

# Product EntityCube: A Recommendation and Navigation System For Product Search\*

Jongwuk Lee <sup>#1</sup>, Seung-won Hwang <sup>#2</sup>, Zaiqing Nie <sup>†3</sup>, Ji-Rong Wen <sup>†4</sup>

<sup>#</sup>*Pohang University of Science and Technology*

*Pohang, Republic of Korea*

<sup>1</sup>julee@postech.ac.kr

<sup>2</sup>swhwang@postech.ac.kr

<sup>†</sup>*Microsoft Research Asia*

*Beijing, P. R. China*

<sup>3</sup>znjie@microsoft.com

<sup>4</sup>jrwen@microsoft.com

**Abstract**— We demonstrate Product EntityCube, a product recommendation and navigation system. While the unprecedented scale of a product search portal enables to satisfy users with diverse needs, this scale also complicates product recommendation. Specifically, our target application poses a unique challenge of overcoming insufficient user profiles and feedbacks. To address this problem, we organize query results into clusters representing different user perceptions of similarity, and provide a navigational UI to handle personal interests. Specifically, we first discuss *hybrid object clustering* capturing diverse user perception from millions of Web pages and disambiguating different senses using feature-based similarity. We then discuss *skyline object ranking* to highlight interesting items at each cluster. Our demonstration illustrates how Product EntityCube can enrich user product shopping experiences.

## I. INTRODUCTION

Product search portals host millions of products sold worldwide. This scale potentially enables to satisfy users with varying personal information needs. However, at the same time, due to this scale, it is extremely challenging to guide users to discover their personal favorite products.

To address this problem, the closest research area studies *recommender systems*. Existing approaches, surveyed in [1], can be categorized into *content-based recommender systems* (CBR) and *collaborative filtering systems* (CF). Specifically, CBR exploits *user profiles*, such as the history of products purchased by the particular user, to recommend similar items. Meanwhile, CF exploits *user feedbacks* such as product ratings, to recommend items based on the preferences of other users with similar feedbacks. A well-known problem is the Netflix challenge [6] of building a system to recommend movies, using 100 million user ratings on 100,000 movies, in which *hybrid approaches* exploiting both user profiles and feedbacks are leading the competition. To illustrate, Fig. 1 categorizes existing recommender systems.

In clear contrast, recommendations for product search portal pose unique challenges. First, since object-level search engine [7] simply crawls and extracts the product information from

\*This work was done when the first two authors visited Microsoft Research Asia.

		User feedbacks	
		Yes	No
User profiles	Yes	Hybrid recommender systems	Content-based recommender systems (CRF)
	No	Collaborative filtering systems (CF)	

Fig. 1. Taxonomy of recommender systems

documents, it cannot access user profiles and feedbacks. Similar challenges have been raised in using public web data for recommendation [9]. Second, unlike a document represented as a TF-IDF vector where every feature belongs to a homogeneous domain, an item in product search is often represented as numerical features on *heterogeneous* domains, e.g., sensor size, price, and weight. The appropriate similarity notion well-reflecting user perception thus depends on both domains and user profiles.

To address these challenges, we propose to *diversify results* to minimize the dissatisfaction of diverse user profiles and provide a navigational UI to allow users to drill down to personal favorites. In particular, we achieve the goal by adopting both user-perceived and feature-based similarity collected from the Web corpus.

We believe that the key contributions of our proposed system are as follows:

- **Hybrid object clustering:** In order to provide diversified results, we propose a hybrid clustering approach combining two complementary similarity notions of items. First, we extract *co-occurrences* of items in Web documents, and use as a robust similarity measure for clustering, which can be interpreted as *implicit feedbacks* of document creators viewing two items as relevant. Second, we exploit *feature values* of items to explain and disambiguate different reasons of co-occurrences.
- **Skyline object ranking:** Contrary to existing recom-

