E-tail Product Return Prediction via Hypergraph-based Local Graph Cut

Jianbo Li
Three Bridges Capital
New York, New York
jianboliru@gmail.com

Jingrui He
Arizona State University
Tempe, Arizona
jingrui.he@asu.edu

Yada Zhu
IBM Research
Yorktown Heights, New York
yzhu@us.ibm.com

ABSTRACT

Recent decades have witnessed the rapid growth of E-commerce. In particular, E-tail has provided customers with great convenience by allowing them to purchase retail products anywhere without visiting the actual stores. A recent trend in E-tail is to allow free shipping and hassle-free returns to further attract online customers. However, a downside of such a customer-friendly policy is the rapidly increasing return rate as well as the associated costs of handling returned online orders. Therefore, it has become imperative to take proactive measures for reducing the return rate and the associated cost. Despite the large amount of data available from historical purchase and return records, up until now, the problem of E-tail product return prediction has not attracted much attention from the data mining community.

To address this problem, in this paper, we propose a generic framework for E-tail product return prediction named *HyperGo*. It aims to predict the customer's intention to return after s/he has put together the shopping basket. For the baskets with a high return intention, the E-tailers can then take appropriate measures to incentivize the customer not to issue a return and/or prepare for reverse logistics. The proposed *HyperGo* is based on a novel hypergraph representation of historical purchase and return records, effectively leveraging the rich information of basket composition. For a given basket, we propose a local graph cut algorithm using truncated random walk on the hypergraph to identify similar historical baskets. Based on these baskets, *HyperGo* is able to estimate the return intention on two levels: basket-level vs. product-level, which provides the E-tailers with detailed information regarding the reason for a potential return (e.g., duplicate products with different colors). One major benefit of the proposed local algorithm lies in its time complexity, which is linearly dependent on the size of the output cluster and polylogarithmically dependent on the volume of the hypergraph. This makes *HyperGo* particularly suitable for processing large-scale data sets. The experimental results on multiple real-world E-tail data sets demonstrate the effectiveness and efficiency of *HyperGo*.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219829

CCS CONCEPTS

• **Information systems** → **Clustering**; **Online shopping**; • **Computing methodologies** → *Ranking*; • **Applied computing** → *E-commerce infrastructure*;

KEYWORDS

E-commerce; product return; hypergraph; graph partitioning; clustering

ACM Reference Format:
Jianbo Li, Jingrui He, and Yada Zhu. 2018. E-tail Product Return Prediction via Hypergraph-based Local Graph Cut. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219829

1 INTRODUCTION

Generally speaking, E-commerce includes 4 segments of business, Business-to-Consumer (B2C), Business-to-Business (B2B), Consumer-to-Consumer (C2C), and Consumer-to-Business (C2B). E-tail belongs to the B2C segment, and it has been rapidly growing over the years, expected to reach $4 trillion dollars in 2020[1]. To further attract online customers, many E-tailers start to offer free shipping and hassle-free returns, such as Kohl's and Macy's. Such a policy adds to the convenience of online shopping by providing the customers with peace of mind regarding potential discrepancy between expectation and reality. However, the downside of this generous policy is the rocketing return rate and the associated costs of handling returned online orders, including both direct costs such as shipping, re-stocking and re-furbishing, and indirect costs, such as call center demand and customer satisfaction. Therefore, it has become a major challenge for E-tail to accurately predict product returns, especially in the early stage when the order has not been placed yet. In this way, for the customers with a high predicted return intention, the E-tailers can take proactive measures to decrease the return intention, such as popping up a chatbot to provide guidance regarding size and fit, offering discount coupons on the products that contribute to the high predicted return intention, and/or prepare for reverse logistics.

Up until now, this challenging problem has not attracted much attention from the data mining community. The limited existing techniques on E-tail product return prediction suffer from two major issues: (1) they are designed to work **after** the customers have made the purchase, and (2) they are not able to make **customer-product** level predictions, leaving the E-tailers with less options

[1] http://www.huffingtonpost.com/michael-lazar/retailers-people-want-eas_b_12759542.html

when taking proactive measures to counter the return intention and/or prepare for reverse logistics. To address these issues, in this paper, we propose a generic framework for E-tail product return prediction named *HyperGo*. It is based on a novel hypergraph representation of historical purchase and return records, where each node corresponds to a shopping basket, and each hyperedge corresponds to a unique product, connecting all the nodes associated with the baskets containing this product. Compared with regular graphs, the proposed hypergraph representation is able to leverage the rich information of basket composition in a more effective way. For a given basket, we further propose a local graph cut algorithm to identify similar historical baskets. It is based on truncated random walk starting from a given basket, and gradually explores the neighborhood of this seed node on the hypergraph. Different from most existing graph cut algorithms, the time complexity of the proposed local algorithm depends linearly on the size of the output cluster and polylogarithmically on the volume of the hypergraph, making it particularly suitable to be applied on the large amount of data consisting of historical purchase and return records, easily reaching hundreds of thousands of products and millions of baskets. Notice that the proposed *HyperGo* framework is designed to work **before** the customers have made the purchase, potentially leading to a changed basket composition with a decreased return intention. Finally, based on the output cluster from the hypergraph, we propose to estimate the return intention on two levels: basket-level vs. product-level, providing more insights into a potential return, such as the basket containing duplicate products with different colors or sizes.

The main contributions of the paper can be summarized below.

(1) **Problem Setting.** Different from the limited existing work on E-tail product return prediction, in this paper, we aim to predict the return intention before the customers have made the purchase, allowing the E-tailers to take a variety of counter measures with respect to potential returns;
(2) **Generic Framework.** We propose a generic framework named *HyperGo* for predicting product returns, which consists of a novel hypergraph representation, a local graph cut algorithm that works in an effective and efficient way, and dual-level return prediction based on the output cluster from the algorithm;
(3) **Experiments on Real Data Sets.** We evaluate the proposed *HyperGo* on multiple real-world E-tail data sets with promising results.

The rest of the paper is organized as follows. After a brief review of the related work in Section 2, we introduce the proposed *HyperGo* framework in Section 3, including the novel hypergraph representation of historical records, the local graph cut algorithm for identifying similar historical baskets, as well as the dual-level return intention prediction. Section 4 provides some experimental results demonstrating the effectiveness and efficiency of *HyperGo* on real-world E-tail data sets from leading omni-chanel (both online and in-store) E-tailers. Finally, we conclude the paper in Section 5.

2 RELATED WORK

In this section, we briefly review the existing work on predicting product returns, hypergraph models and graph partitioning.

