PRESS: A Personalised Approach for Mining Top-k Groups of Objects with Subspace Similarity

Tahrima Hashem [a], Lida Rashidi [a], Lars Kulik [a], James Bailey [a]

[a] *Department of Computing and Information Systems, The University of Melbourne, VIC, Australia*

Abstract

Personalised analytics is a powerful technology that can be used to improve the career, lifestyle, and health of individuals by providing them with an in-depth analysis of their characteristics as compared to other people. Existing research has often focused on mining general patterns or clusters, but without the facility for customisation to an individual's needs. It is challenging to adapt such approaches to the personalised case, due to the high computational overhead they require for discovering patterns that are good across an entire dataset, rather than with respect to an individual. In this paper, we tackle the challenge of personalised pattern mining and propose a query-driven approach to mine objects with subspace similarity. Given a query object in a categorical dataset, our proposed algorithm, PRESS (**Pe**r**s**onalis**e**d **S**ubspace **S**imilarity), determines the top-k groups of objects, where each group has high similarity to the query for some particular subspace. We evaluate the efficiency and effectiveness of our approach on both synthetic and real datasets.

Key words: Subspace mining, Similarity search, Association rules, Mining methods and algorithms, Personalisation.

1. Introduction

The study of similarities among people (or more generally among objects) enables us to gain a broader and deeper understanding of them. It can provide individuals with personalised feedback and guidance for building their career, identifying the role in social networks and diagnosing rare/unusual diseases. For example, a newly graduated student might select a career pathway by identifying employees with a similar profile. Alternatively, a personalised property recommendation system could help a first-time home

*Corresponding Author: Tahrima Hashem, Tel.: +61403053684

Email address: tahrimah@student.unimelb.edu.au, rashidi.l@unimelb.edu.au, lkulik@unimelb.edu.au, baileyj@unimelb.edu.au (Tahrima Hashem [a], Lida Rashidi [a], Lars Kulik [a], James Bailey [a])

patients	gender (*gr*)	age (*age*)	(high) blood-pressure (*hbp*)	pain (*pn*)	cough (*cg*)	skin-condition (*sk*)	medicine (*med*)
p1: Mary	female	adult	no	chest	dry	cold-sores	M-A
p2: Sam	male	toddler	no	chest	wet	discolored	M-B
p3: Peter	male	teenager	yes	back	none	normal	M-C
p4: Rose	female	adult	no	muscle	dry	rashes	M-D
p5: David	male	adult	yes	none	none	normal	M-E
p6: Jack	male	senior	no	chest	dry	wart	M-E
p7: Jeny	female	senior	no	chest	dry	wart	M-A
p8: Brad	male	adult	no	chest	dry	acne	M-A
q: Lisa	female	teenager	yes	chest	dry	normal	?

Table 1: Example of medical records of existing patients.

buyer in finding a villa unit with spacious lounge nearby a city and a supermarket while being flexible on having an open yard or a car space. In this paper, we aim to identify groups of objects that share similarities in subspaces (subset of attributes) with respect to the given query object in a categorical dataset. Our approach is unsupervised as it does not require any prior knowledge of the class/label information of the objects. The inherent groupings among objects are discovered by exploiting the subspace similarity between the objects and the query.

Most of the existing query ranking approaches (1; 2; 3) highlight the need for similarity search over a fixed subspace that is supplied by the user. Authors in these works made an assumption that the query user must have certain knowledge about the subspace he/she is looking for. However, in some scenarios the user may not have any domain knowledge or may have little to no preference about the subspace (e.g., non-expert laptop buyer) and wants to identify the objects/persons similar to a given query object/person. It is very unusual to find an object that matches with all the characteristics/features of the query object. Hence, subspace similarity search is essential specially in the presence of large number of attributes. Nevertheless, it is computationally very expensive to enumerate all the possible combinations of subspaces (exponential in numbers). Our proposed approach would be suitable for these cases as we dynamically determine the groups of objects with corresponding subspaces, which are similar to the given user query.

1.1. Motivating Examples

We present two different scenarios illustrating the significance of the proposed research problem. The first scenario illustrates the challenges of providing treatment to a patient with unusual symptoms. Our proposed approach can play an effective role in this case by providing insights about the query patient characteristics. In the second scenario, we mention the difficulties faced by a novice customer with flexible needs in the electronics or real estate market. Then, we discuss how inexperienced customers can be guided carefully to make their own decisions by using our proposed approach. We label

the first and second scenario as ***Query Insights*** and ***Query Exploration***, *respectively, throughout the paper*.

Scenario-1 (Query Insights): Consider the sample medical records of eight patients (in Table 1) and a female patient named *Lisa* who has recovered from an injury of a broken arm three months ago but suffers from high blood pressure. Lisa also has light chest pain with dry cough. The medical practitioners find it difficult to diagnose Lisa due to the lack of clear medical practice guideline for her treatment. In such a scenario, treatment options can be provided to the medical practitioners by identifying patients with health problems similar to Lisa. In the medical literature, this decision making scenario is known as the "green-button", referring to the wish of a clinician to have available a magic green button, that once pressed, could identify cohorts of patients who are similar to an individual patient who is proving difficult to diagnose (4).

In this scenario, our approach would suggest a group of two *female* patients, *Mary* and *Jeny* to be most similar to Lisa who are suffering from *chest* pain and *dry* cough. However, we would recommend another group to consider that comes with a new *female* patient *Rose* in addition to the patients, i.e., Mary and Jeny, by relaxing the value on the pain attribute (*Rose* is having *muscle* pain in contrast to Lisa's *chest* pain). Rose could be a potential candidate for Lisa because Lisa's chest pain might be the effect of the accident that had happened to her arm. Hence, the practitioners can be guided with more information by identifying "groups of patients" that combine the matching and non-matching symptoms with respect to the given query. For instance, if the practitioners are aware of the medicines used for the identified "groups of patients", they can apply this knowledge for diagnosing the query patient. In this example, Lisa can be treated according to the medicines, i.e., 'M-A' and 'M-D', which were prescribed to *Mary, Jeny* and *Rose*, respectively. Section 5.3.3 and Section 5.4 provide more discussions about achieving "Query Insights" on two real datasets, i.e., Laptop and Academic Citation Network, respectively.

Scenario-2 (Query Exploration): Our approach is also applicable to the first-time home buyers i.e., buying homes in *zillow*[1], who find it difficult to determine suitable homes with their desired features. In this scenario, the user might have *strict* requirements for some attributes while being *flexible* on others. Our approach is able to identify "groups of properties" satisfying all such strict requirements, where each group is the best option for the user within a given subspace of flexible attributes. Moreover, the non-expert users of the electronic products, e.g., laptops and mobiles, may find our approach helpful. We construct the reference query from a sample product supplied by the user and provide them with a personalised ranked list of products with interpretable subsets of features, which can help the customers in deciding their most desirable product Section 5.3.1 provides a real-world example of this scenario on Laptop dataset.

[1]https://www.zillow.com

1.2. Contributions

To the best of our knowledge, we are the first to introduce query-oriented approach for mining object clusters in subspaces for categorical data. Our approach requires no knowledge about the type/class of the objects in the dataset while identifying the clusters of objects, hence it is unsupervised. Existing (unsupervised) query ranking techniques (1; 2; 3) identify objects from the whole dataset for a fixed subspace whereas our approach dynamically determines the groups and subspaces by maximising the similary with respect to the query characteristics. Inlying or outlying aspects mining (5; 6; 7) approaches also focus on a query, but they identify a subspace that makes the query most inlying or outlying with respect to *all* objects in the dataset. However, we identify the subset of attributes for a cluster of objects such that the objects become most similar (inlying) to a *specific* query object.

Subspace clustering (8; 9) is closely related but not directly applicable as it identifies clusters of objects such that the intra-cluster similarity is maximised while the inter-cluster similarity is minimised. In contrast, our approach groups objects such that they share similar characteristics in subspaces with respect to a particular query. Thus, if one were to apply subspace clustering to solve our proposed problem, it would likely result in poor quality of clusters in terms of similarity with the given query. Bi-clustering (10; 11) is another related research topic where simultaneous clustering of rows/objects and columns/attributes of data matrix is performed. However, this technique is also not suitable for our task as it mines general patterns across the matrix instead of discovering patterns significant for a particular (query) object.

Our proposed algorithm efficiently enumerates the subspaces to mine personalised clusters, where the objects are homogeneous with respect to the query subspaces. We exploit the strictness (if required by the user) and flexibility of the query subspaces and, effectively reduce the search space by eliminating redundant groups that can never be a part of the answer. Our effective pruning strategies adopt a 'top-k' enumeration approach. We further provide a guideline for the determination of divergent objects from the discovered personalised groups to make it more user friendly. As there is no ground truth available for the query focused object-clusters, we would not be able to use the cluster validity metrics (e.g., NMI, purity and F-score) for evaluating the performance of PRESS. Instead, we conduct qualitative case studies to assess the effectiveness and efficiency of our approach. The key contributions can be summarised as follows:

- Novelty: We define a novel personalisation framework, where we aim to maximise subspace similarity for a given query and a group of objects.
- Effectiveness: We develop effective strategies to mine subspaces for datasets possessing categorical attributes.
- Scalability: Our experimental study demonstrates that our proposed algorithm is highly scalable with respect to the number of attributes and objects in the dataset.

The rest of the paper is organised as follows. Section 2 reviews existing works related to this problem. We introduce new terminologies and formally define the problem in Section 3. In Section 4, we propose our algorithms to solve the problem and in Section 5, we present extensive experiments and perform case-studies to verify the effectiveness of our algorithms. Section 6 concludes the paper with future research directions.

2. Related Works

2.1. Query Oriented Similarity Search

*k*NN Classification or regression techniques (12; 13) predict the class (value of the target attribute) of a given test/query object using its k nearest neighbours. However, in our approach we neither have an intention to predict the class of the given query object, nor do we use the class information of the objects. We determine the inherent groupings of objects in an unsupervised manner such that the objects in a group share the highest similarity with the query object.

Ranking queries (14; 15) is a related topic to our target research where k objects are identified in terms of exact and approximate matches with the criteria of the query. Query-relaxation (1), query-refinement (16) and ranked-retrieval techniques (2; 17) are applied either in interactive or non-interactive manner to get rid of the empty/many answers problems. All these approaches perform the similarity search for a given fixed set of attributes. There exist a number of approaches (3; 18; 19) that support similarity search for any arbitrary subset of attributes but these attributes need to be specified along with the user query at run-time.

