IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 25, 2017, accepted April 3, 2017, date of publication April 17, 2017, date of current version June 7, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2694490

Query-Based-Learning Genetic Algorithm to Construct Mobile-Oriented Catalogs in M-Commerce

HUNG-MIN HSU[1,2], RAY-I CHANG[1], (Member, IEEE), and JAN-MING HO[3], (Senior Member, IEEE)

[1]National Taiwan University, Taipei 10617, Taiwan
[2]Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan
[3]Institute of Information Science and Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan

Corresponding author: Hung-Min Hsu (tsbear1@gmail.com)

This work was supported in part by the Academia Sinica Digital Center: System Management and Content Retrieval Technologies for Supporting Cloud-based Digital Archive Systems and Services Projects through the Academia Sinica, and Ministry of Science and Technology, Taiwan, under Grant 105-2410-H-002-099-MY3.

ABSTRACT This paper seeks to optimize customer attraction for mobile apps in m-commerce. We model this problem as a mobile-oriented catalog (MOC) segmentation problem. We use query-based learning (QBL) to develop MOCs and aim to attract most number of customers through minimal number of MOCs. This paper illustrates how to design attractive MOCs to recommend items by using QBL genetic algorithm (QBLGA). We propose preference modeling, Product2Vec, Transaction2Vec, and their variations as our oracles of QBLGA. These oracles can aggregate similar purchasing experiences to optimize combination of products for MOC construction. We divide these oracles into major oracle and minor oracles, and then, QBLGA uses these two types of oracles to produce high-attractive products. Experiments show that QBLGA outperforms the state-of-the-art algorithm to attract the greatest number of customers.

INDEX TERMS M-commerce, Product2Vec, Transaction2Vec, mobile-oriented catalog, query-based-learning genetic algorithm

I. INTRODUCTION

Among mobile-commerce (m-commerce) service innovations, mobile commerce has emerged as a business model with significant potential and opportunity to monetize the one-quarter of the global population who own or use smartphones [1]. However, a better understanding is needed of the impact of barriers inherent in mobile device such as small mobile device screens, time consuming processes, difficulty completing transactions, difficulty visualizing product attributes, and the lack of a secure medium [2]. Therefore, mobile shopping applications should be designed to further simplify access to product information as compared to their desktop counterparts. That is to say, the traditional browsing model of desktop internet usage does not fit well in mobile shopping applications. Mobile application must prioritize the provision of highly relevant content, with relevance determined by personal preferences as revealed through the use of using data mining techniques.

This paper aims to optimize customer attraction for mobile shopping application to address obstacles due to small mobile device screens and time-consuming processes. The idea is to use customer preference information to minimize the number of operations required to present customers with optimally attractive products, thus minimizing the number of input actions (e.g., tapping and swiping) the customer must perform.

We model this problem as a mobile-oriented catalog (MOC) segmentation problem. The concept of MOCs is similar to that of product lines in that companies produce multiple catalogs to maximize customer attraction. MOCs have been shown to be an effective tool for presenting products [3] and the contents of each catalog can be customized to appeal to the preferences of various customer segments. The main goal of this paper is to use minimal number of MOCs to interest the most number of customers. Therefore, the number of operations can be reduced through minimal MOCs.

The MOC segmentation task is to build group recommendation model (i.e., a classifier) capable of optimizing these catalogs to cover most number of customers. In typical applications of data mining and machine learning

algorithms (e.g., greedy algorithm, associated based algorithm and genetic algorithm) to catalog segmentation, group recommendation models are using transaction information and random strategy to optimize catalogs since the MOC segmentation problem is NP-Complete [4]. Therefore, their success usually depends on the quality of the product selection set. If the product selection set contains extraneous and irrelevant product candidates, data mining and machine learning algorithms may trap in local minima. To address this shortcoming, we apply the Query-Based-Learning (QBL) concept to discover high-attractive product as oracle. In our previous research, we applied the topic modeling to develop Latent Dirichlet Allocation based self-adaptive genetic algorithm (LDASAGA) [3]. Topic modeling is used to discover a set of ''preferences'' from the transaction database depending on the distribution of the products in the transactions. In this study, we explore the QBL concept to develop a novel MOC construction system. We designed three main types of oracle which are preference modeling, Product2Vec and Transaction2Vec in the high-attractive product discovering loop. The oracle can actively and repeatedly add high-attractive products into MOC for higher covered customers.

The main contributions of this paper are as follows. (1) We formally propose a framework to optimize the customer attraction for m-commerce applications based on a MOC segmentation problem. (2) We present a QBL Genetic Algorithm (QBLGA) to optimize the design interface of m-commerce applications.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. In Section 3, we define and formulate MOC segmentation problem. In Section 4, we describe the framework of our methodology and provide a detailed description of the proposed algorithms. Section 5 compares performance of the proposed algorithms and reviews the experimental results. Finally, we draw conclusion in Section 6.

II. RELATED WORKS

Many studies have examined issues related to interface design for mobile commerce. Lee and Benbasat [5] used Rayport and Jaworski's 7 Cs (context, content, community, customization, communication, connection, and commerce) [6] and identified 2 Ms (mobile setting and mobile device constraints) to serve as the basis for a new framework for mobile commerce interfaces. Persson and Berndtsson [7] found a positive relationship between intention to shop through smartphones and self-reported past smartphone shopping habits. Xu *et al.* [8] presented our work on the design and evaluation of vision-based mobile interfaces for shopping in physical stores. Most analyses approach related issues from the human-computer interaction (HCI) perspective.

Moreover, some studies focused on using clickstream to cluster interest patterns [9], [10] to provide significant assistances on webpage optimization and personalized recommendation. In this paper, we model interface design for mobile commerce as a MOC segmentation problem [3] which requires MOCs to feature recommendation systems and minimize user operations.

The MOC segmentation problem arises from catalog segmentation. In 1998, Kleinberg *et al.* [11] proposed catalog segmentation as a means of optimizing product recommendations for customers. It was later proved to be NP-Complete [4]. Xu *et al.* [12] used semi-definite programming techniques to solve a 2-catalog segmentation problem. Steinbach *et al.* [13] proposed Indirect Catalog Creation (ICC), Direct Catalog Creation (DCC) and Hybrid Catalog Creation (HCC) to create promotional catalogs. The Customer-Oriented Catalog (COC) segmentation problem is addressed by Ester *et al.* [14] who found that the covered customer must buy a threshold number of products from a catalog to be covered by that catalog. COC designs k catalogs, in which each catalog is represented as a set of products. If the customer is interested in at least t (threshold) products in the catalog, the customer is assigned to the catalog. In other words, each catalog can be referred as a cluster of products. The assigned customer is called a 'covered' customer and each covered customer belongs to only one catalog which contains the largest number of products of interest to him/her. Amiri [15] used a greedy algorithm and associated based algorithm to yield COC. Mahdavi *et al.* [16] proposed the Self-Adaptive Genetic Algorithm (SAGA) to solve the COC problem in e-commerce (e-COC). They suggested that each catalog should be weighted individually according to the order in which they are presented to the user and SAGA was used to cluster customers into various catalogs. SAGA encodes the e-COC problem as a single chromosome input.