2.1 Product Returns in E-commercce

Numerous studies have shown that at least 30% of all products ordered online are returned every year[2]. Returned products directly cost U.S. e-tailers on shipping and re-stocking well over $200 billions annually [5]. In today's competitive environment, as e-tailers strive to improve customer engagement by offering generous return policies, effective measures to reduce product return rate has significant impact on the sustainability of e-commerce industry [15].

However, today the limited work on product return prediction in literature narrowly focuses on *end-of-life* returns. [14] estimates the total return quantity within a time period for electronic products that have the potential of generating addition revenue through remanfaucturing or reducing environment hazard. [7] reports simple and naive approaches using the proportion of returns to total sales with a known life cycle length and autoregressive-type statistical models to forecast return quantity and time. [14] introduces Bayesian inference model and expectation maximization algorithm to compute the return delay distribution. These methods are not designed for customer-product level return predictions. [15] compares Mahalanobis feature extraction to a total of six dimension reduction techniques for processing sparse feature matrices when return orders are characterized with respect to consumption patterns. The large set of features are originated from categorical variables such as customer's return history, payment method, device and operating system from which a returned basket is ordered. Some information usually is available only when customers have finished their online shopping journey. This could limit the opportunity for e-tailers to take proactive actions. The challenges for predicting customer return intention before they make purchases lie in the large problem scale that usually involves hundreds of thousands of products and millions of historical purchase and return records.

2.2 Hypergraph Modeling and Partitioning

Graphs are widely used in data mining to capture the relationships between entities, where the entities are considered as nodes and the relationships as edges. Such graphs serve as the undelrying data model in many tasks as ranking, clustering, and recommendation. Simple graph, however, can only capture pair-wise relationships between nodes preventing us from modeling complex relationships.

Recently hypergraph based models have become an active research area [11]. A hypergraph is a generalization of simple graphs that can accurately model high-order relations. Early work [1, 10, 19] transforms hypergraphs into simple graphs by mapping the high-order affinities to pairwise relationships. Such transformation could lead to information loss. [17] extends spectral clustering methods for graphs to hypergaphs and further develops a semi-supervised transductive inference setup for embedding. This work has inspired many researchers to apply random walk on hypergraphs [4]. [13] transforms a hypergraph to a normal graph by treating each hyperedge as a star in ranking video hyperlinks, and uses a random walk in a start-shaped graph. [6] applies ranking on a hypergraph that includes users, news articles, and topics for personalized news recommendation. [4] applies random walk in hypergraphs for sentence ranking in document summarization.

[2] http://www.huffingtonpost.com/michael-lazar/retailers-people-want-eas_b_12759542.html

However, all these methods work in a global manner in the sense that they need to explore the entire hypergraph, which can be computationally prohibitive, especially in E-tail product return prediction with hundreds of thousands of products and millions of baskets.

More recently, truncated random walk based local algorithms for graph partitioning have achieved polylogarithmic time complexity in the number of edges. The first of such methods is *NIBBLE* [12] that attempts to minimize the clustering quality metric *cut conductance* for undirected binary graphs. Given a starting vertex, it provably finds a cluster near that vertex in time that is proportional to the size of the output cluster. Finding a cluster in time proportional to its size is an extremely valuable routine for practical problems. Later [2] extends *NIBBLE* using PageRank vectors and develops an algorithm for local unweighted binary graph partitioning. [3] further generalizes their work to directed binary graphs by defining conductance in terms of the stationary distribution of a random walk. More recently [16] adapts *NIBBLE* to undirected and weighted bipartite graphs by introducing a constraint on the number of user nodes in the output clusters for advertisement campaign recommendation. As far as we know, there is no truncated random walk based local graph cut algorithms for hypergraphs as needed in the scale of our problem. Our proposed framework *HyperGo* fills in this gap and provides an effective and efficient solution for E-tail product return prediction.

3 THE PROPOSED *HYPERGO* FRAMEWORK

In this section, we will present our proposed framework *HyperGo* for E-tail product return prediction. More specifically, we start by introducing the major notation used throughout the paper, followed by the hypergraph-based representation of historical purchase and return records. Then we will introduce the local graph cut algorithm to identify similar historical baskets for any given basket, and analyze its performance from various perspectives. Finally, we will discuss the estimation of dual-level return intention based on the output cluster from the proposed algorithm.

3.1 Notation

Let $G = (V, E, w)$ denote the hypergraph that we construct to represent historical purchase and return records, where V is the node set consisting of n nodes, and E is the hyperedge set consisting of m hyperedges. For any hyperedge $e \in E$, it can be considered as a subset of V, i.e., $e \subset V$, indicating that it connects 2 or more nodes together, and $w(e) \geq 0$ is the non-negative weight associated with this hyperedge. Let $\boldsymbol{H}$ denote the $n \times m$ incidence matrix of G. Its entries $H(v, e) > 0$ if any only if $v \in e$, and $H(v, e) = 0$ otherwise. Notice that $H(v, e)$ can vary from product to product even within the same basket, reflecting the quantity of the product in this basket. Based on $\boldsymbol{H}$, we define the degree of a node $v \in V$ as $d(v) = \sum_{e \in E} w(e)H(v, e)$, the degree of a hyperedge $e \in E$ as $\delta(e) = \sum_v H(v, e)$, and the volume of a node subset $S \subset V$ as $\mathrm{vol}\, S = \sum_{v \in S} d(v)$. In particular, the volume of the hypergraph $\mathrm{vol}\, V$ is $\sum_{v \in V} d(v)$. Furthermore, following the definition in [18], we consider a cut on the graph as a partition that separates the node set V into set S and $\bar{S}$, where $\bar{S}$ is the compliment of S. The boundary of S is defined as the set of hyperedges that connect S and $\bar{S}$, i.e. $\partial S = \left\{e \in E | e \cap S \neq \varnothing, e \cap \bar{S} \neq \varnothing\right\}$. The volume of S is the sum of degrees of all the nodes in S, i.e. $\mathrm{vol}\, S = \sum_{v \in S} d(v)$ and the volume of the boundary is

$$\mathrm{vol}\, \partial S = \sum_{e \in \partial S} w(e) \frac{|e \cap S||e \cap \bar{S}|}{\delta(e)}. \tag{1}$$

Throughout this paper, we use boldface upper case letters to denote matrices, and boldface lower case letters to denote vectors.