In contrast, our approach is for those users who have little or no knowledge about the subspace they are searching for. For a given query in m attributes and a database of n objects, our approach identifies m' attributes ($m' < m$) where the query appears to be most similar to a particular group consisting of n' objects ($n' \ll n$). In other words, we identify the top-k groups of objects with corresponding subsets of attributes where they exhibit highest similarity to the given query object. Though we do not build any similarity index, we find our approach highly scalable for large number of attributes.

Mining skylines in subspaces (20; 21; 22; 23) is another relevant problem to our study. Skylines are those objects that are not dominated by others in the dataset. An object dominates others when it is strictly better for at least one attribute provided that the attribute values are ordered. This definition is also applicable to the subspaces. The approaches in (24; 25) further integrated user constraints with this dominance relationship and applied the skyline or selection operator over the whole data-space. In contrast, our approach intends to determine the subspace where a specific group of objects becomes similar to the query. Even in the presence of some user constrained attributes, we would further explore the remaining (flexible) attributes to identify the group of objects similar to the query.

Outlying/inlying aspects mining (5; 7) is a related research area where subsets of attributes are identified to make a query most outlying or inlying as compared to all the objects in the database. In our approach, we aim to discover the subset of attributes for a group of objects such that the query becomes most inlying to the objects in that group. Another relevant topic is *contrast subspace* mining (26), which determines the top-k subspaces maximising the probability of a query belonging to a target class. However, there is no such constraint on identifying subspaces in our approach.

2.2. General Similarity Search

Clustering objects based on their attribute similarities has been extensively studied. Variants of such clustering approaches include projected clustering, subspace clustering and pattern based clustering (bi-clustering) (see (27; 28)).

The intuition behind subspace clustering approaches is that the objects depict more similarity in a smaller subspace in comparison with the full attribute space, particularly in a high dimensional dataset. A variety of strategies (8; 9; 29) have been applied to mine effective subspace clusters of objects. However, subspace clustering is different from our approach as we aim to discover groups of objects where they exhibit similarities with respect to the query subspaces. Thus, we need to choose a subset of attributes on behalf of the query while grouping objects into a cluster. A related research topic is bi-clustering (10; 11) that clusters genes (rows/objects) and samples (columns/attributes) of a data matrix based on the pattern observed over the sub-matrices. This approach does not address our problem as it only captures general patterns among objects rather than doing individual object analysis.

The approaches above are tailored for numerical data. However, the object attributes can be both numerical or categorical. There are a few studies, i.e., CLICKS (30), SUBCAD (31), that cluster categorical data in subspaces. In contrast to them, we find subspace clusters of objects similar to a specific query in categorical data. To the best of our knowledge, there is no query oriented study for discovering categorical subspace clusters.

Subgroup discovery (32; 33) aims to mine subgroups with unusual distribution. The relationship among the objects is expressed in terms of rules where the antecedent comprises the dependent attributes and the consequent is the class attribute. However, our approach mines groups of objects similar to a query for some subsets of input attributes. There is no need of the class information in our approach. We do not have any intention to mine groups with atypical distributions.

3. Terminologies and Problem Definition

Let $O_f = \{o_1, o_2, \ldots, o_n\}$ be a set of n objects and $A_f = \{a_1, a_2, \ldots, a_m\}$ be a set of m categorical attributes. Each object $o_i \in O_f$ including the query q consists of m attributes. For each attribute $a \in A_f$, there are t possible categorical values, e.g., attribute *cough* in

Table 1 has t=3 values. We find the query q as {*gender=female*, *age= teenager*, *high blood-pressure=yes*, *pain=chest*, *cough=dry*, *skin-condition=normal*} in Table 1.

Assume v_a represents the value of attribute a. For an attribute-set $A = \{a_1, a_2, \ldots, a_i\} \subseteq A_f$ for $i \leq m$, O_A corresponds to those objects $O_A \subseteq O_f$ sharing the same values with the query on A, i.e., $v_a^o = v_a^q$, $\forall o \in O_A$, $\forall a \in A$. The attribute-set, A is called closure (34) for the given query, if there remains no attribute-set $A' \subseteq A_f$ such that $A' \supset A$ and O_A=$O_{A'}$, e.g., A={*age*, *pain*}, A'={*age*} and A''={*pain*} where $O_A = O_{A'}$ ={*p2*} and $O_{A''}$ ={*p1,p2,p6,p7,p8*} for the values v_{age}^q=*teenager* and v_{pn}^q=*chest*. A'' is a closure but A' is not as $A \supset A'$ with $O_A = O_{A'}$.

A group of objects can be described using two different types of categorical attributes, i.e., (i) *completely* and (ii) *partially* matched attributes, respectively, with regards to the query q. For a *completely* matched attribute, all objects of the group share the same values with q. For a *partially* matched attribute, objects in a group contain both matching and non-matching values with q. A_c and A_p represent the sets of completely and partially matched attributes, respectively. Let $G = (O, S)$ represent a *personalised* group of objects $O \subseteq O_f$ in subspace $S \subseteq A_f$ for the query q where S consists of *completely* (A_c) and *partially* (A_p) matched categorical attribute-sets, i.e., $S = A_c \bigcup A_p$.

Consider a group of patients O={*p1:Mary*, *p6:Jack*, *p7:Jeny*, *p8:Brad*} in Table 1, where the patients have complete matches on A_c = {*pain*, *cough*} corresponding to the values v_{pn}^q = *chest* and v_{cg}^q = *dry* of q while having partial matches on A_p = {*gender*} given the value v_{gr}^q = *female*. Hence, S = { *pain*, *cough*, *gender*} for the movies O. We find O_{A_c}={$p1$, $p6$, $p7$, $p8$} and O_{A_p}={$p1$, $p7$}.

We consider both A_c and A_p for measuring the similarity of a group with the query q. We use the notion of *confidence*; a widely used metric in association rule mining (35), for defining the score of a group $G = (O, S)$ in subspace S consisting of A_c and A_p in Eq. 1.

$$score(O, S, q) = \sum_{o^i \in O} match(o^i, S, q)/|O| \tag{1}$$

$$match(o, S, q) = \begin{cases} 1, & \text{if } v_a^o = v_a^q, \forall a \in S \\ 0, & \text{otherwise} \end{cases}$$

However, if $A_p \neq \emptyset$ then the $score(O, S, q)$ (Eq. 1) reduces to $score(O, A_p, q)$ (Eq. 2), since objects sharing the same values with q on A_c is same as O, i.e., O_{A_c}=O. Therefore, A_c does not play role to determine the value of *match* function and we only need to calculate the number of objects that are sharing the same values with q on A_p, i.e., $|O_{A_p}|$ where $O_{A_p} \subset O$.

$$score(O, S, q) = score(O, A_p, q)$$
$$= \sum_{o^i \in O} match(o^i, A_p, q)/|O|$$
$$score(O, S, q) = |O_{A_p}|/|O| \quad (2)$$

For the group of patients (as discussed above), O = {*p1*, *p6*, *p7*, *p8*} and A_p = {*gender*}, we find *match(p1,* {*gender*}*, q)* = 1, *match(p6,* {*gender*}*, q)* = 0, *match(p7,* {*gender*}*, q)* = 1 and *match(p8,* {*gender*}*, q)* = 0. Thus, $score(O, S, q)$=$\frac{2}{4} = \frac{1}{2} = 0.50$.

In this paper, we develop an algorithm **PRESS** for mining the *top-k* personalised groups of objects having the k largest scores in some categorical subspaces with respect to the query q for a given minimum subspace length d and minimum similarity score min_S.

Problem Statement. Given a query q and a set of n objects, $O_f = \{o_1, o_2, \ldots, o_n\}$ where each object o_i and the query q comprises m categorical attributes, $A_f = \{a_1, a_2, \ldots, a_m\}$, a minimum subspace length d and a minimum similarity score min_s, PRESS finds the top k groups of objects $\{O^1, O^2, \ldots, O^k\}$, $O^i \subseteq O$ and their corresponding subspaces $\{S^1, S^2, \ldots, S^k\}$, $S^i = (A^i_c, A^i_p) \subseteq A_f$ having the largest scores $\{sc_1, sc_2, \ldots, sc_k\}$ with respect to q where $A^i_c \neq \phi$, $| S^i | \geq d$ and $sc_i \geq min_S$.

Note that PRESS groups objects only if at least one attribute is similar to the query. For our medical database, the practitioners will only find a group significant if the patients have at least one completely matched attribute. However, if the patients do not share any charateristic/attribute with the query, our approach does not suggest any patient since the query is an outlier with respect to the existing records of patients. In summary, we enforce that a completely matched attribute-set, A_c can never be empty for a group of objects to be similar to the query.

4. Algorithms

4.1. The Framework

We present a framework of our approach in Algorithm 1 to mine k groups of objects illustrating similarity with the query q in their corresponding subspaces. There are three key steps.

Step 1: We determine the groups of objects with respect to the completely matched attributes (A_c) given q. Function *FindCG* retrieves the non-redundant groups of objects in terms of A_c using the closure mining strategy (Lines 1.1 - 1.2) as described in Section 4.2.

Algorithm 1: The Framework

Input : A query q and a set of n objects O_f described by m categorical attributes A_f, k, d, min_s

Output: k groups of objects and attributes having k largest scores.

1.1 $\mathbb{C} \leftarrow FindClosures(q,A_f,O_f)$; // remove redundancy (A_c)
1.2 $Ans \leftarrow FindCG(\mathbb{C}_{\geq d}, O_f)$
1.3 **if** $|Ans| \geq k$ **then return** Ans;
1.4 $k \leftarrow k\text{-}|Ans|$; // update k
1.5 $kScore \leftarrow min_s$; // initialise lower bound of score
1.6 *initialize a queue Q of size k to* ϕ
1.7 **for** *each closure* $C^i \in \mathbb{C}_{<d}$ **do**
1.8 $\quad (Q, kScore) \leftarrow FindCPG(q, C^i,A_f,Q,kScore,k,d)$
1.9 $Ans_{tmp} \leftarrow \Phi$
1.10 **while** *Q is not empty* **do**
1.11 $\quad G \leftarrow Dequeue(Q)$
1.12 $\quad Ans_{tmp} \leftarrow Ans_{tmp} \bigcup \{G\}$
1.13 $reorder(Ans_{tmp})$
1.14 $Ans \leftarrow Ans \bigcup Ans_{tmp}$; // merge groups
1.15 **return** Ans

Step 2: The candidate groups with both completely (A_c) and partially (A_p) matched attributes are generated using *FindCPG* Function in Lines 1.6-1.8. We maintain a priority queue Q to retrieve k groups with subspaces having k highest scores. We use *kScore* representing the current minimum score of the candidate groups explored so far, to prune groups with low scores. The process is elaborately discussed in Section 4.3.