III. PROBLEM DEFINITION AND FORMULATION

The MOC segmentation problem is concerned with clustering customers into appropriate catalogs. Customer preferences can be extracted by aggregating their transaction records. Assume that each catalog consists of n products. If a customer is interested in at least t products in a catalog, we say that this catalog *covers* the customer. The goal of the MOC segmentation problem is to optimize *revenue* by maximizing the number of individual customers covered by these catalogs. Each covered customer is only counted once. Let $C = \{c_1, c_2, c_3, \ldots.\}$ be the set of all customers and $P = \{p_1, p_2, p_3, \ldots.\}$ be the set of products in the database. The retailer's APP features multiple webpages for product display. $L = \{l_1, l_2, l_3, \ldots\}$ is the set of layers and there are many screens $S_l = \{s_{l,1}, s_{l,2}, s_{l,3}, \ldots.\}$ in the l-th layer. The s-th screen in the l-th layer is composed of catalogs and is expressed as $K_{l,s} = \{k_{l,s,1}, k_{l,s,2}, k_{l,s,3}, \ldots.\}$. Each catalog $k_{l,s,k}$ consists of n products. Each screen size of layer l can be represented by $I_l = \{|K_{l,s}||\forall s \in S_l\}$. Thus, mobile APP interface design is expressed as $\mathrm{MI} = \{I_l|\forall \mathrm{l} \in \mathrm{L}\}$. Table 1 lists the parameter notations for MOC including dataset variables and control variables. Customers and product information is obtained from the dataset, while catalog size and threshold are control variables. Fig. 1 and Fig. 2 illustrate the concept of layer and screen. This mobile application interface

TABLE 1. Notation of MOC parameters.

Data variable	Meaning
C	set of customers
P	set of products
$K_{l,s}$	set of catalogs in screen s in layer l
S_l	set of screens in layer l
L	set of layers
Control variable	
n	catalog size
t	minimum customer interest threshold

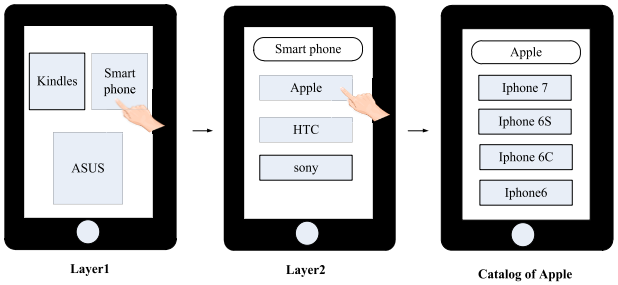

FIGURE 1. Layer concept: Layer 1 features two catalogs (Kindle and ASUS), along with a directory to layer 2 called "Smart Phone". When the user taps smart phone icon, the APP will show three catalogs (i.e, Apple, HTC and Sony). Then user taps the Apple icon to launch the Apply product catalog.

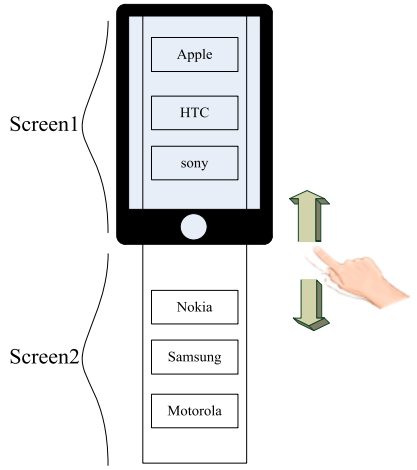

FIGURE 2. Screen concept: Assume there are a total of six catalogs in Layer2, but only three can be displayed at one time due to the small screen size. Thus the user has to swipe to see additional catalogs.

design example from Fig. 1 and Fig. 2 can be expressed by $\mathrm{MI} = \{I_1, I_2\} = \{\{2\}, \{3, 3\}\}$.

In the MOC segmentation problem, the foremost catalog is assigned a higher priority since customers will not be bothered to search through multiple screens for a product. Thus, we give the catalog of the first layer and the first screen the heaviest weight. The subsequent catalogs are assigned weights in descending order. Therefore, we assume $W_l = |L| - l + 1$ for the layer weight and $W_s = Max(|S_l|) - s + 1$ for the screen weight. We use the largest screen size from all layers to normalize the screen weight in each layer. Thus, the weight of the catalog in screen s of layer l equals $W_s * W_l$. In this design, the weight of the last screen of the first layer might be lower than the first screen of second layer. This is reasonable approach since swiping the screen five times is slower than tapping to access the second layer. Our object function is designed by commission. Commission means the remuneration paid to m-commerce platform provider. Therefore, the m-commerce platform provider can optimize the MOCs to maximize revenue. We define two kinds of commission for the m-commerce platform: commission of screen and commission of layer. CO_s denotes the commission of screen which means the m-commerce platform provider's transaction fees in the screen level, by contrast, CO_l is the commission of layer. *fee* is the unit price of commission. The commission of screen and commission of layer are

$$\mathrm{CO}_s = W_s * fee \tag{1}$$
$$CO_l = W_l * fee \tag{2}$$

Therefore, the commission for each catalog can be calculated by $CO_s * CO_l$.

