Query Result Clustering for Object-level Search[*]

Jongwuk Lee, Seung-won Hwang
Pohang University of Science and Technology
Pohang, Republic of Korea
{julee, swhwang}@postech.edu

Zaiqing Nie, Ji-Rong Wen
Microsoft Research Asia
Beijing, P. R. China
{znie, jrwen}@microsoft.com

ABSTRACT

Query result clustering has recently attracted a lot of attention to provide users with a succinct overview of relevant results. However, little work has been done on organizing the query results for object-level search. Object-level search result clustering is challenging because we need to support diverse similarity notions over object-specific features (such as the price and weight of a product) of heterogeneous domains. To address this challenge, we propose a hybrid subspace clustering algorithm called Hydra. Algorithm Hydra captures the user perception of diverse similarity notions from millions of Web pages and disambiguates different senses using feature-based subspace locality measures. Our proposed solution, by combining *wisdom of crowds* and *wisdom of data*, achieves robustness and efficiency over existing approaches. We extensively evaluate our proposed framework and demonstrate how to enrich user experiences in object-level search using a real-world product search scenarios.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering; H.3.4 [**Systems and Software**]: Performance evaluation

General Terms

Algorithms

Keywords

object-level search, subspace clustering

1. INTRODUCTION

An important goal of Web search engines is to provide end-users with relevant results. However, as different users often have different intents on the same query keyword, two conflicting pillars can be defined to achieve this goal. The first pillar, *personalization*, aims at maximizing the satisfaction of a particular user, while the second pillar, *diversification*, aims at minimizing the dissatisfaction risk of varying user intents. Our goal is thus to pursue both pillars with complementary strengths, to allow users to not only view a big picture of result space but also quickly drill-down to specific results meeting their personalized needs.

Toward this goal, the most relevant research area is *query result organization* for document-level search. Specifically, [9, 25, 22, 26, 7, 13, 11, 20] proposed clustering techniques to partition a set of result documents into several subsets covering different topics. Well-known commercial sites include Vivisimo (*www.vivisimo.com*), Grokker (*www.grokker.com*), iBoogie (*www.iboogie.com*), and Kartoo (*www.kartoo.com*), organizing query result clusters into an expandable topic hierarchy or visualizing query results as interconnected topic terms on a map. Such interfaces help users see the overview of the entire data visually then drill-down to the specific category of an interest. More recently, [12] studied how such combination of clustering and ranking can be simultaneously computed for efficiency.

This paper focuses on query result clustering for *object-level search engines* [16, 15, 14] that automatically extract and integrate the information on *Web objects*. In particular, we focus on organizing the query results for Microsoft Live Product Search[1] extracting product information from the Web document corpus. Unlike a document represented as a TF-IDF vector where every feature value belongs to a homogeneous domain, an object is often represented as numerical feature values of heterogeneous domains, *e.g.*, *sensor size*, *price*, and *weight*. As a result, an appropriate similarity for Web object pairs is highly data- and intent-specific [5], requiring domain expertise to be defined appropriately, while the similarity notion for documents is rather well-agreed, which poses an additional challenge to our problem.

To address this challenge, we now allow clusters representing different search intents to have different salient features, *e.g.*, *sensor size* is important when searching for DSLR cameras and *weight* for compact cameras. As a result, similarity notions also vary significantly in different clusters. This problem, known as *subspace clustering* [4, 6, 8, 2, 3, 21, 23, 17], has been actively studied. Specifically, these algorithms search through possible subspaces, and then determine a subspace that best presents the "local similarity" of objects, though search and selection schemes vary over algorithms. However, these algorithms typically suffer from the following two major disadvantages:

[*]This work was done when the first two authors were at Microsoft Research Asia.

[1]http://search.live.com/products

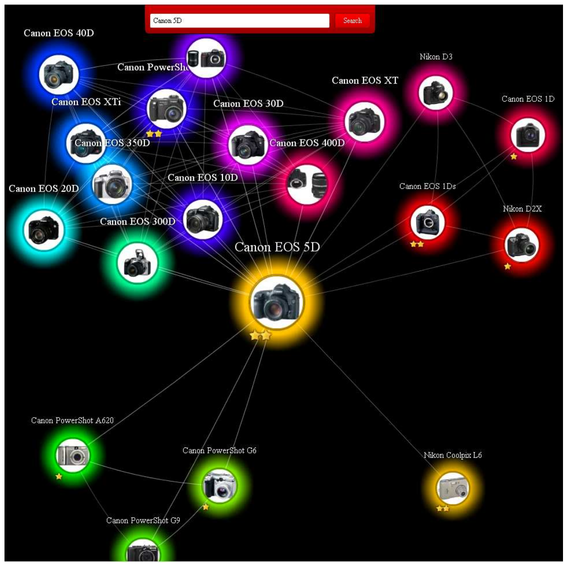


Figure 1: Clustering results for query "canon 5d"

- **Complexity**: Since the subset generation problem is inherently intractable, proposed approximate solutions tend to (1) incur computationally intensive subspace search or (2) require expensive parameter tuning to restrict search space.

- **Cluster quality**: Clustering, being an unsupervised learning technique, may or may not capture locality meaningful to users, depending on the quality of similarity function and feature selection. To train the user perception of locality, recent work discusses semi-supervised clustering, adopting explicit human judgements on cluster boundary [1] or whether to link object pairs [19], often in a limited amount to reduce user intervention.

In a clear contrast, we propose a novel framework overcoming these two drawbacks, by adopting large-scale implicit human feedbacks on *pairwise similarity* of all object pairs. More specifically, we use *co-occurrences* of object pairs appearing in the same Web document together. An instance of such co-occurrence can be viewed as an implicit relevance feedback reflecting the document creator's decision to see two objects as relevant and display in the same page. Co-occurrences can be mined from the entire Web corpus to build "ground-truth" *pairwise similarity matrix* S where S_{ij} indicates the co-occurrences of objects o_i and o_j.

However, such matrix reflecting human perception of similarity represents varying notions of local similarity. To illustrate, since the definition of S_{ij} between a camera object o_i and its accessory o_j and that of S_{ik} between two cameras o_i and o_k capture different senses of similarity, clustering these three objects into a cluster should be discouraged. To disambiguate these two different senses, we adopt the intuition of *subspace clustering* for qualification. That is, we put objects into the same cluster, only if they share enough "value locality" in the same feature subspace. Summing up, our proposed framework combines two notions of relevance, co-occurrences and feature-based local similarity, representing *wisdom of crowds* and *wisdom of data* respectively, to achieve the robustness and efficiency over existing approaches:

- **Robustness**: We use large-scale wisdom of crowds to train a clustering algorithm to well capture varying user perceptions of similarity.

