Optimizing User Exploring Experience in Emerging E-Commerce Products

Xiubo Geng
Yahoo! Labs Beijing
Beijing, 10083, P.R.China
gengxb@yahoo-inc.com

Xin Fan
Yahoo! Labs Beijing
Beijing, 10083, P.R.China
xinfan@yahoo-inc.com

Jiang Bian
Yahoo! Labs
Sunnyvale, CA, USA
jbian@yahoo-inc.com

Xin Li
Yahoo! Labs Beijing
Beijing, 10083, P.R.China
xinli@yahoo-inc.com

Zhaohui Zheng
Yahoo! Labs Beijing
Beijing, 10083, P.R.China
zhaohui@yahoo-inc.com

ABSTRACT

E-commerce has emerged as a popular channel for Web users to conduct transaction over Internet. In e-commerce services, users usually prefer to discover information via querying over category browsing, since the hierarchical structure supported by category browsing can provide them a more effective and efficient way to find their interested properties. However, in many emerging e-commerce services, well-defined hierarchical structures are not always available; moreover, in some other e-commerce services, the pre-defined hierarchical structures are too coarse and less intuitive to distinguish properties according to users interests. This will lead to very bad user experience. In this paper, to address these problems, we propose a hierarchical clustering method to build the query taxonomy based on users' exploration behavior automatically, and further propose an intuitive and light-weight approach to construct browsing list for each cluster to help users discover interested items. The advantage of our approach is four folded. First, we build a hierarchical taxonomy automatically, which saves tedious human effort. Second, we provide a fine-grained structure, which can help user reach their interested items efficiently. Third, our hierarchical structure is derived from users' interaction logs, and thus is intuitive to users. Fourth, given the hierarchical structures, for each cluster, we present both frequently clicked items and retrieved results of queries in the category, which provides more intuitive items to users. We evaluate our work by applying it to the exploration task of a real-world e-commerce service, i.e. online shop for smart mobile phone's apps. Experimental results show that our clustering algorithm is efficient and effective to assist users to discover their interested properties, and further comparisons illustrate that the hierarchical topic browsing performs much better than existing category browsing approach (i.e. Android Market mobile apps category) in terms of information exploration.

Categories and Subject Descriptors

H.5.2 [**Information Interface and Presentation**]: User Interfaces

General Terms

Design, Algorithm, Experimentation, Performance

Keywords

Hierarchical clustering, category browsing, user experience

1. INTRODUCTION

Recent years have witnessed the rapid growth of e-commerce, which refers to the buying and selling of products or services over electronic systems such as Internet. An increasing population of Web users have started using e-commerce services to conduct transactions on the Web. In e-commerce services, users usually prefer to explore interested properties via category browsing. Compare to searching, users only need to choose among provided options, which will save users's own efforts of formulating queries, especially when they do not exactly know what they are looking for. Besides, as more and more e-commerce services accessible on mobile smart phones, category browsing is more convenient than typing queries on mobile devices.

As a result, many e-commerce services provide hierarchical taxonomies for users to explore. For example, Amazon[1] provides different channels, e.g. books, electronics & computers, kindle, toys, kids & baby, etc., for category browsing beyond general search; Yahoo! shopping[2] also divides all items into several categories, like clothing & accessories, flowers & gifts, computers, etc., and provides a category browsing interface for users to explore. In these scenarios, a user-friendly hierarchical structure is necessary for assisting users to discover their interested items effectively and efficiently.

Although hierarchical structures for category browsing have been well-defined in some e-commerce services (e.g. books, clothing, etc.), this invaluable information is still missing in some other services, especially those emerging ones. For example, with the rapid growth of smart mobile phone's apps, apps exploration has become a non-trivial task. Currently, almost all existing app services only provide very coarse categories. As an example, the android market provides 26 categories for non-game apps, but there is no further classification within each category. The missing of well-defined

[1]http://www.amazon.com
[2]http://shopping.yahoo.com

hierarchical structures will lead to bad user exploration experience considering the following aspects,

- There are a large amount of apps included in each category. For example, hundreds of, and even thousands of apps comprise of one category in Android Market[3], which makes it quite difficult for users to reach their interested apps efficiently.

- In many cases, users intend to explore interested apps without any specific target. However, they may feel quite frustrated when facing so many apps with no hint on what these apps are used for.

To address these problems, we propose building a hierarchical structure for emerging e-commerce products according to users' behavior preference, which can be recognized from searching logs. Specifically, in order to identify users' interests, we model the similarity between queries according to users' corresponding click-through information in their searching log. However, since searching logs in e-commerce are usually quite sparse and noisy, we combine them with another two signals representing query similarity, i.e. term similarity and click-through information in general web search. Based on the derived query similarity, we propose a hybrid hierarchical clustering approach for query clustering.

In addition to building the hierarchical structure for category browsing, we further propose an intuitive approach to classify items into each category automatically. Specifically, we consider two information. First, those frequently clicked items of queries in the category are classified as the current category. Second, we also use popular queries of each category to retrieval items, and those highly ranked items are included in the category too. This approach is straightforward, while it saves tedious human efforts and machine resource to conduct classification, and provides quite effective performance.

The hierarchical structure and classified items for each category are integrated into a new user-friendly interface to assist users to browse interested items. By using this new interface, users can browse along hierarchical categories, in each level of which we provide both hot clicked items and highly ranked items for each cluster at current level, and users can also narrow down their interests by clicking the sub-cluster and entering the next level in the hierarchical structure.