The following notation is used in the MOC segmentation problem:

$$\rho_c \mathrm{P} = \begin{cases} 1 & If\ customer\ c\ has\ interest\ in\ product\ p \\ 0 & Otherwise \end{cases} \tag{3}$$

The decision variables are as follows.

$$X_{l,s,k,c} = \begin{cases} 1 & If\ customer\ c\ is\ covered\ by\ catalog\ k \\ & \quad in\ screen\ s\ in\ layer\ l \\ 0 & Otherwise \end{cases} \tag{4}$$

$$Y_{l,s,k,p} = \begin{cases} 1 & If\ the\ catalog\ k\ in\ screen\ s\ in\ layer\ l \\ & \quad includes\ product\ p \\ 0 & Otherwise \end{cases} \tag{5}$$

$$revenue = \sum_{l \in L} \sum_{s \in S} \left(\left(\sum_{k \in K_s} \sum_{c \in C} \sum_{p \in P} \rho_c \mathrm{P} * Y_{l,s,k,p} \right) * CO_s \right) * CO_l \tag{6}$$

The decision variables are subject to

$$\sum_{p \in P} Y_{l,s,k,p} = n, \quad \forall k \in K_{l,s},\ s \in S_l,\ l \in L \tag{7}$$

$$\mathrm{t} * X_{l,s,k,c} \le \sum_{p \in P} \rho_c P * Y_{l,s,k,p}, \quad \forall c \in C,\ k \in K_{l,s},\ s \in S_l,\ l \in L \tag{8}$$

$$\sum_{l \in L} \sum_{s \in S} \sum_{k \in K_{l,s}} X_{l,s,k,c} \le 1, \quad \forall c \in C,\ k \in K_{l,s},\ s \in S_l,\ l \in L \tag{9}$$

$$X_{l,s,k,c}, Y_{l,s,k,c} \in \{0, 1\}, \quad \forall c \in C,\ p \in P,\ k \in K_{l,s},\ s \in S_l,\ l \in L \tag{10}$$

The following notation is used in the MOC problem and our goal is to optimize the *revenue* as in Eq.(6). Eq.(7) ensures that each catalog k in screen s in layer l includes n products. Eq.(8) guarantees that a catalog $k \in K_{l,s}$ covers a customer when the customer is interested in at least t products in the catalog. All of the control variables can be expressed in the APP interface except that the threshold is used to ensure that the customer had purchased at least t products to qualify as a covered customer. Eq.(9) ensures that a customer is covered by only one catalog. Each covered customer can only be covered by one catalog to ensure that the algorithm covers more customers. Eq.(10) must be satisfied to ensure integrality for the decision variables.

IV. METHODOLOGY

The MOC segmentation problem is NP-Complete because it is reduced from a well-known NP-Complete *Set Cover* problem [17]. Thus, the proposed method only provides good feasible solutions. In this paper, we propose QBLGA to deal with the MOC segmentation problem. Traditional learning scenarios exploit observed data to formulate a model. In contrast, QBL uses active learning to process data. QBL is framed as a dialogue between a teacher and learner [18] through which learner can be trained effectively and efficiently since teacher teaches students in accordance with their aptitude. Teacher in the QBL framework is named the *oracle* which is a data source designed to help the system learn correctly. The oracle can also take the form of a natural system, artificial simulation, mathematical equation or expert experience. There are many researches adopting QBL in their methods to obtain excellent performance [19]. In this section, we firstly illustrate the framework of QBLGA to solve MOC segmentation problem. After that, we will describe oracles of QBLGA.

A. QBLGA

We introduce the QBL framework and propose different customer preference generation approaches to produce the oracle of the QBL framework.

Before turning to illustrate how to use QBLGA to construct MOCs, we must draw attention to explain SAGA. The process of SAGA [16] is as follows.

1) INITIALIZE POPULATION

Initialize population: For randomly generated products, QBLGA computes the fitness of each chromosome. We generate chromosomes randomly and then reserve chromosomes with the top r *revenue* in the original population. Fig. 3 illustrates an example for representing solution of a MOC. The APP interface includes two layers, each of which has a hierarchical structure which contains a screen, catalog and products. The size of each chromosome is the number of catalogs multiplied by the catalog size (n = 2).

2) CHOOSE PARENT STRATEGY

A binary tournament in which two chromosomes are chosen randomly; the chromosome with the better *revenue* is retained as one parent and the process is repeated to produce the other parent.

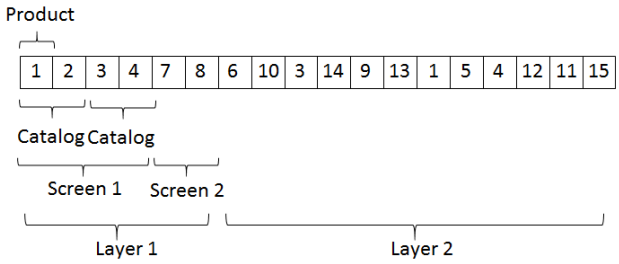

FIGURE 3. A sample chromosome of a MOC.

3) MUTATION

This operation aims to obtain a better combination by replacing m products in each catalog of a chromosome. These replaced products are selected from the products with m lowest *popularity* in a catalog and the altered products are generated randomly from a set of eligible products with the highest *goodness*. *popularity* is the number of customers covered in a given catalog who are interested in the product and *goodness* means the number of still uncovered customers who are interested in the product [16]. The parameter m is dynamic. If *revenue* does not improve in this iteration, m will be reduced by one in the next iteration until m equals to zero, then m returns to the default value. When product duplication occurs through mutation, QBLGA alters products (genes) with m lowest popularity in a catalog by applying the products from the oracle.

4) CROSSOVER

After selecting two chromosomes as parents, each catalog of the pair of parent chromosomes generate a number x randomly from *1* to *n-1*, and switch the products from position 0 to x for the two chromosomes. In the crossover stage, we refresh r chromosomes of the chromosome population. QBLGA will eliminate redundant products (genes) in a catalog by applying the products from the oracle.

5) REPRODUCTION

This operation reserves r chromosomes which have r best fitness.

Let us now attempt to extend SAGA into the idea of QBL. In this paper, we apply QBL into SAGA to develop QBLGA. The emphasis is on developing the interaction between SAGA and oracle (see Fig. 4 for an illustration of the QBLGA framework).

In our QBL, there are two types of oracles. One is major oracle which means the high complexity solution. In contrast, the other one is minor oracle with lower complexity. However, major oracle can also adopt high complexity solution but minor oracle only can be lower complexity solution. The QBL remains minor oracle through a specific number of epochs then switches to major oracle. Once major oracle finds better

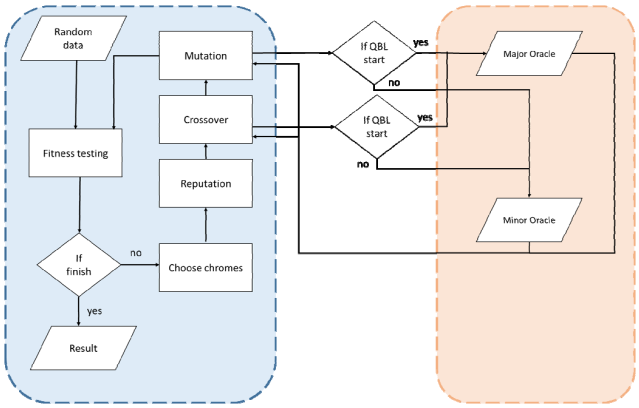

FIGURE 4. Workflow of QBLGA.

solution, the QBL switches to minor oracle immediately. Otherwise, the QBL remains major oracle. After a specific number of epochs, the QBL will switch to minor oracle. The advantage of our QBL is to possess a chance to query high complexity solution and still have an opportunity to apply another oracle. Multiple oracles have a beneficial effect on QBLGA as a consequence of approaching the optimal solution. Detailed account of QBLGA has already discussed, and the procedure of using QBLGA to construct MOCs is given in TABLE 2.

