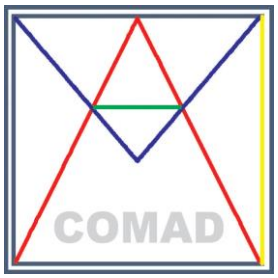


COMAD 2016

Proceedings of the 21st International Conference on Management of Data
Mar 11-13, 2016
Pune, India

Editors

B. Ravindran, IIT Madras

Amol Deshpande, University of Maryland, College Park

Sayan Ranu, IIT Madras

Preface

This volume contains the papers presented at COMAD2016: 21st International Conference on Management of Data held on March 11-13, 2016 in Pune.

For over two decades, the International Conference on Management of Data (COMAD), modelled along the lines of ACM SIGMOD, has been the premier international database conference hosted in India by Division II of Computer Society of India. The first COMAD was held in 1989, and it has been held on a nearly annual basis since then (except for a few breaks such as in the years when VLDB and ICDE were held in India). COMAD has always had a significant international participation, with about 30% of the papers being from outside India, including Europe, USA and East/South-East Asia.

This year COMAD decided to co-locate with the IKDD Conference on Data Science (CoDS). The joint conference proved to be a big success attracting many joint registrations and a very active common day with several keynote talks and tutorials. The invited talks this year were by Minos Garofalakis (University of Crete), Ravi Kumar (Google), Krithi Ramamritham (IIT Bombay) and Deepak Agarwal (LinkedIn), the last two talks being shared with IKDD CoDS. We continued the COMAD tradition of inviting Indian authors of papers presented in premier database conferences. Seven such premier papers were presented this year from venues such as ICDE, SIGMOD, ICDM, and VLDB.

This year there were 27 submissions from around the world. The committee decided to accept 7 full papers and 2 demos. There were 5 *work in progress* posters accepted. We had a fantastic program committee of 23 people from 4 continents, including good representation from the Asia-Pacific region.

We also ran a Data Challenge, Dataview 2016, which was one of the first to focus on open data. The participants were asked to answer questions on public health with the help of open data put out by the Indian government and were also asked to develop an app to facilitate delivery of the information. The winning entries were presented as demos as well.

Pune, India

Amol Deshpande
Balaraman Ravindran
Program Co-Chairs

Sayan Ranu
Proceedings Chair

Table of Contents

Organising Committee

General Chair	P. Sreenivasa Kumar, IIT Madras
Program Chairs	B. Ravindran, IIT Madras
	Amol Deshpande, University of Maryland
CSI DIV II Chair	Dr. R. Nadarajan, PSG College of Technology
Industry Chair	Sudipto Das, Microsoft Research
Tutorial Chair	Rajasekar Krishnamurthy, IBM Research
Data Challenge Chairs	Biplav Shrivastava, IBM Research India
	Debtanu Dutta, Latentview Analytics
	Hemant Mittal, Latentview Analytics
Publication Chair	Sayan Ranu, IIT Madras
Organizing Chair	Arun Kadekodi, Soft Corner, Pune
Web Chair	Anand Joglekar, Ameya Software

Program Committee

Srikanta Bedathur	IBM Research
Arnab Bhattacharya	Indian Institute of Technology, Kanpur
Mahashweta Das	HP Labs, Palo Alto
Amol Deshpande	University of Maryland
Prasad Deshpande	IBM Research - India
Alan Fekete	University of Sydney
Vikram Goyal	IIIT-Delhi
Manish Gupta	Microsoft
Kamalakar Karlapalem	IIT Gandhinagar
Ashwin Kayyoor	Veritas Labs
Udayan Khurana	IBM T. J. Watson Research Center
Qiong Luo	Hong Kong University of Science and Technology
Karin Murthy	IBM Research
Dhaval Patel	IIT Roorkee
Krishna Reddy Polepalli	IIIT-H
Maya Ramanath	IIT Delhi
Balaraman Ravindran	Indian Institute of Technology Madras
Jagan Sankaranarayanan	NEC Labs America
Ralf Schenkel	Universitaet Passau
Kyuseok Shim	Seoul National University
Srinath Srinivasa	International Institute of Information Technology, Bangalore
Divesh Srivastava	AT&T Labs-Research
S. Sudarshan	IIT Bombay

SkyCover: Finding Range-Constrained Approximate Skylines with Bounded Quality Guarantees

Shubhendu Aggarwal
shubhu@cse.iitk.ac.in

Shubhadip Mitra
smitr@cse.iitk.ac.in

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Dept. of Computer Science and Engineering,
Indian Institute of Technology, Kanpur.
India

ABSTRACT

Skyline queries retrieve promising data objects that are not dominated in all the attributes of interest. However, in many cases, a user may not be interested in a skyline set computed over the entire dataset, but rather over a *specified range of values* for each attribute. For example, a user may look for hotels only within a specified budget and/or in a particular area in the city. This leads to *constrained* skylines. Even after constraining the query ranges, the size of the skyline set can be impractically large, thereby necessitating the need for *approximate* or *representative* skylines. Thus, in this paper, we introduce the problem of finding *range-constrained approximate* skylines. We design a grid-based framework, called SkyCover, for computing such skylines. Given an approximation error parameter $\epsilon > 0$, the SkyCover framework guarantees that every skyline is "covered" by at least one representative object that is not worse by more than a factor of $(1 + \epsilon)$ in all the dimensions. This is achieved by employing a non-uniform grid partitioning on the data space. We also propose two *new* metrics based on the covering factor to assess the quality of an approximate skyline set. Experimental evaluation reveals that SkyCover outperforms the competing methods in both quality and running time.

1. INTRODUCTION

Since their introduction to the database community by [3], skyline queries have attracted a significant interest from researchers, and have also found their way into commercial databases such as PostgreSQL [11]. Skylines are especially suitable for situations where there are multiple attributes of interest but no clear optimization function that can help choose a single preferred object. The skyline set returns all promising objects that are *not worse* than another object in *all* the attributes. There is typically no ordering among the skyline objects.

A preference function needs to be specified for every dimension of interest for the skyline query. For example, while looking for good hotel deals online, the attributes of interest may be cost and distance to city center, and the preference functions are *less than* for both. A user *never* chooses a hotel H_1 that costs more *and* is farther from city center than another hotel H_2. The hotel H_1 is, hence, not a skyline.

Formally, assume that there is a dataset D with d skyline attributes having the preference function in each attribute to be less than ($<$).[1] In other words, for every skyline dimension i, a smaller value dominates a larger one. A point t *dominates* another point u, denoted by $t \succ u$, if and only if $\forall i,\ t_i \leq u_i$ and $\exists j,\ t_j < u_j$. A point t is in the *skyline* set $S \subseteq D$ if and only if there does not exist any other point u in the dataset that dominates it, i.e., $t \in S \iff \nexists u \in D$ such that $u \succ t$.

Note that in the definition of skylines, the *exact* values of the objects are not important; only the relative ordering matters. Thus, the ranges of the values can be modified (such as scaling and shifting) as long as the preference relationships are maintained.

When the number of dimensions is large, the skyline query suffers from the cardinality problem in the sense that the size of the skyline set may be too large. Since one of the basic utilities of the skyline query is to deduce the set of useful objects by discarding the useless ones from the dataset that cannot be good, the entire exercise of finding skylines may suffer.

In many cases, a user may not be interested in all the objects over the entire range of the dataset. For example, while searching for hotels, the user may have a certain budget and/or a particular area of the city in mind. Hence, the skyline query needs to be appropriately modified as well. The query should return the skyline set considering only the specified range as the dataset. Informally, we refer to such skyline objects over a specified query range as *range-constrained skyline* objects.

Given a query range Ω, specified by d ranges of the form $[l_i, u_i)$ for $i = 1, \ldots, d$, let $D(\Omega)$ denote the induced set of objects (in D) that fall within Ω. An object t is referred to as a *range-constrained skyline* of $D(\Omega)$ if there does not exist any object $s \in D(\Omega)$ that dominates it. In other words, t is a skyline object of the set $D(\Omega)$. Note that t may not be a skyline object in D, as it may be dominated by some object not in $D(\Omega)$. Therefore, the range-constrained skylines cannot be simply computed from the skyline set of D by applying the range Ω.

Although the range-constrained skylines have been addressed in the literature [10, 17], they still do not address the cardinality issue satisfactorily. For high dimensions, even the range-constrained skylines can be very large in number. The cardinality may be prohibitively large also when the query range Ω is large.

To address such high number of range-constrained skylines, in this work, we propose *approximate range-constrained* skyline computation. As there are several existing works on *approximate* or

[1] A greater than preference function can always be converted by inverting or negating the values.

representative set of skylines [6,23,32], a natural proposition would be to employ one of these techniques to compute approximate range-constrained skylines. Given a query range Ω, a range query is first performed over D to compute the induced dataset $D(\Omega)$. Following this step, skylines $S(\Omega)$ are computed over the set of objects $D(\Omega)$. Finally, approximate skylines are computed from $S(\Omega)$.

However, this strategy has the following limitations:

- **Low performance:** The approximation is performed only *after* computing the skyline set which needs the range query to be solved before that. Thus, this can be fairly time consuming, especially when the range query answer set and the skyline set are large.
- **Low quality:** Most of the existing approximate skyline computation techniques do not cover the skyline set, i.e., for every skyline object in the original dataset, there may not exist a suitable representative object in the reported set. Moreover, they consider optimizing metrics such as k most diverse skylines or k most dominating skylines which are NP-hard problems and, therefore, resort to heuristics that yield sub-optimal solutions.
- **High storage:** The storage of the sets $D(\Omega)$ and $S(\Omega)$ may be fairly large.

Motivated by these challenges, in this paper, we propose a new indexing framework called SKYCOVER that reports a set of approximate range-constrained skylines. The approximation is pushed *within* the skyline operator and not afterwards. Further, the approximation is achieved by using a grid structure that *guarantees* the coverage up to a user-defined error threshold. In other words, the reported representative data objects *cover* the *entire* set of range-constrained skylines with bounded approximation errors.

The SkyCover framework employs a hashing scheme that ensures the bounded quality guarantees, without requiring any separate approximate skyline technique. The proposed index structure stores a sampled set of objects that is sufficient to offer the quality guarantees. As a result, the number of input objects to the skyline finding routine is reduced, thereby making it faster. The set returned by the skyline query on this reduced input set is reported as the SkyCover set, which is an approximate and succinct representation of the range-constrained skyline set. In addition to its efficiency, the proposed SkyCover framework is simple and easily extensible to several practical variations of skyline queries, such as top-k skyline queries [4, 31], nearest-neighbor skyline queries [19], progressive skyline queries [24, 29], skyline queries over dynamic and streaming settings [22, 28], etc.

The basic idea in SkyCover is that if there are two or more objects that are quite similar to each other, the information added beyond the first object is low, and therefore, the subsequent objects can be filtered out. The similarity is based on the proximity of the values in the underlying data space. For all such objects that have values close to each other, the SkyCover index structure retains only one of them.

Ideally, the value of a skyline object that is not reported should be within some threshold of tolerance within the skyline object that is reported. The approximation error is captured by an error parameter ϵ. SkyCover guarantees that every range-constrained skyline object is *covered* by at least one object in the SkyCover set that is not worse by a factor of $(1 + \epsilon)$ in all the dimensions. Such an approximation factor is natural in many applications. For example, when a skyline hotel H_1 has no vacancies, a user may tolerate a hotel H_2 that has larger cost and distance than H_1 within a factor of, say, 5%. This translates to finding H_2 with cost less than or equal to 105 units and distance less than 2.1 units when the H_1 has a cost of 100 units and distance 2 units. The value of $\epsilon = 0.05$ guarantees this.

In sum, our contributions in this paper are as follows:

1. To the best of our knowledge, this is the first work to directly address the problem of approximate range-constrained skyline computation.
2. We propose a novel SkyCover framework for efficiently computing approximate range-constrained skylines with bounded quality guarantees.
3. We propose two new metrics for measuring the quality of an approximate skyline set based on the concept of *covering factor*.
4. Experimental evaluation, on both real and synthetic datasets, reveals that SkyCover outperforms the other competing methods in both quality and running time.

The outline of the rest of the paper is as follows. Section 2 discusses the related work. Section 3 describes the SkyCover framework, the properties of which are analyzed in Section 4. In Section 5, we propose two new quality metrics. The results are described in Section 6. Section 7 concludes.

2. RELATED WORK

We first discuss the works in the area of skyline computation and then describe the related literature for range-constrained skylines and approximate skylines.

2.1 Skylines

The skyline problem has been first studied as that of determining the maxima of a set of vectors. The algorithms in this field are typically based on the divide-and-conquer [20] and parallel approaches [25]. These methods assume that the entire data fits in the main memory.

The notion of skyline for relational database systems was introduced by [3] who proposed the "skyline" operator. Since then, a large number of algorithms, both index-based and non-index-based, have been proposed for skyline computation. Index-based methods include B-tree based schemes [3], bitmap and index [29], nearest-neighbor (NN) [19] and branch-and-bound (BBS) [24]. More recently, a trie-based index structure has been proposed in [27] to improve scalability and maintenance costs in case of data updates. In non-index-based methods, block-nested-loop (BNL) and divide-and-conquer (DC) were proposed in [3]. Chomicki et al. [7] proposed a variant of BNL algorithm called the sort-filter-skyline (SFS) algorithm that involves pre-processing of sorting the data with respect to a monotone scoring function.

For high dimensional data, there has been a lot of focus on reducing the number of skylines reported. Since it becomes increasingly difficult for a point to dominate another in high dimensions, k-dominance [6] was introduced to relax the dominance criteria where it is sufficient to be better in k dimensions in order to dominate another point.

The strategy of grid-based partitioning has been used in the distributed environment [1, 8, 21, 26, 33]. Essentially, in these works, a grid is created on the data such that each grid cell has almost the same amount of data. The same grid structure is imposed by all the distributed nodes on the data space. Each node computes the skylines in a specific grid (or a set of grids), and finally the local results are merged to compute the desired skyline set.

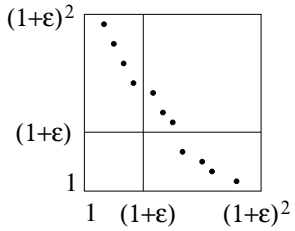


Figure 1: Example of ϵ-SkyCover.

2.2 Range-Constrained Skylines

The problem of constrained subspace skyline computation was introduced in [10]. In order to support constrained skylines for arbitrary subspaces, they presented approaches exploiting multiple low-dimensional indexes instead of a single high-dimensional index. Range-constrained skylines in two dimensions was investigated in [17]. None of these works address the issue of high cardinality of the skyline set.

2.3 Approximate Skylines

There are several works on approximate or representative skylines. One one hand, [9, 13, 23] focused on finding k skyline points that together dominate the maximum number of non-skyline points. On the other hand, [14, 30] considered returning k skyline points that best represents the skyline contour. Other works such as [32] focused on reporting the k most diverse skyline points, where the diversity is measured using Jaccard distance.

Approximately dominating representatives (ADRs) [18] reports ϵ-approximate skyline for some ϵ given by the user. An ϵ-ADR query returns a set of points which together when boosted by ϵ in all dimensions dominate all other points (they considered greater than $>$ as the preference function for all dimensions). This is similar to the notion we implement as well. However, their work is significantly different from ours as they assume that the skyline set has been pre-computed and the task is to post-process and determine the smallest possible ADR. We, on the other hand, push the approximation scheme before the skyline operation. They also show that for a given ϵ, the problem of minimizing the cardinality of ϵ-ADR is NP-hard for three or more dimensions.