Step 3: The groups of objects with their associated subspaces are dequeued from Q iteratively into Ans_{tmp}, and after re-ordering Ans_{tmp}, we merge these groups with Ans to obtain the final results of personalised groups (Lines 1.9-1.15).

4.2. Finding Groups of Objects

Objects are grouped in terms of the completely matched attribute-set, A_c. Our approach avoids redundancy among the groups. Consider two groups G^1 and G^2 (in Table 2) comprising the same set of patients {*p3:Peter*, *p5:David*}, share the same values on A_c^1 = {*high blood-pressure*} and A_c^2 = {*high blood-pressure*, *skin-condition*}, respectively, with q where v^q_{hbp} = *yes* and v^q_{sk} = *normal*. We find G^1 redundant with respect to G^2 from Lemma 1.

Lemma 1. *Let* G^1=(O^1, S^1) *and* G^2=(O^2, S^2) *be two personalised groups of objects with subspaces,* $S^1 = A_c^1 \bigcup A_p^1$ *and* $S^2 = A_c^2 \bigcup A_p^2$, *respectively. If (i)* $A_c^1 \subset A_c^2$ *and (ii)*

Algorithm 2: FindCG($\mathbb{C}_{\geq d}$, O_f)

Output: Groups with completely matched attribute-sets

2.1 $Ans_{CG} \leftarrow \phi$

2.2 **for** *each closure* $C^i \in \mathbb{C}_{\geq d}$ **do**

2.3 $\quad G \leftarrow (O : S)$ where $O = O_{C^i}$, $S.A_c = C^i$ and $S.A_p = \phi$

2.4 $\quad Ans_{CG} \leftarrow Ans_{CG} \bigcup \{G\}$

2.5 **return** Ans_{CG}

G^1		G^2		
Patients	A_c	Patients	A_c	
	high blood-pressure		*high blood-pressure*	*skin-condition*
p3:Peter	✓	*p3:Peter*	✓	✓
p5:David	✓	*p5:David*	✓	✓

Table 2: Redundancy in completely matched attribute-set (A_c)

A_c^2 is a closure of completely matched attribute-set, then G^1 is redundant with respect to G^2.

Proof. Since $A_c^1 \subset A_c^2$ and A_c^2 is a closure attribute set with respect to q, both the clusters G^1 and G^2 comprise the same set of objects (by definition). Thus, G^1 is redundant with respect to G^2 as it covers the same cluster of objects with a smaller number of attributes.

Hence, we mine non-redundant object groups using the closures of A_c ($\mathbb{C}$). Any closure mining algorithm can be applied to determine these groups. We use CHARM (34), a well-known scheme, to mine closure attribute-sets. A list of complete matching groups Ans_{CG} is formed from the closures of length $\geq d$ ($\mathbb{C}_{\geq d}$) in Function *FindCG* to ensure that the subspaces of the discovered groups contain at least d completely matched attributes. (see Algorithm 2).

Constrained A_c. We might omit the closures from $\mathbb{C}_{\geq d}$ that do not contain certain attributes which are strictly required by the query to be present in A_c.[2]

4.3. Enumerating Partially Matched Attributes

Considering attributes with partial match (A_p) as well as the complete match (A_c) provides a better understanding of the query q while mining the personalised groups. Our approach determines partially matched attribute-sets in Function *FindCPG* for those groups of objects that share less than d completely matched attributes with q, $\mathbb{C}_{<d}$ (see Lines 1.6-1.8 of Algorithm 1). We first discuss the monotonicity property of *score*

[2]Please note that one can apply the strategies of a constraint closed itemset mining algorithm (36) in the pre-processing step (before applying CHARM (34)) to handle these monotone user(/item) constraints for the early removal of an object(s) that does not have such strict properties.

G^1			G^2			
Patients	A_c	A_p	Patients	A_c	A_p	
	pain	*gender*		*pain*	*gender*	*cough*
p1:Mary	✓	✓	*p1:Mary*	✓	✓	✓
p2:Sam	✓	✗	*p2:Sam*	✓	✗	✗
p6:Jack	✓	✗	*p6:Jack*	✓	✗	✓
p7:Jeny	✓	✓	*p7:Jeny*	✓	✓	✓
p8:Brad	✓	✗	*p8:Brad*	✓	✗	✓

Table 3: Redundancy in partially matched attribute-set (A_p)

(Eq. 2) followed by the definition of redundancy in A_p. Then, we present two baselines and our approach PRESS. Table 5 summarises these three approaches.

Given two personalised groups of objects $G^1 = (O, S)$ and $G^2 = (O, S')$ where: (i) $O \subseteq O_f$, (ii) $S, S' \subseteq A_f$, (iii) $S = A_c \bigcup A_p$, $S' = A_c \bigcup A'_p$, and (vi) $A_p \neq A'_p$, we assert the monotonicity of *score* and redundancy of A_p through Lemma 2 and Lemma 3, respectively.

Lemma 2. *(Monotonicity) If the number of objects that have the same values as the query q with respect to A_p is larger than that of A'_p, i.e., $|O_{A_p}| \geq |O_{A'_p}|$, then score(O, S^1, q) ≥ score(O, S^2, q).*

Proof. Since both groups comprise the same set of objects O and $|O_{A_p}| \geq |O_{A'_p}|$ where $O_{A_p}, O_{A'_p} \subset O$, we have score($O$, S^1, q) ≥ score(O, S^2, q) from Eq. 2.

Lemma 3. *If (i) $A_p \subset A'_p$ and (ii) A'_p is a closure with respect to the objects O, then G^1 is redundant with respect to G^2.*

Proof. Since $A_p \subset A'_p$ and A'_p is a closure with respect to O, we find that objects sharing the same values with q in terms of A_p are the same as that of A'_p, i.e., $O_{A_p} = O_{A'_p}$ (by definition of closure) where $O_{A_p}, O_{A'_p} \subset O$. Thus, we have $|O_{A_p}| = |O_{A'_p}|$ which means score(O, S^1, q) = score(O, S^2, q) (from Lemma 2). Therefore, G^1 is redundant with respect to G^2 as it obtains the same score with a smaller number of partially matched attributes.

Consider G^1, G^2 and G^3 comprising the movies O={*p1*, *p2*, *p6*, *p7*, *p8*} with A_c={*pain*} and three different partially matched attribute sets, i.e., A^1_p={*gender*}, A^2_p={*gender*, *cough*} and A^3_p={*cough*}, respectively. We find score(O, A^3_p, q) > score(O, A^1_p, q) since $|O_{A^3_p}| > |O_{A^1_p}|$ where $O_{A^3_p}$ = {*p1*, *p6*, *p7*, *p8*} and $O_{A^1_p}$ = {*p1*, *p7*}. Again, $A^1_p \subset A^2_p$ with $O_{A^1_p} = O_{A^2_p}$ = {*p1*, *p7*} $\subset O$. Thus, we have score(O, A^1_p, q)= score(O, A^2_p, q) which makes G^1 redundant given G^2. Table 3 illustrates this redundancy of partially matched attribute-set A_p.

4.3.1. Baseline-1 (naïve)

Baseline-1 determines the partially matched attribute sets for a group of objects $O = O_C \subseteq O_f$ sharing same values with query on C where C is a closure of less than d completely matched attributes. It uses set-enumeration approach to cover the search-space consisting of all possible combinations of attributes $L_A = A_f - C$ in depth first manner. For a subspace $S_p \subseteq L_A$, Baseline-1 builds a candidate subspace S such that $S = S_p \bigcup C$ and computes score for objects O in subspace S. When $score(O, S, q) \leq kScore$, Baseline-1 uses the downward closure property (35) to discard all the super spaces S'_p of S_p, i.e., $S'_p \supset S_p$. However, there still remains redundancy in the discovered subspaces with respect to the partially matched attributes as stated in Lemma 3.

4.3.2. Baseline-2

Baseline-2 eliminates redundancy by mining closures of partially matched attributes for a group of objects $O_C \subseteq O_f$ that share the same values with query on $C \in \mathbb{C}_{<d}$ (Line 3.2). Baseline-2 mines the set of closures ($\mathbb{P}$) for the attributes in L_A where L_A=A_f-C with respect to the objects in O_C. It then computes the number of matching objects with the query for every P^j, i.e., $|P^j|$, where $P^j \in \mathbb{P}$ and $O_{P^j} \subset O_C$. We use $|O_{P^j}|$ to order $\mathbb{P}$ in a descending manner, i.e., $\{P^1, P^2, ..., P^{k'}\}$ where $P^j \leq P^{j+1}$. A candidate group G^j consisting of objects O with subspace $S^j = (C, P^j)$ is formed for each $P^j \in \mathbb{P}$. The ordering of $\mathbb{P}$ enables Baseline-2 to skip enumerating the closures $\{P^{j+1},\ldots,P^{k'}\} \in \mathbb{P}$, if the score of G^j, $score(O, S^j, q) \leq kScore$ (see Lemma 4). Only the groups consisting of subspaces S with at least d attributes and higher score than *kScore* are stored in the queue Q.

Lemma 4. *A set of candidate groups $\{G^j, G^{j+1}, \ldots, G^{k'}\}$ consisting of the same set of objects $O \subseteq O_f$ with the same completely matched attribute-set C and different sets of partially matched attributes $\{P^j, P^{j+1}, \ldots, P^{k'}\}$, can be pruned if $score(O, S^j, q) \leq kScore$ where $S^j = C \bigcup P^j$.*

Proof. Since the groups $\{G^1, G^{j2}, \ldots, G^{k'}\}$ are sorted in a descending manner using $|O_{Pj}|$ for j=$\{1, \ldots, k'\}$ and $O_{Pj} \subset O$ (Line 3.3 of Algorithm 3), for any two groups G^{j_1} and G^{j_2} with $j_1 < j_2$, we have $score(O, S^{j_1}, q) \geq score(O, S^{j_2}, q)$ as $|O_{Pj_1}| \geq |O_{Pj_2}|$ (see Lemma 2). Thus, if $score(O, S^j, q) \leq kScore$, then the groups $\{G^j, G^{j+1}, \ldots, G^{k'}\}$ can be pruned as all of them will have less than or equal scores to *kScore*.