B. ORACLES OF QBLGA

Oracle plays a vital role in improving QBLGA performance. The purpose here is to explore a little further into oracle. In this paper, we consider preference modeling, Product2Vec, Transatiocn2Vec and their variations to solve MOCs problem. It was mentioned in the preceding section that our QBL has major and minor two types of oracle. This is a question to be considered later.

1) PREFERENCE MODELING

We will begin by considering preference modeling as oracle. LDA is a well-known topic modeling algorithm [20]. It analyzes documents and clusters words based on topic through three hierarchy layers: topic, document and word. In our previous work [3], we exploited the concept of LDA to establish LDASAGA which refers to the document as a transaction and the product as a word; in other words, we refer topics as preferences. Besides LDASGA, we proposed two novel preference modeling algorithms as oracles. I shall have more to say about these three preference modeling algorithms later on. Let us look deeper into preference modeling that illustrates the point that we have been considering. According to LDA, the probability model of oracle can be written as follows.

$$p_i|z_i, \emptyset^{(z_i)} \sim Discrete(\emptyset^{(z_i)}) \tag{11}$$

$$\emptyset \sim Dirichlet(\beta) \tag{12}$$

$$z_i|\theta^{(tr_i)} \sim Discrete(\theta^{(tr_i)}) \tag{13}$$

$$\theta \sim Dirichlet(\alpha) \tag{14}$$

TABLE 2. QBLGA.

Algorithm QBLGA
Input: Transaction data
Output: *revenue* and optimized catalogs
1: Produces customer preference by oracle
2: **While**(stop condition achieved)
3: Generate chromosomes randomly and each chromosome represents a catalog
4: Computes *revenue* of each chromosome
5: Reproduction
6: QBL flag ← false
7: **If** counter = specific times
8: QBL flag ← true
9: counter ← 0
10: Crossover
11: **If** QBL flag = true
12: Check redundancy and apply products from major oracle
13: **Else**
14: Check redundancy and apply products from minor oracle
15: Mutation
16: **If** QBL flag = true
17: Check redundancy and apply products from major oracle
18: **Else**
19: Check redundancy and apply products from minor oracle
20: **If** mutation count=0
21: mutation count ← m
22: **If** current fitness<best fitness then
23: mutation count ← mutation count − 1
24: **Else**
25: QBL flag ← false
26: counter ← counter + 1
27: **Return** *revenue* and catalog sets

θ is the preference distribution for transactions. φ is the product distribution for preferences. α and β are hyper parameters which are used to specify the priors on θ and φ. p_i means the ith product in a transaction. z_i is a latent variable indicating the preference of p_i and tr_i is the current transaction.

There are many approaches to training the LDA model. Gibb sampling [21] is a popular LDA modeling solution and is used to sample the preferences for each product. The formulation is as follows.

$$\mathrm{P}(z_i = \mathrm{j}|z_{-i}, \mathbf{p}) \propto \frac{n_{-i,j}^{(p_i)} + \beta}{n_{-i,j}^{(\cdot)} + P\beta} \frac{n_{-i,j}^{(tr_i)} + \alpha}{n_{-i,\cdot}^{(tr_i)} + Z\alpha} \tag{15}$$

where $n_{-i}^{(\cdot)}$ is a count that does not include the current assignment of z_i and Z means the number of preferences.

Through a sufficient number of iterations, the probability of product p in preference $j\hat{\emptyset}_j^{(p)}$ can be estimated [21] by the following equation where P is the number of products in the database and $n_j^{(p)}$ is the number of times product p has been

assigned to preference j.

$$\hat{\emptyset}_j^{(p)} = \frac{n_j^{(p)} + \beta}{n_j^{(\cdot)} + P\beta} \tag{16}$$

Therefore, φ is the product distribution for preferences in LDA model, which is then referred to as the product item distribution for preferences. After LDA modeling, we generate a preference set $Z = \{z_1, z_2, z_3, \ldots.\}$ and $z_j = \{p_1, p_2, p_3, \ldots\}$ is the product item set which is sorted by preference distribution over product items in descending order. Based on Eq.(4), the number of covered customer of each preference can be expressed as $\sum_{c\in C} X_{z_i,c}$ (i is the index of preference in Z).

We are now able to see these three preference modeling algorithms. The first algorithm is named LDASAGA [3]. Oracle represents the set of each catalog's most similar preference set. A preference set with the most number of overlapping products to a catalog means the catalog's most similar preference set. The insight of this oracle is that customers with similar preferences will buy similar products, therefore the oracle focuses on the preference similarity in constructing the catalogs. While crossover or mutation process requests to alter products, LDASAGA selects products randomly from the oracle. On the other hand, LDASAGA can also be used to create interactive catalogs. For example, if the user taps several products in the APP interface, we refer these products as a catalog. We can then use the LDASAGA to produce a list of products related to the initially selected products. The second preference modeling algorithm so called QBLGA-1 (HG-LDASAGA) is the preference set with highest goodness. The resulting products will then be selected from the preference set in descending order of *goodness*. The third preference modeling is named QBLGA-2 (TOPN-LDASAGA) and selects the top N highest number of covered customer preferences as the oracle. Each selected preference in the oracle has an equal number of products. According to Eq.(4), the number of covered customer of top N covered preference can be expressed as $\sum_{c\in C} X_{topN,c}$. TABLE 3 describes QBLGA-2 (TOPN-LDASAGA) in detail. When the crossover or mutation process requests an alternative product list, QBLGA-2 (TOPN-LDASAGA) selects products randomly from the preference set.

TABLE 3. QBLGA-2 (TOPN-LDASAGA).

Algorithm QBLGA-2 (TOPN-LDASAGA)

Input: Given preference set Z = $\{z_1, z_2, z_3 \ldots.\}$ from LDA.