Approximate skylines over a data stream was computed in [28] which gave a notion of additive errors in skylines by defining strong domination. A point t strongly dominates another point u if for all dimensions i, $t_i + \epsilon \leq u_i$ where ϵ is an acceptable difference distance. Their algorithm guarantees that any two approximate skyline points are at least 2ϵ apart and for every skyline point there is an approximate skyline point within ϵ distance of it. An ϵ-skyline was proposed in [34] where the domination criteria called ϵ-domination was modified to allow additive error of ϵ along each weighted dimension. Thick skylines were proposed in [16] where all skyline points and their ϵ-neighbors (points within ϵ distance) are reported. The algorithm involves computing distances between points which is expensive.

2.4 Comparison with SkyCover

Next we closely compare our SkyCover approach with the k-RSP approach [23] and SkyDiver approach [32], as they are the most popular notions of representative skylines. For experimental evaluation, we compare the quality of the solutions realized by SkyCover against these two approaches.

While all the three approaches attempt to suitably represent the skyline set, the computational challenge in optimally (and also approximately) computing these solutions vary significantly. While the sets reported by k-RSP and SkyDiver approaches are of fixed cardinality of size k, the size of the set returned by our SkyCover approach is not fixed. Due to this reason, while the optimal computation of the representative sets in case of k-RSP and SkyDiver is NP-hard, the computation of the representative set by our SkyCover approach is polynomially solvable. While our approach guarantees that the reported representative points have bounded approximation errors in all dimensions w.r.t. the skylines missed (not reported), there is no such guarantee in the other two approaches. For instance, consider the example shown in Figure 1. Observe that each point in the dataset is a skyline. Suppose a user wants $k = 3$ representative points. Since the dominating sets, i.e., the set of points dominated by the given point, for each of the points is empty, both the k-RSP and the SkyDiver approaches report any 3 points arbitrarily. Therefore, the worst case approximation factor of the representative sets returned by these two approaches can be arbitrarily bad. On the other hand, as it will be described later, our ϵ-SkyCover would select one point from each of the three grid cells containing the points, thereby guaranteeing that every missed skyline has a representative point that is worse by a multiplicative factor of at most $(1 + \epsilon)$.

The k-RSP and SkyDiver problems being NP-hard for dimensions greater than two, it was shown that when greedy heuristic is applied to them, they can be approximated within a factor of $1 - \frac{1}{e}$ and 2 respectively. For scalability and efficiency reasons, each of these heuristics consider randomized techniques that offer theoretical accuracy guarantees. While k-RSP approach employ probabilistic counting using FM algorithm [12], the SkyDiver employs min-hash (MH) [5] and LSH [15]. In contrast, we propose an efficient deterministic algorithm based on the concept of hashing using non-uniform grid-based partitioning, whose complexity depends on the desired value of the approximation parameter.

Most importantly, the k-RSP and the SkyDiver approaches can be applied only *after* the skyline computation which is quite expensive. Our SkyCover approach is applied *prior* to skyline computation, which reduces the size of the dataset considerably, thereby leading to savings in query-time.

In addition, our SkyCover framework is more generic as any of the post-processing approximation techniques including k-RSP and SkyDiver techniques can be applied after the approximation and skyline computations have been done. Hence, our work can be treated both as alternative as well as complementary to the existing works in the space of approximate skyline representation.

3. THE SKYCOVER FRAMEWORK

In this section, we first introduce the problem of approximate range-constrained skyline computation. Then, we describe the proposed SkyCover framework for finding such skylines. Finally, we analyze its running time, and discuss possible extensions of the framework.

3.1 Range-Constrained Skylines

Assume that the dataset D is d-dimensional with the range in each dimension between 1 and R. The total data space, therefore, is $[1, R]^d$. Suppose S is the skyline set of D. All the symbols used in this paper are listed in Table 1 for easy reference.

We first introduce the notion of range-constrained skylines. Given a range $\Omega = \prod_i [l_i, u_i)$, where $(1 \leq l_i \leq u_i \leq R)$, $D(\Omega)$ denotes the induced dataset over Ω, i.e., it contains the points in D that lie within the range Ω. Hence, $p \in D(\Omega)$ if and only if $l_i \leq p_i \leq u_i \forall i$. The skylines $S(\Omega)$ computed over $D(\Omega)$ is the range-constrained skyline set.

Table 1: List of Symbols.

Symbol	Description
D	Dataset
$N = \|D\|$	Cardinality of dataset
d	Dimensionality
$[1, R]$	Range of values along a dimension
S	Skyline set
$s = \|S\|$	Cardinality of skyline set
$\Omega = \prod_i [l_i, u_i)$	Query range
$D(\Omega)$	Induced dataset over the range Ω
$n = \|D(\Omega)\|$	Cardinality of induced dataset
$S(\Omega)$	Range-constrained skyline set of $D(\Omega)$
ϵ	Error parameter
$\Omega_\epsilon = [l'_i, u'_i)$	Stretched range of Ω
$l'_i = l_i/(1+\epsilon)$	Lower range of Ω_ϵ
$u'_i = u_i.(1+\epsilon)$	Upper range of Ω_ϵ
$S'(\Omega_\epsilon)$	SkyCover set of $D(\Omega)$
g	Number of grid partitions along a dimension
$P(\Omega)$	Set of grid representatives that lie in Ω_ϵ
$m = \|P(\Omega)\|$	Total number of grid representatives stored
S'	SkyCover set of D
$k = \|S'(\Omega_\epsilon)\|$	Cardinality of SkyCover set

DEFINITION 1 (RANGE-CONSTRAINED SKYLINE). *Given a range Ω, a point $s \in D(\Omega)$ is in the* range-constrained skyline *set $S(\Omega)$ of $D(\Omega)$ if and only if it is not dominated by any other point in $D(\Omega)$.*

It is important to note that $S(\Omega)$ is not necessarily a subset of the skyline set S, as it may contain a point that is dominated by another in D but not in $D(\Omega)$. A range-constrained skyline query is, thus, a generalization of the regular skyline query.

Although by constraining the data range, the size of the dataset is reduced, there is no guarantee that the size of the skyline set will be reduced. This may happen since there may be a globally dominating skyline outside the range, and not considering it allows many other objects as skylines.

Even otherwise, the number of range-constrained skylines may be too many, especially for high-dimensional spaces. Hence, a flavor of representation or approximation is needed to reduce the number. For that, we introduce the concept of skyline cover next.

3.2 Skyline Cover

Given an $\epsilon > 0$, we first define ϵ-cover.

DEFINITION 2 (ϵ-COVER). *Given $\epsilon > 0$, a point $t \in D$ ϵ-covers a point $s \in D$, if and only if for every dimension i, t is not worse than s by a multiplicative factor of $(1+\epsilon)$*

$$t \text{ } \epsilon\text{-covers } s \iff \forall i,\ t_i \leq (1+\epsilon)s_i \tag{1}$$

The simple dominance is a special case of ϵ-cover when $\epsilon = 0$.

Using this, we next define the concept of ϵ-SkyCover. Informally, a set $S'(\Omega_\epsilon)$ is a *skyline cover* of $S(\Omega)$ if each point in $S(\Omega)$ has a suitable representative in $S'(\Omega_\epsilon)$. The skyline cover $S'(\Omega_\epsilon)$ is an ϵ-SkyCover if every point $s \in S(\Omega)$ is represented within the multiplicative factor of $(1+\epsilon)$, i.e., it is ϵ-covered by at least one point $t \in S'(\Omega_\epsilon)$.

DEFINITION 3 (ϵ-SKYCOVER). *A set $S'(\Omega_\epsilon)$ is an ϵ-*SkyCover *of $S(\Omega)$ if every point in $S(\Omega)$ is ϵ-covered by at least one point in $S'(\Omega_\epsilon)$:*

$$S'(\Omega_\epsilon) \text{ is an } \epsilon\text{-SkyCover of } S(\Omega) \iff \forall s \in S(\Omega), \exists t \in S'(\Omega_\epsilon) \text{ s.t. } t \text{ } \epsilon\text{-covers } s \tag{2}$$

Assume that Ω_ϵ denotes the range of Ω that is stretched by a factor of $(1+\epsilon)$ along each dimension, i.e., $\Omega_\epsilon = \prod_i [l'_i, u'_i) \forall i$ where $l'_i = l_i/(1+\epsilon)$ and $u'_i = u_i.(1+\epsilon)$. The skyline set $S'(\Omega_\epsilon)$ of $D(\Omega_\epsilon)$ is *necessarily* an ϵ-SkyCover of the skyline set $S(\Omega)$ of $D(\Omega)$.

When the context is clear, we refer to an ϵ-SkyCover by simply SkyCover.

The parameter ϵ essentially captures the user's tolerance to approximation. For example, consider the cost attribute of a particular dataset. If the cost of a skyline point is 100 units and $\epsilon = 0.05$, it means that the user can tolerate a point that costs up to 105 units.

Note that ϵ is a multiplicative ratio which makes more sense than an absolute margin (i.e., additive error) such as 5 units in this context. When the cost of an object is more, the difference in cost is more as well. When it is less, the difference automatically shrinks. Hence, in the above example, if a skyline object costs 10 units, it is more likely that a user will tolerate up to 10.5 units and not 15 units. Similarly, if the cost is 1000 units, the user can go up to 1050 units and not 1005 units.

3.3 Problem Statement

In this paper, we address the following query:

PROBLEM 1. *Given a range $\Omega = \prod_i [l_i, u_i)$, where $(1 \leq l_i \leq u_i \leq R)$, report its SkyCover, i.e., $S'(\Omega_\epsilon)$.*

Note that the range Ω is available only at query time. Therefore, since $S'(\Omega_\epsilon)$ need not be a subset of S or its SkyCover S', even if we pre-compute the sets S or S', it is not easy or efficient to compute the set $S'(\Omega_\epsilon)$ using these pre-computed sets.

The goal of the proposed SkyCover framework is to, thus, efficiently compute the SkyCover $S'(\Omega_\epsilon)$. To achieve this, we first build an index structure using sampled data points in D. On receiving the query range Ω, we identify a set of points stored in the index structure that lie in the stretched range of Ω, i.e., Ω_ϵ. Finally, we compute the skylines over this set of points to return the desired SkyCover. We next discuss these steps in more detail.

3.4 Grid Partitioning

We construct a grid-based index structure by employing a non-uniform grid partitioning scheme. The grid boundaries are imposed at multiplicative intervals of $(1+\epsilon)$. Thus, the first grid cell is from $(1+\epsilon)^0 = 1$ to $(1+\epsilon)^1$, the second one from $(1+\epsilon)^1$ to $(1+\epsilon)^2$, and so on. The number of grid partitions, g, along any dimension is, therefore, equal to

$$g = \lceil \log_{1+\epsilon} R \rceil \tag{3}$$

Hence, the total number of grid cells for d-dimensional space is g^d.

Assume that $R = (1+\epsilon)^g$ such that there are exactly g grid partitions along any dimension. (The data space can always be stretched so that this happens.) Otherwise, using the ceiling function for g imposes an error parameter $\epsilon' < \epsilon$, thus, protecting the $(1+\epsilon)$ SkyCover guarantees.

If the range of a dimension is $[min, max]$, it can always be shifted and stretched to fit $[1, R]$. If $min = 1$, then the ratios after the transformation do not change. Otherwise, the error value ϵ in the $[min, max]$ range can be mapped to an ϵ' in the $[1, R]$ range if $min > 0$. If both min and max are negative, the ratios are inverted. If, however, $min < 0$ and $max > 0$, then the

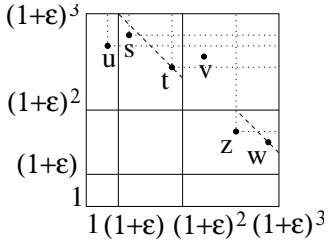


Figure 2: Grid-based partitioning.

concept of multiplicative error makes little sense and we do not consider such cases. As explained in Section 3.10, an additive error is more meaningful there and the SkyCover framework can be appropriately modified to handle that using uniform grid partitions.

A grid cell $x = (x_0, \ldots, x_{d-1})$ is indexed by the corresponding d grid cell numbers along the dimensions. Here, $0 \leq x_i \leq g-1$ denotes the grid cell index for the i^{th} dimension.

A point $t = (t_0, \ldots, t_{d-1})$ falls in grid cell $x = (x_0, \ldots, x_{d-1})$ if and only if it satisfies the condition $\forall i,\ (1+\epsilon)^{x_i} \leq t_i < (1+\epsilon)^{x_i+1}$. The hash function $h_i(t_i)$ that returns the grid cell index for the attribute i of a point t is, thus, $h_i(t_i) = \lfloor \log_{(1+\epsilon)} t_i \rfloor$. The hash function $h(t)$ for a point t is the combination of the corresponding hash functions for each dimension:

$$h(t) = \langle h_0(t_0), \ldots, h_{d-1}(t_{d-1}) \rangle \quad (4)$$

Figure 2 shows an example partitioning of the space with $g = 3$ and $d = 2$.

Note that the grids are imposed on the original $[1, R)^d$ space before the query range is made available at run time.

3.5 Choosing Grid Cell Representative

All the points that hash to the same grid cell are represented by only one of them. In this way, the SkyCover stores a much smaller sample of the dataset. Since any point within the cell ϵ-covers any other point within the same cell, keeping anyone of them would serve the purpose of ϵ-SkyCover (shown in Theorem 1). Thus, this not only ensures quality guarantees, but also significantly contributes towards lower storage and, therefore, higher efficiency.

We next discuss how to choose the cell representative. For each point t, we employ an entropy function similar to that used in the SFS algorithm [7]: $ent(t) = \sum_{i=0}^{d-1} t_i$. Since the ultimate goal is to compute range-constrained skylines, the entropy function is re-used later during the actual skyline computation phase.

The grid representative of a cell x is chosen as the point t with the lowest entropy: $t = \arg\min\{ent(t) | t \in x\}$. Note that no other point in the same cell can dominate t as then its entropy would have necessarily lower. Moreover, the scheme guarantees that t ϵ-covers any other point in the same cell.

3.6 Query Processing

When the query parameters Ω and ϵ arrive, the first step is the retrieval of the set of grid representatives $P(\Omega)$ that map to the *stretched* range of Ω, i.e., Ω_ϵ. Assume the query range is $\Omega = \prod_i [l_i, u_i)$. Along dimension i, we identify the grid cell indices that map to the above space as follows. Assume $x_i = \lfloor \log_{1+\epsilon} l_i \rfloor$ and $y_i = \lfloor \log_{1+\epsilon} u_i \rfloor$. The set $P(\Omega)$ denotes the set of all the grid representatives of the grids that lie in the space $\prod_{i=0}^{d-1} [x_i, y_i]$:

$$P(\Omega) = \{\text{grid representative of cell } z |\ \forall i,\ x_i \leq z_i \leq y_i\} \quad (5)$$

Next, a skyline finding algorithm is employed over the set of points in $P(\Omega)$. The answer thus obtained is reported as the SkyCover $S'(\Omega_\epsilon)$.

Algorithm 1 SkyCover Algorithm

```
1: procedure OFFLINE COMPUTATION (Dataset D, Error parameter ε)
2:     H ← Φ
3:     for all t ∈ D do
4:         h(t) ← grid index of t using ε in Eq. (4)
5:         if h(t) is empty then
6:             Insert h(t), ⟨t, ent(t)⟩ in H
7:         else
8:             ⟨u, ent(u)⟩ ← value(h(t))
9:             if ent(t) < ent(u) then
10:                value(h(t)) ← ⟨t, ent(t)⟩
11:            end if
12:        end if
13:    end for
14:    return H
15: end procedure
1: procedure ONLINE COMPUTATION (Query range Ω)
2:     Compute P(Ω) from H as described in Section 3.6
3:     return S'(Ω_ε) ← FINDSKYLINES(P(Ω))
4: end procedure
```

For our implementation, we choose the SFS algorithm [7] as it is an online algorithm with no requirement of index construction and, is, thus, applicable in all situations. Moreover, it is quite simple, easy to implement, and fairly efficient in terms of running time.

