendation in large-scale e-commerce environments.

3 Problem Definition

Data constraint: Amazon review data is extremely sparse, and co-review patterns become unreliable when users or items have very few interactions. To ensure that graph structure is meaningful, we use the Amazon *5-core* subsets, where every user and every item has written or received at least five reviews. Let $\mathcal{D}_c$ denote a category-specific 5-core dataset (e.g., *Electronics*, *Home & Kitchen*, *All Beauty*).

Product graph G: For each category, we construct an item–item co-purchase graph

$$G = (V, E, W),$$

where V is the set of products (asins) and an undirected edge $(i, j) \in E$ is added when a user has reviewed both items i and j. The weight w_{ij} counts how many users co-reviewed the pair. Each edge (i, j) is assigned a timestamp t_{ij} defined as the earliest time at which a user's review history contains both i and j. For evaluation, we restrict to the giant connected component $G_{\text{GCC}} \subseteq G$ to remove isolated nodes and focus on structurally meaningful regions of the graph.

Time-aware split and leakage: To avoid temporal leakage, we partition edges chronologically using a cutoff t^* defined as the 80th percentile of all edge timestamps:

$$E^{\text{train}} = \{(i, j) \in E : t_{ij} \le t^*\}, \qquad E^{\text{test}} = \{(i, j) \in E : t_{ij} > t^*\}.$$

All neighborhood statistics (Γ, degrees, intersections, unions) used for scoring are computed *only* from $G(E^{\text{train}})$.

Candidate non-edges via 2-hop blocking: Full evaluation over all non-edges is infeasible ($O(|V|^2)$). Instead, following common practice, we restrict to node pairs within two hops in the train graph:

$$S = \{(i, j) : i < j,\ (i, j) \notin E^{\text{train}},\ \Gamma(i) \cap \Gamma(j) \neq \emptyset\}.$$

This reflects the empirical observation that most future edges arise between items that already share a structural vicinity.

Warm-start vs. cold-start edges: Warm-start test edges connect nodes already present in G_{train}. Cold-start edges connect to nodes that appear for the first time after t^*. Classical heuristics depend on neighborhood overlap and therefore cannot reliably score items with no prior connectivity, making cold-start analysis crucial for realistic evaluation.

P1 — Time-aware link prediction on the product graph. Input: G_{train} at time t^* and candidate set S. *Output:* a scoring function $s(i, j)$ ranking candidate pairs by likelihood of forming future edges. *Objective:* maximize AUC, Average Precision (AP), and Precision@K on E^{test} under strict temporal constraints.

Evaluation slices: We report: (i) *warm-start* performance (nodes seen during training), (ii) *cold-start* performance (nodes unseen during training), and (iii) per-category breakdowns. This separation clarifies how sparsity and new-item emergence affect the predictive power of topology-only models.

Symbol	Definition
$\mathcal{D}_c$	Amazon 5-core subset for category c
$G = (V, E, W)$	Product co-purchase graph
t^*	Time cutoff for train/test split (80th percentile)
$E^{\text{train}}, E^{\text{test}}$	Train/test edge sets
S	Two-hop candidate non-edges
$s(i, j)$	Link prediction score for pair (i, j)

Table 1: Notation used in Problem Definition.

4 Related Work

Research on product recommendation networks frequently models items as nodes in a graph and co-purchase or co-review signals as edges. Prior work has shown that simple structural features such as shared neighbors, degrees, or normalized overlap encode meaningful information about future link formation.

4.1 Product graphs and link prediction

Classical link-prediction heuristics such as Common Neighbors, Jaccard, Adamic–Adar, and Preferential Attachment remain widely used due to their low computational cost and interpretability [1, 8]. These heuristics rely purely on topological structure and have been evaluated extensively on social, citation, and co-purchase networks. More expressive models, including Node2Vec, GraphSAGE, and LightGCN [4, 5], learn node embeddings from graph neighborhoods and often outperform heuristic methods on large-scale recommendation benchmarks. Recent studies also benchmark multiple GNNs on Amazon co-purchase graphs [2], demonstrating that embedding-based methods can capture long-range structure.

Our angle: We focus on a *strictly time-aware* evaluation setup, constructing co-purchase graphs directly from Amazon 5-core reviews [11] and enforcing leakage-free chronological splits at cutoff t^*. Unlike work that evaluates baselines on static snapshots or pre-built Amazon meta-graphs, our evaluation reflects temporal constraints that real-world recommender systems must satisfy.