4.3.3. Our Approach

We introduce PRESS in Algorithm 4 to determine the partially matched attribute sets for a group of objects $O = O_C \subseteq O_f$ that share the same values with query on $C \in \mathbb{C}_{<d}$. Since the closures are non-redundant, Baseline-2 mines partial closures from L_A = (A_f-C) for objects O_C. However, such task is computationally expensive. PRESS efficiently prunes redundant attribute sets using the pre-computed closures with at least

Algorithm 3: FindCPG(q,C,A_f,Q,$kScore$,k,d):Baseline2

Output: Groups with completely and partially matched attribute-sets

3.1 $L_A \leftarrow A_f$ - C

3.2 $\mathbb{P} \leftarrow$ *FindClosures*(q,L_A,O_C) ; // remove redundancy (A_p)

3.3 *order*($\mathbb{P}$); // To apply monotonicity property

3.4 **for** *each* $P^j \in \mathbb{P}$ **do**

3.5 $G^j \leftarrow (O : S^j)$ where $O = O_C$, $S^j.A_c = C$, $S^j.A_p = P^j$

3.6 *compute score*(O, S^j, q); // using Eq. 2

3.7 **if** *score*(O, S^j, q) $\leq kScore$ **then**

3.8 *skip all the remaining* $P^j s$; ; // using Lemma 4

3.9 **else**

3.10 **if** $|S| \geq d$ **then**

3.11 *update*(Q, k, G^j, $kScore$)

3.12 **return** (Q, $kScore$)

d completely matched attributes ($\mathbb{C}_{\geq d}$) (Theorem 1) that were initially computed for all objects in the dataset with respect to q (Line 1.1 in Algorithm 1).

PRESS first counts the objects in O_{C^j}, i.e., $|O_{C^j}|$, for $C^j \in \mathbb{C}_{\geq d}$ and uses it to order the set $\mathbb{C}_{\geq d}$ in a descending manner (Line 4.1). Then, for each $C^j \in \mathbb{C}_{\geq d}$, a candidate group G^j consisting of objects O_C and subspace $S^j = (A_c, A_p^j)$ where $A_c = C$ and $A_p^j = C^j - C$ is constructed if $C \subset C^j$ (Lines 4.3-4.5). We note that $O_{C^j} \subset O_C$ as $C \subset C^j$. Thus, $O_{A_p^j} = O_{C^j}$ as $O_{A_p^j}$ represents those objects from O_C that share the same values with q on A_p^j. The ordering of $\mathbb{C}_{\geq d}$ enables PRESS to use the monotonicity property of *score* function (see Lemma 2) in pruning subspaces with low scores. If the score of G^j, *score*(O_C,S^j,q) $\leq kScore$, PRESS skips enumerating the closures $\{C^{j+1}, \ldots, C^{k'}\} \in \mathbb{C}_{\geq d}$ using Lemma 5. Otherwise, G^j is added into the queue Q.

Assume we found a closure C={*pain*} with $|O_C|$=5 with respect to the query q where minimum subspace length d=2 and $kScore$ = 0.60. PRESS iterates $\mathbb{C}_{\geq 2}$ = {C^1, C^2, C^3, C^4, C^5} where C^1={*pain*, *cough*}, C^2={*gender*, *cough*}, C^3={*gender*, *pain*, *cough*}, C^4={*blood-pressure*, *skin-condition*}, C^5={*age*, *pain*} with cardinalities $|O_{C^1}|$=4, $|O_{C^2}|$=3, $|O_{C^3}|$=2, $|O_{C^4}|$=2 and $|O_{C^5}|$=1 for identifying the partially matched attribute-sets for objects O_C in the following manner:

- As $C \subset C^1$, we find $A_p^1 = C^1$-C ={*cough*} and score(O_C, A_p^1, q) = $\frac{|O_{C^1}|}{|O_C|} = \frac{4}{5} = 0.80$ > $kScore$. PRESS enqueues $S^1 = C \bigcup A_p^1$ into Q.
- As $C \not\subset C^2$, PRESS skips C^2 (from Theorem 1)
- As $C \subset C^3$, we find $A_p^3 = C^3 - C$ = {*gender*, *cough*} and score(O_C, A_p^3, q) = $\frac{|O_{C^3}|}{|O_C|}$

Algorithm 4: FindCPG(q,C,A_f,Q,$kScore$,k,d): PRESS

Output: Groups with completely and partially matched attribute-sets

4.1 *order*($\mathbb{C}_{\geq d}$) ; `// To apply monotonicity property`
; `/* apply Theorem 1 to mine non-redundant` A_p `*/`
4.2 **for** *each closure* $C^j \in \mathbb{C}_{\geq d}$ **do**
4.3 **if** $C \subset C^j$ **then**
4.4 $G^j \leftarrow (O : S^j)$ where $O = O_C$,
4.5 $S^j.A_c = C$ and $S^j.A_p = C^j - C$
4.6 *compute score*(O,S^j,q); `// using Eq. 2`
4.7 **if** *score*(O,S^j,q) $\leq$ *kScore* **then**
4.8 *skip all the remaining* C^j*s*; ; `// using Lemma 5`
4.9 *update*(Q, k, G^j, *kScore*)
4.10 **return** (Q, *kScore*)

$= \frac{2}{5} = 0.40 <$ *kScore*. PRESS skips C^3 and the remaining closures, i.e., C^4 and C^5 (from Lemma 5).

Theorem 1. *A closure of partially matched attributes $P \subseteq (A_f$-$C)$ with respect to the objects $O_C \subseteq O_f$ can be found if there exists another closure $C' \subset A_f$ with respect to q such that $C \subset C'$.*

Proof. Since P is a closure of the attributes in (A_f - C) with respect to the objects $O_C \subseteq O_f$, there remains no other closure $\hat{P}$ such that $\hat{P} \supset P$ and $O_P, O_{\hat{P}} \subset O_C$ with $O_P = O_{\hat{P}}$ (by definition). Therefore the attribute set $C' = P \bigcup C$ is a closure with respect to q where $O_{C'} = O_P$, i.e., objects sharing the same values with q on both P and C, and $C \subset C'$.

Lemma 5. *A set of candidate groups $\{G^j, G^{j+1}, \ldots, G^{k'}\}$ consisting of the same set of objects $O \subseteq O_f$ with the same completely matched attribute-set C and different sets of partially matched attributes $\{A_p^j, A_p^{j+1}, \ldots, A_p^{k'}\}$, can be pruned if score(O, S^j, q) $\leq$ kScore, i.e., $S^j = A_c \bigcup A_p^j$, where $A_c = C$, $A_p^j = C^j - C$, $C \subset C^j$ for $C \in \mathbb{C}_{<d}$ and $C^j \in \mathbb{C}_{\geq d}$.*

Description	grp. no.	gender	age	high blood-pressure	pain	cough	skin-condition	objects	score
Query : *Lisa*	n/a	female	teenager	yes	chest	dry	normal	n/a	n/a
Groups with completely matched attribute-set	g_1.	✓	✗	✗	✓	✓	✗	{*p1:Mary*,*p7:Jeny*}	1.0
Groups with completely and partially matched attribute-set	g_2.	✓	✗	✗	✓*	✓	✗	{*p1:Mary*, *p4:Rose*, *p7:Jeny*}	0.67
	g_3.	✓*	✗	✗	✓	✓	✗	{*p1:Mary*, *p6:Jack*, *p7:Jeny*, *p8:Brad*}	0.50

Table 4: Top-3 groups of patients similar to the query patient *Lisa* with $k = 3$, $d = 3$ and min_s=0.50.

Issues	Baseline-1	Baseline-2	PRESS
Redundancy in A_c	*no*	*no*	*no*
Redundancy in A_p	*yes*	*no*	*no*
Enumeration strategy to find A_p	set enumeration	mines closures of A_p	uses pre-computed closures of A_c
Efficiency	moderately fast	slow	fast
Monotonicity of *score*	*yes*	*yes*	*yes*

Table 5: Comparison among Baseline-1, Baseline-2 and PRESS for *FindCPG* function.

Proof. $\{C^j, C^{j+1}, \ldots, C^{k'}\} \in \mathbb{C}_{\geq d}$ is ordered in terms of $|O_{C^j}|$ for j=$\{1, \ldots, k'\}$ (Line 4.1 of Algorithm 4). For any two groups G^{j_1} and G^{j_2} ($j_1 < j_2$), we have $score(O, S^{j_1}, q) \geq score(O, S^{j_2}, q)$ as $|O_{A_p^{j_1}}| \geq |O_{A_p^{j_2}}|$ (from Lemma 2) where $O_{A_p^{j_1}} = O_{C^{j_1}}$ and $O_{A_p^{j_2}} = O_{C^{j_2}}$. Thus, if $score(O, S^j, q) \leq kScore$, the groups $\{G^j, G^{j+1}, \ldots, G^{k'}\}$ can be pruned as the scores of thses groups will be less than or equal to $kScore$.

Table 4 presents the final output of the motivating example in Table 1. Top-3 groups are identified as similar to the query patient *q:Lisa* for the values, i.e., $d = 3$, $k = 3$ and $min_s = 0.50$. Attributes labelled as ✓ and ✓* correspond to complete and partial matches with respect to q, respectively. PRESS finds the group g_1 comprising the patients, i.e., *Mary* and *Jeny*, to be the most similar to Lisa with respect to the subspace $S = (A_c, A_p)$ where A_c=$\{gender, pain, cough\}$ and $A_p = \phi$. Group g_2 finds another patient, i.e., *p4:Rose* after relaxing value on pain attribute and obtains score as 0.67. Similarly, group g_3 relaxes gender attribute and identifies two new *male* patients i.e., *Jack* and *Brad*, as compared to g_1. However, if the medical practitioners do not want to compromise values on two attributes, i.e., *pain* and *cough*, corresponding to the values $v_{pn}^{Lisa} = chest$ and $v_{cg}^{Lisa} = dry$, respectively, then PRESS would discover only the two groups g_1 and g_3 for the same set of values in d, k and min_s.