3.7 The SkyCover Algorithm

Using the above ideas, we now summarize the steps in the SkyCover framework (pseudocode shown in Algorithm 1).

The framework comprises of two phases, the offline phase and the online phase. During the offline phase, we build the SkyCover index structure. A dynamic hash table H is maintained with the grid cell index as the key and the 2-tuple consisting of the representative point and its entropy, as the value. The data points are processed one at a time. For a point t, its grid cell $h(t)$ is computed using Eq. (4). If there is no such key in H, a key-value pair with key $h(t)$ and value $\langle t, ent(t) \rangle$ is inserted. If, however, there is already a key, then the value corresponding to it is extracted. If the entropy of the new point is less than that of the old one, the old value is replaced with the new point (and its entropy). Otherwise, since there is already a representative of this cell with a lower entropy, the new point is discarded.

The online phase is discussed in Section 3.6.

3.8 Analysis of Running Time

In this section, we compare the running time of SkyCover algorithm with that of the naïve approach, discussed in Section 1, that requires range search, followed by skyline computation, and finally approximate skyline computation.

Assume that the cardinality of the dataset D is N, the dimensionality is d, the size of the skyline set S is s, the number of representative points left using the grid-based partitioning is m, and the size of the SkyCover set extracted from m is k.

We assume that if the cardinality of the induced dataset $D(\Omega)$ is n, the range search required to find all the points in the range Ω is at least $O(n)$.

We first analyze the naïve approach. For skyline computation, we consider the SFS algorithm. The SFS algorithm has the following costs: (i) Computing entropy: $O(nd)$ for mapping d dimensions to a single value for all n points, (ii) Sorting the points based

on entropy: $O(n \log n)$, (iii) Skyline computation: $O(ns)$ assuming a window size of $O(s)$. Hence the total time complexity of the SFS algorithm is $O(n.d + n \log n + n.s)$.

Finally, using any existing approximate skyline techniques [6, 23, 32], reporting k representative skylines would require at least $O(sk)$ time. Hence the total time complexity of the naïve approach is at least$O(nd + n \log n + n.s + s.k)$.

The running time of the SkyCover framework comprises of the following components: (i) Computing entropy and hashing during the offline phase: $O(nd)$ assuming $O(1)$ hashing costs, (ii) Retrieving all the hash values during online phase: $O(m)$, (iii) Sorting based on entropy (during online phase): $O(m \log m)$, (iv) Skyline computation during online phase: $O(mk)$. Thus, the offline cost is $O(n.d)$, and the online cost is $O(m \log m + m.k)$.

It was shown in [2], that if the attribute values are chosen independently, then the average number of skylines is $O((\ln n)^{d-1})$. Thus, assuming $s = O((\log n)^{d-1})$, $k = O((\log m)^{d-1})$ and $d = O(1)$, and considering only the dominant terms, the query time of the naïve approach simplifies to $O(n.(\log n)^{d-1} + (\log m)^{d-1}.(\log n)^{d-1})$, and that of the SkyCover becomes $O(m.(\log m)^{d-1})$. As $m < n$, the running cost of SkyCover framework is always better than that of the naïve approach, at least by an additive factor of $O((\log m \log n)^{d-1})$. In Section 4.3, we present an analysis of expected value of m.

3.9 SkyCover using Uniform Grids

The SkyCover framework can be easily modified to work with uniform grid partitions. In this scheme, the total range $R - 1$ in each dimension is split into g equal parts. The grid boundaries for any dimension, therefore, fall at $1, 1 + (R-1)/g, 1 + 2(R-1)/g$, etc. The total number of cells is again g^d, i.e., the same as non-uniform partitioning. The SkyCover framework remains the same for the two cases.

3.10 Extensions of SkyCover

The grid-based hashing mechanism employed in the SkyCover framework is generic enough to capture different kinds of errors. For example, the error parameter can be different for each dimension. The number of partitions, g_i, for dimension i, is computed using the corresponding error parameter ϵ_i.

More importantly, SkyCover is not restricted to only multiplicative errors. For example, guarantees for additive errors can be easily offered using uniform grid partitions. If the additive error tolerance is δ, the width of the grid cells should be at most δ.

Moreover, the SkyCover framework allows seamless integration of skyline queries over attributes demanding different types of error tolerance. While some attributes may tolerate multiplicative errors, some others may require additive errors, and the rest may not tolerate any error. The dimensions for multiplicative error attributes may be partitioned in a non-uniform manner, while those for additive errors may be partitioned using equal grid widths. The dimensions that do not tolerate any error will not be hashed at all using the grid partitions.

We have, however, stuck to the multiplicative SkyCover framework in this paper and have postponed the detailed experimentation and analysis of the generalized framework to a later paper.

We also claim that the SkyCover framework is applicable to streaming data settings [28] with insert-only operations. The multi-dimensional data points arriving in the stream can be efficiently hashed into the SkyCover index structure, and subsequently queried upon. Similarly, the framework is particularly suited for update-heavy workloads where most of the small updates in the values of a point can be absorbed since it lies within the same cell. The update in the skyline set needs to be checked only when a point moves to a cell that was not occupied earlier. In addition, we claim that the framework can be easily adapted for other settings such as distributed environments [26], and moreover, help in computing approximate top-k skylines [4, 31], progressive skylines [24, 29], etc. However, in this paper, we do not evaluate the performance of SkyCover over such settings.

4. PROPERTIES OF SKYCOVER

In this section, we first discuss the correctness of the SkyCover framework, followed by quality guarantees on falsely reported range-constrained skylines. Finally, we present a thorough analysis of the expected number of grid representatives, m.

4.1 Correctness

The following theorem establishes the correctness.

THEOREM 1. *Given any range Ω, the SkyCover framework correctly computes its ϵ-SkyCover, i.e., (1) $S'(\Omega_\epsilon) \subseteq D(\Omega_\epsilon)$, and (2) for every range-constrained skyline point $s \in S(\Omega)$, there exists at least one $t \in S'(\Omega_\epsilon)$ that ϵ-covers s.*

PROOF. Assume the query range to be $\Omega = \prod_i [l_i, u_i)$.

(1) Since the SkyCover set $S'(\Omega_\epsilon)$ is a subset of the set of representative points, $P(\Omega)$, i.e., $S'(\Omega_\epsilon) \subseteq P(\Omega)$, it is sufficient to show that $P(\Omega) \subseteq D(\Omega_\epsilon)$. Consider any point $t \in P(\Omega)$ lying in the grid cell $z = \langle z_0, \ldots, z_{d-1} \rangle$. Assuming non-uniform grid partitioning, the index structure ensures that $\forall i,\ x_i \leq z_i \leq y_i$, where $x_i = \lfloor \log_{(1+\epsilon)} l_i \rfloor$ and $y_i = \lfloor \log_{(1+\epsilon)} u_i \rfloor$. This implies that $\forall i,\ (1+\epsilon)^{x_i} \leq t_i \leq (1+\epsilon)^{y_i+1}$. Plugging in the values of x_i and y_i, and using the fact that for any real number a, $a - 1 \leq \lfloor a \rfloor < a$, we get the following: $\forall i,\ l_i/(1+\epsilon) \leq t_i \leq u_i.(1+\epsilon)$. Hence, $t \in D(\Omega_\epsilon)$. Therefore, $P(\Omega) \subseteq D(\Omega_\epsilon)$.

(2) Consider a range-constrained skyline point $s \in S(\Omega)$. Suppose it lies in a grid cell z. Assume the point t to be the cell representative of z where t may or may not be equal to s. Since s and t are in the same cell, the construction of the non-uniform grid partitioning ensures that t ϵ-covers s. More formally, $\forall i, t_i \leq (1+\epsilon)s_i$. If $t \in S'(\Omega_\epsilon)$, then s is covered by t. If, however, $t \notin S'(\Omega_\epsilon)$, then there must exist a point $u \in S'(\Omega_\epsilon)$ that dominates t, i.e., $u \succ t$. Combining with the above inequality, we get $\forall i, u_i \leq t_i \leq (1+\epsilon)s_i$, i.e., u ϵ-covers s. □

The ramification of this theorem is that even if $s \in S(\Omega)$ was missed, there is another point $t \in S'(\Omega_\epsilon)$ that approximates it in the sense that it is not too bad in any of the dimensions. To be precise, the values of t are within a multiplicative factor of $(1+\epsilon)$ from those of s in *every* dimension.

Figure 2 shows an example of how Theorem 1 works. Assume that $\Omega = [1, R)^2$ where $R = (1+\epsilon)^3$. In this case, any range-constrained skyline in $S(\Omega)$ is a skyline of D. The skyline point u, which is reported, ϵ-covers itself. The skyline t is missed and is represented by s which has a lower entropy (the equi-entropy line is shown as dashed). The reported point $u \in S'$ dominates s, and consequently, ϵ-covers t. Similarly, even though w is not reported, it is ϵ-covered by its cell representative $z \in S'$.

4.2 Falsely Reported Skylines

Even though the set $S'(\Omega_\epsilon)$ returned by the SkyCover framework correctly ϵ-covers the range-constrained skyline set $S(\Omega)$, it may happen that *not* every point returned is a range-constrained skyline itself, i.e., there may exist a point $u \in S'(\Omega_\epsilon)$ such that $u \notin S(\Omega)$. Figure 2 shows such a situation. The point v is not an actual skyline as it is dominated by t. It is also a grid representative, i.e., $v \in$

$P(\Omega)$. However, since $t \notin P(\Omega)$, and there is no other point that dominates v, $v \in S'(\Omega_\epsilon)$.

Unfortunately, the guarantees for the values of such falsely reported range-constrained skylines are not very strict. Along some of the dimensions, the values can be arbitrarily bad as compared to an actual range-constrained skyline point. However, the next theorem shows that it can still be guaranteed that the values of such falsely reported range-constrained skylines *cannot* be bad in *all* the dimensions.

THEOREM 2. *For any $u \in S'(\Omega_\epsilon)$, there does not exist any $s \in S(\Omega)$ such that $\forall i,\ u_i > (1+\epsilon)s_i$.*

PROOF. We prove by contradiction. Suppose such a point $s \in S(\Omega)$ exists, such that $\forall i,\ u_i > (1+\epsilon)s_i$. From Theorem 1, there must exist $t \in S'(\Omega_\epsilon)$ that ϵ-covers s, i.e., $\forall i,\ t_i \leq (1+\epsilon)s_i$. Together, this implies that, $\forall i, t_i < u_i$, i.e., $t \succ u$. Since $t \in S'(\Omega_\epsilon)$, therefore, $u \notin S'(\Omega_\epsilon)$, which is a contradiction. □

In Figure 2, even though the value of v is very large in the x-dimension as compared to the skyline point u, it is not worse than a ratio of $(1+\epsilon)$ in the y-dimension as well.

For real-life applications such as online hotel deals, assuming a value of $\epsilon = 0.05$, this implies that if there is an actual skyline hotel with cost 100 units and distance 2 units, no hotel is reported in the SkyCover set that has both cost more than 105 units and distance more than 2.1 units.

4.3 Expected Number of Representative Points

In this section, we analyze the expected number of representative grid points for the SkyCover framework. For the sake of comparison, we do the analysis for both uniform and non-uniform grids.

We assume that the points are generated independently and the values are independently and uniformly distributed along the dimensions across the data space.

There are two types of grid cells, empty and non-empty. Since no data point hashes to an empty grid cell, there is no representative for such cells. For non-empty cells, however, even if there are more points, exactly one point is chosen as the representative. Thus, we essentially need to calculate the number of non-empty grid cells to get an estimate of m.

4.3.1 Uniform Grids

For uniform grids, there are a total of g^d grid cells having the *same* volume. Hence, the probability that a point lies in a particular grid cell is

$$P_{point} = g^{-d}. \tag{6}$$

The probability that the cell remains empty is equivalent to the probability that none of the n points lie in it. Since the points are generated independently, this is equal to

$$P_{empty} = (1 - P_{point})^n = (1 - g^{-d})^n. \tag{7}$$

Thus, the expected number of empty cells is

$$\mathbf{E}[\text{empty}] = \sum_{\forall \text{cells}} P_{empty} = g^d(1 - g^{-d})^n. \tag{8}$$

The estimate for the total number of non-empty grid cells for uniform grid partitions is, therefore,

$$m_u = g^d - g^d(1 - g^{-d})^n. \tag{9}$$

Expressing Eq. (9) using binomial expansion, we get

$$m_u = g^d - g^d\left(1 - n.g^{-d} + \frac{n(n-1)}{2}.g^{-2d} - \dots\right)$$

$$\therefore\ \frac{m_u}{n} = 1 - \frac{n-1}{2}g^{-d} + \dots \tag{10}$$

When $n < g^d$, the later terms can be ignored, and therefore, with increasing n, the ratio m_u/n decreases. On the other hand, when $n \geq g^d$, since m is constrained to be at most g^d, the ratio will decrease when n increases. Thus, with increasing number of points and fixed number of grid cells, the proportion of representative points decreases.

When n is fixed, and g^d is increased, the proportion m_u/n increases as intuitively there are more options for a point to lie in, and consequently, more representative points are preserved. When $g^d \geq n$, the ratio will saturate to 1.

4.3.2 Non-Uniform Grids

The analysis for non-uniform grid partitions is not so straightforward as the grid cells have *different* volumes. Hence, the probability of a cell being empty depends on its *location*.

The volume of a grid cell $x = (x_0, \dots, x_{d-1})$ where $0 \leq x_i \leq g-1$ is the grid index along dimension i is

$$v(x) = \prod_{i=0}^{d-1}\left[(1+\epsilon)^{x_i+1} - (1+\epsilon)^{x_i}\right] = \epsilon^d(1+\epsilon)^{\sum_{i=0}^{d-1} x_i}$$
$$= \epsilon^d(1+\epsilon)^{\sigma_x} \tag{11}$$

where σ_x denotes the sum $\sum_{i=0}^{d-1} x_i$ for a cell x.

Since the total volume of the data space is $(R-1)^d$, the probability that a point lies in the grid cell x is $P_{point} = v(x)/(R-1)^d$. The probability that it remains empty when n points are generated independently is, therefore,

$$P_{empty} = (1 - P_{point})^n = \left(1 - \frac{v(x)}{(R-1)^d}\right)^n$$
$$= \left(1 - \left(\frac{\epsilon}{R-1}\right)^d(1+\epsilon)^{\sigma_x}\right)^n. \tag{12}$$

Thus, the expected number of empty cells is

$$\mathbf{E}[\text{empty}] = \sum_{\forall \text{cells}} P_{empty} = \sum_{\forall x}\left(1 - \tau^d(1+\epsilon)^{\sigma_x}\right)^n \tag{13}$$
$$= \sum_{\forall x}\left[\sum_{j=0}^{n}(-1)^j\binom{n}{j}\left(\tau^d(1+\epsilon)^{\sigma_x}\right)^j\right]$$
$$= \sum_{j=0}^{n}\left[(-1)^j\binom{n}{j}\tau^{jd}\sum_{\forall x}(1+\epsilon)^{j\sigma_x}\right]. \tag{14}$$

where τ denotes the ratio $\epsilon/(R-1)$.

