

Rank-Aware Clustering of Structured Datasets

Julia Stoyanovich^{*}
Columbia University
New York, NY, USA
jds1@cs.columbia.edu

Sihem Amer-Yahia
Yahoo! Research
New York, NY, USA
sihem@yahoo-inc.com

ABSTRACT

In online applications such as Yahoo! Personals and Yahoo! Real Estate users define structured profiles in order to find potentially interesting matches. Typically, profiles are evaluated against large datasets and produce thousands of matches. In addition to filtering, users also specify ranking in their profile, and matches are returned in a ranked list. Top results in a list are typically homogeneous, which hinders data exploration. For example, a user looking for 1- or 2-bedroom apartments sorted by price will see a large number of cheap 1-bedrooms in undesirable neighborhoods before seeing a different apartment. An alternative to ranking is to group matches on common attribute values, e.g., cheap 1-bedrooms in good neighborhoods, 2-bedrooms with 2 baths, and choose groups in relationship with ranking. In this paper, we present a novel paradigm of rank-aware clustering, and demonstrate its effectiveness on a large dataset from Yahoo! Personals, a leading online dating site.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Clustering; H.3.5 [On-line Information Services]: Web-based services; H.5.3 [Group and Organization Interfaces]: Web-based interaction

General Terms: Design

Keywords: information filtering, information presentation, rank-aware clustering, structured datasets.

1. INTRODUCTION

In online applications with large structured datasets, e.g. Yahoo! Personals and Yahoo! Real Estate, there are often thousands of high-quality items, in this case, persons and apartments, that satisfy a user's information need. Users typically specify a structured target profile in the form of attribute-value pairs, which is used to *filter* items. On dating sites, a target profile may specify the age, height, income, education, political affiliation, and religion of a potential

^{*}Research supported in part by National Institute of Health grant 5 U54 CA121852-03

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'09, November 2–6, 2009, Hong Kong, China.

Copyright 2009 ACM 978-1-60558-512-3/09/11 ...\$10.00.

match. In real estate applications, a profile describes a user's dream home by its location, size, and number of bedrooms. The number of matches to a specific profile is often very high, making *data exploration* an interesting challenge.

Typically users also specify ranking criteria which are used to *rank* matches. In Yahoo! Personals potential matches may be ranked by decreasing income or increasing age, while in Yahoo! Real Estate houses may be ranked by increasing price or decreasing size. Ranking helps users navigate the set of results by limiting the number of items that they see at any one time, and making sure that the items users see first are of high quality. However, ranking also brings the disadvantage of *match homogeneity*: the user is often required to go through a large number of similar items before finding the next different item. This is illustrated in the following fictional example inspired by Yahoo! Personals.

EXAMPLE 1.1. *Mike is looking for a date with some college education, 20 to 30 years old, and he wants to see matches sorted on decreasing income. When inspecting the results, Mike notices that the top ranks are dominated by women in their late-twenties with a Master's degree. It takes Mike a while to scroll down to the next set of matches which are different from the top-ranking ones. In doing so, he skips over some unexpected cases such as younger women with higher education and income levels, or women with high income who did not graduate from college. After additional data exploration, Mike realizes that there is a correlation between age, education (filtering), and income (ranking.) Such correlations would have been obvious if data were presented in labeled groups such as [20–24] year-olds with a bachelor's degree, [25–30] year-olds who make over 75K, etc. The user would then explore the interesting groups by looking at the ranked lists of results within the groups.*

A key takeaway is that a user browsing a result set sequentially, item by item, is only able to understand trends in the data after seeing a significant number of items. Moreover, the difficulty of sequential exploration increases with more sophisticated rankings. For example, the score of a match on a dating site could be proportional to its income and inversely proportional to the distance from the user's geographic location. Helping users navigate their results more effectively is even more important in this case, given that correlations between filtering and ranking are less obvious.

1.1 Limitations of Clustering Algorithms

There are many families of clustering algorithms that produce labeled groups from a dataset. In domains like Yahoo! Personals, where datasets are large and all items are

# beds	# baths	size (ft ²)	price (\$)	# apts
1	1	600	1800	5
	1.5	700	2100	55
	1	750	2900	25
2	1.5	700	2200	30
	1	800	2400	60
	2	800	2850	10
3	2	950	3500	100
	1.5	950	2900	5
	2	1000	3200	10

Table 1: A fictional real estate database.

described by a large number of attributes, it is intuitive to use *subspace clustering* to find clusters of items with meaningful descriptions.