Output: Oracle

1: $Best\ score \leftarrow -1, score \leftarrow 0, count \leftarrow 0, N \leftarrow 1$
2: Best topN $\leftarrow \emptyset$, topN $\leftarrow \emptyset$
3: Sort Z based on $\sum_{c\in C} X_{z_i,c}$ in descending order to obtain $Z' = \{z'_1, z'_2, z'_3, \ldots.\}$
4: **While** score > Best score
5: preference_size $\leftarrow \frac{|z'_N|}{N}$
6: **For** each z'_j in oracle
7: **If** j < N
8: **For** each product p in z'_j
9: **If** count<preference_size
10: topN $\leftarrow$ topN $\cup$ {p}
11: count $\leftarrow$ count + 1
12: count $\leftarrow$ 0
13: score = $\sum_{c\in C} X_{topN,c}$
14: **If** score>Best score
15: $Best\ score \leftarrow$ score
16: Best topN $\leftarrow$ topN
17: $N \leftarrow N + 1$
18: **Return** Best topN

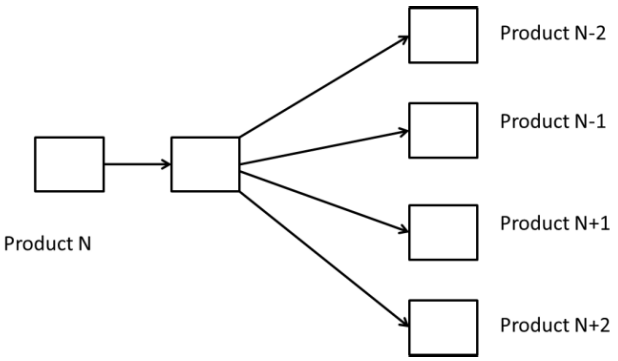

FIGURE 5. The architecture of Product2vec. The objective function is to learn product representation to discover regularities between products.

2) Product2Vec

We have discussed earlier that how to use preference modeling as oracle. Hence, we will now take a look at Product2Vec. The idea of the oracle is to adapt Word2Vec algorithm [22], [23] to generate product vectors as shown in Fig. 5. Word2Vec is a word embedding algorithm using two-layer neural networks. We aim to use neural network to learn the representation of products so that we can estimate the product similarity. We refer each transaction as a sentence so that each transaction can be concatenated as a paragraph. Then, each product is referred as a word.

The training objective of the Product2Vec model is to find product representations that are useful for predicting the surrounding products. More formally, given a products of a transaction $tr = \{p_1, p_2, p_3, \ldots, p_T\}$ and p_i means the ith product in tr. The objective of the Product2Vec model is to maximize the average log probability.

$$\frac{1}{T}\sum_{i=1}^{T}\sum_{-a\le j\le a, j\neq 0} logp(p_{i+j}|p_i) \tag{17}$$

where a is the size of surrounding products of p_i. Product2vec adopts negative sampling to calculate $p(p_O|p_I)$:

$$p\left(p_O \mid p_I\right) = \log\sigma\left(v'_{p_O} v_{p_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{p_i\sim P_n}[\log\sigma\left(-v'_{p_i} v_{p_I}\right)] \tag{18}$$

where σ (x) $= 1/(1 + \exp(-x))$. v_{p_I} and v'_{p_O} are the "input" and "output" vector representations of product p. Thus the task is to distinguish the target product p_O from draws from the noise distribution Pn (w) using logistic regression, where there are k negative samples for each data sample. Through estimating the similarity of product vectors, we can find a product which has the shortest distance to a catalog. Nevertheless, we shall concentrate on how to use Product2Vec to improve QBLGA. Here we have three distinct strategies. First, we can obtain top N nearest products by product vectors then select a product randomly from these products. Secondly, we select the product with highest goodness of the top N nearest products. Lastly, we select the product with highest product price of the top N nearest products.

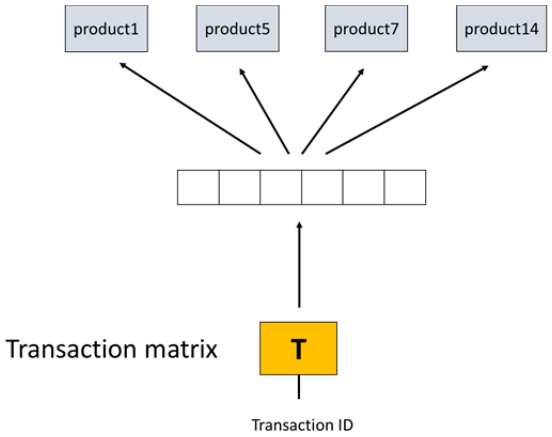

FIGURE 6. Transaction2vec is trained to learn transaction representation and predict the products in a transaction. Based on based on transaction representation, we can discover regularities between transactions.

3) Transaction2Vec

Besides product vector, another way is to learn the representation of transaction. The idea is the same as paragraph vector [24]. In Fig. 6, we refer a transaction as a paragraph. Therefore, we enhance the similarity between products if these products appearing in the same transaction. Based on cosine similarity between transaction vectors, we extract the most nearest transaction. The same as Product2Vec, there are three kinds of strategies. For one thing, we randomly select a transaction from the top N nearest transactions. Next, we select the transaction with highest goodness of the top N nearest transactions. Only in the final place, we select the transaction with highest product price of the top N nearest transactions. There is something noticeable Transaction2Vec is to retrieve the most similar transaction thus we have to extract one product of the transaction. There is room for further investigation. Here we just apply random selection strategy to cope with this issue.

We are now able to see the complexity of oracle. Based on complexity, oracle can be divided into two types: heavy oracle and light oracle. In Transaction2Vec model, a transaction has to be inferred to generate transaction vector since it is rare to have the same transaction in the model. Then, we can extract

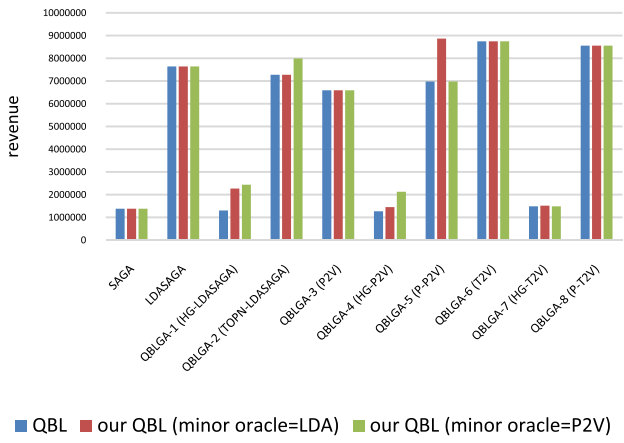

FIGURE 7. Evaluation results of different minor oracle.

top N nearest transactions of this transaction. Therefore, Transaction2Vec is a heavy oracle. In contrast to Transaction2Vec, Product2Vec is easier to find a product vector of a product. We can direct access the product vector instead of inference so Product2Vec is a light oracle. On the other hand, preference modeling is to select products from preference set. Likewise, preference modeling is a light oracle.