By using the new proposed hierarchical structure, we are able to enhance users exploration experience with respect to several aspects:

1. The provided fine-grained structure can help users reach their interested items in a more efficient way.

2. Our hierarchical structure can provide explicit intuition for users exploration as it is derived from users' interaction logs. It also provides informative hints for users who have a general interest without any clear target, since we recommend them what most users are interested in.

3. For each category, we provide both frequently clicked items (which are highlighted) and highly ranked items of popular queries in the category. This gives users more intuitive items and richer information about these items, which help them judge whether they are interested in them.

[3]http://market.android.com

We implement and deploy this hierarchical query clustering approach and the user-friendly interface to a real-world mobile apps discoving system. The experiments on this system demonstrate that our hybrid clustering approach has a high success rate in providing a reasonable and user-oriented hierarchical category. Further analysis illustrates that the user-friendly interface based on the derived query taxonomy can offer users better exploration experience than the current category browsing one.

To sum up, our contributions include,

1. proposal of a hybrid query clustering approach, which can build user-oriented hierarchical structures automatically,

2. proposal of an intuitive approach to construct browsing lists for derived clusters automatically,

3. verification of the proposed approach on an emerging e-commerce area–mobile app browsing.

2. RELATED WORK

This study is related to two general areas, category browsing in e-commerce and query clustering.

2.1 Category Browsing in E-Commerce

In mature e-commerce services, category browsing is a popular way for users to explore interested items. As an example, Amazon provides hierarchical categories to assist users to browse products, as shown in Figure 1. As another example, Yahoo! shopping also classifies items into different dimensions, as shown in Figure 2. From these examples, we can observe that a sound hierarchical structure can effectively enhance user experience of exploring interested items via online browsing.

However, such important hierarchical structure is still missing or not well-defined in many other e-commerce services, especially those new emerging ones. For example, the category browsing within mobile apps is so coarse that there are too many items in each category, which makes it difficult for users to explore and lead to bad user experience consequently. What's worse, as there have been large amount of items in the new emerging services, it is quite expensive and almost infeasible to define a hierarchical structure by humans. To address these problems, in this paper, we propose to conduct hierarchical clustering on user queries to organize products in e-commerce service automatically, and then apply the derived structure to a new hierarchical category browsing interface.

2.2 Query Clustering

There has been much work on conducting query clustering. Many previous studies make use of users' searching log to define similarity between queries. For example, in [2] Befferman and Berger propose building a query-url bipartite graph according to the click-through log, and apply an agglomerative clustering approach to conduct query clustering and url clustering simultaneously. The derived query clusters are employed for query suggestion. Wen et al. [10] also leverage the click-through information. Besides, they add

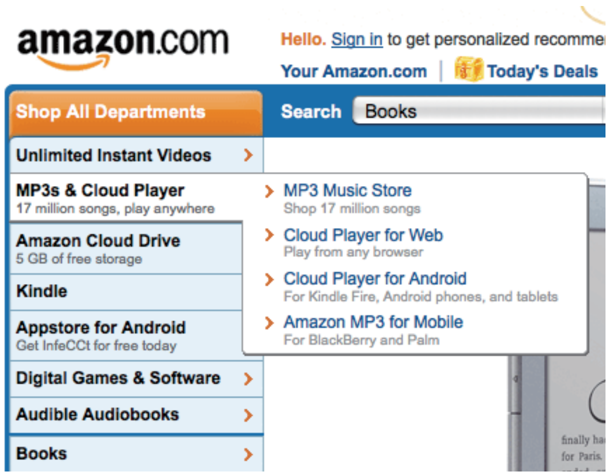

Figure 1: Category browsing in Amazon

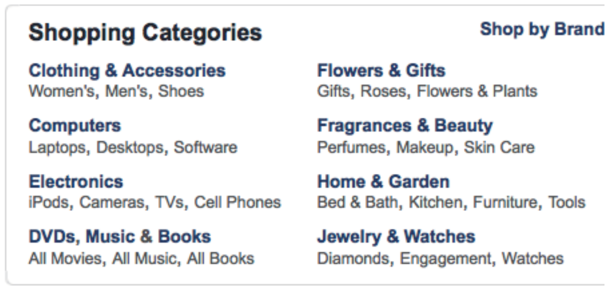
Shopping Categories
Shop by Brand
Clothing & Accessories
Women's, Men's, Shoes
Flowers & Gifts
Gifts, Roses, Flowers & Plants
Computers
Laptops, Desktops, Software
Fragrances & Beauty
Perfumes, Makeup, Skin Care
Electronics
iPods, Cameras, TVs, Cell Phones
Home & Garden
Bed & Bath, Kitchen, Furniture, Tools
DVDs, Music & Books
All Movies, All Music, All Books
Jewelry & Watches
Diamonds, Engagement, Watches

Figure 2: Category browsing in Yahoo! shopping

three more signals: keywords similarity of queries, string matching similarity, and query similarity derived from the click-through signal and the distance between clicked urls. They apply a density-based clustering approach DBSCAN [6], and apply the clustering result to identifying frequently asked queries.

Fonseca et al. [7] divide query log into sessions, where each session contains a sequence of related queries in a time interval. They propose discovering related queries based on association rules. Zaiane and Strilets [12] propose using more information including the similarity of clicked urls to detect related queries. There are also some studies inferring related queries by query expansion [1][11].

Most of existing works apply the query clustering result to query suggestion, finding frequently used queries, or to enhance search results. To our best knowledge, there has been no work applying it for building e-commerce hierarchical product structure. To achieve this, there are some special properties we need to consider. First, the searching log data in e-commerce products is much sparser and noisier compared to general web search. Second, since we aim at enhancing users' browsing experience, it becomes necessary to build a user-friendly, and self-explanatory structure.