4.2 Time-aware evaluation and candidate selection

A growing body of research emphasizes the importance of chronological evaluation to avoid temporal leakage, which can significantly inflate link-prediction metrics when future information is implicitly incorporated into training. Time-based train/test splits have been adopted in dynamic network analysis and recommendation pipelines to ensure that only past information is used for predicting future structure.

Scalability is another key concern. Because the number of possible non-edges grows quadratically with the number of nodes, many temporal link-prediction studies rely on candidate-reduction methods such as two-hop blocking. This strategy restricts evaluation to item pairs that share at least one neighbor in the train graph, focusing on structurally plausible edges and reducing computational costs. Our methodology follows this standard approach.

4.3 Synthetic graph baselines

Synthetic graph models, particularly Barabási–Albert (BA) networks, are commonly used to contextualize real-world results. BA graphs exhibit heavy-tailed degree distributions similar to many social and product networks, allowing researchers to test whether heuristics behave as expected on idealized structures. Our synthetic BA experiment aligns with this tradition and demonstrates that classical heuristics perform strongly when the underlying graph is dense and displays clear hierarchical structure—clarifying that poor performance on Amazon cold-start items arises from data sparsity rather than from inherent weaknesses of the heuristics themselves.

5 Methodology

5.1 Pipeline overview

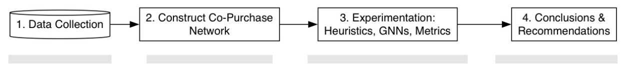
1. Data Collection
2. Construct Co-Purchase Network
3. Experimentation: Heuristics, GNNs, Metrics
4. Conclusions & Recommendations

Figure 1: End-to-end pipeline from data loading to graph construction, time-aware splitting, candidate generation, baseline scoring, and evaluation.

5.2 Dataset: Amazon 5-core

We use three Amazon 5-core category subsets *Electronics*, *All_Beauty*, and *Home_and_Kitchen* where each user and product has at least five reviews [11]. To prevent memory overload, we load at most 100,000 rows for the larger categories while loading the full All_Beauty dataset (5,269 rows). We drop rows with missing reviewer IDs or product IDs. Each valid row contains `reviewerID`, `asin`, `overall` (rating), and `unixReviewTime` (timestamp).

Category	$\lvert V\rvert$	$\lvert E\rvert$	Density
Electronics	720	3144	0.0121
Home & Kitchen	579	3849	0.0230
All Beauty	58	236	0.1428

Table 2: Graph statistics on the giant connected component (GCC) for each category.

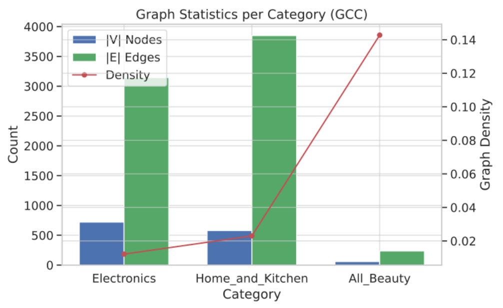

Figure 2: Category-wise node counts in the GCC.

5.3 Product graph G

We construct a weighted item–item co-purchase graph

$$G = (V, E, W)$$

from co-review patterns. For each user, we sort their reviews by time and connect every pair of products they reviewed. The weight w_{ij} counts how many users co-reviewed items i and j. The timestamp t_{ij} is the earliest time when a user's chronological review history contains both products. We extract the giant connected component G_{GCC} for all subsequent steps.

5.4 Time-based split and leakage guards

To enforce temporal correctness, we choose a cutoff t^* equal to the 80th percentile of all edge timestamps and define

$$E^{\text{train}} = \{(i,j) : t_{ij} \le t^*\}, \qquad E^{\text{test}} = \{(i,j) : t_{ij} > t^*\}.$$

All features used for link prediction (degrees, neighborhoods, intersections, unions) are computed exclusively on $G(E^{\text{train}})$.

5.5 Warm-start and cold-start edges

We separate test edges into:

- **Warm-start:** both endpoints appear in G_{train},
- **Cold-start:** one or both endpoints appear only after t^*.

Classical heuristics cannot score nodes with no neighbors, making cold-start evaluation essential.