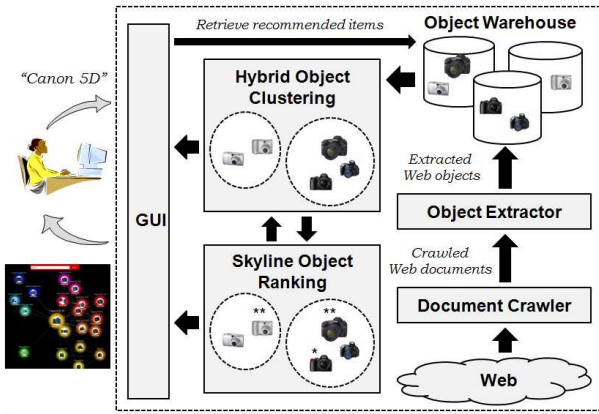


Fig. 2. Overall architecture of Product EntityCube

recommender systems learning user-specific ranking, we aim to generate ranking that is “universally desirable” for all users. While *skyline queries* [2] generate such ranking, its naive adoption for our problem qualifies almost all products as skyline. We thus propose to effectively narrow entire results to cluster-specific interesting results, *i.e.*, *strong* and *weak* skyline at each cluster, using cluster-specific salient features identified from clustering results.

The rest of the paper is organized as follows. Section II describes an overview of our framework, and explains key components for clustering and ranking recommended items. Section III presents our demonstration scenarios.

## II. SYSTEM OVERVIEW

This section first overviews Product EntityCube supporting diversified results, and then presents its key components. As Fig. 2 depicts overall architecture with respect to camera “Canon 5D”, our tool builds upon crawlers that collect product pages and extract co-occurrences and features values of products, and shows diversified results with clustering and ranking modules.

Recently, such needs of diversification have been similarly raised in recommender systems, to address “serendipity recommendations” to help users discover new tastes [1]. For instance, some commercial recommender systems, such as Youtube and Strands, have recently released navigational UIs to explore diversified recommendations in 2D graph or map structures, *e.g.*, Warp [11] and recommendation map [8]. However, these UIs cannot support to explain the reason for recommendations or disambiguate different reasons, which has been recently raised as important challenges for recommendation systems [5].

In clear contrast, we develop a more systematic approach for presenting diversified recommended items, by grouping them into clusters and labeling each with proper explanations to guide user navigation. A similar approach has been explored for document searches [3]: as users may have different search intents on the same query keyword, this line of research studies how to cluster related documents based on topics, and generate representative phrases for each topic cluster, where

users navigate their personal favorites. However, little work has been done on object-level search, such as product search.

In particular, the challenging issue in organizing results for object-level search is *domain heterogeneity*. In our application, product  $p$  is represented as *term phrase*, *i.e.*, product names, and a numeric *feature value vector*  $(p_1, \dots, p_n)$  extracted from the Web corpus, for a set of features  $\mathcal{F} = \{f_1, \dots, f_n\}$ , *e.g.*, sensor size, price, and weight. Since these features belong to heterogeneous domains and user-perceived similarity varies over user profiles, it is non-trivial to identify an appropriate measure that is both domain- and user-specific.

To address this problem, we collected diverse notions of user-perceived similarity from product *co-occurrences* in the review, and clustered them into representative groups based on *feature values* extracted from the Web corpus. This combined notion of similarity thus has complementary strengths of reflecting both user-perceived and data-inherent similarity.

More formally, we define our problem as organizing  $k$  clusters of recommended items in ranked order.

**Problem 1 (Object-level query result organization)** *Given query  $q$  and the number of clusters  $k$ , identify a set of  $k$  clusters  $\mathcal{C} = \{C_1, \dots, C_k\}$ , such that each  $C_i$  defined on feature subspace  $\mathcal{S}_i$  is ranked by cluster-specific ranking function  $\mathcal{R}_i$  on  $\mathcal{S}_i$ .*

### A. Hybrid Object Clustering

This section present our clustering method to organize recommended items to satisfy users with diverse needs, using product features and co-occurrences extracted from Web documents.

We first present how to use feature-based similarity exploiting the intuition of *subspace clustering* [10]. For instance, as  $c_1 = (\text{“Canon 40D”}, \text{“Canon 50D”})$  shares similar high values in feature subspace  $\{\text{sensor size}\}$  and  $c_2 = (\text{“Canon 40D”}, \text{“Canon Powershot SD850”})$  in  $\{\text{release year}\}$ , a group of user looking for high-end DSLRs will find “Canon 50D” in  $c_1$  more relevant and navigate to find more such items, while another group looking for new models will drill down to a group with “Canon Powershot SD850”.