- **Efficiency**: We use wisdom of data to disambiguate different senses of similarity and achieve efficiency by restricting the use of feature-based notion only to qualify whether to merge into the same cluster.

Figure 1 shows an example screenshot of our implementation built on Live Product Search. This illustrates how our proposed clustering algorithm, capturing both co-occurrence and feature-based similarity notions, enriches user experiences in product shopping scenarios. In the figure, a user starts search using her initial interest "Canon 5D" as a query keyword. Before making a purchase decision, she would be interested in browsing and comparing with other related products. In particular, we graphically visualize the related items with varying degrees of similarity represented as distances between nodes, captured from co-occurrences, *e.g.*, a closely displayed pair of "Canon Powershot G6" and "Canon Powershot A620" is highly related. However, such similarity between DSLR cameras has a different meaning from the similarity between compact cameras "Canon EOS 5D" and "Canon EOS 40D", which we distinguish as different cluster locations (and colors) as illustrated in the figure. Such organization enables customers to browse related items with varying complementary strengths then drill-down to the category of interest to make an informed purchase decision.

To summarize, we believe that this paper has the following contributions:

- We build an effective tool to visualize both diversified and personalized search results.

- We propose Algorithm Hydra combining two relevance notions with complementary strengths to achieve robustness and efficiency.

- We extensively evaluate Algorithm Hydra and empirically validate the quality of results.

- We demonstrate Algorithm Hydra developed for a working real-life product search, and illustrate how such tool can enrich user experiences in product shopping scenarios.

The rest of this paper is organized as follows. Section 2 reviews related work to our framework. Section 3 states basic notions to develop our framework. Section 4 designs a hybrid clustering algorithm Hydra. Section 5 validates our proposed framework both using user study on real-life datasets and a larger-scale synthetic data evaluation. Section 6 finally concludes this paper.

2. RELATED WORK

This section overviews the related prior research efforts to our problem context. We then conclude by stating how our work distinguishes itself from these efforts.

2.1 Query Result Organization

Query result organization aims at providing a succinct overview by categorizing the entire results for end-users. As the pioneering work, [9] proposed Scatter/Gather algorithm to cluster documents from search results. [25] proposed STC (Suffix Tree Clustering) algorithm to identify representative phrases from the snippets, based on which documents are associated with each cluster. Furthermore, [26] transformed the given problem as a supervised ranking problem for phrases in order to get meaningful labels and the corresponding clusters. Meanwhile, some research efforts have addressed document clustering based on dimensionality reduction techniques such as SVD (Singular Value Decomposition) [13], NMF (Non-negative Matrix Factorization) [22], and Spectral Analysis [7]. Recently, [12] proposed a spectral clustering algorithm that simultaneously computes clustering and ranking.

2.2 Subspace Clustering

Subspace clustering has been studied to address the "curse of dimensionality" in clustering, *i.e.*, distances between all object pairs become similar in a high-dimensional space. Due to this problem, traditional clustering algorithms, using distance measures over the entire feature space, fail to identify meaningful clusters. In contrast, subspace clustering algorithms, by considering distances in all subspaces and selecting only the most meaningful one, generate high-quality clusters. As the subset generation is intractable, proposed algorithms approximate the search, which can be categorized into bottom-up and top-down greedy searches [17].

- **Bottom-up approach**: The search starts from one dimensional subspace and expands to combine dense region units into a cluster in higher-dimensional subspaces, until no more dense region is found. CLIQUE [4] was the pioneering work, using a grid as a unit of dense region. ENCLUS [6] then used entropy values as a measure for identifying a dense region. Meanwhile, MAFIA [8] enhanced CLIQUE with an "adaptive" grid to the given data distribution.

- **Top-down approach**: The search starts from full feature space then iteratively expands to lower-dimensional subspaces. PROCLUS [2] started from a set of clusters in the full feature space, with respect to k randomly chosen medoids, and iteratively refined the clustering by considering lower-dimensional subspaces. ORCLUS [3] extended PROCLUS to consider non-axis subspaces. FINDIT [21] proposed a unique distance measure called dimension-oriented distance (DOD). Recently, HARP [23], a hierarchial agglomerative clustering, was reported to outperform existing top-down algorithms. We thus adopt HARP as a baseline in this paper.

2.3 Incorporating Feedbacks

To enhance unsupervised clustering algorithms to better reflect user- and domain-specific notions of similarity, [19] proposed semi-supervised k-means clustering adopting feedbacks such as *must-link* and *cannot-link* between objects, obtained from domain or human knowledge. [1] proposed a human-computer cooperative subspace clustering framework where users provide feedbacks on cluster boundaries, based on which the computer iteratively generates satisfactory results. Recent work [24] proposed a semi-supervised projected clustering algorithm based on feedbacks, such as membership labels of some objects to specific cluster or the feature space of some clusters. Similarly, [11] proposed a semi-supervised document clustering model with predefined user feedbacks. More recently, [20] used web search logs to generate more informative document cluster labels, and exploited such labels to guide clustering.

2.4 Our Work

Our work extends query result organization to object-level retrieval. To address the challenge of supporting feature values of heterogeneous domains, we adopt the intuition of subspace clustering, but ensure the robustness and efficiency of our proposed solution, by adopting large-scale user feedbacks and restricting the use of feature-based distance for qualification. Our work distinguishes itself from existing work incorporating user feedbacks, typically assuming explicit and consistent feedbacks, by allowing implicit and inconsistent user feedbacks then disambiguating them using feature-based similarity notion.

3. PRELIMINARIES

This section states preliminaries on modeling Web objects (Section 3.1) and measuring the similarity between Web objects using co-occurrences (Section 3.2) and feature values (Section 3.3).

3.1 Web Object Model

Web object is a concise, recognizable search unit extracted from the Web corpus under object-level search. In general, Web object is represented both as a *term phrase* as a unique identification, *e.g.*, "Canon 5D", and as a set of specific *numeric features* describing objective contents, *e.g.*, (0.81kg, \$2,149). (Though the feature extraction process may introduce errors as discussed in [14], such issue is beyond the scope of this paper.) We formally define Web object as follows: Given a set of Web documents $\mathcal{D} = \{D_1, \ldots, D_{|\mathcal{D}|}\}$, a Web object $o \in \mathcal{O}$ is represented both as a representative term phrase o_T and a feature value vector $(o_1, \ldots, o_n)$ on a domain-specific feature set $\mathcal{F} = \{f_1, \ldots, f_n\}$.