3. OVERVIEW

In this paper, we take mobile app exploration as an example, and propose a general approach for optimizing user exploring experience by providing a more intuitive taxonomy for app browsing.

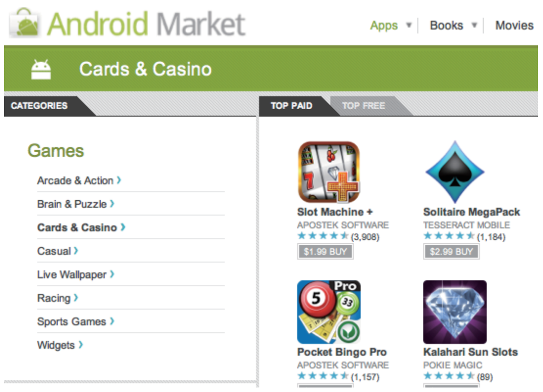

Figure 3: Category browsing in Android Market

3.1 Mobile App Exploration

As mobile devices including smartphones and tablet PCs become more and more popular in our daily life, mobile apps have exploded in popularity in the last three year. There are only 28,000 apps in iTunes app store in March 2009, while this number has grown to 500,000 in October 2011. We can foresee that mobile app service has great potential in the e-commerce market. In this paper, we use a newly launched mobile app discovery service[4] by Yahoo! as an example to show how the multi-level clusters generated by our proposed approach can improve the traditional category-based browsing experience.

Most of current mobile app discovery services have provided search and recommendation functions. However, the browsing methods are generally limited to looking through category-based top apps. For example, users can only browse the top free and top paid apps in about 30 pre-defined categories in Android Market, as shown in Figure 3. It is very difficult to define a complete taxonomy for such a new market. On one hand, mobile apps do not provide clear clues in a semantic hierarchical structure. On the other hand, thousands of new mobile apps emerge every day. It would be a highly laborious task to maintain such a taxonomy by manual work.

3.2 Approach and Evaluation

In this paper, we first propose to mine actual user queries and click-through logs to build a user-friendly hierarchical topic structure under the original app categories. The hierarchy is built by adopting a hybrid query clustering approach and can be further revised by editors.

Then we consider leveraging the derived hierarchy to assist users discovering interested items. To achieve that, we classify items for each cluster. In this paper we propose an intuitive and light-weight approach to conduct classification.

Finally we incorporate the hierarchical cluster and classified items into a new explore interface, by which users can take advantage of the fine-grained app categorization to navigate through large numbers of various mobile apps.

We evaluate both quality of hierarchical topic structure and actual usability of the proposed app exploration method. The professional search editors were invited to modify the hierarchical topic structure which was automatically generated by our clustering algorithm. We observe that only a

[4]http://apps.search.yahoo.com/

small number of modifications were made by editors to form a user-friendly app topic hierarchy. Furthermore, our app browsing method based on the topic hierarchy showed obvious superiority over the traditional category based ones.

4. A HYBRID APPROACH FOR QUERY CLUSTERING

4.1 Define Query Similarity

A critical issue for query clustering is how to define similarity between queries. A natural idea is to leverage queries' co-click information, which has been used in many existing works [2, 10, 6, 12]. Compared to human-defined taxonomy, a hierarchical structure derived based on users' click-through log is more intuitive to users, and more likely to catch users' preference. For example, in our mobile app exploration experiments, we detect several sub-categories, like "love", "quotes", "music", "ringtone", "yahoo", etc. for category "entertainment", which are all obvious user interested topics; while it is difficult for editors to define such taxonomy.

To take use of searching logs of new emerging e-commerce services, we investigated the data, and have several observations as follows,

- The click-through information in the product search of those emerging e-commerce services is very noisy and sparse. Taking the mobile app search as an example, we collected the click-through information of our AppSearch[5], and find that almost 85% query-app pairs result in only one click. Using this information directly will introduce too much noise, while filtering out query-app pairs with small click number will make the data too sparse.

- Keyword based query similarity yields many false positive example query pairs. For example, despite that "google talk" and "talking tom cat" is similar in term space, they belong to quite different topics. For another example, "word search" and "words with friend" are neither similar to each other though they yield overlaped keywords.

- It is sound to define query similarity by combining both co-click based and keyword based similarity. For instance, according to the app searching logs, all of three queries, "remote controller", "remote desk connection", and "t-mobile", result in clicks on the app "PC Remote Controller". However, "t-mobile" is not similar to the other two, which can be detected by checking keyword based similarity. In contrast, "remote desk connection" and "remote controller" share a word besides co-click an app, by which we can judge they are similar to each other.

Based on above observations, we define query similarity according to both their term similarity and co-click information,

$$cts(q_1, q_2) = ts(q_1, q_2)cs(q_1, q_2), \tag{1}$$

where q_1, q_2 are two queries, $cts(q_1, q_2)$ denotes the similarity between q_1 and q_2 derived from both click information and term similarity, $ts(q_1, q_2)$ represents term similarity between q_1 and q_2, and $cs(q_1, q_2)$ represents co-click based similarity between q_1 and q_2.