To simplify the above equation, we denote $(1+\epsilon)^j$ as α. For the first term, i.e., when $j=0$, the sum $\sum_{\forall x}(1+\epsilon)^{j\sigma_x}$ is simply the total number of cells, which is g^d. For other terms, the summation can be computed by unrolling one dimension at a time. Thus,

$$\sum_{\forall x}\alpha^{\sigma_x} = \sum_{\forall x}\alpha^{\sum_{i=0}^{d-1} x_i}$$
$$= \sum_{\forall i=0,\dots,d-1;\forall x_i=0,\dots,g-1}\left(\alpha^{\sum_{i=0}^{d-1} x_i}\right)$$

Table 2: Values of β and m for different combinations.

g	d	g^d	ϵ	n	β	m_{est}
50	3	125000	0.01	10000	0.16	9588
20	4	160000	0.05	10000	0.34	9594
10	5	100000	0.05	10000	0.28	9471
7	6	117649	0.05	10000	0.19	9564

$$= \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left[\sum_{x_0=0,\ldots,g-1} \left(\alpha^{\sum_{i=1}^{d-1} x_i} . \alpha^{x_0} \right) \right]$$

$$= \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left[\left(\alpha^{\sum_{i=1}^{d-1} x_i} \right) . \sum_{x_0=0,\ldots,g-1} \alpha^{x_0} \right]$$

$$= \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left[\left(\alpha^{\sum_{i=1}^{d-1} x_i} \right) . \left(\frac{\alpha^g - 1}{\alpha - 1} \right) \right]$$

$$= \left(\frac{\alpha^g - 1}{\alpha - 1} \right) \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left(\alpha^{\sum_{i=1}^{d-1} x_i} \right)$$

$$= \cdots = \left(\frac{\alpha^g - 1}{\alpha - 1} \right)^d . \quad (15)$$

Using Eq. (15) in Eq. (14), and since $(1+\epsilon)^g = R$, we get

$$\mathbf{E}[\text{empty}] = g^d + \sum_{j=1}^{n} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{(1+\epsilon)^{jg} - 1}{(1+\epsilon)^j - 1} \right)^d$$

$$= g^d + \sum_{j=1}^{n} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d . \quad (16)$$

The computation of the exact value of the above expression is infeasible due to large n. However, if the expression $\beta_x = n(\tau^d(1+\epsilon)^{\sigma_x}) \ll 1$ for all cells x (from Eq. (12)), then the binomial series converges rapidly and can be bounded with very low error using only the first few terms. For example, the second term, (i.e., for $j = 1$) is simply n. Thus, a very crude estimate for the number of empty cells is $\mathbf{E}[\text{empty}] = g^d - n$.

Assume that $\beta = \max_{\forall x} \beta_x$ denotes the maximum value all among the grid cells, i.e., $\beta = n(\tau^d(1+\epsilon)^{\max_{\forall x} \sigma_x})$. When $\beta < 1$, the series can be cut-off before any positive term to get a lower bound. Thus, for example, by retaining only four terms (i.e., from $j = 0$ to 3), the estimate for the expected number of empty cells is at least

$$\mathbf{E}[\text{empty}] > g^d + \sum_{j=1}^{3} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d$$

$$= g^d - n + \binom{n}{2} \tau^{2d} \left(\frac{R^2 - 1}{(1+\epsilon)^2 - 1} \right)^d - \binom{n}{3} \tau^{3d} \left(\frac{R^3 - 1}{(1+\epsilon)^3 - 1} \right)^d . \quad (17)$$

This translates to an upper bound on the estimate of the expected number of representative points:

$$m_{nu} = g^d - \mathbf{E}[\text{empty}]$$

$$\lessapprox n - \sum_{j=2}^{3} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d . \quad (18)$$

where $\lessapprox$ signifies less than or equivalent.

Even if $\beta \not< 1$, the number of empty cells can be approximated by the first k terms of the binomial expansion:

$$\mathbf{E}[\text{empty}] \simeq g^d + \sum_{j=1}^{k} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d . \quad (19)$$

An appropriate estimate of m_{nu} can then be obtained.

Table 3: Estimates of m using Eq. (19) and their empirical counterparts; k denotes the number of terms after which m converges ($\beta = 1.5 \times 10^{-6} \times n$, $g = 8$, $d = 7$, $\epsilon = 0.05$, $g^d = 2097152$).

n	k	m_{est}	m_{emp}
1×10^4	3	9975	9977
5×10^4	4	49357	49354
1×10^5	5	97449	97463
5×10^5	7	440688	440592
1×10^6	10	782406	782116
5×10^6	23	1857515	1857442
1×10^7	40	2059755	2059738

Table 3 shows that even when $\beta > 1$, the series converges within a few iterations and the estimates thus obtained are quite close to the empirical ones.

Since the maximum σ_x for any cell is $(g-1)d$, the condition under which $\beta < 1$ is $n(\tau^d(1+\epsilon)^{(g-1)d}) < 1$. Note that evaluating Eq. (13) is computationally inefficient when the number of cells, i.e., g^d is too high. Hence, the approximation using Eq. (17) is needed for only such cases. Table 2 shows that $\beta < 1$ for typical values of d, g, ϵ and n when g^d is high. It also lists the corresponding estimates for m.

Similar to the case of uniform grids, when the number of grid cells is fixed, but the number of points n is increased, the ratio m_{nu}/n decreases. Again, this happens as more points now hash to the same cell. The ratio increases when number of grid cells is increased keeping n fixed.

The more interesting observation is when the error parameter ϵ is changed, keeping all the other parameters fixed. When ϵ increases, the ratio of the largest grid cell to the overall volume increases and that of the smallest decreases. In other words, the distribution of the volume becomes more skewed. Thus, more points are likely to lie on a smaller number of cells, thereby increasing the expected number of empty cells. Hence, the ratio m_{nu}/n decreases with increasing ϵ.

5. QUALITY METRICS

In this section, we describe the various metrics that can be used to assess the quality of an approximate or reduced skyline set.

The first metric, proposed in [32], is the maximum Jaccard similarity between any two sets of points dominated by a pair of skyline points. The lesser the Jaccard similarity, the more diverse the two skyline points are. For a particular size k, the more diverse the reduced skyline set is, the better.

The second measure, proposed in [23], is the ratio of points dominated by a subset of size k to the total number of points. The more this ratio is, the better.

5.1 Covering Factor

In this paper, we introduce two more quality metrics based on *covering factor*. Intutively, the covering factor is the ratio by which a skyline needs to be stretched in order to be dominated by a returned point in SkyCover. Hence. if $s \in S'$, $cf_s = 1$. Otherwise, it is the (maximum) approximation ratio over all dimensions, max-

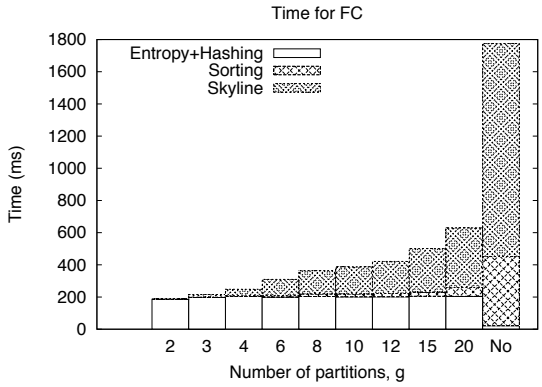


Figure 3: Running times for real data (FC).

imized over all the points $t \in S'$:

$$cf_s = \min_{t \in S'} \left\{ \max_i (t_i/s_i) \right\} \quad (20)$$

Thus, the covering factor essentially captures how well a skyline point $s \in S$ is covered by an approximate skyline point $t \in S'$.

5.2 Metrics Based on Covering Factor

Based on the covering factor, we propose two quality metrics:

1. *Worst Covering Factor (WCF)*, is the maximum covering factor of all the skylines.

2. *Cumulative Covering Factor (CCF)*, is the area under the curve of the cumulative distribution function (cdf) of the covering factors of all the skyline points.

Lower values of WCF indicate better quality of the solution. On the other hand, higher values of CCF signify that the covering factors attain their highest values earlier and are, thus, preferred.

6. EXPERIMENTAL RESULTS

In this section, we report in detail the results of experimenting with both real and synthetic datasets. All the experiments were done in Java on an Intel Core i7 CPU 870 @ 2.93 GHz machine with 8 GB RAM running Ubuntu Linux 12.04 LTS (64-bit).

For our empirical evaluation, we assume the worst case scenario of the query range Ω, i.e., $\Omega = [1, R)^d$. Hence, $D(\Omega) = D$, $S(\Omega) = S$, and $S'(\Omega_\epsilon) = S'$.

6.1 Real Dataset

The real dataset was the Forest Covertype (FC) dataset, containing different attributes of forest cover types, available from http://archive.ics.uci.edu/ml/datasets/Covertype. The size of the dataset is $n = 581012$. Similar to [32], only the first $d = 5$ attributes were considered. All the attributes were normalized to $[1, 2]$ space. The number of skylines is $s = 1356$.

6.1.1 Running Time

Figure 3 shows the running times when the number of grid partitions, g, is varied. It also includes the no-grid scheme. The total time is broken down into three components. The first is for grid partitioning (and entropy computation), the second is for sorting the m representatives, and the third is for the skyline computation routine. When g increases, m increases, thereby increasing the second and third components as well. As expected, the difference in skyline computation time is the main source of difference in the running time. When no grids are employed (i.e., the basic SFS method), the sorting and skyline finding times are too large. Overall, the SkyCover method runs 3-9 times faster.

6.1.2 Cardinality

Table 4 shows the statistics of the cardinality of skylines for the real dataset over the same range of g values. When m is low due to low g, a lot of actual skylines can be missed; however, a large proportion of them (the ratio w/s') always have an approximate skyline from the same grid cell. For high values of ϵ, k' approaches s. In this case, the number of skylines falsely reported ($v = k - k'$) is low as well.

6.1.3 Quality

The last set of experiments on real data measures the quality of the three methods, SkyDiver, k-RSP and our SkyCover, on the four quality metrics. Using a particular value of g, we first used our method to derive the SkyCover set. Assume that the cardinality is k. We then reduce the actual skyline set using SkyDiver and k-RSP to the same number k.

Figure 4 shows the covering factor metrics of the three methods. Expectedly, SkyCover is the best. Although the differences do not look significantly large, for the same k, while we ensure the WCF to be within $(1+\epsilon)$, SkyDiver and k-RSP routinely violate it. Thus, for these methods, no multiplicative error ratio can be guaranteed.

We also conducted a second set of experiments where we took a particular SkyCover set and reduced it to a subset of lower size, f, using the MH method of SkyDiver and the FM method of k-RSP. The skyline set is also reduced to the same size, f, using both SkyDiver and k-RSP. We then quantized all the four quality measures for varying f. We do it for $g = 8$ that produces $k = 333$.

Figure 5 shows the results. The MH method produces much better Jaccard similarity measures. However, very interestingly, when f is very small, using MH on SkyCover produces a better Jaccard measure than on SkyDiver. This shows that the SkyCover representation of the skyline set is extremely useful not only for the covering measures but also for other quality metrics. Similarly, FM produces better domination ratios and using FM on SkyCover is almost as good as k-RSP.

6.2 Synthetic Datasets

The synthetic datasets were generated using the code available from http://pgfoundry.org/projects/randdataset/. The parameters based on which the experiments were done are: (1) number of grid partitions per dimension, g, (2) error parameter, ϵ, (3) dataset cardinality, n, (4) dimensionality, d, and (5) type of dataset. While experimenting, we varied one parameter at a time while fixing the others to understand the effect of that single parameter better.

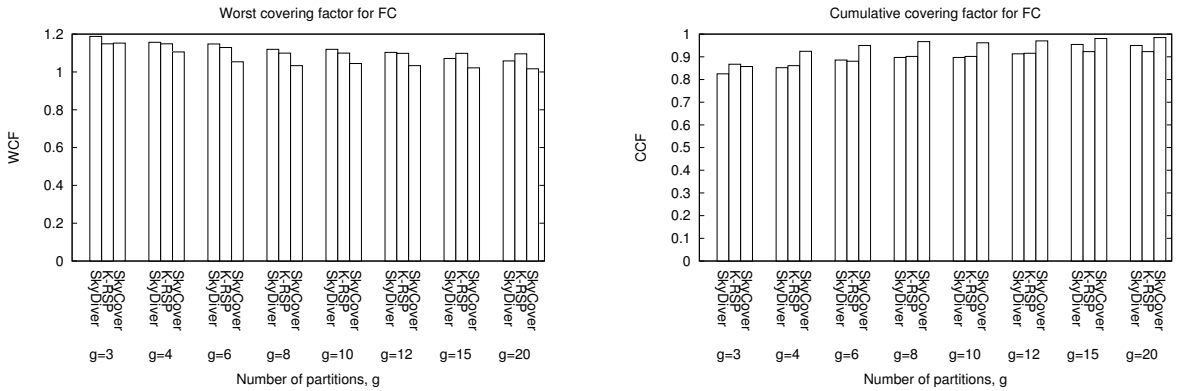
In the graphs, "NU" represents non-uniform grid partitioning, "Uni" represents uniform grid partitioning, and "No" represents the base method of computing the skylines without using grids.

6.2.1 Effect of Number of Grid Partitions

The first set of experiments (Figure 6a) measures the effect of number of grid partitions, g. For low values of g (till 5), the number of grid cells is so low ($g^d = 5^4 = 625$) that the ratio of the number of representative points m to the total number of points n is negligible. Consequently, the cardinality of the approximate skyline is very low as well and the entire method runs very fast. For medium values of g ($= 10$), when the number of grid cells ($g^d = 10^4$) is comparable to (but still less than) $n(= 50000)$, $m \to g^d$. In other words, almost all the grid cells are occupied. The running times are still lower than the base non-grid method. For high values of g

Table 4: Statistics of approximate skylines for the FC dataset: $n = 581012$, $s = |S| = 1356$, $d = 5$, $R = 2$.

Partitions per dimension	Error	Grid representatives	SkyCover size	Skylines truly reported	Skylines falsely reported	Skylines falsely missed	Covered by skyline in same cell	Covered otherwise
g	ϵ	$m = \|P\|$	$k = \|S'\|$	k'	$v = k - k'$	$s' = s - k'$	w	$t = s' - w$
2	0.414214	32	13	1	12	1355	1276	79
3	0.259921	172	30	11	19	1345	811	534
4	0.189207	515	87	45	42	1311	1102	209
6	0.122462	2086	173	108	65	1248	558	690
8	0.090508	5481	333	252	81	1104	651	453
10	0.071773	12014	427	285	142	1071	545	526
12	0.059463	22015	638	483	155	873	513	360
15	0.047294	44740	783	693	90	663	412	251
20	0.035265	103734	975	869	106	487	271	216



(a) Worst covering factor.

(b) Cumulative covering factor.

Figure 4: Variation of quality with number of partitions, g.

$(= 25)$, the number of grid cells is much larger $(\approx 4 \times 10^5)$ compared to n and consequently, almost all the points lie in a separate grid cell by itself. As a result, $m \to n$. The running times for the grid-based mechanisms become worse due to the additional overhead of hashing, etc. There is little to choose between uniform and non-uniform grids as far as running time is concerned.

6.2.2 *Effect of Error Parameter*

For a fixed R, the error parameter ϵ is complementary to g. When g increases, ϵ decreases. Hence, the effect of ϵ is the reverse of g.