We then consider *object-level search* as identifying relevant Web objects to user-specified query keyword q, which represents either a specific object or a descriptive general term. We define initial set $\mathcal{O}_q$ as a set of Web objects appeared together in Web documents where term q appears. Our goal is thus to organize such initial result set $\mathcal{O}_q$ to satisfy both diverse user intents and specific user needs. Towards this goal, the next sections will present two different similarity measures between Web objects.

3.2 Co-occurrence-based Similarity

We first discuss how the similarity of two objects o and o' can be quantified based on their *co-occurrences* in document set $\mathcal{D}$. For such counts, the proximity of o_T and o'_T in co-occurring documents should be considered, as illustrated in one example form:

$$s(o, o') = 1 - argmin_{D \in \mathcal{D}} proximity(D, o_T, o'_T), \qquad (1)$$

where *proximity* is the minimum number of words between o_T and o'_T in D, normalized by the number of words in D. (Depending on application semantics, *argmin* and *proximity* can be replaced with other more suitable functions, orthogonally to our proposed framework.)

$s(o, o')$ reflects the document creator's perception of the relevance inferred from displaying the two objects closely together in the document. This notion can generalize to quantify cluster similarity $S(C, C')$, by *averaging* distance for all possible object pairs. We formally define the co-occurrence score between two clusters $S(C, C')$ as follows:

$$S(C, C') = \frac{\sum_{o \in C, o' \in C'} s(o, o')}{|C| * |C'|}, \qquad (2)$$

where $|C|$ is the size of clusters.

Clustering challenge: While co-occurrence closely captures the user perception of similarity, it is hard to judge whether to merge two highly co-occurred object pairs into cluster C_i, which is only meaningful when both pairs share enough value locality in the same feature subset S_i.

3.3 Feature-based Similarity

We now discuss how to quantify the degree of "value locality" of clusters. As the notion "enough locality" is defined differently over heterogeneous attributes of arbitrary distributions, all proposed measures inevitably depend on data- and cluster-specific thresholds to distinguish relevant features from the rest.

Specifically, *relative index notion* [23] quantifies the value locality on f_j of cluster C_i as:

$$R_{ij} = 1 - \frac{\sigma_{ij}^2}{\sigma_j^2}, \qquad (3)$$

where σ_{ij} is "local deviation" of f_j values within the cluster and σ_j is "global deviation" of all objects in $\mathcal{O}_q$[2]. This score is close to maximum value 1, when values within the cluster shares high locality such that local variance is small compared to the global one, and 0 when local and global variances are identical. Similar notion, *dimension-oriented distance* [21], was studied which quantifies R_{ij} as binary values 1 and 0 with respect to the given threshold ϵ. Since the two metrics share the same intuition and the former generalizes the latter, we focus on *relative index notion*.

Such locality on f_j should be aggregated on all the relevant feature set S_i for the cluster. For selecting S_i, data-specific parameter R_{min} is used:

$$\mathcal{S}_i = \{f_j | R_{ij} \geq R_{min}\} \qquad (4)$$

Clustering algorithms can then decide whether to merge C_{i_1} and C_{i_2} into C_i, using the following similarity measure $\delta(C_{i_1}, C_{i_2})$:

$$\delta(C_{i_1}, C_{i_2}) = \sum_{f_j \in \mathcal{S}_i} R_{ij}. \qquad (5)$$

As a general assumption, all subspace clustering algorithms encourage resulting clusters to share value locality in as many features as possible, *i.e.*, stronger evidences for the cluster quality. To reflect this assumption, the following qualification condition is checked before each merge, which introduces another data-specific parameter d_{min}.

Definition 1 (Qualification condition) *For the given parameter d_{min} and R_{min}, any merge of C_{i_1} and C_{i_2} into C_i should satisfy $|\mathcal{S}_i| \geq d_{min}$.*

[2]We adopt refined R_{ij} in [23] to discourage an extreme case of merging two distance clusters with large size difference, *i.e.*, $|C_{i_1}| \gg |C_{i_2}|$. More details on this refined notion can be found in [23].

Clustering challenge: While feature-based similarity notion captures different local similarity notions in different clusters, the quality of similarity depends heavily on d_{min} and R_{min}, which ideally should be tuned differently for different clusters.

4. ALGORITHM

This section first discusses baseline algorithms in Section 4.1, then proposes our clustering algorithm in Section 4.2.

4.1 Baselines

As a basis for all algorithms, we adopt a bottom-up *agglomerative hierarchial clustering method* [10], where every object initially corresponds to a singleton cluster, then iteratively merged into k clusters. In particular, we consider two baseline algorithms, using co-occurrence and feature-based similarity respectively.

1. **Base1**: At each iteration, cluster pair (C, C') with the highest co-occurrence similarity $S(C, C')$ is merged into C^{agg}, after which the similarities with other clusters need to be updated.

2. **Base2**: Similarly, we can iteratively merge a pair with the highest feature similarity δ, if it satisfies the qualification condition (Definition 1).

4.2 Proposed Algorithms

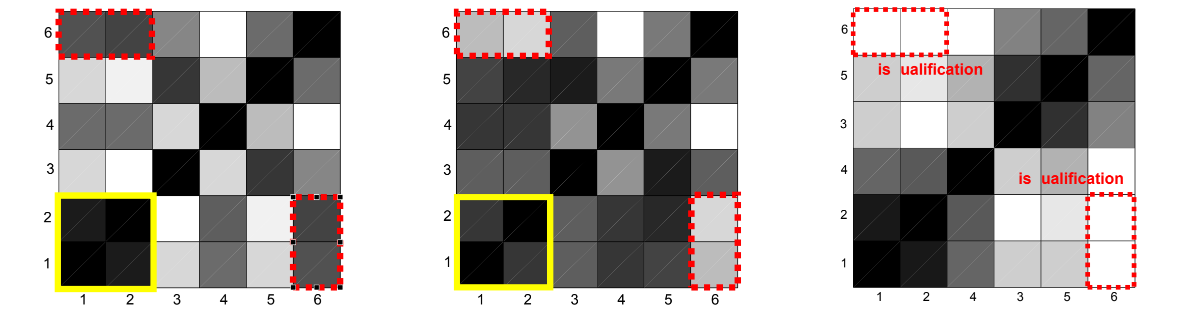
This section discusses how to design a hybrid clustering algorithm using both co-occurrence and feature-based similarity notions, to address the clustering challenges discussed in Section 3. We name our algorithm Hydra for **HY**bri**D** p**R**ojected clustering **A**lgorithm.