Based on this intuition, we allow clusters to have “value locality” in different “feature subspaces”. Specifically, one successfully deployed measure [10] for value locality is a *relative index* as follows:

$$R_{ij} = 1 - \frac{\sigma_{ij}^2}{\sigma_j^2}, \quad (1)$$

where  $\sigma_{ij}$  is “local deviation” of feature  $f_j$  on a cluster  $C_i$ , and  $\sigma_j$  is “global deviation” of all related products on  $f_j$ . Based on the locality notion, cluster-specific subspace  $\mathcal{S}_i$  can be determined as follows:

$$\mathcal{S}_i = \{f_j | R_{ij} \geq R_{min}\}, \quad (2)$$

where  $R_{min}$  is a minimum threshold parameter for  $R_{ij}$ .

In addition, subspace clustering algorithms generally encourage clusters to share value locality in as many features

as possible, *i.e.*, stronger evidences for the cluster quality. To ensure that, each cluster should satisfy a qualification condition:

$$|\mathcal{S}_i| \geq d_{min}, \quad (3)$$

where  $|\mathcal{S}_i|$  is the size of subspace, and  $d_{min}$  is a minimum threshold parameter for  $|\mathcal{S}_i|$ .

While the above notions enable to distinguish different notions of value locality by selecting feature subset  $\mathcal{S}_i$ , we can observe that the quality of similarity heavily depends on parameter  $R_{min}$  and  $d_{min}$  that are data- and cluster-specific, which is hard to tune. To illustrate, determining whether pair  $c_1$  (“Canon 40D”, “Canon 50D”) is more similar than  $c_2$  (“Canon 40D”, “Canon Powershot SD850”) is reliable, only if both parameters are appropriately tuned for two pairs respectively. As a result, the quality of clustering algorithms is also highly sensitive to the quality of these parameters.

To fight such sensitivity, we propose to adopt *co-occurrences* of items from Web corpus. Specifically, pair-wise similarities of pairs  $c_1$  and  $c_2$  can be judged by their co-occurrences in Web documents, *e.g.*, reviews. This notion, by not requiring any parameters, robustly quantifies the magnitude of pair-wise similarities, as successfully adopted in [12]. However, using only co-occurrences for clustering may result in combining two pairs, *e.g.*,  $c_1$  and  $c_2$ , although the relationships for two pairs can be very different. Note that the relationships can be inferred as feature subspaces.

We thus propose a hybrid scheme, using robust co-occurrence to determine merging order for *agglomerative clustering*, while we apply feature-based similarity to decide if the cluster to be merged shares enough value locality. Due to space limitation, we refer readers to [4] for the details on our hybrid clustering algorithm. We also empirically validated in [4] that our proposed clustering algorithm outperforms existing algorithms both in terms of accuracy and efficiency.

### B. Skyline Object Ranking

This section discusses how to address the challenge of presenting results in meaningful ranked order. Toward the goal, existing recommender systems typically exploit user profiles to “learn” user-specific ranking function  $\mathcal{R}$ . However, as such user histories are lacking in our target scenarios, we cannot adopt this approach.

Instead, we aim to identify “universally good”  $\mathcal{R}$  to general users. This goal is consistent with that of *skyline queries* [2], to identify items that are not “dominated” by any other item. Formally,  $p$  dominates  $q$  on total feature set  $\mathcal{F}$  if:

$$\forall f_j \in \mathcal{F} : p_j \geq q_j, \exists f_k \in \mathcal{F} : p_k > q_k \quad (4)$$

Based on the notion, *skyline* is a set of items which is not dominated by any other items. Since such item  $p$  in the skyline is guaranteed to rank higher than  $q$  in any monotonic class of ranking functions  $\mathcal{R}$ , it can be identified as one of universally good candidates for all users, without requiring any user profiles or feedbacks.

However, as a side effect for not requiring any user-specific information, skyline queries do not allow to control the size of

results. According to the above dominance notion, a product is dominated, only if it is worse in all features, which is highly unlikely, as most popular products in the market typically excel at least in one feature to stay competitive in the market. In this case, skyline results end up including most products, which is known as the “curse of dimensionality” problem in the skyline query literature.