We use oracle to improve the chances of obtaining an optimal solution. However, all good combinations in all catalogs are might eventually come to resemble the preference sets from the oracle. In other words, QBLGA might run for a specified number of generations without changing the best revenue value. In this case, the oracle has no meaningful suggestions to provide, therefore the oracle operation will degenerate to random. That is to say QBLGA uses applied uncertainty to deal with this issue.

V. EXPERIMENT AND DISCUSSION

Our experiments use Ta-Feng which is a grocery shopping dataset released by ACM RecSys. Ta-Feng dataset covers products from food, office supplies to furniture. The dataset includes 32266 users and 23812 products. The total count of transactions in this dataset is 817741 and the transaction data was collected from November 2000 to February 2001. Here we used the data form November 2000 for MOCs construction and we tested the predication performance of these MOCs by the data from December 2000. In the data from November 2000, there are 16760 users and 17556 products. For the data from December 2000, there are 15447 users and 16684 products. In our experiments, we use LDASAGA as the baseline algorithm since it is the state-of-the-art algorithm for designing MOCs.

TABLE 4 shows the experimental settings. Moreover, we only reserve the product which sales volume is over 300 in population initialization in our experiments and *commision* of these experiments are set as 10. The specific number of epochs for QBL to switch between major oracle and minor oracle is 10 epochs in all of our experiments. For Product2Vec and Transaction2Vec, top N nearest vectors is set as 100. We consider the real designs of two APPs in 2016 and the parameter of these APPs are listed in TABLE 5. eBay has the max

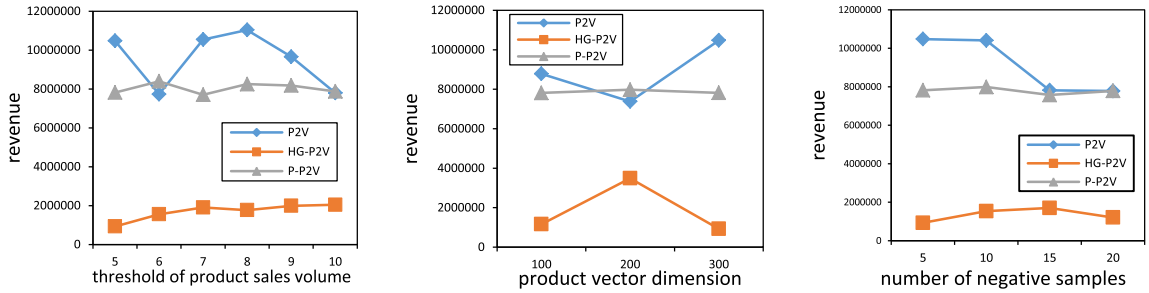

FIGURE 8. Parameter analysis for Product2Vec.

TABLE 4. Parameter settings.

Number of chromosomes	Generations	Reserved chromosomes	Mutations	Size of catalog	Customer interest threshold
20	30	10	20	50	7

TABLE 5. APP design for two companies.

company	MI
eBay 2016	{{1}{6,6,6,6,6,4}}
Taobao 2016	{{4,10,3,1}}

screen size 6 as shown in TABLE 5. eBay has only one catalog on the first layer and puts the rest of catalogs in the second layer, and then divides the second layer into 6 screens since 6 catalogs can simultaneously fit on the screen. Taobao puts all catalogs in the first layer and the first screen contains only 4 catalogs which serves as their promotional focus. The rest of catalogs in the first layer are shown in the following three screens, with 10 catalogs on the second screen, 4 catalogs on the third screen and one catalog on the last screen.

The proposed method and compared methods are brief described as follows. It can be noted that these QBLGAs are concerned about major oracle. The selection of minor oracle in these QBLGAs is taking up in the first experiment.

- **SAGA** is the state-of-the-art approach used to solve the COC problem in e-commerce (e-COC).
- **LDASAGA** is the state-of-the-art solution to construct MOC.
- **QBLGA-1 (HG-LDASAGA)** uses the preference set with highest goodness.
- **QBLGA-2 (TOPN-LDASAGA)** considers the top N preference with highest goodness.
- **QBLGA-3 (P2V)** exploits Word2Vec algorithm to establish Product2Vec model then selects a product from top N nearest products randomly.
- **QBLGA-4 (HG-P2V)** is similar as QBLGA 3 (P2V) except for selecting a product with highest goodness from top N nearest products.
- **QBLGA-5 (P-P2V)** is similar as QBLGA 3 (P2V) except for selecting a product with highest price from top N nearest products.
- **QBLGA-6 (T2V)** uses paragraph vector concept to construct transaction vector selects a transaction from top N nearest transactions randomly. After that, QBLGA-6 (T2V) randomly selects a product from the selected transaction.
- **QBLGA-7 (HG-T2V)** is similar as QBLGA-6 (T2V) except for selecting the transaction with highest goodness from top N nearest transactions. Then, QBLGA-6 (T2V) randomly selects a product from the selected transaction.
- **QBLGA-8 (P-T2V)** is similar as QBLGA-6 (T2V) except for selecting the transaction with highest price from top N nearest transactions.

A. COMPARISON OF DIFFERENT ORACLE PAIRS

The first experiment aims to compare the performance of original QBL and our QBL. We adopt the APP design of Taobao 2016 in this experiment. Original QBL only has one oracle. In contrast, our QBL has major and minor oracle. In this experiment, we only focus on minor oracle, as I said earlier, the QBLGAs already decide the major oracle. We introduce our empirical evaluation in different minor oracle. In terms of our QBL, we only use light oracle as our minor oracle. We need mention here only preference modeling and Product2Vec. Since we focus on finding the best minor oracle in this experiment, the parameters of LDA, Product2Vec and Transation2Vec are using the same parameters of [21] and [23]. For LDA, we used $\beta = 0.1$ and $\alpha = 50/Z$. Z is the number of topics and we set Z as 100. The size of each topic is 100. On the other hand, Product2vec has following three parameter: threshold of product sales volume, product vector dimension and the number of negative samples. In terms of Transaction2Vec, Transaction2Vec only has product vector dimension and the number of negative samples. Since it is hard to occur the same transaction, there is no threshold for occurrence of transaction. Reference [23] suggested the threshold of subsampling of frequent words is 5. Number of negative samples is in the range 5–20 and the vector dimension is 300. Therefore, we set threshold of product sales volume as 5, product vector dimension as 300 and the number of negative samples as 5. Fig. 7 shows the evaluation results of different minor oracle on Ta-Feng