4.3.4. Complexity of PRESS

PRESS uses CHARM (34) to find closures with respect to the query that takes time $O(nc)$ for c=♯closures. For each closure $C^i \in \mathbb{C}_{<d}$, PRESS determines closed supersets from $\mathbb{C}_{\geq d}$ taking $O(mc_1c_2)$ time in total where c_1=$|\mathbb{C}_{<d}|$ and c_2=$|\mathbb{C}_{\geq d}|$. In the worst case, PRESS needs to update k entries of queue Q consuming O($log(k)$) time (using heap implementation) for each $C^i \in \mathbb{C}_{<d}$. Hence, the worst-case time-complexity of PRESS is $O(nc + (mc_1c_2 + (c_1 logk))) = O(nc) = O(n2^m)$ with max. value of c being 2^m. In our experiments with real and synthetic data, we find $c \ll 2^m$ in most cases. PRESS mines k groups of objects with subspaces from c closures. For each group PRESS needs to keep track of a subspace S. The maximum length of the subspace can be m. Therefore, the worst-case space-complexity of PRESS would be $O(c + km) = O(2^m + km)$ where $c = 2^m$.

4.3.5. Identifying the Divergent Objects

The objects within each personalised group share similarity in terms of the completely and partially matched attributes with the query. Sometimes the discovered groups might consist of a large number of objects. In that case, it would be more desirable to provide the user with a smaller number of objects from that group. One option is to identify *skylines* (discussed in Section 2.1), i.e., objects that are not dominated by any other object, which may return all objects within the dataset if there are many objects with different dominant attributes. Alternatively, one can determine only the *dominating* objects which can often yield no result since there is usually no one object that is dominant in all features within the dataset. Hence, we are interested in determining those objects having the highest/lowest (most preferred) values for the maximum number of attributes within each personalised group. We refer to these objects as *Divergent* objects. Algorithm 5 gives the overview of such an approach to identify a short list of the divergent objects from a personalised group $G = (O, S)$.

We consider all attributes $\hat{A}$ from A_f except the completely matched attributes ($S.A_c$) while identifying the divergent objects from the group G because all objects share the same categorical values with the query on $S.A_c$ (see Line 5.1). We assume that the user provides a preference order for the categorical attributes so that the attribute values can be assigned numerical weights according to the preferences. The lower the preference order, the larger the numerical weight would be. We further assume that all attributes have same priority. Hence, values from different attributes with the same preference order would receive the same numerical weights in our weighting scheme as illustrated in Lines 5.2 – 5.7. Please note that, if the user has a specific priority for the attributes, our weighting scheme can be easily modified to adjust the weights according to the user preferences. However, for the numerical attributes, we first need to discretise the attributes in order to apply PRESS for mining the personalised groups. Then, we can use the original numerical values for identifying the divergent objects from the groups. In this case, no weighting scheme is required.

We provide a straight-forward technique for determining the divergent objects from the given group G in Lines 5.8 – 5.13. At first, we build a list L that sorts all the objects of the group in descending order of the weights (or numerical values) for an attribute $a_i \in \hat{A}$ in Line 5.10 i.e., objects with the most preferred attribute value are placed at the top of the list L. Then, we use a matrix $\hat{D}$ to track the objects $o_{\{1,\ldots,j\}}$ with the highest weights $w_{\{1,\ldots,j\}}$ by assigning value 1 to $\hat{D}[a_i][o_{\{1,\ldots,j\}}]$ where $(o_{\{1,\ldots,j\}}, w_{\{1,\ldots,j\}}) \in L$ and $1 \leq j \leq |O|$ (see Line 5.11). We repeat these two steps for all attributes in $\hat{A}$. After filling the matrix $\hat{D}$, Function *findMaxObjs()* returns the divergent objects R_{div} from $\hat{D}$ which have the value 1 (the most preferred value) for the maximal no. of attributes (see Line 5.15).

Complexity: Assignment of numerical weights to the categorical attributes (Lines 5.1 – 5.7) from the preference matrix P takes $O(n'm')$ time where $|O| = n'$ and $|\hat{A}| = m'$. The time complexity of sorting and finding objects with the highest weights for each

Algorithm 5: Identifying Divergent Objects from a Personalised Group

Input : $G = (O, S)$, A_f, $P_{|A_f| \times (cardinality\ of\ attribute)}$ (Preference Matrix)
Output: A set of divergent objects with subspaces.

5.1 $\hat{A} \leftarrow A_f$ - $S.A_c$; `// obtains the attributes where objects do not necessarily share the same values with respect to the query`
5.2 $\widehat{W}_{|\hat{A}| \times |O|} \leftarrow \phi$; `// stores numerical representation of the group` G
5.3 **for** *each* $o_j \in O$ **do**
5.4 **for** *each* $a_i \in \hat{A}$ **do**
5.5 $v := v_{a_i}^{o_j}$; `// obtains the value on attribute` a_i `of object` o_j
5.6 $p := P[a_i][v]$; `// obtains the preference order from` P
 `/* assigns numerical weights to the categorical values */`
5.7 $\widehat{W}[a_i][o_j] := 1/(2^{(p-1)})$
5.8 $\hat{D}_{|\hat{A}| \times |O|} \leftarrow [0]_{|\hat{A}| \times |O|}$; `// initialises with 0 values`
`/* Tracking of objects having the highest attribute-values in` $\hat{D}$ `*/`
5.9 **for** *each* $a_i \in \hat{A}$ **do**
 `/* retrieves an ordered list of pairs of objects and weights */`
5.10 L:$\{(o_1, w_1), \ldots, (o_{|O|}, w_{|O|})\} \leftarrow sort(\widehat{W}[a_i])$
5.11 $\hat{D}[a_i][o_{\{1,\ldots,j\}}] := 1$ where $o_{\{1,\ldots,j\}}$ have the highest weights $w_{\{1,\ldots,j\}}$ for $1 \leq j \leq |\ O\ |$
`/* finds object(s)` $T \in O$ `with the maximum` $\sum_{i=1}^{|\hat{A}|} \hat{D}[a_i][T]$ `*/`
5.12 R_{div}=$\{(T^1, S^2), \ldots, (T^t, S^t)\} \leftarrow$ *findMaxObjs*($\hat{D}$, $\hat{A}$)
5.13 **return** R_{div}

attribute of $\hat{A}$ (Lines 5.10−5.11) is O($n'logn'$). Hence, for m' attributes, the tracking loop (in Lines 5.9 − 5.11) takes in total O($m'n'logn'$) computation time, which can be easily parallelized. Function *findMaxObjs()* consumes O($m' + n'$) time in identifying R_{div} from $\hat{D}$. Therefore, the overall time complexity of Algorithm 5 is O($n'm'+m'n'logn'+m' + n'$) = O($m'n'logn'$).

Example: Assume the medical practitioners provide the preference orders for all attributes describing the medical records in Table 1. Table 7 assigns weights $w \in (0,1]$ corresponding to the preferences listed in Table 6 by applying the strategy mentioned in Algorithm 5. For example, $\{\frac{1}{2^0}, \frac{1}{2^1}, \frac{1}{2^2}\}$ = {1, 0.5, 0.25} are the numerical weights cor-

Attrib.	gender	age	(high) blood-pressure	pain	cough	skin-condition
Preference Order	female	teenager	yes	chest	dry	normal
	male	adult	no	muscle	wet	acne
		toddler/ senior		back/ none	normal	cold-sores
						discolored/ wart/rashes

Table 6: Preference Matrix (P) listing the preferred orders for the attributes describing the medical records in Table 1

Attrib.	gender	age	(high) blood-pressure	pain	cough	skin-condition
Preference Weights	1	1	1	1	1	1
	0.5	0.5	0.5	0.5	0.5	0.5
		0.25		0.25	0.25	0.25
						0.125

Table 7: Weights of the attributes corresponding to the preference orders of P in Table 6

patients	A_c		A_p
	pain	cough	gender
p1:Mary	✓	✓	✓
p6:Jack	✓	✓	✗
p7:Jeny	✓	✓	✓
p8:Brad	✓	✓	✗

Table 8: Group g_3 selected from Table 4 comprises the four patients with A_c= {*pain*, *cough*} and A_p = {*gender*}.

patients	A_p	$(A_f$-$A_c)$			no. of highest values
	gender	age	high blood-pressure	skin-condition	
p1:Mary	1	0.5	0.5	0.25	3
p6:Jack	0.5	0.25	0.5	0.5	2
p7:Jeny	1	0.25	0.5	0.125	2
p8:Brad	0.5	0.5	0.5	0.5	3

Table 9: Numerical representation of the group g_3 according to the preference weights in Table 7

responding to the values {*chest*, *muscle*, *back/none*} which are ordered according to the paractioners' preferences for the attribute *pain* given the query patient *Lisa*. We consider the group g_3 (in Table 8) selected from Table 4 comprising the four patients with A_c= {*pain*, *cough*} and A_p = {*gender*} for the determination of divergent objects. We present the numerical description of this group according to the preferences of the practitioners in terms of the four attributes, i.e., *gender*, *age*, *high blood-pressure* and *skin-condition*, as shown in Table 9. We find *Mary* and *Brad* as divergent objects with max. no. of highest preference values = 3 with respect to two different subspaces, i.e., {*gender*, *age*, *skin-condition*} and {*age*, *high blood-pressure*, *skin-condition*}, respectively.

5. Experimental Results

Since PRESS mines query specific object-clusters (not the general object-clusters in the literature) and there is no ground truth available for these personalised clusters, we are unable to apply cluster validation techniques, e.g., Normalized Mutual Information (NMI), purity, F-measure and precision/recall, for the evaluation of PRESS's performance. Instead, we perform qualitative case studies on two different datasets, i.e., AMiner[3] (Academic Citation Network) and Laptop[4] to investigate the interpretability of the discovered groups of objects by PRESS. AMiner enables us to evaluate PRESS's feedback to the identified clusters of scholars for a given well-known prolific researcher chosen as a query object. On the other hand, the Laptop dataset allows us to measure the effectiveness of PRESS's performance in terms of providing guidelines for the inexperienced/novice tech-users in the market so that they can purchase desirable products within their constraints.

We compare the performance of PRESS with a benchmark approach *CLICKS**, which is based on a general subspace clustering algorithm *CLICKS* (30), after making modifications so that it can retrieve personalised (not general) clusters. Section 5.2 provides analytical comparisons among PRESS, *CLICKS**, Baseline-1 and Baseline-2 in terms of time and space complexity. We also conduct an enhanced scalability test of

[3]https://aminer.org/data

[4]https://www.kaggle.com/ionaskel/laptop-prices

Complexity	CLICKS*	PRESS	Baseline-1	Baseline-2
Time	$O(t^m\|L\|)$	$O(nc)$	$O(n2^m)$	$O(nc(c + clogc)) = O(nc^2)$
Space	$O(m^2t^2 + \|L\|)$	$O(mk + c)$	$O(mk + 2^m)$	$O(mk + c^2)$

Table 10: Analytical comparisons of CLICKS*, PRESS, Baseline-1 and Baseline-2 in terms of run-time and memory consumption.