Term similarity is defined as

$$ts(q_1, q_2) = \begin{cases} 1, & \text{if } q_1 \text{ and } q_2 \text{ share at least one word} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

And co-click based similarity is defined as

$$cs(q_1, q_2) = \sum_i \min\{r(q_1, d_i), r(q_2, d_i)\}, \tag{3}$$

where d_i denotes document i, $n(q_1, d_i)$ represents the click number of query q_1 and document d_i, and $r(q_1, d_i)$ denotes the normalized click number for (q_1, d_i) pair, which is defined as

$$r(q_1, d_i) = \frac{n(q_1, d_i)}{\sum_j n(q_1, d_j)}, \tag{4}$$

similar normalization approach has been applied in previous work [3].

In Eq. (1), if two queries share at least one word, and both of them result in clicks at least one document, then we can say they are quite similar to each other. This is in accordance with our observation.

The similarity derived by Eq. (1) is much more reliable than using only click or term similarity. However, it still faces the challenge of sparseness, which means there are only a small portion of query pairs yielding similarity larger than zero. For example, in the AppSearch experiment, there are 12,191 queries, while only 6,263 queries have non-zero similarity with others. To further alleviate the problem, we consider using click-through log in general web search. We investigate web clicks whose queries are from mobile app's searching log, and find that,

- The click numbers for query-url pairs are much larger than those of emerging e-commerce searching log. For example, we find that the click number of 26% query-url pairs is larger than 5, while 51% query-url pairs larger than 2, which indicates that web click is a good signal to infer query similarity.

- The coverage of searching queries in e-commerce products is not very large in web search click log. In particular, we find that only about 50% queries in AppSearch result in clicks on at least one same document with other queries in general web search. Therefore, using only web search clicks is too sparse to define query similarity.

- About 84% query pairs derived from web search click are not covered in those derived from the e-commerce searching log. For example, we find that both query "yahoo chat" and query "messenger" result in clicks on the app "messenger" more than 50,000 times. However, there is very few of such signal in e-commerce searching log., which indicates that query similarity derived from general web search is helpful to enrich that derived from e-commerce searching log.

Based on above observations, we first filter query-url pairs whose click number is smaller than 2, and then define query similarity as follows,

$$wcs(q_1, q_2) = \sum_i \min\{r(q_1, u_i), r(q_2, u_i)\}, \tag{5}$$

[5]http://apps.search.yahoo.com

where $wcs(q_1, q_2)$ denotes the similarity between q_1 and q_2 derived from web search click, and

$$r(q_1, u_i) = \frac{n(q_1, u_i)}{\sum_j n(q_1 u_j)}$$

is the normalized clicked number for query q_1 and url u_i.

Finally, we combine these two similarities to compute the final similarity between two queries,

$$s(q_1, q_2) = cts(q_1, q_2) + wcs(q_1, q_2). \quad (6)$$

4.2 Hierarchical Query Clustering

In this section, we describe our hierarchical clustering approach. We first analyze the special properties of the clustering problem in our application, then we propose a hybrid hierarchical clustering approach, which can handle the properties.

Our hierarchical clustering problem has the following properties:

- Since we intend to employ the query clustering result to guide users to explore their interested queries, exclusive clustering is risky. For example, a user might want to search "talking tom cat ringtone", which is clustered into "ringtone"; however, there might be another cluster called "cat". In this case, users might feel unsatisfied with the cluster "cat" since they cannot find "talking tom cat ringtone" in this cluster. On the other side, if each query occurs in too many clusters, users will not gain good exploration experience either since it cannot tell apart different clusters and thus fails to save users efforts. To balance users efforts and hitting ratio is important.

- Usually there are two objectives for query clustering: minimizing similarity between clusters, and maximizing similarity within each cluster. In our problem, the second objective is more important than the first one. In one aspect, to reduce user hitting error, we allow one query belonging to multiple different clusters. In another aspect, it is necessary to avoid clustering dissimilar queries into one cluster as it will give rise to bad exploration experience for users.

Based on above observations, we propose a two-step hierarchical clustering approach,

1. Clustering queries into small clusters (referred to as *cluster L2*);

2. Clustering *cluster L2* into larger clusters (referred to as *cluster L1*).

Table 1 lists an example of the clustering result. Next we discuss how to conduct clustering in details.

4.2.1 Generating Cluster L2

In the first step, we apply the spectral clustering approach to cluster user queries into *cluster L2*. Spectral clustering [4][9] is a state-of-the-art approach to conduct clustering on graph. It converts the clustering process into an integer optimization problem, which is NP-hard. To conduct clustering efficiently, one has to relax the integer constraints [8][9] or make other approximation [5]. These relaxation or approximation will introduce errors to clustering results.

Considering the problem introduced by relaxation or approximation, and our emphasis on maximizing similarity within cluster, we conduct a postpone process. That is, for each cluster resulted from spectral clustering, we check whether the query similarity graph within each cluster has only one connected component. If not, we split the cluster to ensure that each cluster has only one connected component.

This will generate many small clusters (e.g. which has fewer than 3 user queries). For example, a *cluster L2* contains queries "love tester", "positive quotes", "friendship quotes", and "missing you quotes". We checked the similarity graph, and found that "love tester" is not connected to other queries. Therefore, we split this cluster into two clusters. Meanwhile, we find there is another cluster containing "love test games", "love compatibility test", "I love you", "I love you sms", which are all very similar to "love tester". In such a case, we will merge the query "love tester" into the larger one.

To name each *cluster L2*, we take the following steps. First, we build a vocabulary for each cluster, and select k most frequent words. Then we re-order the selected words according to its original order in queries within the cluster. For example, for the cluster with queries "gta cheats ps2", "cheats for grand theft auto cheat codes ", "cheat in gta", and "gta san andreas cheats", we name it as "gta cheats".