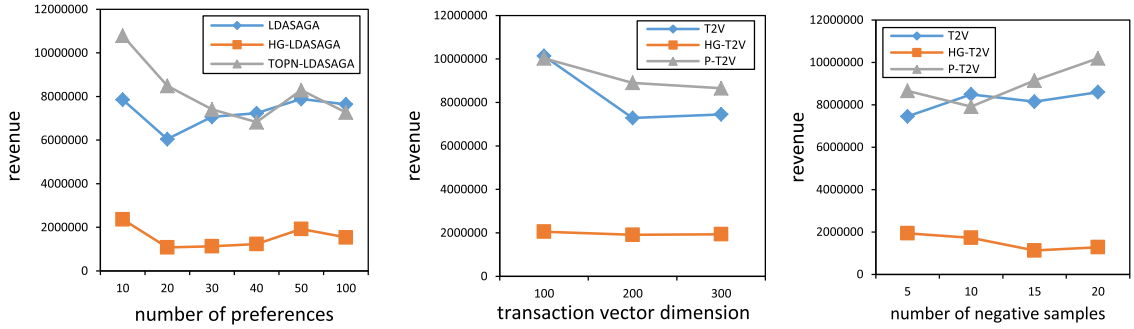

FIGURE 9. Parameter analysis for LDA and Transaciton2Vec.

dataset. We can observe that QBLGA using Product2Vec as minor oracle almost outperforms than other approaches, especially for QBLGA-2 (TOPN-LDASAGA). This indicates that Product2Vec can capture more customer preference information of products as compared to original QBL and QBLGA using LDA as minor oracle. Therefore, we adopt Product2Vec as minor oracle of QBLGAs in the following experiments.

B. PARAMETER ANALYSIS

In the second experiment, we present the parameter analysis of these proposed algorithms. The APP design of Taobao 2016 is adopted in this experiment. First, we study the effect of threshold of product sales volume. Fig. 8 shows the performances of Product2Vec for different threshold of product sales volume. We cannot observe any improved performance as the threshold of product sales volume increases. The best thresholds of product sales volume for these Product2Vec - based approaches are not equal. QBLGA-3 (P2V) has the best performance while threshold is 8. Then, QBLGA-4 (HG-P2V) achieves the best performance in 10 and the best threshold of product sales volume of QBLGA-5 (P-P2V) is 6. In terms of product vector dimension, the QBLGA-3 (P2V) has better performance while product vector dimension is 300. And both QBLGA-4 (HG-P2V) and QBLGA-5 (P-P2V) perform well in product vector dimension 200. Furthermore, the number of negative samples is a parameter of Product2Vec. Our experiments show that the best number of negative samples is 5 for QBLGA-3 (P2V), 15 for QBLGA-4 (HG-P2V), and 10 for QBLGA-5 (P-P2V). Then,

Fig. 9 is to find the best parameter of preference modeling and Transaction2Vec. The parameter of minor oracle adopts the tuned parameter from Fig. 8. Based on our observation, the number of preferences (i.e., topics of LDA) is an important factor for performance in LDASAGA, QBLGA-1 (HG-LDASAGA) and QBLGA-2 (TOPN-LDASAGA). Therefore, our experiments consider different number of preferences and the results are represented in

Fig. 9. The figure indicates that LDASAGA achieves the best performance in 50 preferences. Then, QBLGA-1 (HG-LDASAGA) and QBLGA-2 (TOPN-LDASAGA) have the best revenue are 10 preferences. In the contrast, the other number of preferences does not affect the performance

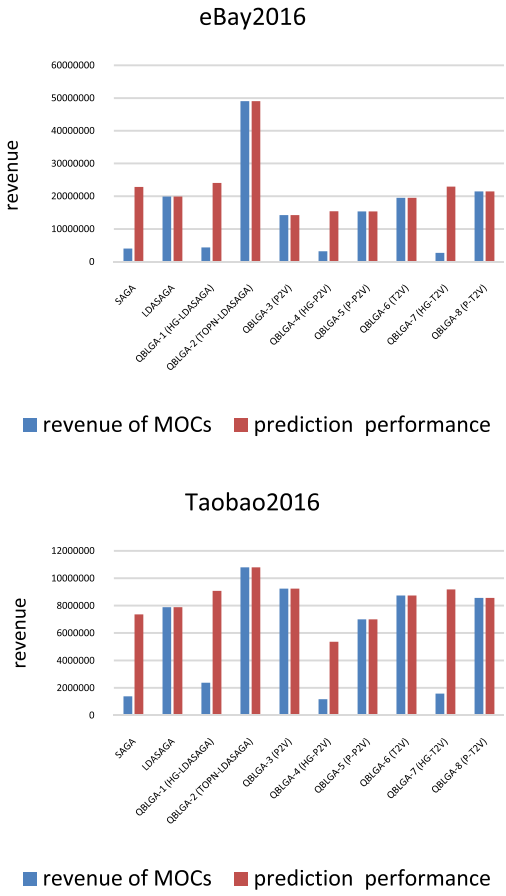

FIGURE 10. Revenue of two companies using real data.

markedly. We are now ready to consider Transaction2Vec, as mentioned before, Transaction2Vec only has to consider transaction vector dimension and the number of negative samples. In product vector dimension, the result clearly shows that all Transaction2Vec-based approaches reach the best revenue in 100. Furthermore, the results of the number of negative samples are shown in

Fig. 9. The best the number of negative samples of QBLGA-6 (T2V) and QBLGA-8 (P-T2V) are 20. For QBLGA-7 (HG-T2V), the number of negative samples is appropriate to set as 5.

C. PERFORMANCE OF SEVERAL COMPANIES BY REVENUE

The third experiment compares the performance of several companies using *revenue* in MOCs optimization problem as well as testing the prediction performance of these MOCs. Fig. 10 compares the performance of our proposed approaches and the baseline algorithm, and shows that various Apps based on QBL-based approaches outperform LDASAGA. The bar plots in Fig. 10 demonstrate that with more customers covered in MOCs, it is able to achieve outstanding *revenue* in the future. The result show that QBLGA-2 (TOPN-LDASAGA) is superior to all compete algorithms. Therefore, it seems reasonable to conclude that using major oracle as top N preference and applying Product2Vec as minor oracle can achieve better result. On the other hand, it may be worth pointing out, in passing, that the scope of revenue is due to the different interface design concepts. Some companies aim to cover more customers thus they place more MOCs in the first few screens and layers. In contrast, low scope of *revenue*, the market strategy is focused on specific groups of customers to arise the sales volume. In that case, there are less catalogs in this kind of market strategy. Therefore, *revenue* cannot be used to compare the interface design of these companies directly.