To motivate our approach, we visualize the two similarity notions on a real-life dataset for q="Canon 5D". More specifically, Figure 2(a) visualizes pairwise *co-occurrence* similarity using a checkerboard plot where each cell represents object pair and its color represents similarity (the darker, the more similar). For example, the cell in the upper-left corner represents object pair (o^1, o^6) with high co-occurrence represented by a dark cell. Figure 2(b) similarly visualizes pairwise *feature-based* similarity when $R_{min} = 0.5$ and $d_{min} = 3$.

These plots illustrate the complementary strengths of the two notions. Co-occurrence, being parameter independent, robustly distinguishes the similarity differences, while it is unclear, from co-occurrence similarity alone, whether the high co-occurrences of $(\{o^1\}, \{o^2\})$ and $(\{o^1\}, \{o^6\})$ can be explained with the same reason. Meanwhile, feature-based similarity, as the given parameters are suitable only for some pairs, tends to over- or under-estimate the distance of the remaining pairs.

This observation naturally motivates Algorithm Hydra to use robust and parameter-independent co-occurrence metrics to determine the merge order then use feature-based similarity to qualify each merge decision. Algorithm Hydra thus accesses pair $(\{o^1\}, \{o^2\})$ with the highest co-occurrence first, such as four cells marked by the solid box in the lower-left corner in Figure 2(a). Before merging this pair, we check their feature-based similarity, *i.e.*, the matching solid box in Figure 2(b), which is consistently high. This consistency supports that this merge is meaningful.

In contrast, consider the next pair $(\{o^1, o^2\}, \{o^6\})$. As co-occurrences of $(\{o^1\}, \{o^6\})$ and $(\{o^2\}, \{o^6\})$ are both



(a) Checkerboard plot of pairwise co-occurrence matrix (Darker cells indicate closer pairs.)

(b) Checkerboard plot of pairwise feature similarity (When $R_{min} = 0.5$ and $d_{min} = 3$)

(c) Clustering results of Algorithm Hydra combining (a) and (b)

Figure 2: Illustration of Algorithm Hydra for q="Canon 5D"

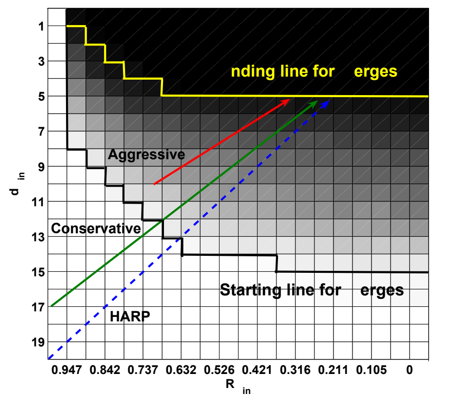


Figure 3: Over varying (R_{min}, d_{min})

Algorithm 1 Hydra($\mathcal{O}_q$)

```
Input
    $\mathcal{O}_q$: A set of objects co-occurred with query $q$.
Output
    $\mathcal{C}$ : A result set of $k$ clusters
1:  $\mathcal{C} \leftarrow \{\}$.
2:  $\mathcal{L} \leftarrow \{\}$. // Sorted queue in the descending order of $S(C, C')$.
3:  $\forall i (1 \leq i \leq m) : \mathcal{C}.add(C_i)$.
4:  $\forall i, j (1 \leq i < j \leq m) : \mathcal{L}.add((C_i, C_j))$.
5:  $\tilde{R}$=EstimateR($\mathcal{O}_q$)
6:  $\tilde{d}$=EstimateD($\mathcal{O}_q, \tilde{R}$)
7:  for $l \leftarrow 0$ to $\tilde{d} - 1$ do
8:      $R_{min} \leftarrow \tilde{R} - \frac{l \times \tilde{R}}{\tilde{d}-1}$.
9:      $d_{min} \leftarrow \tilde{d} - l$.
10:     while $\mathcal{L}.movenext()$ do
11:         $(C, C') \leftarrow \mathcal{L}.current$.
12:         if $(C, C')$ satisfies qualification condition in Def 1 then
13:             Merge $(C, C')$ into $C^{agg}$ and update $\mathcal{C}$ and $\mathcal{L}$.
14:             if $|\mathcal{C}| = k$ then
15:                 Terminate.
16:             end if
17:         end if
18:     end while
19:     $\mathcal{L}.movefirst()$
20: end for
```

high, as represented by cells in dotted boxes in Figure 2(a), this merge appears promising from co-occurrence. However, Figure 2(b) shows clear differences in feature-based similarity between $(\{o^1\}, \{o^2\})$ and $(\{o^1\}, \{o^6\})$, which suggests the reasons for two co-occurrences are different. Merging o_1 and o_2 into a single cluster will thus be disqualified by qualification condition in Definition 1, as represented by white cells in the dotted boxes in Figure 2(c).

After iterative qualified merges, Algorithm Hydra renders the two clearly separated clusters of $\{o^1, o^2, o^4\}$ and $\{o^3, o^5, o^6\}$ as the final results, which corresponds to the cluster of related DSLR cameras {Canon 20D, 30D, 40D} and that of lower-end DSLR cameras {Canon 350D, 400D, XT} respectively. Meanwhile, **base1**, using only co-occurrence, mixes up two categories by performing the above unqualified merge $\{o^1, o^2, o^6\}$, while **base2** returns a low quality cluster $\{o^1, o^2, o^3, o^4, o^5\}$ and $\{o^6\}$, basing merge orderings on blurred similarity matrix.

4.2.1 Hydra with Linear Tuning

As the notion of feature-based similarity heavily relies on data-specific parameters d_{min} and R_{min}, tuning determines the quality of feature-based clustering. Toward the goal, we first discuss *linear tuning* developed in [23] for HARP as preliminaries, based on which we later propose a more sophisticated tuning approach.

Parameter tuning problem can be abstracted as a multivariate interpolation problem of finding (R_{min}, d_{min}) for generating k clusters. Figure 3 illustrates an example search space, where the darkest cells represent parameter pairs generating k clusters and white cells represent pairs which cannot generate any merge. HARP interpolates data points in the space, by searching diagonal cells starting from $R_{min} = 1$ and $d_{min} = n$, *i.e.*, the lower-left corner in Figure 3.