4.2.2 Generating Cluster L1

Now we consider clustering *cluster L2* into *cluster L1*. As discussed in the beginning of section 4.2, exclusive clustering is risky for users to browse. We propose generating *cluster L1* as follows. First we generate a vocabulary according to names of *cluster L2*. Then for each word in the vocabulary, we build a *cluster L1*, which consists of *cluster L2* whose names contain the word.

The idea behind this approach is that, *cluster L2* with shared name words may talk about different aspects of the same topic. For example, "birthday quotes", "love quotes", "funny quotes", and "famous quotes" all talk about quotes, while they specify different aspects of "quotes". Similarly, "love letters", "love poems", "love quotes", and "love songs" are all related to "love", but of different aspects of "love".

Directly using above approach has some problems. First, two words might have very similar *cluster L2* sets. For example, "angry" and "bird" both correspond to different aspects of the similar *cluster L2*, i.e. "angry bird" games. Second, some word might contain very few *cluster L2*. For example, "4shared" only contains one *cluster L2*.

To address these problems, we also conduct a postpone processing step. First, if subsets of two *cluster L1* are very similar to each other, we merge them together. Second, those *cluster L1* which contains fewer than two *cluster L2* are all grouped into one cluster, which is named as "others".

To sum up, we propose a hybrid hierarchical clustering approach, which satisfies the following properties,

- It allows a *cluster L2* belonging to multiple different *cluster L1*, which can help users find their interested queries quickly,

- It checks query similarity within each cluster, which ensures that queries within each cluster are similar to each other.

We also introduce the editorial intervention to ensure the better readability and structure of clusters. The professional

Table 1: An example part of our new hierarchical clusters

category	*cluster L1*	*cluster L2*	*user queries*
Entertainment	movies	bruce-lee-movies	bruce lee martial arts movies ...
		netflix-movies	netflix movies instantly netflix for tv ...
		...	...
	music	mp3-music	mp3 music mp3 songs ...
		yahoo-music	yahoo music music ...
		...	...

Table 2: Guidelines for cluster editing

Level	Action	Description
user queries	delete	Remove the queries which are supposed not to belong to the *cluster L2*
cluster L2	delete	Remove the clusters which are supposed not to belong to the *cluster L1*
	rename	Rename the cluster name if the editor think the current name is not really suitable or readable for this cluster.
	merge	Merge the highly similar clusters in the *cluster L2* level into one
cluster L1	rename	Rename the cluster name if the editor think the current name is not really suitable or readable for this cluster
	merge	Merge the highly similar clusters in the *cluster L2* level into one

search editors are asked to re-formulate the cluster names if they think the automatically generated ones are incorrect or not readable. In addition, two other cluster editing actions *delete* and *merge* are requested to make the hierarchical topics look like a neat mobile apps taxonomy. The allowed editorial actions on each topic hierarchy level are list in Table 2. The evaluation result in Section 6.2 indicates that only minor editorial effort is needed to complete a user-friendly taxonomy for mobile apps based on the generated query cluster hierarchy.

5. AUTOMATIC CONSTRUCTION OF BROWSING LISTS

To assist users browsing interested items, there is still another critical challenge, besides building an intuitive hierarchical structure, about how to build browsing lists for all clusters, i.e. grouping items into new generated clusters, and ranking them within each cluster. Intuitively, this can be viewed as a classification plus ranking problem. However, generic classification and ranking problems usually require great human effort and machine resource. Especially, in our new hierarchical structure, each cluster takes individual learning process to obtain its own dedicated model, which is too expensive in practice. To address this challenge, we propose a straightforward approach for automatic product classification and ranking, which requires quite little effort.

Specifically, for each cluster, we include those items from two sources. First, we rank frequently clicked items of queries in the cluster on the top. Next, we retrieve highly ranked items using queries in the cluster, and blend the results of all queries according to each item's position and the frequency of its queries.

This idea is quite intuitive, since both frequently clicked items and highly ranked items of top queries are relevant to the corresponding cluster; meanwhile it succeeds in saving tedious classification effort and providing sound results.

We then aggregate the multi-level clusters, which are generated by our proposed algorithm, and the classified items into a hierarchical category based browsing interface. We design such an interface for mobile apps exploration to demonstrate its practical potentials to enhance user exploration experience. As shown in the Figure 4, we introduce a four level hierarchy in the left panel, including *category*, *cluster L1*, *cluster L2* and *user queries*, to assist users to browse the mobile apps by category. For each cluster in any level, we list classified items in the right panel.

6. EVALUATION

In this section, we present several experiments to evaluate the hierarchical structure derived in Section 4 and the browsing lists constructed as in Section 5. The experimental results demonstrate how effectively our new hierarchical topics can organize queries according to user intent and actual clicks, as well as how efficiently our new hierarchical topics can assist users to discover their interested items. In particular, we first introduce the datasets we used in our experiment in Section 6.1, then we present the quantitative evaluations of our new hierarchical topics based on user query clusters in Section 6.2; and finally we demonstrate a couple of quantitative evaluations of the efficiency that users can achieve by using our new hierarchical topics in Section 6.3.