5.6 Two-hop candidate generation

Enumerating all non-edges is infeasible. We restrict candidates to structurally plausible pairs:

$$S = \{(i,j) : (i,j) \notin E_{\text{train}},\ \Gamma(i) \cap \Gamma(j) \neq \emptyset\}.$$

This dramatically reduces candidate size while capturing the majority of future edges.

5.7 Link-prediction baselines

We evaluate four standard heuristics:

$$\text{CN}(i,j) = |\Gamma(i) \cap \Gamma(j)|, \qquad \text{Jaccard}(i,j) = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|},$$

$$\text{Adamic–Adar}(i,j) = \sum_{z \in \Gamma(i) \cap \Gamma(j)} \frac{1}{\log \deg(z)}, \qquad \text{PA}(i,j) = \deg(i) \cdot \deg(j).$$

Scores are computed on G_{train} and evaluated against E_{test}.

5.8 Runtime measurement

We measure:

- graph construction time,
- two-hop candidate generation time,
- baseline scoring time.

Runtimes are recorded per category and merged into a unified summary.

5.9 Synthetic Barabási–Albert sanity check

To contextualize Amazon results, we generate a synthetic Barabási–Albert graph and apply the same 80/20 temporal split and baseline evaluation. The strong performance of all heuristics on the BA model confirms that poor cold-start performance on Amazon arises from sparsity and new-item emergence rather than algorithmic limitations.

5.10 Complexity and implementation notes

With two-hop blocking, scoring complexity is

$$O\left(\sum_{(i,j) \in S} \min\{\deg(i), \deg(j)\} \right).$$

Experiments were run over five seeds and all intermediate results were cached for reproducibility.

6 Evaluation

6.1 Setup

We evaluate three Amazon **5-core** categories (*Electronics*, *Home and Kitchen*, *All Beauty*) under the time-aware split of Section 5.4, using the 80th percentile timestamp $t^\star$ to form E^{train} and E^{test}. All neighborhood statistics, edge weights, and suspicion indices are computed strictly with data $\leq t^\star$ to avoid leakage. Candidate non-edges S are obtained via 2-hop blocking on $G(E^{\text{train}})$. Four classical link-prediction heuristics—Common Neighbors (CN), Jaccard, Adamic–Adar (AA), and Preferential Attachment (PA)—score $(i, j) \in S$.

Unless stated otherwise, results are averaged over five seeds $\{42, 43, 44, 45, 46\}$ with standard deviations included for multi-seed aggregates.

6.2 Metrics and reporting

We report two families of ranking metrics: (i) AUC and Average Precision (AP) over the full candidate pool, and (ii) Precision@K for $K \in \{10, 20, 50, 100, 200, 500\}$. All warm-start metrics are macro-averaged across categories unless per-category breakdowns are shown. Cold-start results are reported where the category contains test edges whose endpoints were unseen during training.

6.3 Train/test chronology

Figure 3 shows category-wise CDFs of edge timestamps. The 80th-percentile cutoff produces a sufficiently large holdout across categories, with *Electronics* and *Home & Kitchen* providing especially deep test sets.

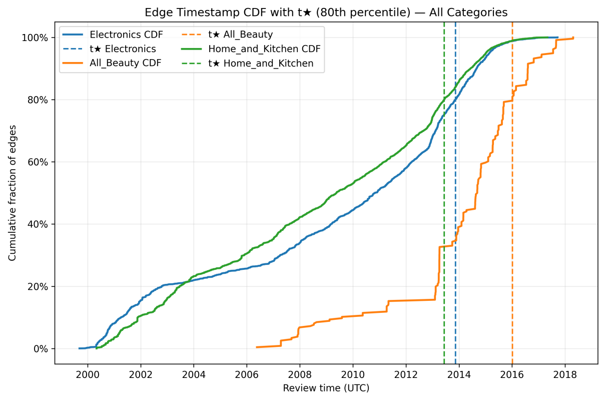

Figure 3: Edge-timestamp CDFs per category with the time cutoff $t^\star$ (80th percentile).

6.4 Warm-start performance

Warm-start considers only test edges whose endpoints appear in the training graph. Figures 4 and 5 show AP and AUC across baselines and categories.

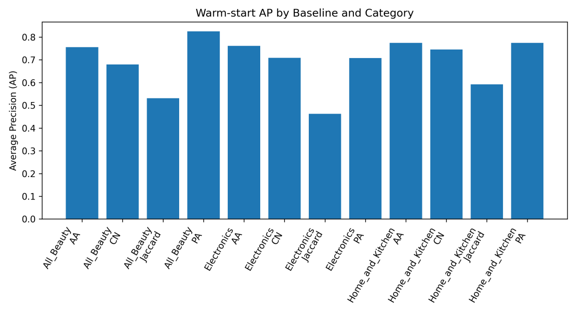

Figure 4: Warm-start AP by baseline and category (mean over 5 seeds).