6.2.3 *Effect of Dataset Cardinality*

We next show how the algorithms scale with increasing size of dataset (Figure 6b). The cardinality is varied from 10^4 to 10^7 with $g^d = 8^7 \approx 2 \times 10^6$ grid cells. Interestingly, the ratio of skyline points to n, i.e., s/n decreases with increasing n (but not the absolute number). When n is small compared to g^d, $m/n \to 1$, and there is no gain either in running time or otherwise. (Note that both the scales in the figure are logarithmic.) When n approaches g^d (e.g., for $n = 5 \times 10^5$, i.e., when $n/g^d \approx 1/4$), the ratio m/n starts falling off, and an appreciable difference in the running time between using the grids and otherwise starts showing up. When n is larger than g^d, the ratio of m/n is quite low, and consequently, the grid-based mechanisms exhibit much better running times. At high n $(= 10^7)$, the grid-based mechanisms are faster by a factor of more than 3. Also, since the hashing functions for the uniform case (which are division operations) are simpler than that for the non-uniform case (logarithms), the difference becomes significant at large n as there are $n.d$ such operations.

6.2.4 *Effect of Dimensionality*

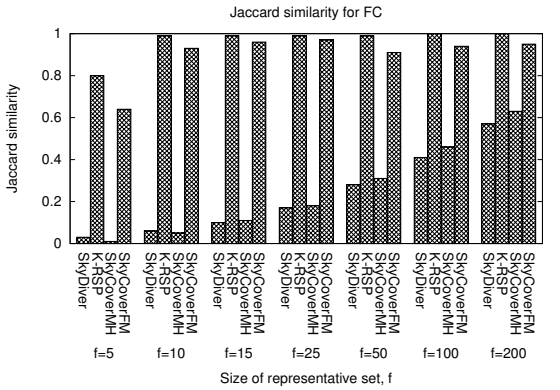
Figure 7a shows the effect of dimensionality, d. For independent datasets, the number of skylines grows exponentially with d. The running times follow the same behavior (note that the y-axis is logarithmic). At high d (from 10 onward), $m \to n$ due to too many grid cells $(g^d \sim 10^7)$. Consequently, there is no gain in the running time by employing grids. For medium to low values of d (≤ 7), the running times are much better as both m and k are lower.

6.2.5 *Effect of Type of Data*

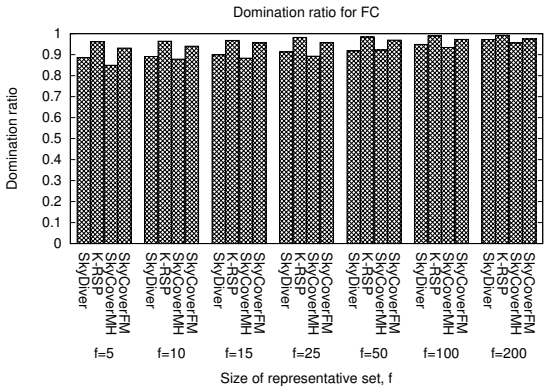
The last set of experiments is to gauge the effect of the type of the data. We generated three standard data types, independent, correlated and anti-correlated. For correlated data, the number of skylines is low as one point is likely to dominate many points. In an anti-correlated dataset, the skyline cardinality is high as it is unlikely that a point dominates another.

Interestingly enough, the number of representative grid cells, m, decreases for both correlated and anti-correlated datasets in relation to the independent one. The reason is that the data is not spread over the entire space uniformly, but is concentrated along certain directions. The correlated dataset is spread along the main diagonal; hence, many points fall on the largest grid $(g-1, \ldots, g-1)$ and m is lower. The anti-correlated dataset is spread along the main anti-diagonal and, hence, m is significantly larger than that of the correlated one but still lower than the independent distribution.

The running time for the anti-correlated dataset is much greater than independent though (Figure 7b). This is due to the fact that the high number of skylines makes the comparison step in SFS of an

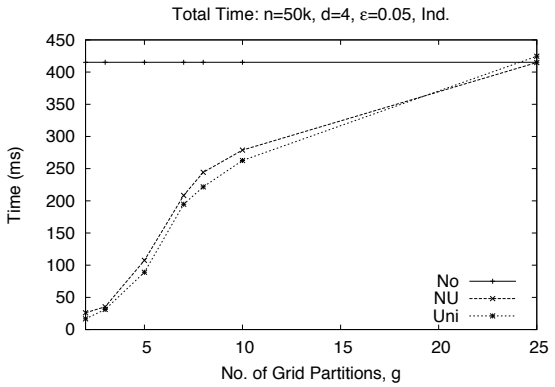


(a) Jaccard similarity.

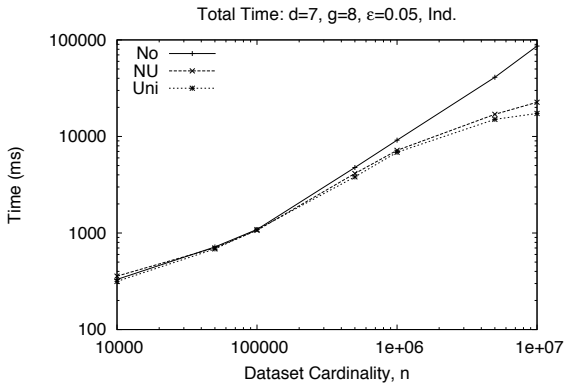


(b) Domination ratio.

Figure 5: Variation of quality with size of representative set, f.



(a) Number of partitions, g.



(b) Dataset cardinality, n.

Figure 6: Effect of g and n for synthetic data.

object against the current skylines much less efficient as the window size is larger. The gain in running times over the base method is the least for the independent dataset.

6.3 Summary of Experiments

We can summarize the empirical observations obtained from the experiments on synthetic data as follows. If all the other parameters are fixed, it is better to increase ϵ up to the factor that the application can tolerate. When g^d is much larger than n, then $m \rightarrow n$ and the grid-based mechanisms are not beneficial. On the other hand, if g^d is much less than n, then m is constrained by g^d and the number of approximate skylines retrieved is much lower than the actual skyline cardinality. The SkyCover guarantees of bounded multiplicative errors still holds, though. In general, the uniform grid-based partitioning method is faster than the non-uniform counterpart although it provides no guarantees. Finally, the number of grid representatives is the largest when the data dimensions are independently distributed, and our method works better for correlated and anti-correlated datasets.

7. CONCLUSIONS AND FUTURE WORK

Range-constrained skyline queries retrieve skyline points over a query range and is a generalization of the skyline query. In spite of having several applications, range-constrained skyline queries have not received much research attention. The reason is that even with a constrained range, the size of the skyline set can be impractically large. To address the above, in this paper, we introduced the concept of approximate range-constrained skyline queries, and proposed the SkyCover framework for efficiently computing them. The framework employs a hashing scheme based on non-uniform grid partitioning. The hashing itself guarantees the approximation bound on the desired skyline set, thus avoiding any separate approximate skyline computation technique. The hashing also significantly contributes towards efficiency.

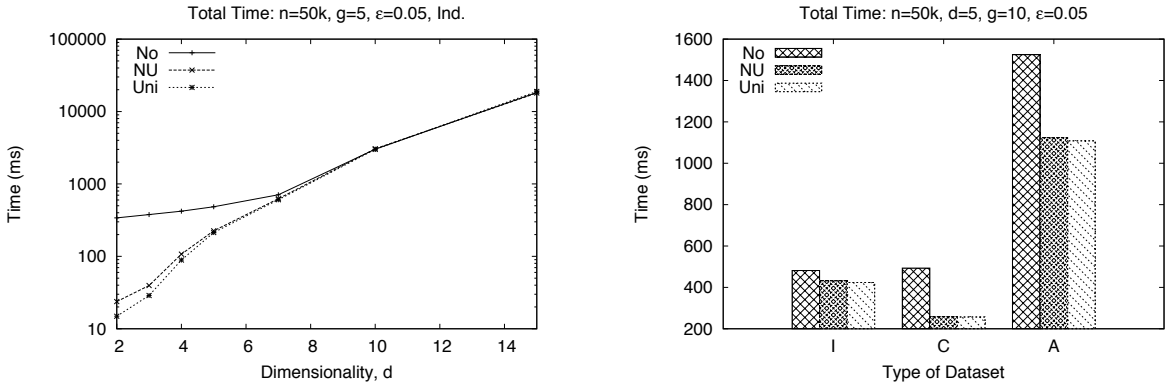
We also proposed two new metrics that can be used to measure the quality of an approximate skyline set. Empirical evaluation of our framework shows that it is significantly faster than the competing techniques, and yields solutions with high quality.

The grid-based hashing mechanism is generic enough to capture other kinds of errors including additive errors and no errors. More importantly, this framework can be applied in various settings such as data streams, and distributed environments. Detailed experimentation and analysis of such schemes remain a future work.

8. REFERENCES

[1] F. N. Afrati, P. Koutris, D. Suciu, and J. D. Ullman. Parallel skyline queries. In *ICDT*, pages 274–284, 2012.

[2] J. L. Bentley, H. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM (JACM)*, 25(4):536–543, 1978.



(a) Dimensionality, d.

(b) Type of dataset.

Figure 7: Effect of d and type of dataset on synthetic data.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[4] C. Brando, M. Goncalves, and V. González. Evaluating top-k skyline queries over relational databases. In *DEXA*, pages 254–263. Springer, 2007.

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, pages 327–336, 1998.

[6] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *SIGMOD*, pages 503–514, 2006.

[7] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, pages 717–719, 2003.

[8] A. Cosgaya-Lozano, A. Rau-Chaplin, and N. Zeh. Parallel computation of skyline queries. In *HPCS*, page 12, 2007.

[9] A. Das Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based preference distributions. In *ICDE*, pages 387–398, 2011.

[10] E. Dellis, A. Vlachou, I. Vladimirskiy, B. Seeger, and Y. Theodoridis. Constrained subspace skyline computation. In *CIKM*, pages 415–424. ACM, 2006.

[11] H. Eder. On extending PostgreSQL with the skyline operator. Master's thesis, Vienna University of Technology, 2009.

[12] P. Flajolet and G. Nigel Martin. Probabilistic counting algorithms for database applications. *J. Computer and System Sciences*, 31(2):182–209, 1985.

[13] Y. Gao, J. Hu, G. Chen, and C. Chen. Finding the most desirable skyline objects. In *DASFAA*, pages 116–122, 2010.

[14] Z. Huang, Y. Xiang, and Z. Lin. l-SkyDiv query: Effectively improve the usefulness of skylines. *SCIENCE CHINA Information Sciences*, 53(9):1785–1799, 2010.

[15] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[16] W. Jin, J. Han, and M. Ester. Mining thick skylines over large databases. In *PKDD*, pages 255–266, 2004.

[17] A. K. Kalavagattu, A. S. Das, K. Kothapalli, and K. Srinathan. On finding skyline points for range queries in plane. In *CCCG*, 2011.

[18] V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. *Theor. Comput. Sci.*, 371(3):148–154, 2007.

[19] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: an online algorithm for skyline queries. In *VLDB*, pages 275–286, 2002.

[20] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975.

[21] H. Li, Q. Tan, and W.-C. Lee. Efficient progressive processing of skyline queries in peer-to-peer systems. In *InfoScale*, 2006.

[22] Z. Li, Z. Peng, J. Yan, and T. Li. Continuous dynamic skyline queries over data stream. *J. Computer Research and Development*, 1:014, 2011.

[23] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDE*, pages 86–95, 2007.

[24] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, pages 467–478, 2003.

[25] C. Rhee, S. K. Dhall, and S. Lakshmivarahan. An optimal parallel algorithm for the maximal element problem (abstract). In *CSC*, page 435, 1990.

[26] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Nørvåg. Agids: A grid-based strategy for distributed skyline query processing. In *Int. Conf. Data Management in Grid and Peer-to-Peer Systems (Globe)*, pages 12–23, 2009.

[27] J. Selke and W.-T. Balke. Skymap: a trie-based index structure for high-performance skyline query processing. In *DEXA*, pages 350–365, 2011.

[28] L. Su, P. Zou, and Y. Jia. Adaptive mining the approximate skyline over data stream. In *ICCS*, pages 742–745, 2007.

[29] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.

[30] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *ICDE*, pages 892–903, 2009.

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

[32] G. Valkanas, A. N. Papadopoulos, and D. Gunopulos. Skydiver: a framework for skyline diversification. In *EDBT*, pages 406–417, 2013.

[33] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. El Abbadi. Parallelizing skyline queries for scalable distribution. In *EDBT*, pages 112–130, 2006.

[34] T. Xia, D. Zhang, and Y. Tao. On skylining with flexible dominance relation. In *ICDE*, pages 1397–1399, 2008.

Short Text Matching in Performance Management

Manoj Apte
Tata Consultancy Services
manoj.apte@tcs.com

Sachin Pawar*
Tata Consultancy Services
sachin7.p@tcs.com

Sangameshwar Patil†
Tata Consultancy Services
sangameshwar.patil@tcs.com

Sriram Baskaran
Tata Consultancy Services
sriram.baskaran@tcs.com

Apoorv Shrivastava
Tata Consultancy Services
apoorv.shrivastava@tcs.com

Girish K. Palshikar
Tata Consultancy Services
gk.palshikar@tcs.com

ABSTRACT

In "Role" based Performance Appraisal process the evaluation of Individuals is done based on the meeting of target for "Goals" given to that individual in the specified time period. Standardization of goals with the help of a pre defined "template" is important for completeness and correctness of role definition and comparing two individuals. Since a goal is a short textual description of expected activity we pose this as a matching problem and explore two different approaches that use minimal human supervision. First approach is based on co-training framework which uses goal description and available additional information in the form of self comments. The second approach uses semantic similarity using weak supervision framework. We demonstrate superior performance of the two approaches as compared to multiple baselines. First approach gives better recall while second approach scores better in precision. Based on the objective of the end application any of the approaches can be used.

1. INTRODUCTION

Quality, utilization and productivity of the workforce are very important factors to be considered by Human Resource (HR) Department of an organization. Performance Appraisal (PA) of the workforce focusses on the Quality factor and is used for identifying Top Performers and laggards in the organization. As organizations become larger it becomes difficult for the HR to measure each individuals performance and so process oriented approach needs to be followed. One approach followed by large organizations in this direction is to move to Role based Performance Management. In Role based Performance Management, each person is mapped to certain Role based on the expected Responsibilities and Activities to be performed. The responsibilities and activities for each role have to be defined in such a way that they represent a large amount of tasks done by the workforce in the specified role. Once the Role, Responsibilities and Activities are identified it is important to set *Goals* with specified targets over a particular time period and then measure the performance based on the inputs given by the individual as well as the supervisor. Recently with the emergence of Analytic tools the PA process is also tracked for improvements based on process parameters. The results of this analysis also drive the process changes in the PA process.

We have worked on analyzing the PA process of a large Information Technology (IT) company employing more than $324,000$ associates and has a robust Performance Appraisal Process defined. The process involves the following:

1. Role is assigned for each person. A Goal template is associated with each Role. This goal template contains a list of standardized goals based on the role definition.
2. Supervisor sets the goals for each time period along with targets. This is known as Goal Setting Process. Here, the supervisor has to set at least 5 goals from the Goal Template for the corresponding role of the individual. These goals are known as *Template Goals*.
3. At the end of each time period, the individual writes the achievements and the remarks about the working environment from his/her perspective in *Self Comments* for each goal.
4. Based on the Self Comments, the Supervisor writes his/her opinion on the achievements of the individual and shortcomings if any.
5. The supervisor scores the individual based on the performance.

In the goal setting process, supervisors expect some freedom to assign goals which are not exactly fitting the Goal Template based on the activities the individual is expected to perform. As a result the supervisors are given freedom to set goals manually in addition to the template goals.