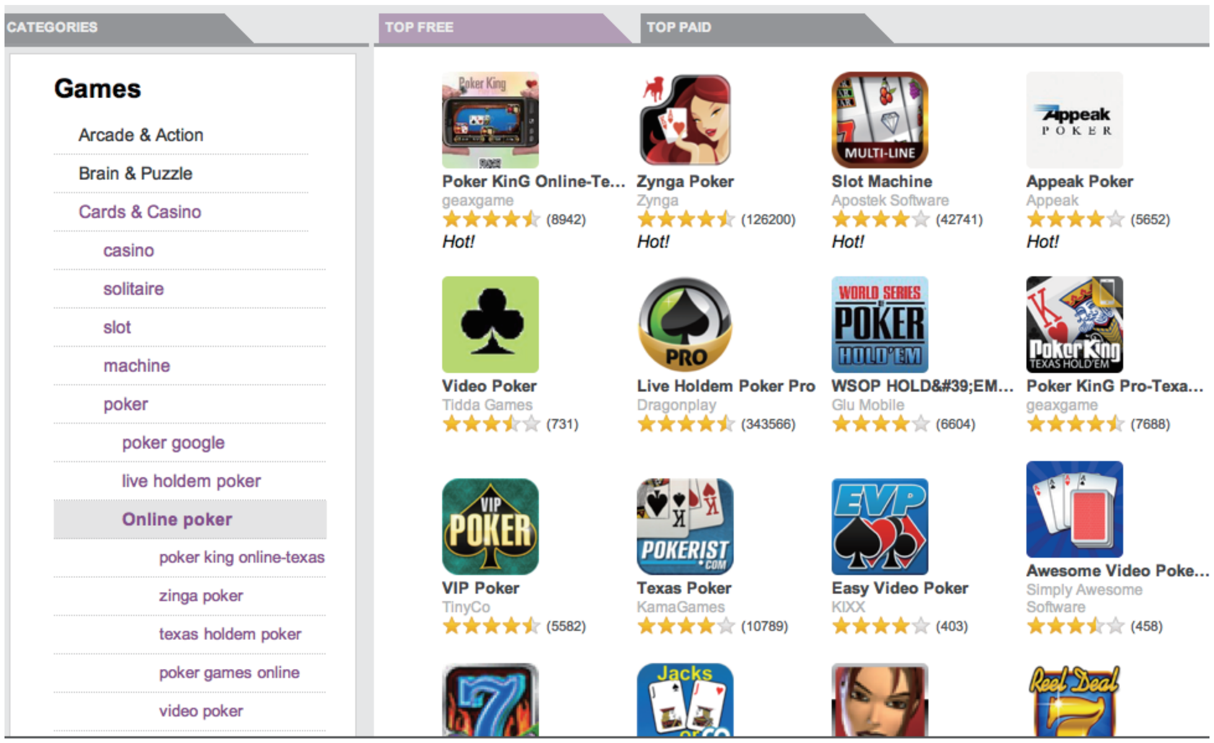

Figure 4: Mobile apps category browsing interface

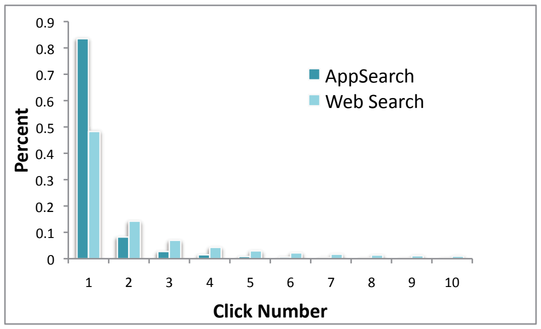

Figure 5: Click number distribution

6.1 Datasets

We extracted user queries and click information from the most recent user log of Yahoo! App Search (a vertical search channel to find mobile apps), ranging from September 2nd to October 12th in 2011. There are 12,191 queries, and 40,912 query-app pairs in total. The distribution of the click number for query-app pairs is described in Figure 5. As discussed in section 4.1, the click through data is highly sparse since this vertical search channel is launched very recently and still has not accumulated large amount of user data yet.

We also extracted query-url pairs from Yahoo! Web Search[6] click-through log. The distribution of click numbers is plotted in Figure 5. We can find that the click number is much larger than that of App Search, which indicates co-click in web search is a good signal for computing query similarity.

We filtered App Search data according to Eq. (1), and removed query-url pairs with click number less than 2 from Yahoo! Web Search click-through log. The statistics of query similarity derived from App Search and Web Search are listed in Table 3. After filtering, there were 21,910 query pairs for App Search, which include 6,263 queries. In addition, we extracted 54, 342 query pairs from the Web Search log, which include 5,919 queries.

Table 3: Query similarity statistics

	#query	#query pair
AppSearch	6,263	21,910
Web search	5,919	54,342

6.2 Qualitative Evaluation of Hierarchical Clusters

We use Graclus [5] to conduct spectral clustering. The number of clusters was set to ensure that there are 5 queries in each *cluster L2* on average. The statistics of clustering results are listed in Table 4. Our proposed clustering approach generated 3.0 queries per *cluster L2* and 3.7 *cluster L2* per *cluster L1* on average. The relatively small number was selected to ensure items within each cluster are similar to each other. There are on average 12.5 *cluster L1* for each category, indicating variety of topics within each category.

In order to evaluate the quality of the hierarchical clusters automatically generated by our proposed algorithm, we recorded the editorial cluster editing actions which are de-

[6]http://search.yahoo.com

Table 4: Clustering statistics before and after editorial intervention

	Before	After
#*cluster L1* per category	12.5	11.8
#*cluster L2* per *cluster L1*	3.7	3.7
#*user queries* per *cluster L2*	3.0	2.9

fined in in Table 2. As described in Section 4.2, the professional search editors were requested to judge the quality of each level of the generated hierarchical clusters based on cluster names and the corresponding included queries. The editors were able to perform three possible actions (*rename*, *delete* and *merge*) in different levels to optimize the hierarchical clusters and ideally build a complete taxonomy of mobile apps. The editorial guidelines are shown in Table 2.