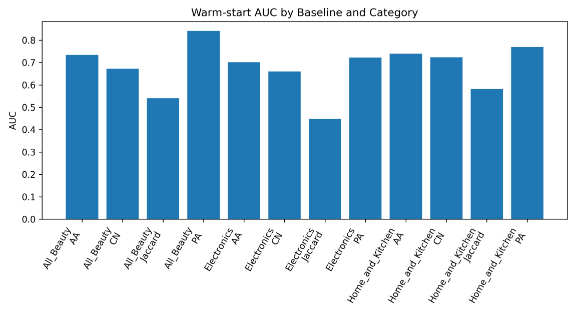

Figure 5: Warm-start AUC by baseline and category (mean over 5 seeds).

Across datasets, AA and CN provide the strongest warm-start performance, with PA close behind. Jaccard trails at small K on sparse graphs (e.g., *Electronics*, *All Beauty*), but narrows the gap as neighborhoods expand. These trends are consistent with co-purchase graphs, where shared-neighbor evidence is highly predictive and degree-normalized measures can suffer when neighborhoods are tiny.

6.5 Cold-start performance

Cold-start evaluates test edges involving nodes unseen at training time. Only *Electronics* exhibits a non-trivial cold-start test set under the 80th-percentile timestamp split; *Home & Kitchen* has none, and *All Beauty* is too small for stable reporting at large K. Figure 6 visualizes ΔPrecision@100 = (Cold − Warm) for *Electronics*.

Baseline	Warm P@100	Cold P@100	ΔP@100
AA	0.9020	0.4800	−0.4220
CN	0.8980	0.4800	−0.4180
Jaccard	0.3580	0.4800	+0.1220
PA	0.8000	0.4800	−0.3200

Table 3: Cold-start effect on Precision@100 for *Electronics*. Jaccard is unusually robust; degree-based heuristics degrade sharply.

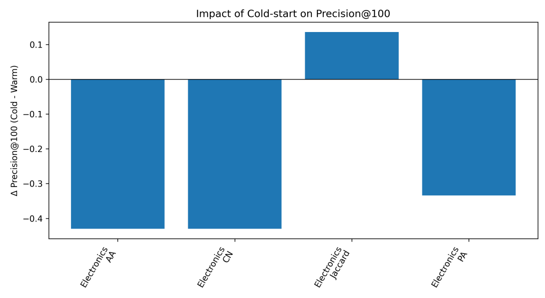

Figure 6: Cold-start impact on P@100 (*Electronics*). Negative values indicate drop from warm-start.

Cold-start degradation table.

Interpretation. Degree-based heuristics (AA, CN, PA) rely heavily on observed neighborhoods and therefore suffer substantial drops under cold-start. Jaccard, due to its union-normalized denominator, is more conservative and consequently more robust when sparse or unseen nodes appear.

6.6 Candidate pool scale

The size of the 2-hop candidate set S affects metric resolution and the number of negative distractors. Larger pools (e.g., *Home & Kitchen*) naturally depress P@K at high K.

Category	# Candidates (2-hop)
Electronics	34,308
All Beauty	438
Home & Kitchen	42,110

Table 4: Candidate non-edge counts under 2-hop blocking.

6.7 Runtime comparison

We report wall-clock times for scoring the full candidate set S per category.

6.8 Warm-start summary

Table 6 consolidates AP, AUC, and P@100 for all categories. As expected, AA and CN dominate across datasets.

6.9 Limitations and next steps

Our evaluation centers on classical, topology-based heuristics under a strict temporal protocol. Several extensions remain:

- Integrating suspicion-filtered graphs (e.g., removing or down-weighting high-burstiness users).
- Adding learned baselines (Node2Vec, GraphSAGE, or Light-GCN) under the identical time-aware split.
- Studying node-embedding drift and community-aware temporal diffusion models.

These extensions constitute the core of D3 and support a complete investigation into temporal and integrity-aware recommendation signals.