To address this problem, we propose to restrict the notion of dominance on  $\mathcal{F}$  into cluster-specific feature subspace  $\mathcal{S}_i \subseteq \mathcal{F}$ , which can significantly reduce the number of skyline points. Our work, building upon subspace clustering results identifying meaningful subspace  $\mathcal{S}_i$  for  $k$  clusters, can leverage cluster-specific salient features identified from clustering results. In particular, we propose to identify *weak* and *strong* skyline as follows:

- **Weak skyline:** No product in cluster  $C_i$  dominates weak skyline on  $\mathcal{S}_i$ .
- **Strong skyline:** No product in cluster  $C_i$  dominates strong skyline, not only on salient features  $\mathcal{S}_i$  but also on the remaining features  $\mathcal{F} - \mathcal{S}_i$  among the weak skyline.

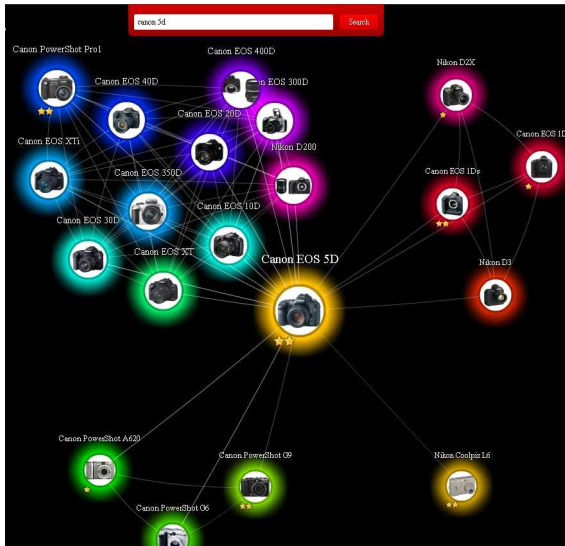
With the above criteria, products in each cluster are now categorized into three groups of strong skyline, weak skyline, and non-skyline, annotated with two, one, and zero stars respectively. These new notions are unique, by combining skylining with cluster-specific feature selection.

However, as even weak skyline can be too many, we also propose strong skyline, considering both strengths in not only salient feature subspaces  $\mathcal{S}_i$  but also the remaining feature subspaces  $\mathcal{F} - \mathcal{S}_i$ . We observed that strong skyline is effective in further narrowing down to interesting results, as we can observe from our demonstration.

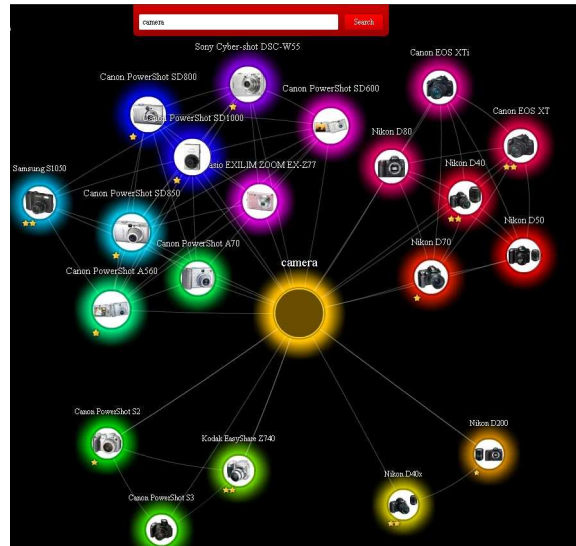
For processing, we consider a two-phase solution using a *non-index* skyline algorithm such as Block Nested Loop (BNL) algorithm [2] as clusters are determined dynamically and thus cannot be indexed a priori. That is, we perform BNL once on  $\mathcal{S}_i$  to identify weak skyline, and then scan weak skyline again to perform BNL on  $\mathcal{F} - \mathcal{S}_i$  to identify strong skyline. This processing can be expedited by a sorted skyline list. More specifically, at the first phase, the skyline list can be sorted by a monotonic aggregation function on  $\mathcal{S}_i$ , *e.g.*,  $f(p, \mathcal{S}_i) = \sum_{f_j \in \mathcal{S}_i} p_j$  such that the first few items dominate the rest to effectively prune out non-skyline. Similarly, at the second phase, the skyline list can be sorted by  $f(p, \mathcal{F} - \mathcal{S}_i)$  for efficiency.