At each interpolation point, HARP performs clustering using corresponding parameter pairs. For example, for the initial value of $R_{min} = 1$ and $d_{min} = d$, no cluster pair can be merged, unless two identical singleton clusters exist in the dataset, which suggests the qualification condition is too "tight" to generate k clusters. The condition is thus gradually "loosened", in particular, by choosing the next diagonal cell for the next round of HARP clustering, *i.e.*, $R_{min} = 1 - \frac{l}{n-1}$ and $d_{min} = n - l$ after l rounds. Such rounds continue, as the dotted diagonal arrow in Figure 3 illustrates, until it hits the darkest cell. Algorithm 1 corresponds to Hydra when the starting values $\tilde{R}$ and $\tilde{d}$ are statically set as 1 and n in line 5 and 6.

4.2.2 HydraAdaptive *with Adaptive Tuning*

However, from the linear tuning, we observe the following two optimization opportunities.

- **Efficiency:** At each round, all pairwise cluster similarities need to be recomputed, while all such computations are wasted if no merge qualifies, *i.e.*, in all white cells in Figure 3, which suggests us to restrict search space to dark cells.

- **Quality:** By restricting search space, similarity recomputations can be done at a finer granularity, which significantly enhances the cluster quality, as Section 5 will empirically validate.

We take these optimization opportunities, by proposing to prune out white cells from search space. Our goal is to efficiently identify a tighter bound for $\tilde{d}$. As such parameter is data-specific, we first develop an adaptive scheme with correctness guarantee, and then improve it into an efficient approximation scheme.

Guaranteed bounding: One way to estimate $\tilde{d}$ is to actually perform HARP with $\tilde{R} = 1$ then set $\tilde{d}$ large enough to qualify all the merges performed, *i.e.*, $\tilde{d} = argmax_{C_i \in \mathcal{C}} |\mathcal{S}_i|$ when $R_{min} = 0$. As depicted in Figure 3, $\tilde{d}$ monotonically increases as R_{min} decreases.

Since performing HARP is expensive, we show that performing an efficient alternative, *i.e.*, *single-linkage* clustering, can also identify $\tilde{d}$ with correctness guarantee. In single-linkage clustering, $\tilde{R}_{ij}$ for merging two clusters is computed as the distance between the two closest elements $o \in C_{i1}$ and $o' \in C_{i2}$ in the two clusters.

$$\tilde{R}_{ij} = \delta_j(o, o') = 1 - \frac{(o_j - o'_j)^2}{\sigma_j^2}, \tag{6}$$

By considering only the closest value pair, $\tilde{R}_{ij}$ is the upper bound of the actual feature-based cluster distance:

$$\forall C_i : \tilde{R}_{ij} \geq R_{ij} \tag{7}$$

Single-linkage clustering is essentially Kruskal's algorithm for finding minimum spanning trees, by merging element pairs in the order of δ, until k spanning trees are generated, which correspond to k resulting clusters. $\tilde{\mathcal{S}}_i$ of the resulting cluster C_i can thus be computed as:

$$\{f_j | \exists \delta_j(o, o') \geq 0\} \tag{8}$$

for every connected pair (o, o') such that $o, o' \in C_i$.

It is immediate from the above upper bounding property that, when Kruskal's and HARP algorithms return the same clustering results $\mathcal{C} = \mathcal{C}'$, $\tilde{\mathcal{S}}_i$ of every Kruskal cluster subsumes $\mathcal{S}_i$ of the corresponding HARP cluster:

$$\tilde{d} = argmax_{C_i \in \mathcal{C}} |\tilde{\mathcal{S}}_i| \geq argmax_{C'_i \in \mathcal{C}'} |\mathcal{S}'_i|. \tag{9}$$

We can extend this for the general case when $\mathcal{C} \neq \mathcal{C}'$. We can transform the clustering results of HARP into an equivalent graph of k spanning trees, by generating a local minimum spanning tree within each resulting cluster C'_i where the maximum $\tilde{d}'$ for every connected node pair satisfies:

$$\tilde{d}' \geq argmax_{C'_i \in \mathcal{C}'} |\tilde{\mathcal{S}}'_i|. \tag{10}$$

Compared to $\tilde{d}$ obtained from the globally minimum spanning tree, $\tilde{d} \geq \tilde{d}'$ holds, which ensures Equation 9 to remain correct.

Conservative starting point can be thus $d_{min} = \tilde{d}$ and $R = 1$, which is guaranteed to start from a white cell. However, this scheme still incurs high computational overheads, *i.e.*, $O(m^2 logm^2)$ for Kruskal's algorithm. We thus develop an approximation of the above procedure in $O(m^2)$– With no correctness guarantee, such approximation may introduce potential mistakes in early merges by loose qualification, which we empirically observe not to negatively affect the quality much. To the contrary, we observe that, by aggressively reducing the search regions and searching the focused region more thoroughly, approximation significantly improves the cluster quality.

Aggressive bounding: We now discuss how we approximate Kruskal's algorithm discussed above. Specifically, we aim at estimating the lower bound $\tilde{R}_j$ of $\delta_j(o, o')$ of all connected object pairs in the resulting spanning trees.

Toward the goal, we pick an estimated value of $\tilde{R}_j$ from the list $\mathcal{L}$ of δ_j for all $\frac{m(m-1)}{2}$ possible pairs. However, the rank of estimated value in the list can vary significantly, depending on the topologies of the resulting spanning trees. To illustrate, consider an extreme case, when k resulting clusters consist of $k-1$ singleton clusters and one big cluster C_0. When every object pair in C_0 is extremely close, $\delta_j(o, o')$ values of all such pairs are higher than $\tilde{R}_j$, which lowers the estimated rank close to $\frac{m(m-1)}{2}$. In another extreme case, where only connected pairs are close, the estimated rank can be as high as $m-(k-1)$. As a moderate estimate of the two, we thus pick the median in $\mathcal{L}$, based on which, $\tilde{d}$ can also be estimated as the median of $|\mathcal{S}_i|$ for every singleton cluster pair. Figure 3 illustrates our aggressive search strategy in the search space. The next section will empirically show how this simple adaptive tuning significantly improves both the effectiveness and efficiency of Hydra.

5. EXPERIMENTS

This section reports our experimental results to validate the accuracy and efficiency of Algorithm Hydra. Our experiments were carried out on a Intel(R) Core 2 machine with 2.13 GHz processor and 2GB RAM running Windows XP. All algorithms were implemented in C$^\#$ language. Our proposed algorithm is compared against two baseline approaches– **Base1** using only co-occurrences which we name HAC and **Base2** implementing Algorithm HARP [23], a subspace clustering algorithm known to have the highest accuracy [18].