Subspace clustering is an extension of traditional clustering that seeks to find clusters in different subspaces of a dataset [7]. Clusters of items are *high-quality* regions identified in multiple, possibly overlapping, subspaces. Many subspace clustering algorithms use the *density of a region* as a quality measure. In the simplest case, density is the percentage of data points that fall within a particular region, and the algorithm aims to find all regions that have density higher than a pre-defined threshold. We now illustrate with an example how one of the first subspace clustering algorithms, CLIQUE [1], uses density to identify clusters.

EXAMPLE 1.2. Consider a fictional real estate example in Table 1: a database of 300 rental apartments, listing the number of bedrooms, number of bathrooms, size in ft², monthly rental price, and the number of such apartments currently on the market. Mary is looking for an apartment that is at least 600-ft² in size and has at least one bedroom, and she wants the matches sorted on price in increasing order. All apartments in Table 1 match Mary’s profile.

Assume a density threshold $\theta = 0.1$. A density-based subspace clustering algorithm starts by dividing the range of values along each dimension (attribute) into cells, and by computing the density in each cell. For example, each distinct value of #beds, size, and #baths may correspond to a cell, and price may be broken down into intervals (1500, 2000], (2000, 2500], (2500, 3000], and (3000, 3500]. Cells that do not pass the density threshold are pruned at this stage. The algorithm immediately prunes 600-ft² apartments ($\frac{5}{300} < \theta$), 750-ft² apartments ($\frac{25}{300} < \theta$), 1000-ft² apartments ($\frac{10}{300} < \theta$), and apartments in the (1500, 2000] price range ($\frac{5}{300} < \theta$.) Given Mary’s interest in cheaper apartments (price is her ranking condition), it is problematic that the cheapest apartments in the dataset, the 600-ft² apartments that cost \$1800, are pruned.

Having identified dense cells, the algorithm proceeds to *merge* runs of neighboring cells in each dimension.

Next, the algorithm progressively explores clusters in higher dimensions by *joining* lower-dimensional ones. For example, the 1-dimensional cluster of 800-ft² apartments (70 items) can be joined with the 1-dimensional cluster of apartments in the (2000, 2500] price range (145 items.) The result of this join is a region with 60 800-ft² 2-bedrooms at \$2400 per month, which qualifies as a cluster since it passes the density threshold. However, the region that results from joining the 950-ft² apartments (105 items) with apartments in the (2500, 3000] price range (40 items) does not qualify as

a cluster (it contains only 5 items) and is pruned, losing the potentially interesting 3-bedrooms for a relatively low price (\$2900.) Density decreases in higher dimensions and the algorithm stops when there are no more clusters to explore.

1.2 Challenges of Rank-Aware Clustering

A lower density threshold would evidently guarantee that some of the regions pruned using a higher threshold would be preserved. However, if the threshold is set too low, the algorithm would keep merging regions, ultimately identifying much larger clusters, and possibly one cluster containing the entire dataset. Moreover, not all regions that pass a typical clustering *quality metric*, e.g., density or entropy, are equally interesting to the user. Indeed, given a scoring function, some items, and hence some clusters, are more desirable than others (e.g., *Mary* has little interest in the 2-bedroom apartments that cost \$3500, but would like to see the 1-bedrooms that cost \$1800.) Even when the density of a region is high, as is the case with 2-bedroom apartments for \$3500, *Mary* would probably have less interest in them than in cheaper apartments. Therefore, we propose to explore density measures that account for item scores and ranks when assessing the quality of a cluster.

2. RANK-AWARE CLUSTERING

In an on-line data exploration scenario, a user specifies a profile composed of filtering and ranking criteria. We assume that the user’s filtering conditions result in a dataset \mathcal{D} (and can thus take users out of the notation since we are interested in one user at a time.) Items in \mathcal{D} are described using attribute-value pairs, including a special attribute *id* which uniquely identifies each item. Attributes belong to a set \mathcal{A} . Ranking is expressed by a scoring function \mathcal{S} which assigns a score *i.score* to each item $i \in \mathcal{D}$. We denote by $\mathcal{S}(\mathcal{D})$ the set of all items from \mathcal{D} augmented with *i.score*. Typically, items are presented to the user as a single ranked list sorted by score. We first argue that rank-unaware clustering algorithms (see Section 1.1) are inappropriate when users are interested in exploring ranked datasets.