Category	Baseline	Runtime (s)
Electronics	CN	0.030
Electronics	Jaccard	0.034
Electronics	AA	0.055
Electronics	PA	0.004
All Beauty	CN	0.0015
All Beauty	Jaccard	0.0020
All Beauty	AA	0.0102
All Beauty	PA	0.0007
Home & Kitchen	CN	0.0226
Home & Kitchen	Jaccard	0.0264
Home & Kitchen	AA	0.0648
Home & Kitchen	PA	0.0044

Table 5: Runtime of classical baselines. PA is consistently fastest due to its degree-only computation.

Category	Baseline	AP	AUC	P@100
Electronics	AA	0.7526	0.6934	0.9020
Electronics	CN	0.7011	0.6535	0.89f80
Electronics	Jaccard	0.4618	0.4480	0.3580
Electronics	PA	0.7005	0.7178	0.8000
Home & Kitchen	AA	0.7793	0.7390	0.9360
Home & Kitchen	CN	0.7502	0.7226	0.9340
Home & Kitchen	Jaccard	0.6015	0.5860	0.7000
Home & Kitchen	PA	0.7763	0.7678	0.9240
All Beauty	AA	0.7454	0.7065	—
All Beauty	CN	0.6700	0.6490	—
All Beauty	Jaccard	0.5039	0.4943	—
All Beauty	PA	0.7905	0.8198	—

Table 6: Warm-start AP, AUC, and P@100 (mean over 5 seeds). P@100 is omitted for All_Beauty due to insufficient test edges.

A Top-20 Predicted Edges (Qualitative Examples)

This appendix provides qualitative insight into the types of product pairs favored by each link-prediction heuristic. For each category, we list the top-20 highest-scoring candidate edges under CN, Jaccard, Adamic–Adar (AA), and Preferential Attachment (PA). These examples illustrate structural bias: CN/AA emphasize shared-neighbor patterns, Jaccard favors tight mutual neighborhoods, and PA amplifies high-degree nodes.

A.1 Electronics — CN

A.2 Electronics — Jaccard

A.3 Electronics — Adamic–Adar

A.4 Electronics — PA

References

[1] Lada A. Adamic and Eytan Adar. 2003. Friends and Neighbors on the Web. In *Proceedings of the Social Networks Workshop.*

Item *u*	**Item** *v*	**CN score**
B00004T8R2	B0000510R4	24
B00004Z5CP	B000067S60	21
B000065BP9	B000067RC4	21
B00001P4ZH	B00004Z5M1	20
B00004Z5CP	B000065BPB	19
B00004SABB	B0000668YX	19
B000065BP9	B0000668YX	18
B0000668YX	B000068O4N	18
B000062VUQ	B000067RTB	18
B00004Z5CP	B000068O3C	18
B00004ZCJI	B00005ATMB	18
B00004T8R2	B000068O4N	18
B00004ZCJJ	B00005ATMB	18
B00004Z5CP	B00005ATMB	17
B00001WRSJ	B000068O18	17
B000065BPB	B0000668YX	17
B00004Z5CP	B000065BP9	17
B00004T8R2	B00005ATMB	16
B00002EQCW	B00004T8R2	16
B00001P4ZH	B00004SB92	16

Table 7: Top-20 CN predictions for *Electronics*.

u	*v*	AA
B00004T8R2	B0000510R4	7.258
B000065BP9	B000067RC4	6.258
B00001P4ZH	B00004Z5M1	6.236
B00004Z5CP	B000067S60	5.909
B000067RTB	B000068O4N	5.893
B00004ZCJI	B00005ATMB	5.755
B00004ZCJJ	B00005ATMB	5.755
B00004SABB	B0000668YX	5.469
B00004Z5CP	B000065BPB	5.423
B00004T8R2	B000068O4N	5.402
B00004SB92	B00005LB8P	5.399
B0000668YX	B000068O4N	5.341
B000067RC4	B000068O49	5.290
B000067RC4	B000068O41	5.290
B00004Z5CP	B00005ATMB	5.267
B000062VUQ	B000067RTB	5.213
B00004Z5CP	B000068O3C	5.140
B000065BP9	B0000668YX	5.078
B00001WRSJ	B000068O18	4.942
B00001P4ZH	B00004SB92	4.871

Table 9: Top-20 AA predictions for *Electronics*.