5.1 Real-life User Study

First, we performed user study over a real-life product dataset of size 83.9 GBs, including more than 1.1 millions documents crawled in the September of 2008 for Live Product Search (*http://search.live.com/products*). To obtain co-occurrence scores, we extracted product co-occurrences from the product reviews in the dataset. We also extracted feature values, linearly normalized to [0,1]. We conducted a real-life user study for 32 people (Microsoft Research Asia interns and POSTECH students). Due to the expensive nature of user studies, we limit to 6 search tasks for users, in particular, on the currently most popular cameras and laptops (3 tasks each) in Korea, according to *www.danawa.com*, among the products in our dataset.

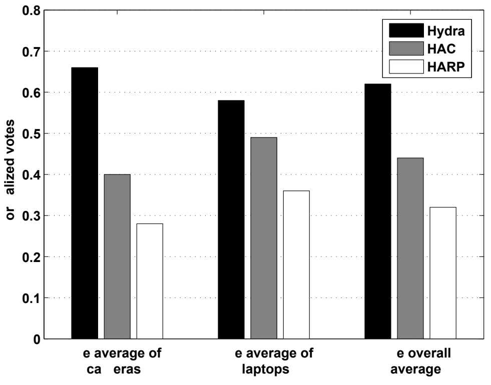


Figure 4: User study results

- **Popular cameras**: Canon Powershot SD850, Nikon D80, Nikon D2Xs
- **Popular laptops**: Lenovo Thinkpad R61, Lenovo Thinkpad T60, Lenovo Thinkpad T61

For each search task, clustering results from the two algorithms are displayed to users without labels, where one is ours and the other is a baseline (randomly chosen between HAC and HARP). Users then blindly vote for the clustering results with higher quality. Figure 5.2 shows the number of user votes, normalized by the number of times displayed to the users. Observe that, both in camera and laptop categories, our clustering results obtain the highest number of votes, which suggests that the hybrid approach of combining two relevance notions improves the user-perceived quality of clustering.

5.2 Larger-scale Simulated Study

As the scale of user study is inherently limited, we generate a large-scale synthetic dataset with ground truth clusters and validate our results over larger and more diverse experimental settings.

Synthetic dataset: We synthetically generate a dataset with ground truth clusters in mind. For synthetically generating feature values, we first determine the size of each cluster, by randomly choosing values in the range of $[\frac{m}{k*1.5}, \frac{m}{k*0.8}]$. To make sure every object belongs to at least one cluster, we randomly generate the size of the first $k-1$ clusters in the above range and decide that of the last cluster as the number of all remaining objects. Once the size of each cluster is decided, we next randomly choose the dimensionality of each cluster, in the range of $[\mathcal{S}_{avg}-3, \mathcal{S}_{avg}+3]$ based on the average dimensionality $\mathcal{S}_{avg}$. Once the dimensionality $\mathcal{S}^i$ of each cluster C^i is determined, we randomly pick $\mathcal{S}^i$ relevant features, and then generate values as follows:

- **Relevant feature** $f_j \in \mathcal{S}^i$: We randomly generate the local mean μ_{ij} and local deviation σ_{ij} to generate f_j, based on which local values generating a cluster are uniformly distributed in the range of $[\mu_{ij} - \sigma_{ij} \times 0.01, \mu_{ij} + \sigma_{ij} \times 0.01]$.
- **Irrelevant feature** $f'_j \notin \mathcal{S}^i$: We uniformly generate values in the entire range [0, 1].

We then synthetically generate co-occurrence scores. While the co-occurrence score can be arbitrarily generated, it is non-trivial to decide the ground-truth clusters when feature-based and co-occurrence similarity scores disagree. Due to this difficulty, we consider a special scenario, where co-occurrences are generated based on feature-based similarity. Specifically, the co-occurrence between objects o and o' in cluster C and C' is represented as $\sum_{f_j \in \mathcal{S} \cup \mathcal{S}'} (1 - |o_j - o'_j|)$. However, as real-life user feedbacks are prone to noises, we add controlled noises, by choosing only $r = 80\%$ of f_j from $|\mathcal{S} \cap \mathcal{S}'|$ then rest from the remaining features, *i.e.*, the smaller r is, the noisier co-occurrences are.

Parameter	Values
The number of data m	**1000**, 2500, 5000, 7500
The number of features n	**20**, 40, 60, 80, 100
Cluster size $\lvert C^i\rvert$	$[\frac{m}{k*1.5}, \frac{m}{k*0.8}]$
Average feature size $\mathcal{S}_{avg}$	**6**, ..., $\lvert\mathcal{F}\rvert - 5$
Local deviation σ_{ij}	**[2, 8]**, [4, 10], [6, 12], [8, 14]
The number of clusters k	5, **10**, 15
Sparsity $sp\%$	90, ..., **50**, ..., 10

Table 1: Parameters of synthetic datasets

We stress that this scenario is somewhat unfavorable to our proposed algorithm (and favorable to HARP), since the co-occurrence scores, generated from feature-based relevance, cannot provide any extra information to our hybrid approach. Our intention of using this scenario is to show that, even in this unfavorable setting, Algorithm Hydra outperforms baselines. Lastly, to reflect the "sparsity" of co-occurrence scores in real-life data where co-occurrence scores are zero for the majority of object pairs, we control sparsity sp to generate $sp\%$ of co-occurrences as zero.

Quality metrics: As quality metrics, we adopt three representative metrics extensively used from prior works [2, 21]: Clustering Error (CE), F_1-value, and FF_1-value.

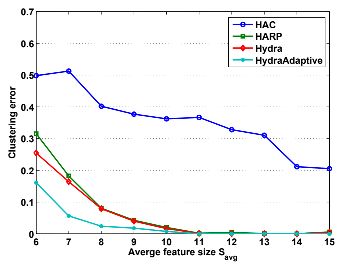
First, CE [2] is generally used to show the difference between ground-truth clustering $\mathcal{C}$ and our clustering results $\mathcal{C}'$. This measure is based on $|\mathcal{C}'| \times |\mathcal{C}|$ matrix M, where M_{ij} is the number of elements shared by clusters C^i and C^j from $\mathcal{C}'$ and $\mathcal{C}$ respectively. Note that we use *confusion matrix* M' [2] maximizing the sum of diagonal elements $D = \sum_{i=1}^{|\mathcal{C}|} M'_{ii}$ among all possible matrices. In an ideal case when $\mathcal{C} = \mathcal{C}'$, non-diagonal elements of M' will be all zero, and in other cases, the error can be quantified as the difference between the sum of diagonal elements and the overall sum, *i.e.*, $CE(\mathcal{C}, \mathcal{C}') = \frac{U-D}{U}$, where U is the sum of all elements in the matrix such that $\sum_{i=1}^{|\mathcal{C}'|} \sum_{j=1}^{|\mathcal{C}|} M'_{ij}$. When the value of this measure is 0, two clustering results are identical.