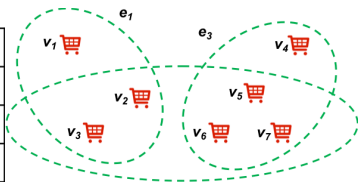
3.2 Hypergraph-based Representation

Given the large number of historical purchase and return records, we propose to build a hypergraph G in order to effectively leverage the rich information from these records. In the hypergraph, each node corresponds to a shopping basket, with or without returned products, and each hyperedge corresponds to a unique product connecting all the basket nodes containing this product. Figure 1 provides a simple example of the proposed hypergraph. Compared with regular graphs where each edge always connects two nodes, the hypergraph constructed this way is able to capture the co-existence of multiple products within the same basket, i.e., multiple hyperedges incident with the same node, which often shed light on potential product returns. For example, if a shopping basket contains multiple similar products such as white T-shirts, more often than not, the customer would like to try them out and only keep one that fits the best while returning the rest. On the other hand, regular graphs are limited to pairs of products and thus lose the big picture of the entire basket.

Intuitively, based on the proposed hypergraph, given a basket, if it contains many shared or similar products with historical baskets with returns, then it is likely to be (partially) returned as well. Given the large number of historical baskets, the challenge is how to identify similar historical baskets effectively (accurately) and efficiently. Our proposed local graph cut algorithm is designed to address this challenge.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
e_1	1	2	1	0	0	0	0
e_2	0	2	1	0	3	1	1
e_3	0	0	0	1	1	1	1

(a)



(b)

Figure 1: An example of the proposed hypergraph built with 3 products $E = \{e_1, e_2, e_2\}$ and 7 shopping baskets $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. (a) The numbers in the table reflect the quantity of each product in a basket. (b) The hypergraph is able to capture the complex relationship among products and shopping baskets.

3.3 Local Graph Cut on Hypergraphs

Let $v \in V$ denote the given basket, for which we would like to predict the return intention. To this end, we first define a random walk on the hypergraph via the following two steps: we first choose

a hyperedge e over all the hyperedges incident with v with probability proportional to $w(e)$, and then choose a node $u \in e$ with probability proportional to $H(u,e)$. The transition probability from node v to node u can be written as follows.

$$P(v,u) = \sum_{e \in E} w(e) \frac{H(v,e)}{d(v)} \frac{H(u,e)}{\delta(e)}. \tag{2}$$

where $P(v,u)$ is the element of the $n \times n$ transition matrix $\boldsymbol{P}$ in the v^{th} row and the u^{th} column. Based on this random walk, we further define the lazy random walk with transition matrix $\boldsymbol{M}$ define as follows.

$$\boldsymbol{M} = (\boldsymbol{P} + \boldsymbol{I})/2 \tag{3}$$

where $\boldsymbol{I}$ is an $n \times n$ identity matrix. For the lazy random walk defined this way, we obtain the stationary distribution according to [18] as follows:

$$\pi(v) = \frac{d(v)}{\operatorname{vol} V}. \tag{4}$$

Based on this random walk, we seek to find a local cluster S near the given basket node v that minimizes the cut conductance on the hypergraph defined below:

$$\Phi_c(S) = \frac{\operatorname{vol} \partial S}{\min\left\{\operatorname{vol} S, \operatorname{vol} \bar{S}\right\}} \tag{5}$$

where the numerator is the volume of the boundary with respect to S, and the denominator is the volume of the smaller side of the partition induced by S.

Let $p(u)$ ($u \in V$) denote the probability distribution of a random walk starting from node v. Following [12], we define

$$I(p,x) = \max_{\substack{\omega \in [0,1]^n \\ \omega(u)\frac{\pi(u)}{\sum \pi} = x}} \sum_{u \in V} \omega(u) p(u). \tag{6}$$

One can easily check that $I(p,0) = 0$ and $I(p,1) = 1$. As the distribution p approaches the stationary distribution, the curve $I(p,\cdot)$ approaches the straight line. Let $S_j(p)$ denote the set of j nodes maximizing $p(u)/\pi(u)$, and let $\lambda_j(q_t)$ denote its volume, i.e., $\lambda_j(p) = \operatorname{vol} S_j(p)$. Furthermore, denote $I_x(p,x)$ as the partial derivate of $I(p,x)$ with respect to x, we have

$$I_x(p,x) = \lim_{\delta \to 0} I_x(p, x-\delta) = \frac{p(\sigma(j))}{\pi(\sigma(j))}, \tag{7}$$

where $\sigma(j) = S_j(p) - S_{j-1}(p)$ is the permutation function, such that

$$\frac{p(\sigma(j))}{\pi(\sigma(j))} \geq \frac{p(\sigma(j+1))}{\pi(\sigma(j+1))}. \tag{8}$$

for all j. As $p(\sigma(j))/\pi(\sigma(j))$ is non-increasing, $I_x(p,x)$ is a non-increasing function in x and $I(p,x)$ is a concave function in x. $I(p,x)$ is used as one convergence measure and $I_x(p,x)$ characterizes the normalized probability mass.

Before introducing our proposed algorithm, let us first look at a few definitions. With slight abuse of notation, let $\boldsymbol{p}$ denote the $n \times 1$ vector whose elements are set to $p(u)$. First of all, let $[\boldsymbol{p}]_\epsilon$ be the truncation operator applied on $\boldsymbol{p}$, such that its u^{th} element $[\boldsymbol{p}]_\epsilon(u) = p(u)$ if and only if $p(u) \geq \pi(u)\epsilon$, where $\pi(u)$ is the stationary distribution at node u, and 0 otherwise. Second, following [12], we define

$$\begin{aligned} t_{last} &= (l+1)t_1 \\ \epsilon &= 1/(c_3(l+2)t_{last}2^b) \end{aligned} \tag{9}$$

where $t_1 = \left\lceil \frac{2}{\phi^2} \ln(c_1(l+2)\sqrt{(\operatorname{vol} V/2)}) \right\rceil$, $l = \lceil \log_2(\operatorname{vol} V/2) \rceil$, b is a positively integer governing the size of the output cluster, and c_1 and c_3 are constants with suggested values in [12].