u	*v*	Jac
B000031WCH	B000058TLP	1.0
B00003CWFA	B00005V8S8	1.0
B00004UFRO	B00005QF7H	1.0
B00004TX77	B00004X0ZO	1.0
B00005AY7G	B00005LEOM	1.0
B00000J434	B00005QCT3	1.0
B00003CWFA	B00005QBUU	1.0
B00005AY7G	B000067O8H	1.0
B00004WCCQ	B00005T3N0	1.0
B00000J434	B00005MNSS	1.0
B00000JCTO	B00005V8S8	1.0
B00004UFRO	B00004X10C	1.0
B000001ON0	B00005TQ1Y	1.0
B00000JCTO	B00005QBUU	1.0
B00004R7OI	B00004RIW8	1.0
B00003CWFA	B00004WCG6	1.0
B00005137P	B00005NWQN	1.0
B00005QF7H	B00005V8S8	1.0
B00005U7RB	B000067RQ2	1.0
B000001OM4	B00000J1UA	1.0

Table 8: Top-20 Jaccard predictions for *Electronics*.

u	*v*	PA
B00004SB92	B0000668YX	7854
B00001P4ZH	B00004SB92	7854
B00004SB92	B00004T8R2	7854
B00004SB92	B000067RC4	7735
B00004SB92	B00004Z5CP	7497
B00004ZCJI	B00005ARK3	7020
B00004ZCJJ	B00005ARK3	7020
B00004SB92	B000067S60	5950
B00003G1RG	B00004ZCJJ	5928
B00003G1RG	B00004ZCJI	5928
B00004SB92	B00004ZCC1	5712
B00004SB92	B000065BP9	4641
B00004SB92	B000068O18	4403
B00002JXBI	B00004ZCJI	4368
B00002JXBI	B00004ZCJJ	4368
B00004SABB	B00004SB92	4284
B00004ZCJI	B000068O33	4212
B00004ZCJJ	B000068O33	4212
B00004ZCJI	B00005ATMB	4212
B00004ZCJJ	B00005ATMB	4212

Table 10: Top-20 PA predictions for *Electronics*.

[2] Mengyang Cao, Frank F. Yang, Yi Jin, and Yijun Yan. 2025. Graph Neural Network for Product Recommendation on the Amazon Co-purchase Graph. arXiv:2508.14059 [cs.IR] https://arxiv.org/abs/2508.14059 Benchmarks LightGCN, GraphSAGE, GAT and PinSAGE on the Amazon-meta dataset.

[3] Linton C. Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry* 40, 1 (1977), 35–41. doi:10.2307/3033543

[4] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS '17)*. Curran Associates, Inc. arXiv:1706.02216 https://cs.stanford.edu/people/jure/pubs/graphsage-nips17.pdf

[5] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of SIGIR*. 639–648. doi:10.1145/3397271.3401063

[6] Nitin Jindal and Bing Liu. 2008. Opinion Spam and Analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM '08)*. ACM, New York, NY, USA, 219–230. doi:10.1145/1341531.1341560

[7] David Kempe, Jon Kleinberg, and Éva Tardos. 2015. Maximizing the Spread of Influence through a Social Network. *Theory of Computing* 11, 4 (2015), 105–147. doi:10.4086/toc.2015.v011a004

[8] David Liben-Nowell and Jon Kleinberg. 2007. The Link-Prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology* 58, 7 (2007), 1019–1031. doi:10.1002/asi.20591

[9] Minghao Liu, Catherine Zhao, and Nathan Zhou. 2025. Building a Recommendation System Using Amazon Product Co-Purchasing Network. https://www.researchgate.net/publication/392371089_Building_a_Recommendation_System_Using_Amazon_Product_Co-Purchasing_Network Preprint (ResearchGate); uses Amazon-meta dataset and a modified GraphSAGE for inductive recommendations.

[10] Jianmo Ni. 2018. *Amazon Review Data (2018)*.

[11] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Amazon Review Data (v2). https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/ Dataset (2018 version): reviews, metadata, and related-product graphs.

[12] OpenAI. 2025. ChatGPT (GPT-5 Thinking) — conversational AI assistant. https://chat.openai.com/ Accessed: 2025-10-09.

[13] Yanping Zheng, Frank de Hoog, Zhewei Wei, Xu Chen, Jiajun Liu, Yuhang Ye, and Jiadeng Huang. 2023. Lighter Graph Convolutional Networks for Recommendation. In *Proceedings of the Workshop on Data-Centric AI (DCAI '23)*. https://dcai-workshop.github.io/assets/pdf/accepted_papers/139.pdf Workshop paper.