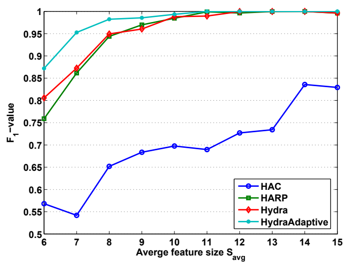
Next, F_1-value is a harmonic mean of *precision* and *recall*, widely used to measure accuracy in IR literatures. Specifically, $precision_j$ and $recall_j$ are defined as $argmax_i\{M'_{ji}\}$ / $\sum_{i=1}^{|\mathcal{C}|} M'_{ji}$ and $argmax_i\{M'_{ij}\}$ / $\sum_{i=1}^{|\mathcal{C}|} M'_{ij}$ respectively. F_1-value is calculated as the average of k clusters, maximized as 1 when two clustering results are identical.

Similarly, FF_1*-value* [21] is an extension of F_1*-value* to quantify the overlap in feature space of two clustering results. The overlapped elements are used from F_1 values. FF*-precision*$_j$ and FF*-recall*$_j$ are defined as $|\mathcal{S}^j \cap \mathcal{S}^j_{ideal}|$ / $|\mathcal{S}^j|$ and $|\mathcal{S}^j \cap \mathcal{S}^j_{ideal}|$ / $|\mathcal{S}^j_{ideal}|$, where $\mathcal{S}^j_{ideal}$ and $\mathcal{S}^j$ are relevant feature sets at j^{th} cluster in $\mathcal{C}$ and $\mathcal{C}'$, respectively.

Experimental results: We present experimental results with synthetic datasets for various parameter settings in Table 1. Each data point reports the average of 50 runs with 50 different datasets.

First, Figure 5 reports the accuracy using three metrics over varying average number of features of clusters. Note that, FF_1-metrics does not apply to HAC which does not


(a) Clustering Error


(b) F_1-value

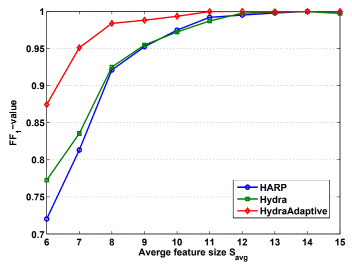
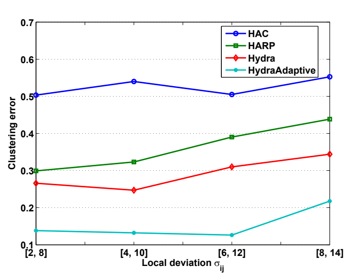
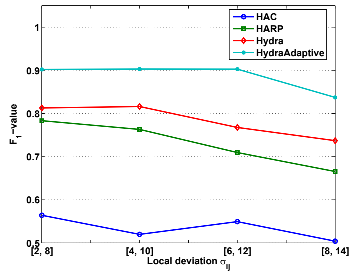

(c) FF_1-value

Figure 5: Over varying $\mathcal{S}_{avg}$


(a) Clustering Error


(b) F_1-value

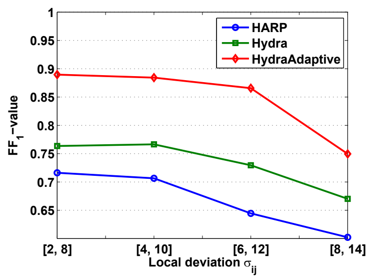

(c) FF_1-value

Figure 6: Over varying σ_{ij}

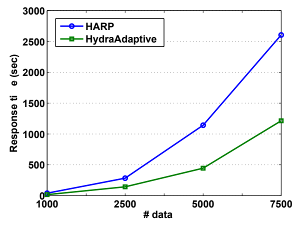
identify relevant feature sets. Observe that in all three metrics, all algorithms were highly accurate when $\mathcal{S}_{avg}$ is high. An interesting difference from user study results in Figure is HAC performs worse in synthetic evaluations, which can be explained by the noises introduced to co-occurrences that negatively affect the accuracy of HAC depending solely on co-occurrences. In particular, for low $\mathcal{S}_{avg}$ the accuracy gaps are apparent. The problem of HARP in such scenario is that, by starting the parameter tuning from the most conservative bound $d_{min} = d$, all similarity score refinements from early rounds are simply wasted, which limits refinement opportunities in the valid range and also affects the quality of merges performed on poorly estimated scores. In a clear contrast, Algorithm Hydra, by using a more robust co-occurrence ordering, shows higher accuracy in all $\mathcal{S}_{avg}$, which is further enhanced by Algorithm HydraAdaptive, searching a focused space at a finer granularity.

Second, Figure 6 reports the accuracy using three metrics over varying local deviation. Observe that, in all three metrics, Algorithm Hydra generates significantly better clustering results than Algorithm HARP and HAC, especially when the local deviation is high. As the ideal R_{min} values for clusters are lowered, HARP searching for the ideal parameter from $R = 1$ wastes refinement efforts in early rounds. In contrast, Algorithm HydraAdaptive, by exploiting co-occurrence ordering and finer-grained parameter tuning on lower R_{min} search region, reduces the CE error of HARP by $\frac{1}{2}$ when $\sigma_{ij} = [8, 14]$.

Third, Figure 7 reports the accuracy using three metrics over varying number of clusters. Observe that, in all three metrics, the accuracy of all algorithms deteriorates as k increases, as a single mistake in merging severely affects the datasets with smaller and many clusters, while such sensitivity is significantly low for Algorithm HydraAdaptive. To illustrate, when $k = 15$, the CE result shows that Algorithm HydraAdaptive shows 3.34, 2.55 and 2.35 times higher accuracy than Algorithm HAC, HARP, and Hydra respectively.

Fourth, Figure 8 reports the accuracy using three metrics for co-occurrence data with varying sparsity. In particular, we vary sparsity sp for Algorithm HydraAdaptive. As references, we plot the accuracy of HARP (as lower bound accuracy) and that of Hydra using perfect co-occurrence data without sparsity (as theoretical upper bound accuracy), both of which are not affected by sp and thus stay constant. Observe that, Algorithm HydraAdaptive, even at the presence of severe sparsity, *e.g.*, $sp = 90\%$, is significantly more accurate than the lower bound and closely approximates the theoretical upper bound. For lower sparsity, *e.g.*, $sp < 70\%$, the accuracy of Algorithm HydraAdaptive converges to the theoretical upper bound.