VI. CONCLUSION

In this paper, we aim to optimize the customer attraction of mobile shopping applications to reduce user obstacles due to small mobile device screens and time consuming operating processes. To build optimized catalogs for m-commerce, we model this problem as a MOC segmentation problem which seeks to minimize the number of operation customers must perform by generating catalogs that can attract the greatest number of potential customers using the proposed QBLGA. In order to achieve this goal, we propose *revenue* as fitness function for QBLGA to optimize. In this work, we develop and compare several different types of QBLGA. Then, QBLGA-2 (TOPN-LDASAGA) are shown to significantly boost performance in MOC construction compared to state-of-art methods. QBLGA-2 (TOPN-LDASAGA) constructs a preference set by Top N preference as major oracle and applies Product2Vec as minor oracle. In conclusion, the experimental result show that QBLGA is a promising solution in MOCs construction.

REFERENCES

[1] R. J.-H. Wang, E. C. Malthouse, and L. Krishnamurthi, "On the go: How mobile shopping affects customer purchase behavior," *J. Retailing*, vol. 91, no. 2, pp. 217–234, 2015.

[2] L. Y. Chen, "Exploring the quality of mobile shopping system and its link to the organizational performance," *Int. J. Inf. Process. Manage.*, vol. 6, no. 2, p. 19, 2015.

[3] H.-M. Hsu, R.-I. Chang, and J.-M. Ho, "Constructing mobile-oriented catalog in m-commerce using LDA-based self-adaptive genetic algorithm," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–7.

[4] J. Kleinberg, C. Papadimitriou, and P. Raghavan, "A microeconomic view of data mining," *Data Mining Knowl. Discovery*, vol. 2, no. 4, pp. 311–324, 1998.

[5] Y. E. Lee and I. Benbasat, "A framework for the study of customer interface design for mobile commerce," *Int. J. Electron. Commerce*, vol. 8, pp. 79–102, Dec. 2004.

[6] J. F. Rayport and B. J. Jaworski, *Introduction to e-commerce*. New York, NY, USA: McGraw-Hill, 2002.

[7] J. Persson and J. Berndtsson, "Determinants of smartphone shopping adoption: Key factors for online shopping of consumer goods through smartphones in Sweden," M.S. thesis, Dept. Bus. Admin., Lund Univ., Lund Univ. Publications, Lund, Sweden, 2015.

[8] Y. Xu, M. Spasojevic, J. Gao, and M. Jacob, "Designing a vision-based mobile interface for in-store shopping," in *Proc. 5th Nordic Conf. Human-Comput. Interactio, Building Bridges*, 2008, pp. 393–402.

[9] Q. Su and L. Chen, "A method for discovering clusters of e-commerce interest patterns using click-stream data," *Electron. Commerce Res. Appl.*, vol. 14, no. 1, pp. 1–13, 2015.

[10] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao, "Unsupervised clickstream clustering for user behavior analysis," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2016, pp. 225–236.

[11] J. Kleinberg, C. Papadimitriou, and P. Raghavan, "Segmentation problems," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, 1998, pp. 473–482.

[12] D. Xu, Y. Ye, and J. Zhang, "Approximating the 2-catalog segmentation problem using semidefinite programming relaxations," *Optim. Methods Softw.*, vol. 18, no. 6, pp. 705–719, 2003.

[13] M. Steinbach, G. Karypis, and V. Kumar, *Efficient Algorithms for Creating Product Catalogs*. Fort Belvoir, VA, USA: Defense Tech. Inform. Center, 2000.

[14] M. Ester, R. Ge, W. Jin, and Z. Hu, "A microeconomic data mining problem: Customer-oriented catalog segmentation," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 557–562.

[15] A. Amiri, "Customer-oriented catalog segmentation: Effective solution approaches," *Decision Support Syst.*, vol. 42, pp. 1860–1871, Dec. 2006.

[16] I. Mahdavi, M. Movahednejad, and F. Adbesh, "Designing customer-oriented catalogs in e-CRM using an effective self-adaptive genetic algorithm," *Expert Syst. Appl.*, vol. 38, pp. 631–639, Jun. 2011.

[17] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, NY, USA: Freeman Company, 1979.

[18] R.-I. Chang, L.-B. Lai, W.-D. Su, J.-C. Wang, and J.-S. Kouh, "Intrusion detection by backpropagation neural networks with sample-query and attribute-query," *Int. J. Comput. Intell. Res.*, vol. 3, no. 1, pp. 6–10, 2007.

[19] R.-I. Chang, S.-Y. Lin, and Y. Hung, "Particle swarm optimization with query-based learning for multi-objective power contract problem," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3116–3126, 2012.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[21] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Acad. Sci. USA*, vol. 101, pp. 5228–5235, Apr. 2004.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[24] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. ICML*, 2014, pp. 1188–1196.

HUNG-MIN HSU is currently a Ph.D. Student with National Taiwan University and a Research Assistant with the Research Center for the Information Technology Innovation. He joined the Taiwan e-Learning and Digital Archives Program from 2009 to 2013 and the Academia Sinica Digital Center: System Management and Content Retrieval Technologies for Supporting Cloud-Based Digital Archive Systems and Services Project from 2014 to 2015. His research interests cover text mining, machine learning, mobile commerce, and related applications.

RAY-I CHANG (M'14) is currently a Professor with the Department of Engineering Science and Ocean Engineering, National Taiwan University, Taiwan. He has authored over 200 original papers in international conferences and journals. His current research interests include machine learning, multimedia networking, and data mining. He is a member of CERP and IICM.

JAN-MING HO received the Ph.D. degree in electrical engineering and computer science from Northwestern University in 1989. He joined the Institute of Information Science, Academia Sinica, as an Associate Research Fellow in 1989, and was promoted to Research Fellow in 1994. He visited IBM's T. J. Watson Research Center in 1987 and 1988, the Leonardo Fibonacci Institute for the Foundations of Computer Science, Italy, in 1992, and the Dagstuhl Seminar on Applied Combinatorial Methods in VLSI/CAD, Germany, in 1993. From 2004 to 2006, he served as the Director General of the Division of Planning and Evaluation, National Science Council.