EXAMPLE 2.1. Consider user *Mary* from Example 1.2. *Mary* is interested in seeing apartments ranked by price in increasing order. *Ann*, another user who shares *Mary*’s filtering conditions, may be interested in seeing the same apartments sorted by size in decreasing order. Which clusters are best for which user depends on the user’s ranking preferences. One reasonable option is to cluster apartments based on the scoring attribute. In particular, *Ann* may appreciate seeing the 950-ft² apartments which cost \$2900 in the same cluster as the same-size apartments for \$3500, while *Mary* may prefer to see 950-ft² apartments grouped together with the same-priced 750-ft² apartments. A subspace clustering measure that does not account for item scores would not distinguish between these two users, and would therefore be inappropriate for rank-aware data exploration.

The *score* of each item can be treated as an additional attribute and can thus be used for clustering. However, as we argue in the following example, using scores as an additional clustering dimension still fails to effectively address data exploration for ranked datasets.

EXAMPLE 2.2. Consider again Example 1.2, where *Mary* wants to sort apartments by price. If item price is used as

a clustering dimension, in the same way as other attributes, then Mary may see a high number of clusters, not all of which are of potential interest to her: e.g. a cluster of expensive 2-bedroom apartments may appear alongside a cluster of cheap 2-bedrooms. If many clusters are discovered by the algorithm, the potentially more interesting ones may go unnoticed. Worse yet, the algorithm may decide to merge together intervals that are of high interest to Mary with those of low interest, resulting in a potentially large heterogeneous cluster with homogeneous results dominating the top ranks.

Hence, we explore new clustering quality measures that use item scores and ranks to assess quality.

A region \mathcal{G} is a set of items labeled with a conjunction of predicates over attributes in \mathcal{A} , which, if evaluated on the dataset \mathcal{D} , results in all items in the region. The dimensionality of a region is the number of predicates that describe it. Any subset of predicates that define a region \mathcal{G} is a *sub-region* of \mathcal{G} . A region qualifies as a *cluster* if it satisfies a *clustering quality measure*. We use $\mathcal{S}(\mathcal{G}, N)$ to denote the N highest scoring items in $\mathcal{S}(\mathcal{G})$. N is a parameter in our formalism that models the user’s *attention span* – the number of items the user is likely to explore sequentially [6]. This parameter can be customized per user, or it can be set to reflect the preferences of an average user.

Our first rank-aware measure, \mathcal{Q}_{topN} , builds on the assumption that users are interested in clusters that contain high-scoring items, and that they will only explore the best items in those clusters. \mathcal{Q}_{topN} states that a multi-dimensional region \mathcal{G} is a cluster if it contains enough items that are in the top- N of each of its one-dimensional sub-regions.

$$\mathcal{Q}_{topN} : \frac{|\mathcal{S}(\mathcal{G}, N) \cap \mathcal{S}(P_1, N) \cap \dots \cap \mathcal{S}(P_m, N)|}{|N|} \geq \theta \quad (1)$$

\mathcal{Q}_{topN} aims to discover attribute correlations among the high-scoring items in the dataset. We illustrate how this measure compares to density using Example 1.2. Recall that user *Mary* specified price as the ranking condition.

The join of the 700- ft^2 cluster with the (2000, 2500] price range cluster preserves the lower-priced 1-bedrooms, since the top- N items in the join correspond to the high-scoring items in the (2000, 2500] cluster (one of the sub-regions) and to the high-scoring items in the 700- ft^2 cluster (its other sub-region.) On the other hand, the join of 2-bathroom apartments with 950- ft^2 apartments would not contain any of the cheapest 2-bathroom apartments in its top- N and would thus not qualify as a cluster.

The second measure, which we call $\mathcal{Q}_{SCORE\&RANK}$, models the intuition that a region with exceptionally high-scoring items in high ranks may be just as interesting to the user as a region in which items have intermediate scores. We define this measure using nDCG (Normalized Discounted Cumulative Gain) [4], with $\mathcal{S}(\cup_k P_k, N)$ as the ideal vector.

$$\mathcal{Q}_{SCORE\&RANK} : AVG_{r \leq N} nDCG(\mathcal{S}(\mathcal{G}, N), \mathcal{S}(\cup_k P_k, N))[r] \quad (2)$$

Consider the *gain vector* $V = \pi_{id, score} \mathcal{S}(\mathcal{G}, N)$. V is sorted by score in descending order. We derive the *cumulative gain vector* CG from V by assigning to each position in $CG[j]$ the sum of scores of items in V up to and including j : $CG[j] = \sum_{k \leq j} V[k]$. (All arrays are 1-based.) To capture the intuition that the user is more likely to pay attention

to items in higher ranks, we derive the *discounted cumulative gain vector* DCG as $DCG[k] = CG[k]/\log_b k$. The constant b controls the rank-based discount: higher values of b correspond to lower discount. There is no discount for positions $k \leq b$, where $DCG[k] = CG[k]$. To make discounted cumulative gain comparable across regions, we introduce a normalization factor: each position in the DCG vector for a region \mathcal{G} is normalized by the corresponding value of DCG_{Ideal} computed w.r.t. the *ideal gain vector*.