## III. DEMONSTRATION

This section demonstrates **Product EntityCube**. In particular, our proposed system was developed with a real-life product dataset of 83.9 GBs, including more than 1.1 millions documents crawled in September, 2008 by crawlers for Live Product Search. Co-occurrences were extracted from review documents in the dataset for six popular product categories such as cameras, laptops, televisions, MP3 players, GPSs, and cell phones. Specifically, given a query, our demo finds top-20 relevant products, based on which clustering and ranking results are presented.



(a)  $q_1 = \text{"Canon 5d"}$



(b)  $q_2 = \text{"camera"}$

Fig. 3. Illustration of query results with different query types

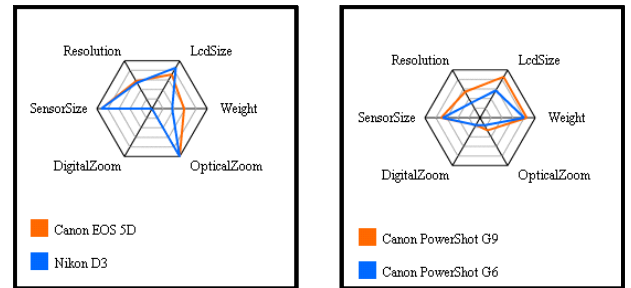
**Running scenarios:** The goal of this demonstration is to show how our proposed system can enrich the shopping experiences for users with diverse needs. Toward the goal, we demonstrate two query types for experienced and novice users: First, users familiar with products may issue a keyword matching a specific product name such as  $q_1 = \text{"Canon 5D"}$ . Second, novice shoppers may issue a keyword presenting a descriptive term such as  $q_2 = \text{"camera"}$ . The results are displayed as a navigational 2D-graph with the query centered and related items connected to the center. Fig. 3 illustrates the snapshots for  $q_1$  and  $q_2$  respectively. Clicking any node will issue another query, representing related items to the clicked node in the center.

**Clustering explanation:** Related items in two different senses will be distinguished by different cluster locations and colors. The length of an edge connecting node pairs represents the magnitude of relevance. This edge, when clicked, visualizes feature subspace where two pairs share value locality, using spidergrams. Fig. 4 illustrates spidergrams showing different relationships in two different clusters, *i.e.*, DSLRs and compact camera clusters. This diagram presents an absolute feature value normalized in  $[0, 1]$ , where 0 and 1 correspond to the center and the outermost layer respectively. In Fig. 4, we can compare and contrast different relationships in two different clusters, *e.g.*, between DSLRs sharing similar values in feature subspace  $\{\text{SensorSize, Resolution, Lcd size, OpticalZoom}\}$  and compact cameras in  $\{\text{SensorSize, Digital Zoom, Weight}\}$ , respectively.

**Ranking explanation:** The cluster-specific ranking is represented by star ratings. This information can help users identify better products without heavy user overheads. The weak and strong skyline is annotated with one and two stars in Fig. 3.

#### ACKNOWLEDGEMENT

The first two authors were supported by Microsoft Research Asia (internet service theme) and Engineering Research Cen-



(a) DSLR cluster

(b) Compact camera cluster

Fig. 4. Illustration of explaining different clusters

ter of Excellence Program of Korea Ministry of Education, Science and Technology (MEST) / Korea Science and Engineering Foundation (KOSEF), grant number R11-2008-007-03003-0.

#### REFERENCES

- [1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 2005.
- [2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, 2001.
- [3] M. A. Hearst and J. O. Pedersen. Re-examining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, 1996.
- [4] J. Lee, S. Hwang, Z. Nie, and J. Wen. Query result clustering for object-level search. In *KDD*, 2009.
- [5] D. Mcsherry. Explanation in recommender systems. *Artificial Intelligence Review*, 2005.
- [6] Netflix. <http://www.netflixprize.com/>.
- [7] Z. Nie, J.-R. Wen, and W.-Y. Ma. Object-level vertical search. In *CIDR*, 2007.
- [8] Recommendation Map. <http://labs.strands.com/recmap/>.
- [9] G. Shani, M. Chickering, and C. Meek. Mining recommendation from the web. In *RecSys*, 2008.
- [10] K. Y. Yip, D. W. Cheung, and M. K. Ng. HARP: A practical projected clustering algorithm. *TKDE*, 2004.
- [11] Youtube Warp. [http://www.youtube.com/warp\\_speed/](http://www.youtube.com/warp_speed/).
- [12] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*, 2009.