Now we are ready to introduce our proposed *HyperGo* in Algorithm 1. Different from the *NIBBLE* algorithm proposed in [12], which works on regular graphs, *HyperGo* is designed to work on hypergraphs. It takes as input the hypergraph G, the seed basket $v \in V$, the upper bound ϕ on the conductance of the local cluster, and the positive integer b. The output is a set of basket nodes within the identified local cluster. In Steps 1, we compute t_{last} and ϵ using Equation 9. Next, in Step 2, we initialize $\boldsymbol{r}_0$ to be an $n \times 1$ indicator vector where only the element corresponding to the seed node is set to one. Steps 4 and 5 generate a sequence of vectors starting from $\boldsymbol{r}_0$ based on the following rule

$$\boldsymbol{q}_t = \begin{cases} \boldsymbol{r}_0, & \text{if } t = 0, \\ \boldsymbol{M}\boldsymbol{r}_{t-1}, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{r}_t = [\boldsymbol{q}_t]_\epsilon$, $t > 0$. That is, at each time step, we let the random walk proceed by one step from the current distribution and then round every $q_t(u)$ that is less than $\pi(u)\epsilon$ to 0. Notice that $\boldsymbol{q}_t$ and $\boldsymbol{r}_t$ are not necessarily probability vectors, as their components may sum up to less than 1. Then Step 7 finds the set $S_j(q_t)$ consisting of j nodes whose corresponding values $\frac{q_t(u)}{\pi(u)}$ are the largest. Step 8 determines whether this set satisfies three conditions: **C.1** in Step 9 guarantees that the output set has cut conductance at most ϕ; **C.2** in Step 10 ensures that it contains a good amount of volume (e.g., not too much and not too little); **C.3** in Step 11 guarantees that the output basket nodes have a large probability mass, where c_4 is a constant with suggested value in [12].

Algorithm 1 *HyperGo* Algorithm

Input: Hypergraph G, seed node v, conductance upper bound ϕ, positive integer b

Output: A local cluster S_v near the seed node v.

1: Compute t_{last} and ϵ using Equation (9).
2: Initialize $\boldsymbol{r}_0$ to be an $n \times 1$ all zero vector except for the element that corresponds to v, which is set to 1.
3: **for** $t = 1{:}t_{last}$ **do**
4: Set $\boldsymbol{q}_t = \boldsymbol{M}\boldsymbol{r}_{t-1}$
5: Set $\boldsymbol{r}_t = [\boldsymbol{q}_t]_\epsilon$.
6: **for** $j = 1 : n$ **do**
7: Let $S_j(q_t)$ denote the set of j nodes whose corresponding elements in q_t/π are the largest.
8: Return $S_j(q_t)$ as S_v if the following conditions are satisfied.
9: – **C.1**: $\Phi(S_j(q_t)) \leq \phi$.
10: – **C.2**: $2^b \leq \lambda_j(q_t) < \frac{5}{6}\operatorname{vol} V$.
11: – **C.3** $I_x(q_t, 2^b) \geq \frac{1}{c_4}(l+2)2^b$.
12: **end for**
13: **end for**
14: Return an empty set.

The following lemma shows the time complexity of the proposed *HyperGo*, which is largely controled by the size of the output cluster 2^b. It also depends on $\operatorname{vol} V$ in a polylogarithmic way.

Lemma 1. *[Time Complexity of* HyperGo*]* HyperGo *runs in time* $O(2^b \log^6(\mathrm{vol}\, V)/\phi^4)$.

Proof. Similar as the *NIBBLE* algorithm [12], it is easy to prove the monotonicity of multiplication by $\boldsymbol{M}$ defined in Equation 3, as well as the upper bound on the escaping mass of a t-step random walk with the volume of a node subset defined in Subsection 3.1. Therefore, following the same line of reasoning, we can prove that the time complexity of *HyperGo* is $O(2^b \log^6(\mathrm{vol}\, V)/\phi^4)$.■

Comparing *HyperGo* with *NIBBLE*, first of all, *NIBBLE* only works on regular graphs, whereas *HyperGo* is designed for hypergraphs. Therefore, *NIBBLE* cannot be applied to our proposed hypergraph representation of historical purchase and return records. Second, the running time of *NIBBLE* depends on the number of (pair-wise) edges in the graph, whereas *HyperGo* depends on the volume of the hypergraph $\mathrm{vol}\, V$. Notice that in hypergraphs, the volume of the hypergraph is not in proportion to the number of hyperedges, as different hyperedges may be incident with different number of nodes.

From Lemma 1, it can be seen that due to the linear dependence of the time complexity on the size of the output cluster, as well as the polylogarithmic dependence on the volume of the hypergraph, *HyperGo* is particularly suitable for large graphs, which is usually the case in E-tail product return prediction.

3.4 Dual-Level Return Prediction

Based on the output of Algorithm 1, next we propose a dual-level return prediction procedure, in order to provide the E-tailers with detailed information regarding the reason for a potential return. To be specific, we first predict if a basket contains products that are likely to be returned (basket-level prediction); then given such a basket, we predict which product(s) could be returned (product-level prediction).

3.4.1 Basket-level prediction. Let *BSK-dupe* denote the type of baskets that contain multiple similar products such as multiple sweaters in the same style, color or size; and *BSK-uniq* denote the type of baskets that only contain distinct products based on product hierarchies. Let $R(v)$ denote the return status of a basket $v \in V$, such that $R(v) = 1$ if basket v contains products that are to be returned, and $R(v) = 0$ otherwise. Suppose the ratio of the return probability between basket type *BSK-dupe* and type *BSK-uniq* is κ, i.e.,

$$\kappa = \frac{\Pr\{R(v) = 1 | v \in \textit{BSK-dupe}\}}{\Pr\{R(v) = 1 | v \in \textit{BSK-uniq}\}}, \tag{10}$$

where $\Pr\{\cdot|\cdot\}$ denotes the return probability corresponding to a specific basket type and can be estimated from historical data.

Given a basket $v \in V$, the algorithm proposed in Subsection 3.3 returns a cluster of baskets $S_v \subset V$ that are similar to the given basket v. Among all the baskets in the cluster, we assume that the return probability ratio of baskets in *BSK-dupe* and *BSK-uniq* is preserved , i.e.,

$$\frac{\Pr\{R(u) = 1 | u \in S_v, u \in \textit{BSK-dupe}\}}{\Pr\{R(u) = 1 | u \in S_v, u \in \textit{BSK-uniq}\}} = \kappa. \tag{11}$$

Assume the return status of baskets in *BSK-dupe* and *BSK-uniq* follows two Bernoulli processes $B(\rho_1)$ and $B(\rho_2)$, where ρ_1 and ρ_2 are the success probabilities of the two processes respectively. Suppose that there are N baskets in the returned cluster S_v, i.e., $|S_v| = N$; N_1 and N_2 baskets without returns from *BSK-dupe* and *BSK-uniq*, respectively. The following lemma shows how to estimate the two parameters ρ_1 and ρ_2 via maximum likelihood estimation.