PRESS against *CLICKS** on synthetic datasets with respect to different dataset-sizes and number of attributes in Section 5.5. We use a 3.40GHz Core i7 PC with 16GB RAM to implement the algorithms in JAVA.

5.1. CLICKS**: Benchmark Approach

As benchmark we use CLICKS (30), which mines clusters for a categorical dataset as maximal dense *k-partite* cliques. It is worth noting that CLICKS results in groupings where the attributes may take on multiple values, i.e., *multi-valued attribute* clusters. Also, it is not directly applicable for retrieving personalised clusters as was not intended for supporting queries. We adapt *CLICKS** by making two significant changes to it. First, we inspect whether the candidate set of vertices/attribute values to be added to the current clique contains any value shared with the query at the clique detection phase. If no such values exists, we stop expanding the clique further and eliminate it. Second, we split the clusters containing multiple values from the same attribute at post-processing stage, since such cliques would allow a large number of objects resulting in poor quality clusters for query. We might require to filter the split clusters again with respect to query attributes.

5.1.1. Complexity of CLICKS*

The cost of mining all maximal k-partite cliques by CLICKS* is $O(mt|L|)$ for m attributes with t values per attribute and L set of cliques (30). The maximum cost of splitting a clique that contains multiple values from the same attribute would be $O(t^m)$. Therefore, the overall time-complexity of CLICKS* in the worst-case scenario would be $O(mt|L| + t^m|L|) = O(t^m|L|)$. The *k-partite* graph used in CLICKS* might have $O(mt)$ vertices and $O(m^2t^2)$ edges (where the graph is dense (30)). Hence, CLICKS* requires maximum of $O(mt + m^2t^2 + |L|) = O(m^2t^2 + |L|)$ space.

5.2. Analytical Comparisons

Table 10 lists the time and space complexity for all four approaches, i.e., PRESS, CLICKS*, Baseline-1 and Baseline-2. We observe that the runtime of CLICKS* is $O(t^m|L|)$ which is exponential in number of attributes m. Further, there can be an exponential number of cliques for certain graphs, e.g., $|L| = 3^{m/3}$ for the Moon-Moser graphs (37). However, PRESS's runtime is $O(nc)$ where $c \ll 2^m$ in most cases and n corresponds to the number of objects. Therefore, PRESS has a much smaller time

AMiner			Laptop				
Description	Attributes	#values	Description	Attributes	#values	Attributes	#values
	publication	20		type	7	price	8
#scholars	citations	20	#laptops	weight	10	screen-size	10
=1.7M	h-index	20	= 1303	OS	10	GPU	103
	co-authors	20		CPU-Model	94	screen-	41
#attributes	co-authors/paper	20	#attributes	CPU-Speed	30	property	
= 7	papers/year	20	= 11	RAM	10		
	experience	20		storage	35		

Table 11: Description of two real datasets, i.e., AMiner (Academic Citation Network) and Laptop.

complexity than CLICKS* as $O(nc) \ll O(t^m|L|)$. In regards to the memory consumption, PRESS requires $O(mk + c)$ which is smaller than the memory cost of CLICKS* $O(m^2t^2 + |L|)$.

The computation time for Baseline-1 and Baseline-2 are $O(n2^m)$ and $O(nc^2)$, respectively. Though the cost of running Baseline-1 algorithm is asymptotically equal to the cost of PRESS for the worst case scenario, i.e., $c = 2^m$, Baseline-1 generates redundant groups. However, Baseline-2 is able to mine non-redundant groups but it takes longer time than PRESS ($O(nc^2) \gg O(nc)$). We also find that PRESS consumes less memory as compared to the baseline approaches in most cases.

5.3. Laptop Case Study

We use a laptop dataset, i.e., a collection of 1303 laptops with 11 attributes describing their configurations in terms of processor, storage, screen, graphics and price (see Table 11). All attributes are categorical except *screen-size*, *weight* and *price* which are discretised into 9, 10 and 7 intervals, respectively. Here, we study the two scenarios (discussed in Section 1.1), i.e., (i) Query Exploration: a non-technical customer wishes to find a Windows counterpart for a *Macbook* and (ii) Query Insights: a small business owner is querying for the typical configuration of *gaming* laptops with certain desirable properties.

5.3.1. Query Exploration

Assume an individual is impressed by his colleague's Macbook in terms of portability as well as weight, and would like to find a similar laptop with Windows OS. Since our user is not a technical person, PRESS retrieves the characteristics of a Macbook to construct a reference query q. Table 12 describes the reference query q in the header row and reports nine groups of laptops from top-13 determined by PRESS for the values $min_s = 35\%$, $d = 7$ and $k = 15$ (see Section 4.3.3). As our user is not willing to compromise on *type*, *weight* and *OS*, all groups must contain at-least these three attributes as a complete match, i.e., portable, light weight and having *Windows10* OS.

The first five groups g_1-g_5 have no partial matches, i.e., 2 laptops per group on average. This is the best possible answer set for a MacBook counterpart as the laptops from

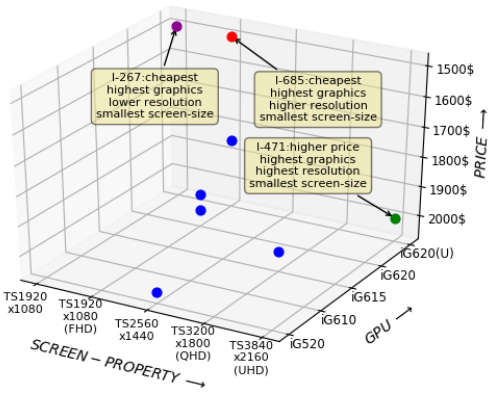

Figure 1: Illustration of three divergent laptops, i.e., l_{267}(purple), l_{685}(red) and l_{471}(green), within group g_{12} with respect to *screen-property*, *GPU* and *price*. The arrows show the direction to the most preferred values in the corresponding axes (attributes).

these groups share the same values on 7 attributes (on average) with the reference query while satisfying all strict requirements. However, we find eight additional groups of laptops (g_6-g_{13}) listing 6 laptops/group where we provide a variety of options exploiting the flexible criterion of the user. Table 12 shows 6 out of the 8 groups that are further explored. The identified groups are different from each other with respect to the partially matched attributes and similar in terms of the constrained attributes. The user may find these diverse groups of laptops useful, e.g., group g_{12} obtains 5 additional laptops by relaxing value of *price* as compared to group g_5 consisting of 3 laptops, with respect to the reference query.

Description	group no.	*type*, *weight*, *OS*	*CPU-model*	*CPU-speed*	*RAM*	*storage*	*price*	*screen-size*, *GPU screen-property*,	♯objs
Attribute values of the query object (q)	n/a	*2in1Convertible*, w_3 (1-1.5)*kg*,*Windows10*	*Intel-Corei7*	*2.7 GHz*	*8GB*	*SSD-512GB*	p_7 (≥$2*K*)	in_5∈ (14”,15”], *AMDRad-eonPro*,*IPSRetina2880x1800*	n/a
Groups with completely matched attributes(score=1.0)	g_1.	✓	✓	✓	✓	✓	✓	✗	1
	g_3.	✓	✓	✓	✓	✗	✓	✗	2
	g_5.	✓	✓	✗	✓	✓	✓	✗	3
Groups with completely and partially matched attributes (ordered by score)	g_6.	✓	✓	✗	✓*	✓	✓	✗	6
	g_8.	✓	✓	✓	✓*	✗	✓	✗	4
	g_{10}.	✓	✓	✗	✓	✓*	✓	✗	6
	g_{11}.	✓	✓	✓	✗	✓*	✓	✗	4
	g_{12}.	✓	✓	✗	✓	✓	✓*	✗	8
	g_{13}.	✓	✓*	✗	✓	✓*	✓	✗	8

Table 12: Nine groups of laptops from top-13 meeting the constraint on attributes *type*, *weight* and *OS* with $k = 15$, $d = 7$ and min_s=35%.

Laptops	Attributes	Groups
l_{267} (Asus Zenbook), l_{685} (Lenevo Yoga 730)	*screen-size*, *GPU*, *price*	g_{12}
l_{471} (Lenevo Thinkpad Yoga)	*screen-size*,*GPU*, *screen-property*	g_{12}
l_{858} (HP Elitebook)	*CPU-speed*, *screen-size*, *GPU*	g_{10}, g_{13}
l_{344} (Lenevo Yoga 920)	*screen-property*, *RAM*, *GPU*	g_6
l_{191} (Lenevo Thinkpad Yoga)	*RAM*, *storage*, *screen-property*	g_8, g_{11}
l_{597} (Lenevo Thinkpad Yoga)	*GPU*, *storage*, *screen-size*	g_{10}, g_{13}

Table 13: List of divergent laptops from different groups of laptops

Finding the right laptop in larger groups might be difficult, thus we report only the divergent laptops using the strategy described in Section 4.3.5. We assume our non-technical user is similar to most customers and prefers to pay as little as possible while retrieving the best quality product. In Table 13, we provide a list of seven diverse laptops that have the most preferred values across the top-ranked groups on maximum number of partial and non-matching attributes, e.g., laptop l_{344} has a better display, larger RAM and quality graphics as compared to the other laptops of group g_6. Table 13 suggests different brands as an alternative for a MacBook such as laptops l_{267} and l_{685}, which are the cheapest, have the smallest screen-size and the highest quality graphics in group g_{12} (see Fig. 1), but our customer might prefer l_{685} over l_{267} for a better screen resolution. Another possible outcome of the divergent objects is l_{471}, which comes at a greater price but offers a laptop with the best display and highest quality graphics as well as the smallest screen size.

5.3.2. *CLICKS* on Laptop*

We run CLICKS* using $\alpha = 2.5$ on Laptop dataset given the same reference query that has been used for PRESS in Section 5.3.1. We know (from Section 5.1) that two major significant changes are required to the CLICKS algorithm for obtaining personalised clusters from the general clusters. The first stage of changes enables us to obtain clusters of laptops that have at least one matching attribute value with the given query. This step results in 1272 *multi-valued attribute* clusters, i.e., clusters where the attributes can have multiple values. However, we would not be able to apply the second stage, i.e., partitioning, for some clusters that are in full space. These clusters may span over the entire attribute set as well as containing attributes with high cardinality, e.g., the no. of possible values for *CPU-model* and *GPU* are 93 and 103 respectively (see Table 11). We encountered memory error while running the prohibitive partitioning process of such clusters. Hence, we chose 5 clusters out of 1272 with the highest similarity scores, and we used them as the final output for CLICKS* on the Laptop dataset.