Consider again Example 1.2 and *Mary*’s scoring function \mathcal{S}_{Mary} , and let us take $N = 5$. Let us compute the nDCG for the 1.5-bathroom apartments with the size of 950- ft^2 . The ideal gain vector consists of scores of the 5 best items that either have 1.5 bathrooms or are 950- ft^2 in size, namely, the 1.5-bath 700- ft^2 apartments that cost \$2100 per month (*i.score* = 0.82.) With $b = 2$ we derive: $DCG_{Ideal} = [0.82 \ 1.64 \ 1.55 \ 1.64 \ 1.77]$.

Let us now compute the nDCG for the 950- ft^2 1.5-bath apartments. The top-5 list of this region consists of five 3-bedroom apartments at \$2900 (*i.score* = 0.35.) We derive $DCG = [0.35 \ 0.7 \ 0.66 \ 0.7 \ 0.75]$. We now normalize each position in DCG by the corresponding position in DCG_{Ideal} , average the values, and arrive at $nDCG = 0.43$.

3. QUALITATIVE ANALYSIS

We studied the properties of several ranking functions on a dataset from Yahoo! Personals, a leading on-line dating service with millions of users. Users create a *personal profile* in which they describe themselves using 30 structured attributes, e.g., age, height, occupation, education, income, etc. Users commonly store one or several *target profiles*, expressed in terms of the same attributes. Attributes in a target profile are designated as *required* or *desirable*. Required attributes are used for *filtering* – exact matching against personal profiles, and desirable attributes are used for *ranking*.

The ranking functions we consider use 6 attributes which induce a natural order on their values: *age*, *height*, *body type*, *education*, *income*, and *religious services* (the frequency with which the user attends religious services.)

Our first scoring function, called *attribute-rank*, assigns equal weights to each ranking attribute, and computes the score of an item as the sum of distances between the item and the ideal item along each attribute dimension. Here, an ideal item has the best possible value for each ranking attribute among items in the filtered dataset. The best value for each attribute is determined by the user’s ranking conditions. Distances along each dimension are normalized by the difference between extreme values for the corresponding attribute found in the filtered dataset. Note that this function is *personalized* in two ways. First, the user specifies which attributes are included in the scoring function. Second, the value of each ranking attribute contributes to the score based on how it compares to the best and worst values for that attribute, from among items that pass the filtering conditions of the target profile.

The second function, *geo-rank*, scales the value returned by *attribute-rank* by the geographic distance between the user and the match: $geo_rank = \frac{attribute_rank}{1 + geo_distance}$.

3.1 Clustering Quality

We now give a qualitative intuition of the kinds of clusters that may be discovered by rank-aware clustering for

the *attribute-rank* ranking function. We use a profile that is in-line with Example 1.1: $age \in [25, 35]$, $height \in [160cm, 175cm]$, $education \geq Bachelor's$, $ethnicity = Caucasian$, $body\ type \in \{slim, slender, average, athletic, fit\}$.

These conditions return over 30,000 matches on Yahoo! Personals. We rank the matches on a combination of *income* and *education*, both from higher to lower. There are about 100 top-matches: women with post-graduate education who make more than \$150K. About two thirds of the top matches are over 29 years old, and so younger matches are not easily visible in a ranked list.

The following rank-aware clusters deal directly with the correlation between income and age, and income and education: $age \in [25, 27] \wedge income \in [\$35K, \$75K]$, $age \in [28, 33] \wedge income \in [\$75K, \$150K]$, $age \in [28, 33] \wedge income \geq \$150K$, $age \in [31, 35] \wedge income \geq \$75K$, $age \in [28, 33] \wedge education = post\ graduate$, $age \in [31, 35] \wedge education = post\ graduate$.

Note that two clusters contain different sets of matches with age between 28 and 33. Note also that younger matches, age 25 to 27, are in a cluster with relatively lower income, and would not have been easily accessible in a single ranked list. Other clusters may be discovered that are not directly related to the ranking conditions, but for which a correlation exists among attributes at top ranks. So, there would be a cluster of matches who are politically *very conservative* or *conservative* and who attend religious services *more than once a week* or *weekly*. Another cluster may contain matches who are politically *middle of the road* or *liberal* and who attend religious services no more often than *monthly*.