Lemma 2. *Given two Bernoulli processes $B(\rho_1)$ and $B(\rho_2)$, assume that the ratio of the success probabilities is a constant, i.e., $\rho_1/\rho_2 = \kappa$. Among a total of N trials from the two Bernoulli processes, N_1 failures are observed from $B(\rho_1)$ and N_2 failures are observed from $B(\rho_2)$. The maximum likelihood estimate of the two parameters are*

$$\hat{\rho}_1 = \frac{1}{2}\left[f_1 + f_2\kappa - \sqrt{f_1^2 + f_2^2\kappa^2 + 2\kappa(2 - 2f_1 - 2f_2 + f_1 f_2)}\right], \tag{12}$$

and

$$\hat{\rho}_2 = \hat{\rho}_1/\kappa. \tag{13}$$

where $f_1 = 1 - \frac{N_1}{N}$ and $f_2 = 1 - \frac{N_2}{N}$.

Proof. Let N_0 denote the number of Bernoulli trials from $B(\rho_1)$. Given a total of N trails from the two Bernoulli processes, there are $N - N_0$ trials from the Bernoulli process $B(\rho_2)$. The likelihood function is given by

$$L = (1 - \rho_1)^{N_1} \rho_1^{N_0 - N_1} (1 - \rho_2)^{N_2} \rho_2^{N - N_0 - N_2}.$$

As $\rho_1 = \rho_2\kappa$, we have

$$L = (1 - \kappa\rho_2)^{N_1} (\kappa\rho_2)^{N_0 - N_1} (1 - \rho_2)^{N_2} \rho_2^{N - N_0 - N_2}.$$

Taking the logarithm of above likelihood function, we obtain

$$\begin{aligned} l &= N_1 \log(1 - \kappa\rho_2) + (N_0 - N_1)\log(\kappa\rho_2) \\ &+ N_2 \log(1 - \rho_2) + (N - N_0 - N_2)\log\rho_2 \end{aligned}$$

Let $\partial l/\partial\rho_2 = 0$, we obtain the maximum likelihood estimate of ρ_1 in Equation (12) and ρ_2 in Equation (13).■

According to Lemma 2, the estimated return probability of the given basket v is

$$\hat{\Pr}\{R(v) = 1\} = \begin{cases} \hat{\rho}_1 & \text{if } v \in \textit{BSK-dupe}; \\ \hat{\rho}_2 & \text{if } v \in \textit{BSK-uniq}. \end{cases} \tag{14}$$

3.4.2 Product-level prediction. For baskets that are predicted to be returned, we further propose a method to predict which specific products are likely to be returned. Let $R_g(g)$ denote the return status of a given product g, i.e., $R_g(g) = 1$ if returned, and $R_g(g) = 0$ otherwise. For a given product $g \in v$ and $v \in V$, let $S_v \subset V$ be the local cluster obtained based on the given basket v using Algorithm 1. The return probability of the product g is estimated by the fraction of baskets with product g returned out of all the baskets that are returned and contain the product g (for these baskets, the returned product may not be g), i.e.,

$$\hat{\Pr}\left\{R_g(g) = 1 \middle| R(v) = 1\right\} = \frac{\sum_{u \in S_v} \mathbb{I}_{R_g(g)=1, g \in u}}{\sum_{u \in S_v} \mathbb{I}_{R(u)=1, g \in u}}, \tag{15}$$

where $\mathbb{I}_X$ is an indicator function, which is equal to 1 if condition X is true, and 0 otherwise.

Based on the steps defined in the above two subsections, the marginal return probability of a product in a given basket $g \in v$ can be calculated as follows.

$$\hat{\Pr}\left\{R_g(g) = 1\right\} = \hat{\Pr}\left\{R_g(g) = 1 \middle| R(v) = 1\right\} \hat{\Pr}\left\{R(v) = 1\right\}. \tag{16}$$

Notice that in the above equation, we omitted $R(v) = 1$ from the left hand side, as $R_g(g) = 1$ indicates $R(v) = 1$, i.e., the return status of a basket is 1 if one of its products has been returned.

By first predicting returns at the basket-level and then at the product-level, the dual-level procedure can leverage the basket-level information, including but not limited to product interactions, into the prediction of each product in the baskets. As a result, despite that no detailed information of product attributes are included in the model, we are still able to predict the return intention at the product-level well, as will be shown in the Subection 4.3.3.

4 EXPERIMENTAL RESULTS

For this study, we collaborate with two leading omni-channel fashion retailers, one in North American and the other in Europe, to collect data and evaluate *HyperGo*. For e-tails in the fashion industry, product returns are a major challenge, where the return rate is often higher than 50% among all purchases[3]. There is a great potential to reduce return rate by taking proactive actions based on predicted product/basket return risk and thus improve the revenue margin.

4.1 Data Sets and Experiment Setup

We obtain about three months historical online purchase and return records from each retailer. The basic data information is summarized in Table 1. For confidential agreement, we name the two data sets as A and B. Data set A involves about 557 thousand products and 3.6 million baskets with return rate of 52% and 65%, respectively. Each product is a unique combination of style, color and size, i.e., the lowest level in the product hierarchies. Each basket is associated with one customer's online transaction and payment at a specific time. The average return rate of *BSK-dupe* that consists of duplicated products in same style, color or size is significantly higher than that of *BSK-uniq* wherein each product is distinct. Data set B shows lower average return rates at both basket and product levels than data set A. Similarly the average return rate of *BSK-dupe* is much higher than that of *BSK-uniq*.

The two E-tailers have a 45-day and 30-day return policy, respectively. In our experiments, we use all the online orders in the first seven days for training and model tuning. To be specific, among these orders we randomly sample 5% to select input parameters for all the methods. We use purchase orders in the subsequent two days as test cases and split them into 10 subsets to estimate variation of prediction performance. All the return orders within 45 (30) days of these purchases either by mail or in-store are included as true returns. This mimics online experiments on e-commerce platforms that process orders sequentially.

To comprehensively investigate the performance of *HyperGo* , we evaluate it using six measures. We use running time to characterize its efficiency. We compare its effectiveness based on precision, recall, $F_{0.5}$, receiver operating characteristic (ROC) curve, and the area under ROC curve (AUC) [8]. ROC curve illustrates the diagnostic ability of a binary classifier as its discrimination threshold is varied. AUC quantifies the overall quality of the ranking algorithm based on the predicted probabilities. $F_{0.5}$ weights precision higher than recall. In our application, precision reflects the correct rate of each target basket/product for proactive actions. In practice, every action for reducing return rate incurs cost and impacts customers' shopping experience, so precision is relatively more crucial. Given this consideration, we use $F_{0.5}$ for parameter tuning in all the experiments.