We compare the hierarchical clusters before and after editor intervention in Table 4. It shows that all statistics do not change much after editorial modification. This indicates that from editor's point of view, only small modifications on the clustering result are needed to make the category structure readable and reasonable for user exploration.

To inspect the evaluation in details, we count the number of each type of action on each level and compute the average value over the category. Table 5 demonstrates the average number of different types of actions per category on each level.

Table 5: Average number of editors' actions to optimize hierarchical taxonomy per category

Action	*cluster L1*	*cluster L2*	*user queries*
delete	-	0.27	23
rename	0.07	2.23	-
merge	0.03	0.1	-

From the table, we can find that the average number of actions editors took to optimize the taxonomy is very small, especially on *cluster L1* and *cluster L2* levels. In particular, there is nearly no *'rename'* or *'merge'* actions on *cluster L1*; and, the average numbers of *'delete'* and *'merge'* actions on *cluster L2* are also close to zero, which indicates that our new hierarchical topics can effectively represent the users' intent. The table also illustrates that, on *cluster L2*, we need about 2.23 actions on average to rename the sub-cluster per category. In most of cases, sub-clusters need to be renamed because the user queries included in the sub-cluster are ambiguous. Moreover, we find that the number of *'delete'* actions on the level *user queries* (avg. 23) is also much smaller than the average number of queries per category (avg. 139).

6.3 Side-by-Side Quantitative Evaluations Compared with Android Market

We also conducted the side-by-side quantitative evaluation, compared with category browsing interface in Android Market, to demonstrate how the new hierarchical clusters can assist users to more efficiently discover their interested mobile apps. To minimize the possible biases in two app browsing interfaces, we tried to adopt the same layout and design as the ones in Android Market, including same categories, same app result tabs (top paid and top free), same app icon sizes and same number of apps per page. The difference between the two interfaces is that Android Market only has one level of flat category but our interface provides three levels of hierarchies (*cluster L1*, *cluster L2* and *user queries*) under each category.

Four professional search editors (different editors from the ones who were involved in editing hierarchical clusters) participated in this side-by-side evaluation. They are from US and all proficient Android app users. Three of them are native English speakers. They had long experience in conducting SBS evaluations for web search. They were invited to browse the apps by category based navigation using both Android Market interface [7] and our app browsing interface designed in Section 5. The editors were asked to open two interfaces side by side, finish a browsing session (2 minutes) for a category on one interface and then start another browsing session for the same category on the other interface. Two editors first used Android Market's interface to browse a category while the other two used our interface first, in oder to reduce the bias of possible stronger first impression.

6.3.1 *Efficiency of Assisting Users to Discover Interested Apps*

The first direct comparison we conducted is efficiency of two types of category structures to assist users to discover their interested apps. After two comparative browsing sessions of a category, the editors were requested to record how many apps which they actually scanned or clicked were not their interested ones in the two interfaces separately, as well as how many interested apps they found in the two interfaces.

We observe that, by using our interface, editors skipped less apps on 56% of the categories and found more interested apps on 60% categories. In both efficiency tests, the categories in which our interface shows superiority include arcade&action, business, cards&casino, communication, entertainment, finance, libraries&demo, lifestyle, news&magazines, photography, sports games and tools. The categories in which our interface performed worse for both tests are brain&puzzle, education, medical, music&audio, shopping, social, travel&local and weather.

6.3.2 *Effectiveness of "Hot" App Symbols*

In our category browsing interface, we added a new type of information, the "hot" symbol, to represent the hotness of apps based on the click number in our user log. To investigate whether this new information intrigues more user's interest, we asked the editors to provide their feedbacks on the hot symbols. In total, the editors reported that those hot symbols did affect and arouse their interests to click the apps in 51.7% of categories. However, we notice the editors showed obvious discrepancy in this test. Two of them showed the preference of the "hot" symbol in about 90% of the categories while the other two didn't show the interest in near 90% of the categories. This initial result indicates the "hot" symbol may be a highly user dependent feature and would be more helpful as a personalized option.

6.3.3 *Effectiveness of Sub-categories*

Based on the hierarchical clusters, our interface provides multi-level sub-categories under each original categories of