In our sample we have $156,904$ confirmed employees across 869 roles. The total number of Goals assigned is $2,176,974$ out of which template goals are $863,465$ (39.66%) and manually created goals are 1313509. The Role representing the largest number of individuals is 'Developer' ($46,476$ Individuals). The key statistics of the dataset of the Role 'Developer' are given in Table 1 below.

We find that supervisors have used their freedom to assign goals outside the template and assigned lot of manually

*Doctoral research scholar at Dept. of CSE, IIT Bombay
†Doctoral research scholar at Dept. of CSE, IIT Madras

#Individuals	46,476
#Goals	638,301
#Template Goals assigned	324,036
#Manually created goals assigned	314,265
#Manually created goals	46,853
Size of Goal Template	41

Table 1: Details for the role "Developer"

created goals. Some of these goals are likely to be very similar to one of the template goals. For example, following manually created goals are very similar to the template goal `Number of Certifications`:

1. `# Certifications`
2. `No.  of certification taken`
3. `No of certifications obtained`
4. `Domain Certification`

High number of manually created goals poses the following problems :

1. Completeness and correctness of Role Definition - If unusually high number of manually created goals that are not similar to any of the template goal are assigned, then the Role definition should be reviewed to see whether some of the responsibilities and activities for the Role as expected by the supervisors should be added. Similarly,
2. Comparison of two individuals based on Goal assignment - Generally organizations follow the method of ranking the individuals and forced distribution where the individuals are compared with each other. In such a scenario, it is important that the expectations from the individuals are comparable. Such a comparison of expectations can be done by comparing the goals set for the individuals. It is not possible unless the goals are mapped against a standard.

To solve the above problems it is important to standardize individual collection of goals. It can be achieved by matching each manually created goal to its equivalent template goal. The manually created goals which are not equivalent to any of the template goals should be kept separate.

There are 3 types of inputs available which help in determining whether a goal description is matching with one of the template goals : (i) Template Goal description, (ii) Description of manually created goal and (iii) Self Comments written by individuals for the goal. For the matching of manually created goals with the template goals, we explore two different classification based approaches. The first approach uses a well-known co-training [3] framework and the second one uses semantic similarity using weak supervision [22] framework. The former approach uses all the 3 available inputs but no human-in-the-loop. The latter uses the goal descriptions along with human feedback and uses active learning to minimize human intervention.

The rest of the paper is organized as follows. Section 2 covers the related literature and its limitations for applying in our domain. Our co-training based approach is described section 3 and weak supervision based approach is described in section 4. We have compared our approaches with different baseline implementations and we present the detailed results in section 5. Finally, we conclude in section 6 with some discussion on future work.

2. RELATED WORK

In performance appraisal systems, generally supervisors set crisp, direct and objective goals. Therefore, most of the goal descriptions are quite short having average length of 10 words. Classification of short documents has been an area of active research in the natural language processing and information retrieval communities. Tweet classification is an increasingly important variant of the short text classification problem. Tweet classification to improve information filtering has been investigated in [26, 25]. They have focused on extracting tweet specific features (such as user call out, shortening of words, presence of time-event phrases, opinionated words etc.) and used standard supervised machine learning approach with naive Bayes algorithm as the learner. Another kind of short text is seen the textual survey responses. Classifying the short text documents seen in survey responses has been tackled by Giorgetti et al [11]. For this type of short documents, Giorgetti et al. observed that supervised learning methods outperform the dictionary based approach. Li and Yamanishi [14] use frequent patterns and association rule mining for classifying short text in automobile survey. Sun [27] proposed a simple, scalable and non-parametric approach for short text classification. It first selects representative query words using bag of words and Clarity score and then searches for a small set of labelled short texts best matching the query words. The predicted category label is the majority vote for the search results. Lu and Li [17] proposed a deep architecture for matching short texts for addressing various matching tasks like finding relevant answers to a given question. Other approaches for short text classification are proposed by Zelikovitz and Hirsh [30], Bobicev and Sokolova [4] and Chen et al. [6].

Due to scarcity of labelled data and high cost involved in construction of it, semi-supervised learning approaches are used. Zhu [31] surveys various semi-supervised learning approaches such as self-training, co-training, Expectation Maximization (EM) and different graph based methods. Nigam et al. [20] proposed an approach for semi-supervised text classification from labelled and unlabelled documents using EM. Co-training [3] is a popular semi-supervised approach for training a classifier using both labelled and unlabelled data. Co-training requires two mutually independent feature views where each individual view is sufficient for classification. Two models are trained with initial labelled data, each using one of the feature views. They are then used to classify the remaining unlabelled records and very high confidence predictions by either of the model, are added to the labelled data. The models are re-trained using the additional labelled data and the process is repeated. Short text with side information can be easily mapped to two separate feature views - i) Features derived from short text itself; and ii) Features derived from the side information. Hence, co-training is one of the natural choices for addressing the problem of short text matching with side information. Nigam and Ghani [19] proposed a hybrid algorithm namely co-EM which is a combination of co-training and EM. Like co-training, it uses two feature views and like EM, it probabilistically labels all the unlabelled data. An-

other approach by Ghani [10] proposes a framework to incorporate unlabelled data in Error-Correcting Output Coding (ECOC) by decomposing multiclass problem into multiple binary problems and then using co-training to learn the individual binary classification problems. This was designed to scale up for multi-class classification with large number of classes.

There are special techniques designed for classification when labelled data is available only for positive class. Li and Liu [15] and Liu et al. [16] consider this problem of text classification with only one class of labelled documents and a set of unlabelled documents. Classifier is built in two steps. First step just identifies a reliable set of negative instances from the set of unlabelled instances. Second step iteratively builds a classifier using algorithms like EM. Fung et al. [9] and Elkan and Noto [7] also propose different approaches which use only positive data for learning.

3. METHOD 1: GOAL DESCRIPTION CLASSIFICATION USING CO-TRAINING

We model the problem of matching manually created goals to template goals as a classification problem. We opt for a semi-supervised learning based approach for two main reasons : i) Initial labelled data is limited in size and ii) unlabelled data is easily available.

Initial labelled data is constructed automatically by using the set of template goals where each template goal is assigned a distinct class label. Optionally, we can manually assign same class label to multiple template goals which are "semantically" similar. In addition to class labels covering the template goals, there is a special class label `NONE` which indicates that none of the template goals are matching.

We propose to use co-training framework [3] for semi-supervised learning. The major motivation for opting for co-training framework was that there is a natural separation of information used for classification of manually created goals. There are 2 different views of each goal (template goal as well as manually created goal):

1. V_1: Goal description itself
2. V_2: Self comments written for the goal

The intuition behind using V_2 is that similar goals are understood by the individuals in similar way and hence the corresponding self comments tend to be similar.

Two different classifiers are trained, each using features generated from only one of the views. We use Maximum Entropy Classifier with real-valued features.

3.1 Classifier C_1: Using Goal Descriptions

Classifier C_1 is trained using features generated only from the goal descriptions. For each goal description, various features are generated as follows:

1. Root-word of each word in the goal description becomes a feature. The value of the feature is set to δ^i where $0 < \delta < 1$ and i is the index of the corresponding word. The intuition is that value corresponding to a word feature is lower if that word appears later in the goal description. In practice, the value of δ equal to 0.95 is used.
2. If a goal description contains any two words such that both the words occur in a single "template" goal description, then such a combination of two words becomes a feature with a fixed weight of 1.5. Since the goal descriptions are generally short, we expect such pair of words to capture its essence in a better way.

3.2 Classifier C_2: Using Self Comments

Classifier C_2 is trained using features generated only from the self comments. For each goal description, various features are generated as follows:

1. A set of self comments is associated with each goal description. Root-words of all the words used in these self comments become features.
2. A bag-of-words is created for each goal description by using the set of associated self comments. This bag-of-words can be viewed as a large document. Following the Information Retrieval (IR) literature, TF-IDF (Term Frequency - Inverse Document Frequency) scores are computed for each word. For each word feature, its value is nothing but the corresponding TF-IDF score.

3.3 Challenge of Negative Instances

Our initial labelled data is constructed automatically using the set of template goals. This process does not label any goal with negative label, i.e. `NONE`. We cannot characterize the `NONE` class using a finite set of labelled examples. Hence, manually labelling examples of `NONE` class is not sufficient. This poses a challenge to train a classifier with just positive examples. We deal with this challenge in following way:

1. **Identifying candidate negative goals:** For each goal, we associate two word vectors which are vector-space representations using the views V_1 and V_2. For each manually created goal, we find its similarity with all the template goals. Both the views are considered and cosine similarity between the word vectors is used as a similarity measure. All those manually created goals having the highest similarity with any template goal, lower than a pre-defined threshold, are identified as "candidate negative" goals. Algorithm 1 describes the detailed procedure.
2. **Co-Training:** The "candidate" negative goals identified in the previous step are not considered as a part of unlabelled goals during the Co-Training iterations.
3. **Disagreement between two views:** After the termination of Co-Training process, the "candidate" negative goals are classified using both the classifiers. If two classifiers predict different class labels for any "candidate" negative goal and none of it is very confident about its classification, then such goals get the final class label as `NONE`.

3.4 Challenge of Class Imbalance

In each iteration of co-training, the unlabelled instances which are classified with high confidence are added to the labelled data along with the predicted label. As a template goal corresponds to a class label in our case, depending on the variation in the manually created goals belonging to a particular template goal, the confidence values assigned for

Data: T (Set of template goals), N (Set of manually created goals), α (Weight given to V_1, lies between 0 and 1 with default value = 0.6), θ (Similarity threshold on cosine similarity with default value = 0.2)
Result: N_{neg} (Set of manually created goals which are candidates for negative (`NONE`) class)

1 $WV_1 := []$; `/* Empty mappings with` $key = goal$ `and` $value =$ `word vector using` V_1 `*/`
2 $WV_2 := []$; `/* Empty mappings with` $key = goal$ `and` $value =$ `word vector using` V_2 `*/`
3 **foreach** $g \in T \cup N$ **do**
4 $\quad WV_1[g] :=$ TF-IDF word vector using goal description of g;
5 $\quad WV_2[g] :=$ TF-IDF word vector using self comments for g;
6 **end**
7 **foreach** $g \in N$ **do**
8 $\quad S_{max} := 0$;
9 $\quad$ **foreach** $g' \in T$ **do**
10 $\quad\quad sim_1 := CosineSim(WV_1[g], WV_1[g'])$;
11 $\quad\quad sim_2 := CosineSim(WV_2[g], WV_2[g'])$;
12 $\quad\quad sim := \alpha \cdot sim_1 + (1 - \alpha) \cdot sim_2$;
13 $\quad\quad$ **if** $sim > S_{max}$ **then** $S_{max} := sim$
14 $\quad$ **end**
15 $\quad$ **if** $S_{max} < \theta$ **then** $N_{neg} := N_{neg} \cup g$
16 **end**
17 **return** N_{neg};

Algorithm 1: Algorithm *GetNegativeCandidates* to determine candidates for negative (`NONE`) class

each class may vary. If the only criteria for adding instances to the labelled set is to check whether the classification probability is more than some threshold, then some class imbalance may get introduced in the labelled set in each iteration. Figures 1 and 2 show the skewed class distribution in the first iteration of co-training algorithm if a fixed threshold is used to select instances to add. In order to prevent this, our algorithm enforces a constraint on maximum and minimum number of instances of any class to be added to the labelled set in each iteration. This ensures that all classes get a fair representation in the labelled data.

3.5 Classification using Co-training framework

Algorithm 2 describes our co-training based approach in detail. The overview of this approach is as follows:

1. The initial labelled data is automatically created by labelling each template goals with different class label. All the manually created goals constitute the initial set of unlabelled goals.

2. A set of "candidate" negative instances is identified **(line 3)** and these instances do not participate in co-training iterations.

3. Two classifiers are trained using the labelled data : (i) C_1 using the goal descriptions view, V_1 and (ii) C_2 using self comments view, V_2 **(lines 6-7)**.

4. All the unlabelled goals which are not candidates for "negative" class are classified using both C_1 and C_2. The predictions along with classification confidence are recorded **(lines 9-17)**.

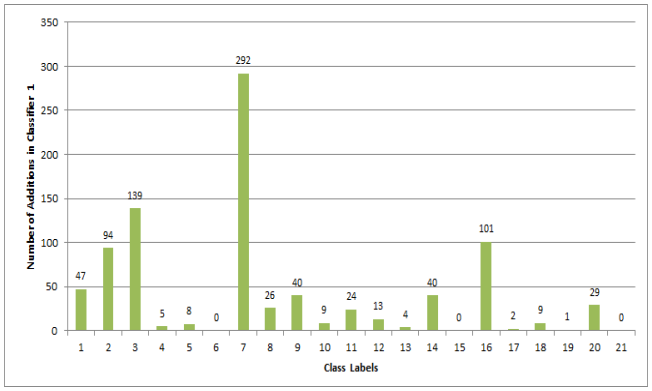


Figure 1: Class Distribution of instances classified by C1 with confidence more than some threshold in the first iteration

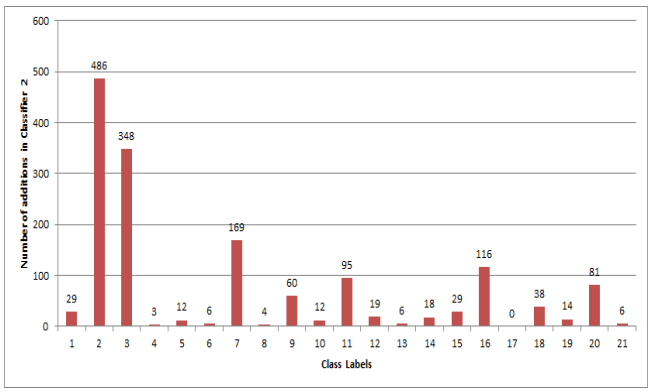


Figure 2: Class Distribution of instances classified by C2 with confidence more than some threshold in the first iteration

5. For each classifier, for each class label, unlabelled instances predicted with high confidence are added to the set of labelled instances such that the constraints on minimum and maximum number of additions are complied with **(lines 19-36)**.

6. The steps 3 to 5 are repeated till number of iterations reaches the specified limit or no new additions take place.

7. After co-training iterations, all the remaining unlabelled instances (including "candidate" negative instances) are classified using both the classifiers. If at least one classifier predicts a class label with the confidence greater than the defined thresholds, then the identified class label is assigned **(lines 41-46)**. Rest of the instances are labelled as `NONE` **(lines 47-48)**.

4. METHOD 2: GOAL DESCRIPTION CLASSIFICATION USING WEAK SUPERVISION

As discussed in earlier section, there are basically three types of inputs which help in determining whether a description of a manually created goal is similar to one of the available goals in the "template of goals" for a given role.

Data: T (Set of template goals), N (Set of manually created goals), I (Maximum number of co-training iterations), ADD_{max}, ADD_{min} (Maximum and minimum number of unlabelled goals per class in each iteration for each classifier), η_1, η_2 (Confidence thresholds for two classifiers), η'_1, η'_2 (Lenient confidence thresholds for two classifiers), η^a, η^f_1, η^f_2 (Confidence thresholds used for final predictions)

Result: Set of tuples of the form (goal,label) where goal is from N and label is the matching template goal in T and `NONE` if none of the template goals match.