Table 1: Summary of e-tail data sets

Retailer	Metrics	Retrun Rate	Ave. Return Rate
A	Product	52%	
	Basket	65%	
	BSK-dupe		81%
	BSK-uniq		48%
B	Product	36%	
	Basket	38%	
	BSK-dupe		60%
	BSK-uniq		24%

4.2 Benchmark Methods

As far as we know, *HyperGo* is the first generic framework on return prediction for e-commerce. It leverages the return and purchase history of similar baskets and basket composition in a principled way. We benchmark its performance over three similarity-based approaches.

(1) *k-NN*: Given a basket, *k-NN* finds its k nearest neighbor baskets in the training set[4] and estimates its return probability by the fraction of total returned baskets out of k.
(2) *JacWght*: Given a basket, *JacWght* calculates its pairwise similarity with every training basket using Jaccard index[5] based on product sets in these baskets. Let $\boldsymbol{j}$ be a $t \times 1$ vector that denotes the simiarity coefficients, where t is the number of the training basket. Let $\boldsymbol{i}$ be a $t \times 1$ vector indicating the return status of each training baskets, where 1 is *return*, and 0 otherwise. *JacWght* estimates a basket's return probability by the average return status of the training baskets wighted by the corresponding pairwise similarity scores, i.e., $\frac{\boldsymbol{i}^T\boldsymbol{j}}{|\boldsymbol{j}|}$, where $|\cdot|$ denotes L1-norm.
(3) *JacNorm*: *JacNorm* estimates a basket's return probability in a similar way as *JacWght*, but it emphasizes the different return rates of *BSK-dupe* and *BSK-uniq*. To be specific, the return probability of *BSK-uniq* is estimated by $\frac{\boldsymbol{i}^T\boldsymbol{K}\boldsymbol{j}}{|\boldsymbol{j}|}$, and *BSK-dupe* by $\frac{\kappa\boldsymbol{i}^T\boldsymbol{K}\boldsymbol{j}}{|\boldsymbol{j}|}$, where κ is defined in Equation (10), and $\boldsymbol{K}$ is a $t \times t$ diagonal matrix with diagonal values 1 or $1/\kappa$ when the corresponding basket belongs to *BSK-uniq* and *BSK-dupe*, respectively.

Note that all the benchmark methods are based on pairwise similarities. For return prediction at the product-level, we first apply the above methods to find similar baskets and calculate basket-level return probabilities. Then we follow the same steps proposed in Subsection 3.4.2 to calculate product-level conditional return

[3] http://www.retourenforschung.de/

[4] http://scikit-learn.org/stable/modules/neighbors.html

[5] https://en.wikipedia.org/wiki/Jaccard_index

probabilities for all the benchmark methods. Finally we obtain the marginal product-level return probabilities for each benchmark method by multiply the conditional return probabilities and the corresponding basket-level return probabilities as in Equation 16.

4.3 Comparison Results

In this section, we present the comparison results at the basket-level as well as the product-level. At the product-level, we present the results based on the conditional probability given basket with return and the marginal probability of a returned product.

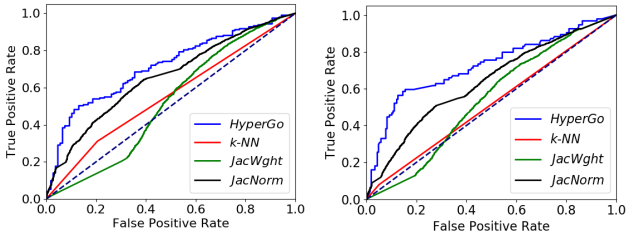
4.3.1 Basket-level Return Prediction. Table 2 summarizes the mean and standard deviation of multiple performance metrics for *HyperGo* and all the benchmark methods. The **bold** numbers highlight the best performers. For both data sets, *HyperGo* outperforms all other methods in terms of mean AUC, $F_{0.5}$, and precision scores. In general, the standard deviation of the performance scores of different methods is small and comparable. These results show that *HyperGo* can improve the prediction performance by more effectively modeling the rich information of historical purchase and return records at the basket-level than pair-wise based methods. In addition, *JacNorm* is the second best method as both *HyperGo* and *JacNorm* leverage the prior knowledge that the return rate of baskets containing duplicated products is much higher than that of baskets with only distinct products. This validates the benefit of differentiating return rates of different types of baskets in the dual-level prediction, as discussed in Subsection 3.4.1.

Fgiure 2 compares the ROC curves across all the methods and again shows the significantly better performance of *HyperGo* than the others on both data sets. In addition, the ROC curves of *k-NN* approximately overlap with the diagonal lines. This implies that the prediction power of *k-NN* is very low and the high recall value of *k-NN* is achieved when it predicts all the baskets as return. In this case, precision values are converging to the average return rate of baskets in the test data. Furthermore, all the methods perform relatively better on data set A than data set B. As the average return rate of data set A at the basket-level is higher than that of data set B.

Table 2: Comparison results at the basket-level: *HyperGo* outperforms all the competitors.

Retailer	Method	AUC	Precision	Recall	$F_{0.5}$
A	*HyperGo*	**.70** (.02)	**.81** (.04)	.57 (.07)	**.74** (.02)
	k-NN	.55 (.02)	.61 (.00)	**1.00** (.00)	.66 (.00)
	JacWght	.52 (.01)	.64 (.00)	.89 (.02)	.68 (.00)
	JacNorm	.65 (.01)	.72 (.01)	.64 (.01)	.70 (.01)
B	*HyperGo*	**.70** (.03)	**.68** (.03)	.37 (.05)	**.58** (.04)
	k-NN	.52 (.00)	.32 (.01)	**1.00** (.00)	.37 (.01)
	JacWght	.54 (.01)	.36 (.01)	.72 (.05)	.40 (.01)
	JacNorm	.64 (.00)	.49 (.02)	.43 (.04)	.48 (.01)

4.3.2 Product-level Return Prediction Given Basket with Returns. In this experiment, we evaluate the prediction performance of the proposed *HyperGo* framework at the product-level only on baskets



(a) Data set A **(b) Data set B**

Figure 2: ROC at the basket-level: *HyperGo* achieves greater performance improvement over other methods on data set B, which has a lower return rate than data set A.

with returns in the test sets. The average product return rate for such baskets is about 0.7 for both data sets. Given a basket, we obtain the basket-level clusters or similarity scores in the same way as the previous experiments for all the methods. Next we follow Equation (15) in Subsection 3.4.2 to calculate the conditional return probability for products within the basket. We report mean and standard deviation of all the performance metrics in Table 3. It shows that *HyperGo* obtains the largest AUC, precision and $F_{0.5}$ scores. All the benchmark methods report mean recall values 1 with standard deviation 0. Given the high product-level return rate for baskets with return, the high recall values imply that benchmark methods obtain their best performance by predicting every product to be returned. Based on the high precision, recall and $F_{0.5}$ scores, the overall performance of all the methods in this experiment setting is reasonable. This provides the foundation to directly predict return at the product-level without knowing the return status of the associated basket, as will be presented in the next subsection.