The top-5 clusters/groups $\{g_1, g_2, g_3, g_4, g_5\}$ for CLICKS* consist of the following number of laptops within each group $\{22, 22, 32, 22, 52\}$ with corresponding similarity scores {59%, 58%, 50%, 48%, 46%}. The average cluster-size is 30 and the number of

Scenes	Constrained Attributes				Flexible Attributes						
	type	screen-size	screen-prop.	price	CPU-model	CPU-speed	RAM	storage	GPU	OS	weight
S-1 S-2	*Gaming*	$in_6 \in$ [15-16")	*IPSFHD-1920x1080*	p_5 [\$1K,\$1.5K) p_6 [\$1.5K,\$2K)	*Corei7*	*2.8GHz*	*16GB*	*SSD128GB +HDD1TB*	*NvdiaG TX1060*	*Windows10*	$w_5 \in$ [2 -2.5kg]

Table 14: Description of the Constrained and Flexible attributes for two scenarios, i.e., S-1 and S-2.

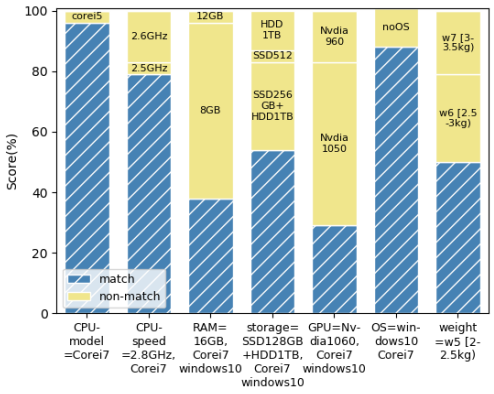

Figure 2: Description of groups of laptops with the maximum scores for each flexible attribute as a partial match in S-1 for d=5, k=30 and min_s=30%

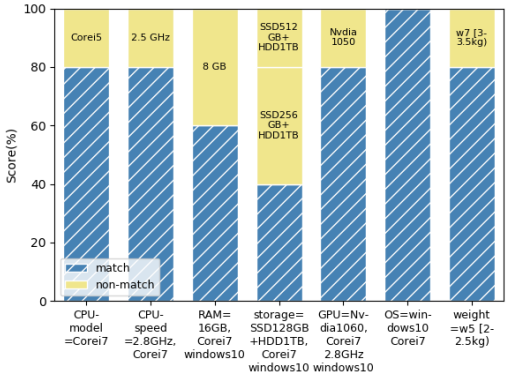

Figure 3: Description of groups of laptops with the maximum scores for each flexible attribute as a partial match in S-2 for d=6, k=20 and min_s=30%

the overall distinct laptops is 75. On average for each cluster, we have 6 attributes where each attribute can take approximately 4 different values. It would be difficult for an inexperienced user to select his/her desirable laptop since each group consists of a large number of laptops to explore and even within a single group, we have multiple values per different attributes. On the other hand, we found the average group-size 3.8 and the distinct number of laptops 10 for the top-5 representative groups of PRESS: {g_1, g_3, g_5, g_6. g_8} from Table 12. Therefore, PRESS mines reasonable size clusters of single-valued subspaces with two different notions of complete and partial matches with respect to the user given specification, and enables the user to identify suitable laptops after his/her choice.

5.3.3. Query Insights

In this scenario, we study a small business owner who wishes to explore the gaming market. Looking for a marketing strategy for his current stock of prototype gaming laptops with the same configurations (in Table 14), he knows that a large display with high resolution is a must for gaming enthusiasts. He wants to find features that may promote sales in a lower price range. Our approach can be effectively applied to this scenario.

Since the owner is strict on screen size and screen property, we consider *type*, *screen-size*, *screen-property* and *price* as constrained attributes for the query q (the values are shown in Table 14). We analyse two scenarios S-1 and S-2: the queries are the same

except for the price range, i.e., $p_5 \in$ [\$1k,\$1.5k) and $p_6 \in$ [\$1.5k,\$2k). Our approach identifies the top 30 and 20 groups of laptops for S-1 and S-2, respectively. For S-1, we find 24 laptops meeting the constraints, and for S-2, we find 5 laptops. For each flexible attribute, we report the subspace that achieves the maximum score for that attribute as a partial match in Fig. 2 and Fig. 3 for S-1 and S-2, respectively. The values of the co-occurring partially matched attributes are reported along the x-axis with the corresponding attribute. The key observations are:

- A model with 16GB RAM and a Core i7 CPU would be a cutting edge for p_5, as the RAM score increases from 30% to 60% as we move from p_5 to p_6. A similar observation applies to a GPU with *NvdiaGTX1060* graphics. Hence, if the owner wants orders at the lower price p_5, 16GB of RAM (or *NvdiaGTX1060* GPU) are desirable.
- *Core i7* processor (with speed 2.8GHz) is popular for gaming laptops and must be treated as a default configuration for CPU-model and CPU-speed of the lower price-ranged (p_5) gaming laptops.
- Since gaming laptops have more storage than asked in the query (obtains a low score=40% for the value SSD128GB+HDD1TB at the higher price range p_6), a competitive offer needs at least 256 GB of storage.
- Higher priced gaming laptops have a 2.8GHz *Core i7* with quality graphics *NvdiaGTX1060* as default as a score of 80% shows for {Corei7, 2.8GHz} co-occurring with *NvdiaGTX1060*.

Analyzing the gaming market for a business owner is merely one of the applications of our framework and might seem obvious for an experienced tech-user. However, our strategy can be useful in many application domains, e.g., electric vehicles and nano-technologies, where the dealers or customers might not know the availability and association of features concerning user demands and budgets.

5.4. AMiner Case Study

AMiner is a large academic social network comprising 1.7M authors, 2.1M papers and 4.3M coauthor relationships. We consider 7 numerical attributes: publications, citations, h-index, papers/year, co-authors, co-authors/paper and research experience. The

groups	*publication*	*citation*	*h-index*	*co-authors*	*co-authors /paper*	*papers /year*	*experience*	♯objects	score
g_1.	✓*	✗	✓	✗	✓	✓	✗	8	88%
g_2.	✓	✗	✗	✓	✗	✓*	✓	5	80%
g_3.	✓	✗	✓	✗	✗	✓*	✓	10	60%

Table 15: Top-3 groups with the completely and partially matched attributes determined by PRESS for the query author: Jiawei Han

Parameter	Range	Default
♯attributes	12, 16, 20, 24, 28	16
dataset-size	20K,40K,60K,80K,100K	60K
k	10, 20, 30, 40, 50	40
min_S	30%, 40%, 50%, 60%	40%

Table 16: Experimental Setup for Synthetic Datasets

attributes are categorised into 20 intervals (see Table 11). We use k-means clustering with a fixed seed number to discretise each attribute and to achieve a pseudo-randomised algorithm. We consider *Jiawei Han*, a prolific researcher as a query author and conduct an illustrative case study to analyse the ***Query Insights***.

PRESS determines the *top*-15 personalised groups of authors and their corresponding subspaces for d=4, k=15 and min_s=*50%*. We find 11 groups of authors who illustrate their similarity with Han in terms of the completely matched attributes. In this case, we have 22 distinct authors in total with 3 authors/group (on avg.). However, PRESS finds another 4 groups with 7 authors/group (on avg.) showing similarity in terms of complete and partial matches with Han. We find 7 new authors from these groups. Among all the groups, PRESS identifies 5 prominent researchers, i.e., 'Tom Henzinger', 'Sushil Jajodia', 'Ian Foster', 'Chris Faloutsos' and 'Jack J. Dongarra', who are the most frequent in top-15 with different subspaces and have research profiles similar to Han.

The top ranked complete matching groups identify those researchers having very high profiles similar to Han that results in small groups. PRESS finds more promising researchers by relaxing certain criteria of the query. We highlight that all authors from the top-15 groups appear in the first 300 DBLP prolific author-list[5] with Han.

Table 15 reports the top 3 groups of authors and their corresponding subspaces. Group g_1 with 8 authors obtains a score = 88% for {*publication*, *h-index*, *co-authors/paper*, *papers/year*}, where *publication* is selected as a partially matched attribute. These 8 authors have on avg. 39 as h-index, 358 publications, 12 papers per year and 0.25 as co-authors/paper whereas Han has 462 publications, 14 papers/year and 0.38 co-authors/paper. 60% of the authors in g_3 would be able to reach the metric papers/year of Han whereas the probability of reaching this metric is high (=80%) for g_2, though the authors from both groups have similar research experience and publications as Han. The authors in g_2 are highly collaborative since their subspace includes *co-authors* as a complete match. Hence, researchers should focus on quality research and collaboration with their peers to become a prolific researcher like Han.

*5.4.1. CLICKS*on AMiner*

We apply *CLICKS** to determine the top-5 groups of authors similar to Jiawei Han. We use threshold α=0.8 and obtain top-5 clusters $\{g_1, g_2, g_3, g_4, g_5\}$ consisting of the

[5]http://dblp.uni-trier.de/statistics/prolific1

following number of authors within each group {25853, 1327, 4886, 15455, 5114} with corresponding similarity scores {50%, 33.33%, 33.33%, 33.33%, 33.33%}. It is very difficult to interpret the notion of similarity between the identified authors with the query Han because of the large groups. Authors from g_1 with 50% similarity score, have 6 papers and 1.54 co-authors/paper (on avg) which also indicates that $CLICKS^*$ fails to identify the prolific authors similar to Han.

The CLICKS algorithm considers maximal k-partite clique as a cluster which might cause objects to belong to the same cluster while having different values on the same attribute. As a result, the cluster size increases. Further, CLICKS does not aim to cluster people based on the attribute values of some particular query object. Although in the CLICKS* algorithm, we consider only those clusters that have at least one matching attribute value with the query, and later we partition the large multi-valued clusters to obtain single-valued small size clusters, we still can not guarantee that the modified clusters have sufficient query features. This explains why CLICKS* yields poor quality personalised clusters with low similarity scores.