$L := T$; `/* Initialize to set of tuples (template goal, class label) */`
$U := N$; `/* Initialize to set of manually created goals */`
$N_{neg} := GetNegativeCandidates(T, N, 0.6, 0.2)$; `/*` $N_{neg} \subset N$ `*/`
$iter := 0$;
while $iter < I$ **do**
  C_1 := MaxEnt Classifier trained using goal descriptions in L;
  C_2 := MaxEnt Classifier trained using self comments for each goal in L;
  $GC_1 := []; GC_2 := []$; `/* Empty two-level mappings with` $key_1 = label$, $key_2 = goal$, $value = confidence$ `*/`
  foreach $g \in U$ **do**
    if $g \in L$ **or** $g \in N_{neg}$ **then continue**;
    (l_1, pr_1) := Classify g using C_1; (l_2, pr_2) := Classify g using C_2;
    if $iter < J$ **and** $l_1 = l_2$ **then** `/* Agreement of classifiers is checked for first` $J(< I)$ `iterations */`
      $GC_1[l_1][g] = pr_1$; $GC_2[l_2][g] = pr_2$;
    else if $iter \geq J$ **then**
      $GC_1[l_1][g] = pr_1$; $GC_2[l_2][g] = pr_2$;
    end
  end
  $N := 0$; `/* Overall number of unlabelled goals added to` L `in current iteration */`
  foreach $label \in GC_1.keys$ **do**
    $N_{label} := 0$; `/* No. of unlabelled goals added by` C_1 `to` L `with` $label$ `in current iteration */`
    foreach $(g, pr) \in GC_1[label]$ **do** `/* selected in descending order of` pr `for the given` $label$ `*/`
      if ($N_{label} < ADD_{max}$ **and** $pr > \eta_1$) **or** ($N_{label} < ADD_{min}$ **and** $pr > (\eta'_1 + (iter * 0.05))$) **then**
        $L := L \cup (g, label)$; $N_{label} := N_{label} + 1$; $N := N + 1$;
      end
      if $N_{label} > ADD_{max}$ **then break;**
    end
  end
  foreach $label \in GC_2.keys$ **do**
    $N_{label} := 0$; `/* No. of unlabelled goals added by` C_2 `to` L `with` $label$ `in current iteration */`
    foreach $(g, pr) \in GC_2[label]$ **do** `/* selected in descending order of` pr `for the given` $label$ `*/`
      if ($N_{label} < ADD_{max}$ **and** $pr > \eta_2$) **or** ($N_{label} < ADD_{min}$ **and** $pr > (\eta'_2 + (iter * 0.05))$) **then**
        $L := L \cup (g, label)$; $N_{label} := N_{label} + 1$; $N := N + 1$;
      end
      if $N_{label} > ADD_{max}$ **then break;**
    end
  end
  if $N = 0$ **then break**;
end
foreach $g \in U$ **do**
  (l_1, pr_1) := Classify g using C_1; (l_2, pr_2) := Classify g using C_2;
  if $l_1 = l_2$ **and** $\frac{1}{2}(pr_2 + pr_2) > \eta^a$ **then**
    $L := L \cup (g, l_1)$;
  else if $pr_1 > \eta^f_1$ **then**
    $L := L \cup (g, l_1)$;
  else if $pr_2 > \eta^f_2$ **then**
    $L := L \cup (g, l_2)$;
  else
    $L := L \cup (g, NONE)$;
  end
end
return L;

Algorithm 2: Method 1 : Our Co-training based approach

The three types of inputs are: (i) the textual description of a goal in the set of "template goals" (i.e., the goals to be used as set of classes/categories in classification process), (ii) the textual description of a manually created goal whose similarity with one of the existing "template goals" is to be determined, and (iii) the text of comments (also called *self-comments* made by an appraisee in support of his/her achievements corresponding to the goal (i.e. the input goal in previous item (ii)).

Apart from the co-training based approach for classification mentioned in another section (which mainly uses the input goal description and self-comments as features), we propose to measure the similarity between the textual description of a "template goal" and the manually created goals to be classified. We note that often the goal descriptions are written more like phrase based text snippets and do not have complete sentence structure. In many ways the textual goal descriptions are similar to short text snippets observed in product review comments or textual responses to open-ended survey questions. Hence, we use the algorithm 3 which is a variant of the short text classification algorithms described in [22, 21] and adapt it to the current problem of manually created goal classification for the employee performance management problem. This algorithm uses weak supervision to minimize the human effort required to create labeled training data. We use a two stage iterative method in which we carry out first level classification using the similarity between the "template goal" description and the manually created goal without any human supervision (i.e., without use of labeled training data). The output of this stage is then passed through a weakly supervised learning stage. We use active learning paradigm for weak supervision to minimize the amount of feedback sought from human expert.

4.1 Stage I - Semantic text similarity based classification

We first compute semantic similarity between the "template goal" description and manually created goals to be classified. For this purpose, we represent each word/phrase in the textual description of "template goal" (i.e., class) using its WordNet [8] synset ids to capture the expected meaning of each word. Further, we assign a numeric weight to measure the relative importance of this word/phrase within the "template goal". For determining expected meaning of a word/phrase, we make use of unsupervised word sense disambiguation techniques [18, 12]. For a given word, this enables us to find out synonyms, antonyms as well as other related words (hypernyms, hyponyms etc.). To estimate relative importance of a word/phrase within the "template goal" description, we use the numeric weight assignment as described in [22]. We then find out word/phrase level overlap between the semantically enriched representation of "template goal" and the input textual description of manually created goal whose similarity with the "template goal" is to be computed. For the set of common words/phrases, the numeric weight signifying their relative importance is combined together. To compute the aggregate value of these possibly multiple relative importance values, we use the certainty factor algebra (CFA) [5]. If this aggregate value is above a pre-determined threshold, the manually created goal is deemed to be similar to the "template goal" and classified accordingly.

Data: T (Set of template goals), N (Set of manually created goals), I (Maximum number of iterations)
Result: Set of tuples of the form (goal, set of labels) where goal is from N and set of labels is the set of matching template goals in T and NONE if none of the template goals match

while $iter \leq I$ **do**
  Stage-I:
  foreach $t \in T$ **do**
    $L_t = \emptyset$
    Let $(w_1, w_2, \ldots, w_t)$ be the sequence of words in the textual goal description of template goal t.
    Let $t' = (w'_1, w'_2, \ldots, w'_t)$ be new semantically enriched representation of t; where w'_i = set of estimated word senses and corresponding related words (synonym, derivationally related words) of w_i determined by using unsupervised WSD techniques [18] as well as considering the feedback received in the weak supervision stage.
    foreach $n \in N$ **do**
      $c_{nt'} = n \cap t'$
      If the combined IDF-based relative importance of words in $c_{nt'}$ is above a threshold θ, then $L_t = L_t \cup n$ (i.e., assign t to n).
    end
  end
  Stage-II:
  foreach $t \in T$ **do**
    $C_t = Output\_of\_Clustering(L_t)$
    foreach $c \in C_t$ **do**
      Seek human feedback about correctness of assignment of t to medoid of cluster c
      Update t' based on feedback.
    end
  end
  iter++;
end
return $\{(t, L_t) | t \in T\}$

Algorithm 3: Method 2: Goal description classification using semantic similarity and weak supervision

4.2 Stage II - Weak Supervision using Active Learning

The classification carried out in the stage-I is vetted using human supervision. To minimize the human involvement and to use the weak supervision machine learning paradigm, we use active learning [24]. The most informative examples from the output of stage-I are selected using the active learning informativeness criteria. These examples are then presented to a human expert to ascertain whether the classification is correct. We cluster the manually created goals which have been deemed similar with a given "template goal" and then select a representative example for each cluster which is then queried to the human expert. To estimate the number of clusters, silhouette coefficient [28, 13, 23] is used. Silhouette Coefficient (ShC) is a practically useful measure to compare the trade-off between intra-cluster cohesiveness and inter-cluster separation. Silhouette coefficient for i^{th} data point is given by $ShC_i = \frac{b_i - a_i}{max(a_i, b_i)}$, where a_i is the

average distance between i^{th} data point and other points in the same cluster; and b_i is average distance between i^{th} data point and all other points in the next nearest cluster. Silhouette Coefficient for a given clustering of data-points is average of individual ShC_i values. At run-time, multiple clusterings are tried out and the clustering having highest silhouette coefficient among the explored is chosen as the final clustering. A representative example from each cluster is chosen as the query to be posed to the human expert. The label verification by human expert is used for updated classification in the next iteration. The specific and detailed description of the technique is given in the algorithm pseudocode.

5. EXPERIMENTAL ANALYSIS

To demonstrate effectiveness of our approaches, we have compared it with classification using standard similarity measures like Cosine, Jaccard and Dice similarities with a defined threshold for each and off the shelf implementation of Nave Bayes and Maximum Entropy classifiers. Before we define the experiment setup we describe the datasets.

5.1 Dataset for Baseline Algorithms

We have used the performance appraisal dataset of an organization for one year. We have taken the goals that are set for the given year to all the associates. Since we are doing the analysis role-wise, we have identified the most frequently assigned role in the organization and run the different algorithms for that role.

The data for the algorithms contains original goal texts. We perform cleaning of the original goal text to get a cleaned goal text. The cleaning process starts with POS Tagging where each word of the text is tagged with the Part of Speech. We have used both Open NLP [1] and Stanford POS tagger [29] for POS Tagging. The choice of the POS Tagger didn't make a significant difference in the final classification results.

The goal names are very short and noisy text. They are bound to have spelling mistakes and human errors. We do a spell check of the goal names based on a corpus of valid words that are highly frequent in the given domain. We used Jazzy [2] for spell check and used the first suggestion (if present) as the valid word. Based on domain knowledge, we also replace known acronyms with their expansions. For example, all occurrences of `CSI` are replaced with `Client Satisfaction Index`.

The cleaning process removes all the punctuation marks and converts all upper case to lower case characters. It removes a set of stop words from the given goal name. Along with the general stop words like `and`, `the`, `on` etc., we also remove domain-specific stop words like `number`, `percentage`, `project`, etc. Based on domain knowledge, we replaced certain words in the goal text with their synonyms. For example, both the words `customer` and `client` (which are synonyms) are quite frequently used and we replace all occurrences of `customer` with `client`. We also performed lemmatization of the remaining words in the goal names to identify the root words based on their POS Tagging. For example, all the verbs like `training`, `trains`, `trained` are replaced with their root word `train`. Also, all the nouns like `trainings` and `training` are replaced with their root word `training`.

5.2 Dataset for Co-training based Approach

The dataset for the baseline algorithm will be used as training data for one of the classifiers. Apart from the goal names dataset created above, the co-training uses another view for the second classifier. The dataset for this view is taken from self comments written by the associates. For every goal we take a list of corresponding self comments and process them to form a TF-IDF based word vector.

5.3 Validation

We have performed all the experiments for the role "Developer". The Goal Template for this role contains 41 goals. We grouped "semantically" similar goals with the template so that they get the same class label. This resulted into 21 distinct class labels for the 41 template goals. For validation, we annotated 1000 manually created goals with one of the 21 newly defined class labels or `NONE` for the role "Developer". Henceforth in the paper we will call it as Gold Standard.

Our gold standard dataset consists of 1000 distinct goals where each goal can be assigned to multiple individuals. Total number of assignments for these 1000 distinct goals is $2,010,627$. From academic point of view, validation of classifiers is generally done on distinct records (goals). But in the industrial scenario, the validation in terms of number of assignments is more meaningful. Hence, we report the results on both distinct goals as well as total assignments.

We measure coverage [1], precision, recall, accuracy and f-measure for both the cases and compare them across different algorithms.

5.4 Evaluation Measures

Our classification algorithms predict one of the K positive classes or 1 negative class (`NONE`). The predictions of various algorithms on gold-standard dataset are compared with the manual annotations and a $(K+1)\times(K+1)$ confusion matrix C is defined as follows. The rows of this matrix correspond to the actual labels whereas the columns correspond to the predicted labels. Without loss of generality, we assume that the 0^{th} row and column correspond to the `NONE` label. Any cell C_{ij} of this matrix represents number of goals having i^{th} true label and j^{th} predicted label. Using this confusion matrix C, we compute following measures.

$$C = \begin{matrix} \\ \\ Actual \end{matrix} \overset{Predicted}{\begin{bmatrix} c_{00} & \cdots & c_{0k} \\ \vdots & c_{ij} & \vdots \\ c_{k0} & \cdots & c_{kk} \end{bmatrix}}$$

1. **True Positives (TP):** Number of goals having the predicted non-`NONE` label same as the gold-standard label.

$$TP = \sum_{i=1}^{K} C_{ii}$$

2. **False Positives (FP):** Number of goals having different gold-standard label than the predicted non-`NONE`

[1]Fraction of goals which are not classified as `NONE`

label.

$$FP = \sum_{j=1}^{K} C_{0j} + \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{K} C_{ij}$$

3. **False Negatives (FN):** Number of goals having different predicted label than the non-`NONE` gold-standard label.

$$FN = \sum_{i=1}^{K} C_{i0} + \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{K} C_{ij}$$

4. **Fraction Correct (FC):** Fraction of goals having the predicted label same as the gold-standard label, including `NONE`.

$$FC = \frac{\sum_{i=0}^{K} C_{ii}}{\sum_{i=0}^{K} \sum_{j=0}^{K} C_{ij}}$$

Using above measures, we calculate micro-averaged precision, recall and F-measure.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Similar set of measures ($TP, FP, FN, FC, Precision, Recall$ and $F-measure$) are defined considering number of assignments instead of distinct goals.

5.5 Baseline Experiments

We compare our algorithms with basic similarity measures with given thresholds. We compare it with classifiers based on standard similarities like Cosine, Dice and Jaccard. Modified "Developer" template with 21 distinct class labels is fed as the training data for the different baseline experiments.

Cosine Similarity: Each goal description is represented with a TF-IDF based word vector. Similarity between any two goal descriptions g_1 and g_2 is computed as follows:

$$CosineSimilarity(g_1, g_2) = \frac{\vec{wv}_1 \cdot \vec{wv}_2}{\|\vec{wv}_1\| \|\vec{wv}_2\|}$$

where $\vec{wv}_1$ and $\vec{wv}_2$ are word vector representations of g_1 and g_2, respectively.

Dice & Jaccard Similarity: Each goal description is initially cleaned as explained earlier in the section 5.1 and represented with a set of words contained in the clean description.

$$DiceSimilarity(g_1, g_2) = \frac{2 \cdot |S_1 \cap S_2|}{|S_1| + |S_2|}$$

$$JaccardSimilarity(g_1, g_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

where S_1 and S_2 are set of words representing g_1 and g_2, respectively.

For the similarity based classifiers, we compare the every manually created goal with each template goal and assign the corresponding class label of the most similar template goal. We define thresholds δ_{cosine}, δ_{dice} and $\delta_{jaccard}$ for assigning class labels to only highly similar goals. All the manually created having the maximum similarity lower than the threshold, are assigned the `NONE` class. This increases the precision of the classifier when validated with the Gold Standard dataset.

For completeness of our baseline implementations, we have also used two off the shelf classifiers, Nave Bayes and Maximum Entropy classifiers. Since the Nave Bayes builds on word probabilities and occurrences, we consider all the words when calculating the TF-IDF for the words and thereafter the word vectors. This pre-processing of data is provided so the model for the classes during training phase is built based on the entire corpus rather than the considering the limited set of words in the modified "Developer" template. We use the implementation present in Weka [**?**] to train and predict the classes for the manually created goals.

5.6 Results

We compare our approaches with various baselines by computing the evaluation measures defined in the section 5.4. It can be seen in the Table 2 shows that our approaches have clearly outperformed the baseline methods. The best F-measure is reported by the method 2 whereas method 1 reports the best recall among all the methods. Our methods report significantly better results when number of assignments are considered as shown in the Table 3.