Table 3: Comparison results at the product-level given baskets with returns: *HyperGo* is the best performer.

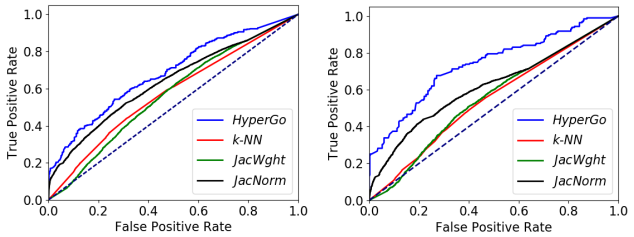
Retailer	Method	AUC	Precision	Recall	$F_{0.5}$
A	*HyperGo*	**.61** (.02)	**.77** (.02)	.98 (.02)	**.81** (.01)
	k-NN	.55 (.01)	.75 (.01)	**1.00** (.00)	.79 (.01)
	JacWght	.56 (.01)	.75 (.01)	**1.00** (.00)	.79 (.01)
	JacNorm	.61 (.01)	.75 (.01)	**1.00** (.00)	.79 (.01)
B	*HyperGo*	**.63** (.02)	**.76** (.01)	.98 (.01)	**.79** (.01)
	k-NN	.54 (.01)	.73 (.01)	**1.00** (.00)	.77 (.01)
	JacWght	.54 (.01)	.73 (.01)	**1.00** (.00)	.77 (.01)
	JacNorm	.58 (.01)	.73 (.01)	**1.00** (.00)	.77 (.01)

4.3.3 Product-level Return Prediction Among All Baskets. In this experiment, we compare the performance of the *HyperGo* framework with other methods at the product-level based on the marginal return probability given by Equation (16) in Subection 3.4.2. Table 4 and Figure 3 present the results. From Table 4, we have following observations. First, *HyperGo* obtains significant higher mean AUC, precision and $F_{0.5}$ scores than all the three benchmark methods. Second, the standard deviation of the scores of all the

methods are small and approximately the same. Third, *JacNorm* is the second best performer. Moreover, ROC curves in Figure 3 show that *HyperGo* achieves the best overall performance in discriminating returns at the product-level, and the ROC curves of *k-NN* and *JacWght* almost overlap with the diagonal lines, i.e., the random guess classifiers. These results are consistent with what we observe at the basket-level. Hence, this experiment again validates the effectiveness of the *HyperGo* framework for improving the prediction performance by leveraging the return and purchase history of similar baskets and basket composition. Furthermore, based on Table 4 and Table 2, the prediction performance of all the methods at the product-level is slightly worse than that at the basket-level. This is because identifying specific products to be returned within a basket is more challenging than classifying a basket with at least one product.

Table 4: Comparison result at the product-level: *HyperGo* outperforms all the benchmark methods.

Retailer	Method	AUC	Precision	Recall	$F_{0.5}$
A	*HyperGo*	**.67** (.02)	**.69** (.03)	.55 (.05)	**.65** (.02)
	k-NN	.55 (.01)	.55 (.01)	**.78** (.06)	.58 (.01)
	JacWght	.56 (.00)	.56 (.01)	.77 (.02)	.59 (.01)
	JacNorm	.64 (.00)	.64 (.02)	.53 (.05)	.61 (.01)
B	*HyperGo*	**.72** (.02)	**.65** (.06)	.35 (.07)	**.53** (.04)
	k-NN	.55 (.01)	.36 (.02)	**.57** (.07)	.38 (.01)
	JacWght	.55 (.01)	.36 (.01)	**.57** (.03)	.39 (.01)
	JacNorm	.62 (.01)	.52 (.01)	.36 (.03)	.48 (.01)



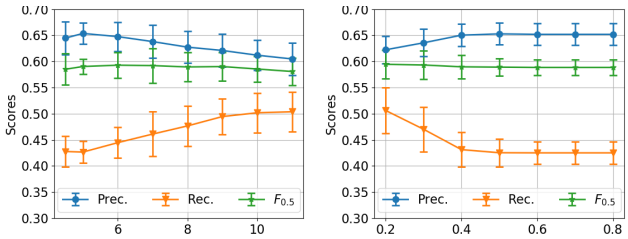
(a) Data set A **(b) Data set B**

Figure 3: ROC at the product-level based on marginal return probability: *HyperGo* performs better than other methods.

4.4 Parameter Sensitivity Analysis

In this section, we evaluate the impact of input parameters on the performance of *HyperGo* at the product-level. We sample 5% baskets from data set A as the evaluation set and the rest for building the hyper graph. We change the input parameters (b, ϕ) in a range of values and calculate the corresponding precision, recall and $F_{0.5}$ scores. We repeat the sampling process 10 times and report the mean and standard deviation of above scores as error-bar plots in Figure 4.

Figure 4(a) shows that as b increases, precision first increases, then decreases gradually. Recall displays an opposite trend. The $F_{0.5}$ first increases and obtains the maximum around $b = 6$, and then starts to decrease. This is because b controls the size of the output cluster, and a larger b leads to larger output clusters. When the output cluster size is small, the amount of baskets used for dual-level prediction is not sufficient for an accurate prediction. As the cluster size increases, more noisy baskets are included for the prediction. This leads to slightly worse performance scores. However, $F_{0.5}$ changes very little. From Figure 4(b) we observe that as ϕ increases, precision increases and then approaches a constant, recall decreases and becomes a constant, and $F_{0.5}$ decreases slightly. As ϕ is the upper bound of the cut conductance on the hypergraph, it controls the quality of the output cluster. When ϕ is too large, it loses the impact on the algorithm. On the contrary, when ϕ is below a threshold and further decreases, the quality of the output clusters increases. However, this usually leads to output clusters in larger size, which offsets the impact of increased cluster quality. In summary, Figure 4 shows that the *HyperGo* framework is robust to small perturbation of the input parameters b and ϕ.



(a) b **(b)** ϕ

Figure 4: Parameter sensitivity analysis: *HyperGo* is robust to small perturbation of the input parameters.