Lastly, we empirically study the scalability of Algorithm Hydra over varying data size m and feature size n. For varying n, we also vary $\mathcal{S}_{avg}$, to avoid cases only $n' \ll n$ features are relevant to clusters, in particular, by varying $\mathcal{S}_{avg} = \lfloor n/3 \rfloor$ as well. Figure 9 shows that Algorithm HydraAdaptive significantly outperforms Algorithm HARP in all settings and scales more gracefully, by using adaptive loosening which enables fast convergence and thus reduces computational overhead.


(a) varying m

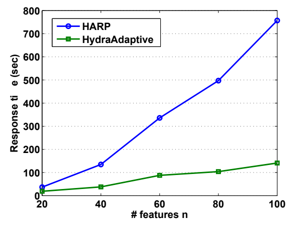
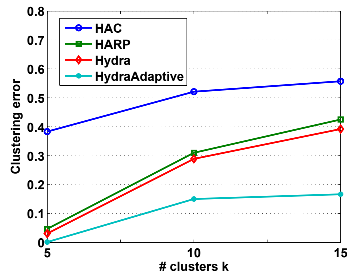

(b) varying n

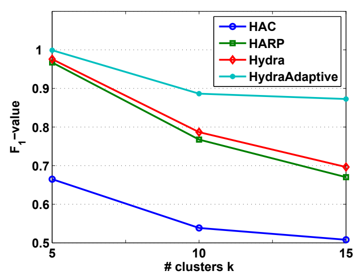
Figure 9: Efficiency between Algorithm HARP and Algorithm Hydra

6. CONCLUSION

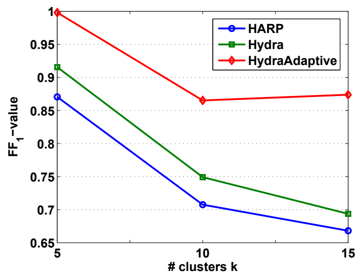
This paper studied query result organization to provide



(a) Clustering Error

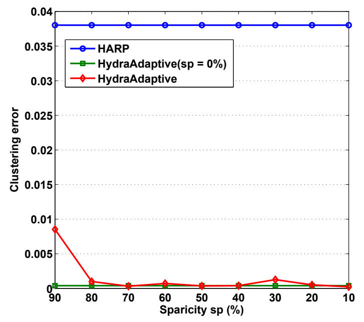


(b) F_1-value

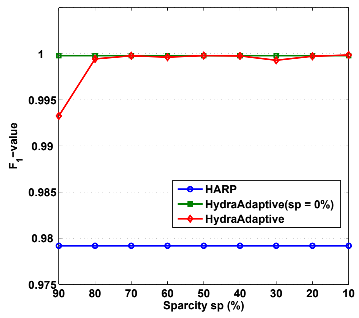


(c) FF_1-value

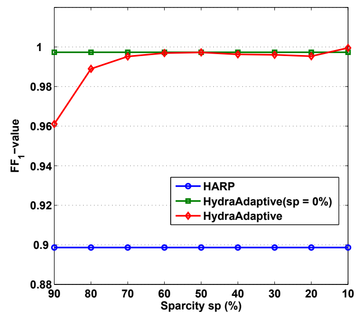
Figure 7: Over varying k



(a) Clustering Error



(b) F_1-value



(c) FF_1-value

Figure 8: Over varying sp

highly relevant results that both cover diverse intents and address the user-specific intent. In particular, we focus on object-level search, which poses a new challenge of combining co-occurrence and feature-based similarity notions with complementary strengths. To address this challenge, we proposed a hybrid clustering algorithm Hydra using large-scale implicit user feedbacks as similarity metrics representing diverse intents, disambiguated by feature-based subspace locality. We extensively validated Algorithm Hydra using both real-life user study and large-scale synthetic datasets.

ACKNOWLEDGEMENT

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

7. REFERENCES

[1] C. C. Aggarwal. A human-computer cooperative system for effective high dimensional clustering. In *KDD*, 2001.

[2] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD*, 1999.

[3] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, 2000.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high-dimensional data for a data mining applications. In *SIGMOD*, 1998.

[5] M. Bilenko, S. Basu, and M. Sahami. Adaptive product normalization: Using online learning for recored linkage in comparison shopping. In *ICDM*, 2005.

[6] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD*, 1999.

[7] D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-merge methodology for clustering. *TODS*, 2005.

[8] S. Goil, H. Nagesh, and A. Choudhary. Mafia: efficient and scalable subspace clustering for very large data sets. In *Technical Report, Northwesthen University*, 1999.

[9] M. A. Hearst and J. O. Pedersen. Re-examining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, 1996.

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 1999.

[11] X. Ji, W. Xu, and S. Zhu. Document clustering with prior knowledge. In *SIGIR*, 2006.

[12] Y. Liu, W. Li, Y. Lin, and L. Jing. Spectral geometry for simultaneously clustering and ranking query search results. In *SIGIR*, 2008.

[13] G. Mecca, S. Raunich, and A. Pappalardo. A new algorithm for clustering search results. *Data and Knowledge Engineering*, 2006.

[14] Z. Nie, Y. Ma, S. Shi, J.-R. Wen, and W.-Y. Ma. Web object retrieval. In *WWW*, 2007.

[15] Z. Nie, J.-R. Wen, and W.-Y. Ma. Object-level vertical search. In *CIDR*, 2007.

[16] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: bringing order to web objects. In *WWW*, 2005.

[17] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Newsletter*, 2004.

[18] A. Patrikainen and M. Melia. Comparing subspace clusterings. *TKDE*, 2006.

[19] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Conttrainted k-means clustering with background knowledge. In *ICML*, 2001.

[20] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, 2007.

[21] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. FINDIT: a fast intelligent subspace clusteing algorithm using diemsnion voting. *Information and Sofeware Technology*, 2004.

[22] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, 2003.

[23] K. Y. Yip, D. W. Cheung, and M. K. Ng. HARP: A practical projected clustering algorithm. *TKDE*, 2004.

[24] K. Y. Yip, D. W. Cheung, and M. K. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. *ICDE*, 2005.

[25] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *SIGIR*, 1998.

[26] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR*, 2004.