[7]https://market.android.com/apps/GAME

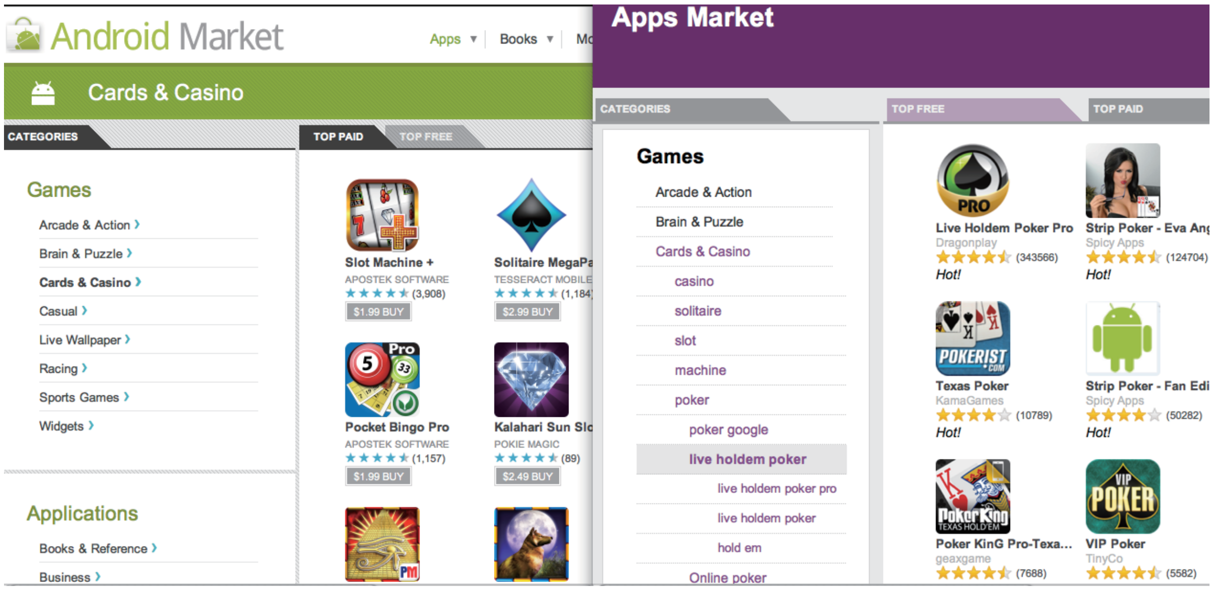

Figure 6: Interfaces in the SBS evaluation

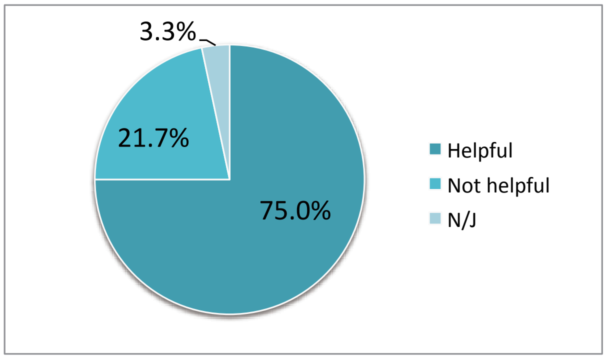

Figure 7: Percentage of sub-categories preference

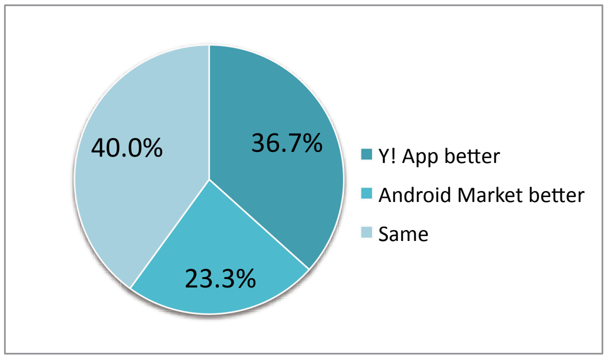

Figure 8: Percentage of browsing method preference

Android Market. The editors were requested to answer whether the added sub-categories were helpful for their app browsing. The editors reached general agreement on this question. As shown in Figure 7, editors thought that the added sub-categories are helpful on about 75% of the categories.

6.3.4 *Overall Preference and Other Feedbacks*

Besides the above quantitative questions, we also asked the editors if they had an overall preference between the two interfaces for each category browsing session. Figure 8 illustrates the respective percentage of categories in which editors preferred one browsing interface to the other or had no preference. From the figure, we can observe that editors were more satisfied with our browsing interface in 36.7% of the categories, with Android Market in 23.3% of the categories, and show no preference in 40% of the categories. This clearly verifies superiority of our browsing interface.

We further questioned the reason why the editors preferred our browsing interface. Most editors mentioned that the sub-category browsing "is quite helpful". Since with so many available apps, they "don't want to search through all the irrelevant apps that come up in the broad categories". Some editors also said that they "like the Hot symbol", and sometimes they "skipped those apps without hot symbol" when they were browsing a category.

7. CONCLUSION AND FUTURE WORK

In this paper we have proposed to optimize user exploring experience in e-commerce services by providing users more intuitive browsing categories. Specifically, on the backend, we proposed to build hierarchical structure automatically based on users' searching log, while on the frontend, we presented users with both frequently clicked items and highly ranked ones for each category, and incorporated the derived structure and associated items for each category to a new browsing interface. Experimental results on a real-world e-commerce service verify the effectiveness of our hierarchical clustering approach, and further comparisons also demonstrate that our new interface is superior than the existing category browsing service.

In the future, we are interested in investigating more query similarity signals (e.g. similarity between retrieved results of queries) beyond the three proposed ones to further alleviate the problem regarding the sparsity and noise of searching log data. Besides, we aim at studying how to evaluate the effectiveness and efficiency of the new browsing interface automatically, rather than leveraging editorial human judgement in our current work.

8. REFERENCES

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[2] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[3] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[4] P. Chan, M. Schlag, and J. Zien. Spectral k-way ratio-cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(9):1088–1096, 1994.

[5] I. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1944–1957, 2007.

[6] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, volume 1996, pages 226–231. Portland: AAAI Press, 1996.

[7] B. Fonseca, P. Golgher, E. De Moura, and N. Ziviani. Using association rules to discover search engines related queries. 2003.

[8] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference*, pages 849–856, 2001.

[9] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[10] J. Wen, J. Nie, and H. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, pages 162–168. ACM, 2001.

[11] J. Xu and W. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems (TOIS)*, 18(1):79–112, 2000.

[12] O. Zaïane and A. Strilets. Finding similar queries to satisfy searches based on query traces. *Advances in Object-Oriented Information Systems*, pages 349–359, 2002.