As we illustrated, the clustering outcome depends on the ranking attributes, and how they correlate with other attributes in the data. Let us now consider how the distribution of scores imposed by the scoring function influences the applicability of rank-aware clustering, and the choice of a clustering quality measure.

3.2 Applicability of Rank-Aware Clustering

Since the clustering outcome depends on the combination of a user’s filtering conditions and the distribution of scores imposed by his scoring function, it is not always possible to find a meaningful clustering. Intuitively, rank-aware clustering does not apply if ranking does not discriminate well between high-quality and low-quality results. So, selecting matches with $income = 50K$, and then ranking on income, is not helpful, since all matches tie for the same score.

We now demonstrate the qualitative difference between Q_{topN} and $Q_{SCORE\&RANK}$, using the profile of one particular user, whom we call $user_1$, as an example.

Ideally, a rank-aware measure should be rich enough to capture the distribution of scores in the result. Q_{topN} treats all items with N highest scores equally, and is appropriate for scoring functions where the *best N items have the highest scores*, but where *there is no significant variability in scores among the top- N* . This is the case for the function *attribute-rank* for $user_1$, as is demonstrated in Figure 1.

Conversely, $Q_{SCORE\&RANK}$ is most meaningful if *there is significant variability in scores among the top- N* . So, for $N = 100$ the best 10 items may have much higher scores than the following 10 items etc. The function *geo-rank* for $user_1$ is one such function, as is shown in Figure 1. Based on the distribution of scores for $user_1$ we conclude that Q_{topN} should be used with *attribute-rank*, while $Q_{SCORE\&RANK}$ is more appropriate for the *geo-rank* scoring function.

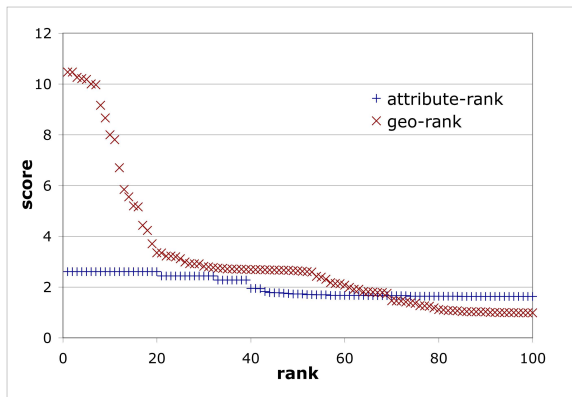


Figure 1: Top-100 scores for $user_1$.

4. RELATED WORK

Li et al [5] argue for native support of clustering and ranking operators in relational databases, and demonstrate how clustering can be implemented by means of a bitmap index over a summary grid with a particular focus on efficiency.

In [3] the authors state the *Many-Answers Problem* and show how correlations among attribute values in a structured datasets can be used to automatically derive a suitable ranking function. To this end, the authors develop a comprehensive probabilistic information retrieval ranking model and present efficient processing techniques. A related *Empty-Answers Problem* is discussed in [2]. Here, the authors present an adaptation of *inverse document frequency* (IDF) that is used for automatic ranking of results.

Sun et al [8] recently presented RankClus, a framework that integrates ranking with clustering in a heterogeneous information network such as DBLP. RankClus is based on a mixture model that uses mutual reinforcement between clustering and ranking. While we share a similar motivation, our application and solution are different.

5. CONCLUSION

In this paper, we formalized new clustering quality measures that are appropriate for rank-aware clustering of structured datasets, and showed their effectiveness experimentally. We believe this work is crucial for more effective exploration of ranked large structured datasets.

6. REFERENCES

- [1] R. Agrawal et al. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.
- [2] S. Agrawal et al. Automated ranking of database query results. In *CIDR*, 2003.
- [3] S. Chaudhuri et al. Probabilistic ranking of database query results. In *VLDB*, 2004.
- [4] K. Järvelin and K. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM TOIS*, 20(4), 2002.
- [5] C. Li et al. Supporting ranking and clustering as generalized order-by and group-by. In *SIGMOD*, 2007.
- [6] U. Manber et al. Experience with personalization of Yahoo! *Commun. ACM*, 43(8), 2000.
- [7] L. Parsons et al. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1), 2004.
- [8] Y. Sun et al. RankClus: integrating clustering with ranking for heterogeneous information network analysis. In *EDBT*, 2009.