4.5 Time Complexity

To empirically evaluate the computational complexity of *HyperGo* we record the CPU time (seconds) and the number of nodes in the output clusters in the experiments to evaluate the basket-level prediction performance based on data set A. The experiments have been performed on a distributed computing environment with 12 Intel Xeon(R) CPU processors (2.30GHz, 8 cores each processor) and a total of 512 GB of RAM, equipped with Red Hat Enterprise Linux 7 operating system. The code has been executed using 64-bit Python 3.6. As shown in Figure 5, the CPU time scales linearly with respect to the number of nodes in the returned cluster. This demonstrates that *HyperGo* is efficient for large scale applications, which is consistent with our theoretical analysis based on Lemma 1 in Subsection 3.3.

5 CONCLUSION AND FUTURE DIRECTIONS

Motivated by e-tail applications where a high return rate directly causes increased costs, we propose a generic framework named *HyperGo* for predicting product returns, which consists of a novel hypergarph representaiton, a local graph cut algorithm, and a dual-level return prediction model based on the output cluster from the

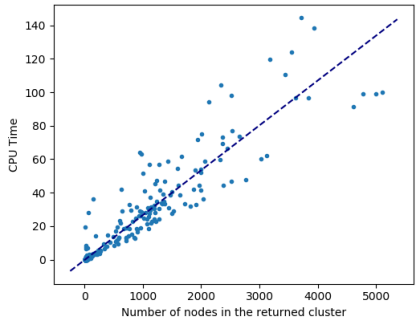


Figure 5: Time complexity: CPU time scales linearly with respect to the size of the output clusters.

local algorithm. Compared with the limited existing work, the proposed hypergraph representation is able to leverage the rich information of basket composition in a more effective way. Based on this representation, the truncated random-walk-based local graph cut algorithm identifies similar historical baskets. Given these baskets, *HyperGo* estimates the return intention on two levels: basket-level vs. product-level, which provides detailed information regarding the reason for a potential return. The main benefit of the proposed algorithm lies in the time complexity, which depends linearly on the size of the output cluster and polylogarithmically on the volumn of the hypergraph, making it particularly suitable to be applied to large amount of historical purchase and return records. Experimental results on multiple real-world e-tail data sets demonstrate the effectiveness and efficiency of *HyperGo*. To further improve the prediction performance, in particular, at the product-level, detailed product attributes, customer information, as well as contextual characteristics such as location, weather and events can be incorporated into the prediction framework, which is our on-going work.

Many online retailers report that if they could achieve a 10% reduction in the rate of product returns, their profitability would increase by more than 20% [9]. The *HyperGo* framework is designed to work **before** the customers have made the purchase. It enables e-tailers to take proactive measures to decrease the return intention, potentially leading to a changed basket composition. A demo for fashion e-tails has been built based on the *HyperGo* framework and several possible intervention measures, such as popping up a chatbot to provide guidance regarding size and fit, offering discount coupons and showing backorder on the products that contribute to the high predicted return intention. Other measures can be designed and implemented based on specific business scenarios (e.g., fashion vs. grocery). In practice, whether and to what extent to intervene customer's online shopping journey based on prediction should be implemented is essentially a business strategy. The success of such a strategy depends not only on the accuracy of the prediction, but also the effectiveness of the intervention measures and customer acceptance. We believe that online A/B testing on the combination of the *HyperGo* framework with different preventive measures can collect more insights and evaluate customer acceptance. In addition, to embed the *HyperGo* framework to e-tailing, it needs be connected to the retailer's order processing system, which could incur a third party's service.

6 ACKNOWLEDGMENT

This work is supported by National Science Foundation under Grant No. IIS-1552654, and Grant No. CNS-1629888, the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001, and an IBM Faculty Award. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Sameer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman, and Serge Belongie. 2005. Beyond pairwise clustering. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. 838–845.

[2] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science*. 475–486.

[3] Reid Andersen, Fan Chung, and Kevin Lang. 2007. Local Partitioning for Directed Graphs Using PageRank. In *Proceedings of International Workshop on Algorithms and Models for the Web-Graph*, Anthony Bonato and Fan R. K. Chung (Eds.), Vol. 4863. Springer, 166–178.

[4] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2013. Random walks in hypergraph. In *Proceedings of the 2013 International Conference on Applied Mathemtics and Computational Methods*.

[5] Peter Grimaldi. 2008. Day of rejects: store clerks gear up for gift exchanges. *Tribune Business News* (Decemeber 26 2008). Washington.

[6] Lei Li and Tao Li. 2013. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*. 305–314.

[7] Jungmok Ma and Harrison M. Kim. 2016. Predictive model selection for forecasting product returns. *Journal of Mechanical Design* (2016).

[8] James W. Perry, Allen Kent, and Madeline M. Berry. 1955. Machine literature searching X. macine language; factors underlying its design and development. *American Documentation* 6, 4 (1955). https://doi.org/doi:10.1002/asi.5090060411

[9] Sabine Pur, Ernst Stahl, Michael Wittmann, Georg Wittmann, and Stefan Weinfurter. 2013. *Retourenmanagement im online handel-das beste daraus machen*. Technical Report. Regensburg: ibi research an der universität Regensburg Gmbh.

[10] Juan Alberto Rodriguez. 2003. On the laplacian spectrum and walk-regular hypergraphs. *Linear and Multilinear Algebra* 51, 3 (2003), 285–297.

[11] Sai Nageswar Satchidanand, Harini Ananthapadmanaban, and Balaraman Ravindran. 2015. Extended discriminative random walk: a hypergraph approach to multi-view multi-relational transductive learning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[12] Daniel A. Spielman and Shang-Hua Teng. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal of Computation* 42 (2013), 1–26.

[13] Hung-Khoon Tan, Chong-Wah Ngo, and Xiao Wu. 2008. Modeling video hyperlinks with hypergraph for web video reranking. In *Proceedings of the 16th ACM International Conference on Multimedia*.

[14] Beril Toktay. 2004. *Business Aspects of Closed Loop Supply Chains*. Carnegie Mellon University Press, Chapter Forecasting Product Returns, 203–209.

[15] Patrick Urbanke, Johann Kranz, and Lutz Kolbe. 2015. Predicting product returns in e-commerce: the contribution of mahalanobis feature extraction?. In *Proceedings of the 36th International Conference on Information Systems (ICIS)*.

[16] Hongxia Yang, Yada Zhu, and Jingrui He. 2017. Local algorithm for user action prediction towards display ads. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2091–2099.

[17] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*. 1601–1608.

[18] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM '08)*. Springer-Verlag, Berlin, Heidelberg, 337–348. https://doi.org/10.1007/978-3-540-68880-8_32

[19] J. Zien, M. Schlag, and P. Chan. 1999. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on computer-aided design of integrated circuits and systems* (1999).