Therefore, PRESS captures useful information for an in-depth analysis of the query author, Jiawei Han. It determines other prolific authors with the subsets of attributes who are similar to Han.

5.5. Scalability Study on Synthetic Datasets

We compare the performance of PRESS with three approaches, i.e., Baseline-1, Baseline-2 and $CLICKS^*$. We generate synthetic datasets using a uniform distribution with values $\in$ [1,100]. To study the scalability, we evaluate the effects on execution time by four parameters, i.e., no. of attributes, dataset size, no. of retrieved groups k and minimum score of the groups min_S. We report the average run time by considering each object as a query for all approaches. Table 16 shows the values used for each parameter in our experiments. We set the cardinality of attributes to 10 and minimum subspace length d to 7.

Effect of no. of Attributes. Fig. 4(a) shows the run time (in log scale) required by the Baseline-1, Baseline-2, $CLICKS^*$ and PRESS for different no. of attributes. The run time of Baseline-1 and Baseline-2 increase rapidly in comparison to PRESS with an increasing number of attributes. However, PRESS's run time decreases as the no. of attributes reaches 20 since it prunes the groups of large-size subspace with low scores. PRESS is 3 times (on avg.) faster than the baselines (Please note that the y-axis in Fig. 4(a) is in log scale). As for $CLICKS^*$, it shows poor performance since it requires significant time for retrieving personalised groups from the general groups within multi-valued subspace. PRESS is two orders (on avg.) faster than $CLICKS^*$.

Effect of Dataset size. Fig. 4(b) shows the run time (in log scale) of the four approaches for different data sizes. The execution time of all approaches increases with data size. Since Baseline-2 generates non-redundant groups, it is slower than Baseline-1. PRESS is significantly faster than both approaches. $CLICKS^*$ is also two orders slower than

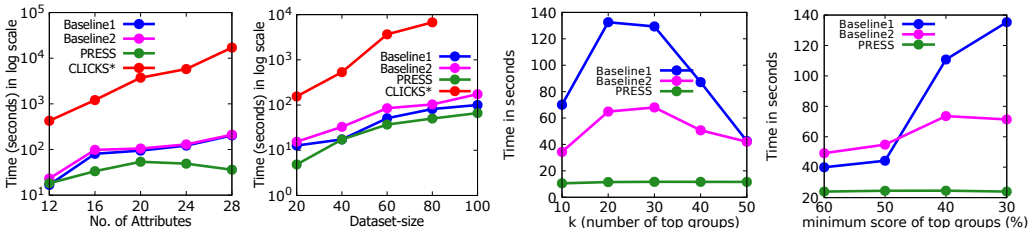

Figure 4: Effect of (a) no. of attributes, (b) dataset size (in KB)[6] (c) k and (d) min_s on run time.

PRESS. We would not be able to compute the run time of *CLICKS** for dataset size=100 KB as we encountered a memory error due to the large number of objects.

Effect of *k*. We vary k from 10 to 50 and study the performance in Fig. 4(c). PRESS's run-time almost remains constant with the increase in number of groups whereas both Baseline-1 and Baseline-2's run-time are largely affected by k. Note that PRESS is two and three times (on avg.) faster than Baseline-2 and Baseline-1, respectively.

Effect of min_S. We vary min_s from 60% to 30% and observe its influence on PRESS and two Baselines, Fig. 4(d). There is a sharp increase in run-time of Baseline-1 and Baseline-2 with respect to the decrease in min_s because both approaches generate large no. of groups with lower scores. However, we observe no significant changes on the run-time of PRESS due to the pruning step used in search space enumeration. PRESS is two times faster than the baselines.

6. Conclusions and Future Work

We introduce a novel problem: finding objects with high subspace similarity with respect to a query. We propose a query-driven approach PRESS for identifying *top-k* most similar groups of objects and subspaces in categorical datasets. We mine closures to enumerate the query subspaces and develop dynamic pruning strategies to address the problem efficiently. We also provide a summarisation strategy to provide the user with more interpretable results, where we offer the most divergent set of objects within each of the top-k personalised groups. Our elaborate case studies on real and synthetic datasets showcase PRESS's ability to identify cohesive groups, its scalability, and its interpretability while providing users with personalised feedback. We plan to extend our work to mine objects with subspace similarity by considering user priorities over attributes, relative ordering and different types of attributes.

[6]Please note that it is infeasible to compute the run-time of CLICKS* algorithm when the dataset size = 100 KB.

References

[1] D. Mottin, A. Marascu, S. B. Roy, G. Das, T. Palpanas, Y. Velegrakis, A holistic and principled approach for the empty-answer problem, The VLDB Journal 25 (4) (2016) 597–622.

[2] S. Chaudhuri, G. Das, V. Hristidis, G. Weikum, Probabilistic information retrieval approach for ranking of database query results, ACM Trans. Database Syst. 31 (3) (2006) 1134–1168.

[3] M. E. Houle, X. Ma, V. Oria, J. Sun, Efficient similarity search within user-specified projective subspaces, Inf. Syst. 59 (2016) 2–14.

[4] e. a. Gallego B., Bringing cohort studies to the bedside: framework for a 'green button' to support clinical decision-making, J Comp Eff Res (2015).

[5] B. Micenková, R. T. Ng, X. H. Dang, I. Assent, Explaining outliers by subspace separability, in: ICDM, 2013, pp. 518–527.

[6] L. Duan, G. Tang, J. Pei, J. Bailey, A. Campbell, C. Tang, Mining outlying aspects on numeric data, DMKD 29 (5) (2015) 1116–1151.

[7] X. V. Nguyen, J. Chan, J. Bailey, C. Leckie, K. Ramamohanarao, J. Pei, Scalable outlying-inlying aspects discovery via feature ranking, in: PAKDD, 2015, pp. 422–434.

[8] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: SIGMOD, 1998, pp. 94–105.

[9] K. Kailing, H.-P. Kriegel, P. Kröger, Density-connected subspace clustering for high-dimensional data, in: SDM, 2004, pp. 246–256.

[10] R. Martinez, C. Pasquier, N. Pasquier, Genminer:mining informative association rules from genomic data, in: BIBM, 2007, pp. 15–22.

[11] R. Henriques, S. C. Madeira, Bicpam:pattern-based biclustering for biomedical data analysis, ALMOB 9 (1) (2014).

[12] Shichao Zhang, X. Li, M. Zong, X. Zhu, D. Cheng, Learning k for knn classification, ACM TIST 8 (3) (2017) 43:1–43:19.

[13] Xindong Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. F. M. Ng, B. Liu, P. S. Yu, Z. Zhou, M. S. Steinbach, D. J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.

[14] D. Xin, J. Han, H. Cheng, X. Li, Answering top-k queries with multi-dimensional selections: The ranking cube approach, in: PVLDB, 2006, pp. 463–475.

[15] S. Hwang, K. C. Chang, Probe minimization by schedule optimization: Supporting top-k queries with expensive predicates, TKDE 19 (5) (2007) 646–662.

[16] C. Mishra, N. Koudas, Interactive query refinement, in: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, 2009, pp. 862–873.

[17] S. Chaudhuri, G. Das, V. Hristidis, G. Weikum, Probabilistic ranking of database query results, in: (e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004, pp. 888–899.

[18] Workshops Proceedings of the 26th International Conference on Data Engineering, ICDE, March 1-6, 2010, Long Beach, California, USA, IEEE Computer Society.

[19] 18th International Conference on Scientific and Statistical Database Management, SSDBM, 3-5 July 2006, Vienna, Austria, Proceedings, IEEE Computer Society.

[20] Y. Tao, X. Xiao, J. Pei, Efficient skyline and top-k retrieval in subspaces, TKDE 19 (8) (2007) 1072–1088.

[21] Y. Tao, X. Xiao, J. Pei, SUBSKY: efficient computation of skylines in subspaces, in: Proceedings of the 22nd International Conference on Data Engineering, ICDE, 3-8 April 2006, Atlanta, GA, USA, p. 65.

[22] A. Vlachou, M. Vazirgiannis, Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships, DKE 69 (9) (2010) 943–964.

[23] J. Lee, S.-W. Hwang, Toward efficient multidimensional subspace skyline computation, The VLDB Journal 23 (1) (2014) 129–145.
[24] D. Papadias, Y. Tao, G. Fu, B. Seeger, Progressive skyline computation in database systems, ACM Trans. Database Syst. 30 (1) (2005) 41–82.
[25] M. Zhang, R. Alhajj, Skyline queries with constraints: Integrating skyline and traditional query operators, DKE 69 (1) (2010) 153 – 168.
[26] L. Duan, G. Tang, J. Pei, J. Bailey, G. Dong, A. Campbell, C. Tang, Mining contrast subspaces, in: PAKDD, 2014, pp. 249–260.
[27] H. Kriegel, P. Kröger, A. Zimek, Clustering high-dimensional data:A survey on subspace clustering,pattern-based clustering and correlation clustering, TKDD 3 (1) (2009) 1–58.
[28] K. Sim, V. Gopalkrishnan, A. Zimek, G. Cong, A survey on enhanced subspace clustering, DMKD 26 (2) (2013) 332–397.
[29] G. Liu, K. Sim, J. Li, L. Wong, Efficient mining of distance-based subspace clusters, Statistical Analysis and Data Mining 2 (5-6) (2009) 427–444.
[30] M. J. Zaki, M. Peters, I. Assent, T. Seidl, Clicks: An effective algorithm for mining subspace clusters in categorical datasets, DKE 60 (1) (2007) 51 – 70.
[31] G. Gan, J. Wu, Subspace clustering for high dimensional categorical data, SIGKDD Explor. Newsl. 6 (2) (2004) 87–94.
[32] F. Herrera, C. J. Carmona, P. González, M. J. del Jesús, An overview on subgroup discovery:foundations and applications, KAIS 29 (3) (2011) 495–525.
[33] S. Wrobel, Inductive Logic Programming for Knowledge Discovery in Databases, Springer, 2001, pp. 74–101.
[34] M. J. Zaki, C. J. Hsiao, Efficient algorithms for mining closed itemsets and their lattice structure, TKDE 17 (4) (2005) 462–478.
[35] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: PVLDB, 1994, pp. 487–499.
[36] F. Bonchi, C. Lucchese, On closed constrained frequent pattern mining, in: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM), 1-4 November 2004, Brighton, UK, pp. 35–42.
[37] J. W. Moon, L. Moser, On cliques in graphs, Israel Journal of Mathematics 3 (1966) 23–28.