Approach	Precision	Recall	F1	FC
Dice	0.55	0.53	0.54	0.45
Jaccard	0.56	0.53	0.54	0.45
Cosine	0.59	0.52	0.56	0.47
Nave Bayes	0.58	0.50	0.54	0.42
Max-Ent	0.54	0.56	0.55	0.44
Method 1	0.73	**0.67**	0.70	0.61
Method 2	**0.86**	0.60	**0.71**	**0.70**

Table 2: Comparative Performance of all approaches considering distinct goals in the Gold-standard dataset

Approach	Precision	Recall	F1	FC
Dice	0.71	0.68	0.69	0.58
Jaccard	0.71	0.68	0.69	0.58
Cosine	0.70	0.65	0.66	0.56
Nave Bayes	0.58	0.52	0.55	0.45
Max-Ent	0.53	0.49	0.51	0.40
Method 1	0.91	**0.95**	**0.93**	**0.88**
Method 2	**0.94**	0.78	0.85	0.81

Table 3: Comparative Performance of all approaches considering number of goal assignments in the Gold-standard dataset

5.7 Discussion

Method 1 based on Co-training framework achieves more recall and coverage. Unlike method 2, it requires no supervision and makes use of additional information from self comments. Consider the goal description: `Effort in process improvement initiatives`. Method 1 is able to match it correctly to the template goal `Re-engineering saves` even

Template Goal	Matched Manually Created Goal
# of Process / Technical / Domain Related Competencies required for the Role	On the track learning of relevant technologies(as applicable like:Flex,Drool,Java.ETL,Tibco,SQL)
Contribution to Focus Groups	Time spent in implementing account level initiatives per quarter
Number of Consulting Engagements/ New Project Wins thru Innovative Ideas	No. of unsolicited proposals leading to the revenue growth
% SLA compliance/ Remedy Compliance	Root Causes in TTs as per Incident Mgmt SOP. Drive for self and onsite and offshore team.
Number of Consulting Engagements/ New Project Wins thru Innovative Ideas	Number of Demand or Bid Management or Pre- Sales Support (Demos, RFPs) participated and contributed effectively
Succession/ Fluidity Planning	Number of working Backup groomed & cross transitiion achieved with resource optimization

Table 4: Examples of manually created Goals matched with their most suitable Template Goals

though there are no explicit keywords for this template goal in the goal description. Self comments provide the knowledge that `process improvement` is semantically similar to `Re-engineering saves`. Likewise, more keywords are from self comments which results in better recall and coverage.

Method 2 achieves better precision and F-measure (distinct) as compared to method 1 as it uses semantic similarity and weak supervision based on active learning. Consider the goal description: `Contribution of articles, discussions to Knowledge sharing platform`. Method 2 matches it correctly to the template goal `No. of reusable components developed/ deployed` whereas method 1 makes an incorrect match to the template goal `Contribution to Focus Groups`. This is because self comments may introduce some noise which may mislead method 1 in making a wrong prediction.

Table 4 shows some examples of manually created goals for various roles in the organization that are matched to relevant template goals. It can be observed that in spite of not having exact word match with the template goals, our methods were successful in matching manually created goals with appropriate template goals. We have got the results validated from HR domain experts.

5.8 Tuning of Parameters for Method 1 (Co-Training Approach)

Since the co-training based approach has a large number of parameters, it is important to empirically determine the best set of parameter values. For this, we randomly divide the gold standard dataset into 5 parts of 200 goals each. Any one of these parts is treated as a "validation" dataset and a set of parameters leading to the best performance in terms of F-measure (distinct) is chosen. Then the classification performance using the same of parameters is measured on the remaining 800 goals. This is repeated for each part of 200 goals identified as "validation" and all the performance measures reported are averaged.

We have tuned the following parameters (in the Algorithm 2) using the above mentioned tuning process.

1. I : The co-training process stops if no new additions in labelled data happens in any iteration. To limit the training time, a limit I on number of iterations is used.

2. J : In the initial iterations of co-training, it is undesirable to add incorrect predictions to the labelled data. In the initial J iterations where $J < I$, we update the labelled data with the classified data point only if both the classifiers predict the same class label with confidences greater than the respective thresholds, η_1, η_2.

3. η_1, η_2 : These are the confidence thresholds for classifiers C_1 and C_2 respectively to add goals into the labelled data during co-training.

4. ADD_{max}, ADD_{min} : These are used to address the problem of Class Imbalance as described in the section 3.4.

5. η_1', η_2' : During co-training, if the condition of ADD_{min} is not satisfied, we reduce the original confidence thresholds to η_1' and η_2'.

6. $\eta^a, \eta_1^f, \eta_2^f$: After co-training, the final predictions for the unlabelled goals are determined using these thresholds.

The best combination of parameters after tuning is : $I = 5, J = 3, \eta_1 = 0.8, \eta_2 = 0.8, ADD_{max} = 10, ADD_{min} = 1, \eta_1' = 0.3, \eta_2' = 0.4, \eta^a = 0.2, \eta_1^f = 0.4, \eta_2^f = 0.5$

The number of iterations are chosen arbitrarily. We have done empirical analysis to identify a correct threshold for the Maximum number of iterations required. We have noticed that the increasing the maximum number of iterations for training has actually classified more goals. But this reduces the F1-Measure of Distinct Goals significantly. The more number of iterations aren't improving the accuracy. but lower iterations will result in poor coverage. Figure 3 shows change in F-measure (distinct) by varying I and setting values of all other parameters to the identified best combination. Empirical results show that $I = 5$ provides a balance between the trade off parameters.

The limits on the number of goals per class (ADD_{min} and ADD_{max}) that can be added reduces the class imbalance that might be introduced in the training iterations. We have performed experiments by increasing the maximum number of goals that can be added to a class in each iteration. This will increase the fraction of goals that are bound to be classified, but this increase is at the cost of reduction in precision and recall. Figure 4 shows change in F-measure (distinct) by varying ADD_{max} and setting values of all other parameters

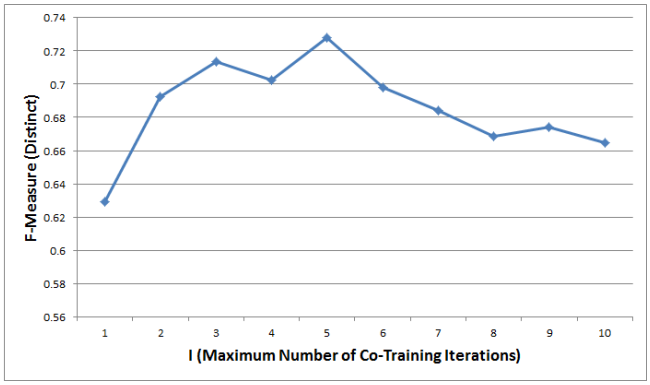


Figure 3: Change in F-measure (distinct) for various values of I (Maximum number of co-training iterations)

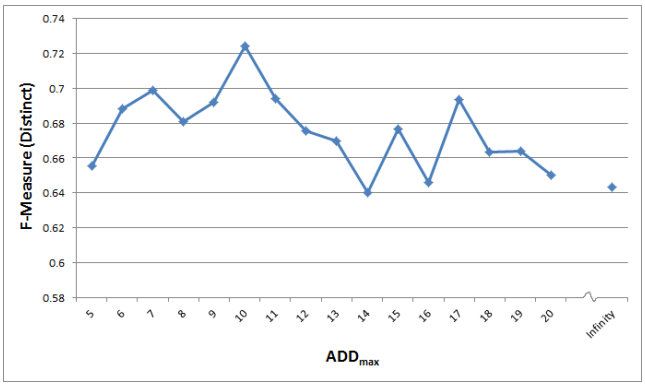


Figure 4: Change in F-measure (distinct) for various values of ADD_{max}

to the identified best combination. It can be seen that the F-measure (distinct) achieves the maximum at $ADD_{max} = 10$ and tends to decrease thereafter.

6. CONCLUSION AND FUTURE WORK

We highlighted the need for standardization of goals in the Performance Appraisal process. We posed this problem as a multi-class classification problem and explored two different approaches with minimal human supervision. The first approach, based on Co-training framework achieves better recall and coverage. This is due to the use of additional knowledge acquired from the self comments. The second approach uses semantic similarity and weak supervision using active learning. It achieves the best precision and F-measure (distinct) as compared to other methods. Both the approaches clearly outperform multiple baseline methods.

From domain perspective, it is important to classify maximum number of manually created goals resulting higher level of standardization of goals. Hence, we also calculated the performance measures based on number of assignments of each goal.

In future, we plan to extend this work to propose new goals to be added to the template by grouping "semantically" similar goals which are not matched to any of the existing template goals.

7. REFERENCES

[1] Apache opennlp. *https://opennlp.apache.org/*.

[2] Jazzy : The java open source spell checker. *http://jazzy.sourceforge.net/*.

[3] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[4] Victoria Bobicev and Marina Sokolova. An effective and robust method for short text classification. In *AAAI*, pages 1444–1445, 2008.

[5] B.G. Buchanan and E.H. Shortliffe. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984. ISBN 978-0-201-10172-0.

[6] Mengen Chen, Xiaoming Jin, and Dou Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781. Citeseer, 2011.

[7] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[8] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998 (ed.).

[9] Gabriel Pui Cheong Fung, Jeffrey X Yu, Hongjun Lu, and Philip S Yu. Text classification without negative examples revisit. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):6–20, 2006.

[10] Rayid Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *ICML*, volume 2, pages 8–12, 2002.

[11] D. Giorgetti and F. Sebastiani. Automating survey coding by multiclass text categorization techniques. *Journal of the American Society for Information Science and Technology*, 54(12):1269–1277, 2003.

[12] Daniel Jurafsky and James Martin. *Speech and Natural Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Education Inc., 2009. ISBN 9780131873216.

[13] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*. Wiley series in Probability and Statistics. John Wiley and Sons, New York, 1990.

[14] Hang Li and Kenji Yamanishi. Mining from open answers in questionnaire data. In *Proceedings of Seventh ACM SIGKDD*, 2001.

[15] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592, 2003.

[16] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE, 2003.

[17] Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2013.

[18] Roberto Navigli. Word sense disambiguation: A

survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[19] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.

[20] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2-3):103–134, 2000.

[21] S. Patil and G. K. Palshikar. Surveycoder: A system for classification of survey responses. In *Proceedings of the 18th International Conference on Application of Natural Language to Information Systems (NLDB 2013), LNCS 7934*. Springer-Verlag, 2013.

[22] S. Patil and B. Ravindran. Active learning based weak supervision for textual survey response classification. In *In Proceedings of 16th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), Part II, LNCS 9042*. Springer, 2015.

[23] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[24] B. Settles. *Active Learning*. Morgan Claypool, Synthesis Lectures on AI and ML, 2012.

[25] Bharath Sriram. *Short text classification in Twitter to improve information filtering*. PhD thesis, The Ohio State University, 2010.

[26] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010.

[27] Aixin Sun. Short text classification using very few words. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1145–1146. ACM, 2012.

[28] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, Upper Saddle River, NJ, 2005.

[29] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[30] Sarah Zelikovitz and Haym Hirsh. Improving short text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the seventeenth international conference on machine learning*, volume 2000, pages 1183–1190, 2000.

[31] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

An Architecture-Oriented Data Warehouse Testing Approach

Neveen ElGamal
Information Systems Department
Faculty of Computers and Information
Cairo University
Egypt
n.elgamal@fci-cu.edu.eg

Ali El-Bastawissy
Information Systems Department
Faculty of Computers and Information
Cairo University
Egypt
aelbastawissy@msa.eun.eg

Galal Galal-Edeen
Information Systems Department
Faculty of Computers and Information
Cairo University
Egypt
galal@acm.org

Abstract

In the past few years, the data warehouse (DW) has regained experts' interest due to the paradigm shift from data storages to data analysis. During the development of DWs data passes through a number of transformations and are staged in multiple storages which might lead to data corruption and/or manipulation. Hence, testing DWs is a vital stage in the DW development life cycle. In this paper, we will present a DW testing approach that is adjustable to fit multiple DW architectures and will present its applicability on three case studies to outline the flexibility and generality of the proposed approach.

1. Introduction

The topic of data warehousing encompasses application tools, architectures, information service, and communication infrastructure to synthesize useful information for decision making from distributed heterogeneous data sources. For this reason, vendors agree that DWs cannot be off-the shelf products but must be designed and optimized with great attention to the customer's situation [20]. Multiple DW life cycle approaches were presented in the literature to discuss how DW systems are built [21, 23]. In those approaches, the architectural design was one of the early and key stages in developing DW systems. On the other hand, testing was not considered in any of the proposed life cycle approaches given that it was always considered in all well-known life cycle approaches like the waterfall and the spiral models. [4, 27]

DW architectural patterns vary from one DW system to another based on user requirements [14]. However, The most common idea in all DW projects is that data is available in one or more data sources and this data needs to be integrated in order to give useful information to assist decision makers to base their decisions on historical behavior of their systems [17].

In the beginning, the data stored in the *Data Sources* (DS) are extracted, transformed, and loaded in the so called *Data Warehouse* (DW). Sometimes this DW is then specialized into a group of business area specific structures each of which contains data that target a specific business area which are called *Data Marts* (DM).

Data passes through several transformations and integration stages before they are loaded from the DSs to the DW or DMs which in most cases force the DW developers to use an intermediary data storage called *Data Staging Area* (DSA); where all the data is transferred to it then transformed and loaded to the DW.

From another perspective, the DW consists of historical data that accumulates years of operational data in one place. Preparing this type of information requires some time, that's why the data stored in the DW are not up to date or even close to that. In some decision making situations, the decision makers want rapid information about data that is not historical, for example; data that is one or two days old. However, they want this type of information to be accumulated from all DSs just like the ones stored in the DW but with less historical dimension. If this type of information is expected to be frequently asked by the decision makers then an *Operational Data Store* (ODS) is required to be part of the DW selected architecture.

Figure 1 shows the most generic and detailed DW architecture that includes most commonly used components and transformations in a DW project. This architecture was proposed under the name of "*Kim-mon Architecture*" which refers to the representation of both Ralph Kimball and Bill Inmon's architectures combined [1]. Data is wrapped from the DSs to the DSA then it

travels to the ODS then to the DW then it is specialized into domain specific DMs then finally it reaches the user/decision maker through User Interfaces (UI) for example; OLAP reports, Analysis, and/or DSS tools.

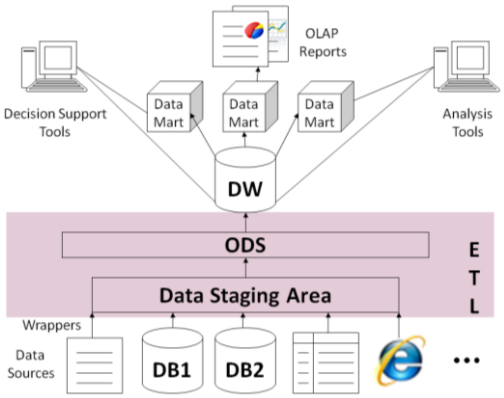


Figure 1. DW Generic Architecture (adapted from [1])

The DW architecture differs from one project to the other based on the specific business requirements. However, the basic component that is available in all DW projects is the DSs. Any other component is included or excluded in the DW project according to the need for it [14]. Variations from the above architecture have been proposed in [19, 14, 21]. These DW architectural patterns simply eliminate or duplicate one of the existing components that are discussed in the Kim-mon architecture. Table 1 summarizes the architectural patterns, discussed in [14], and shows participating components in each.

Table 1. DW Architectural Patterns

Architectural Pattern Name	Architectural Pattern Components
One Layer	DSs→UI
Two Layer	DSs→DW→UI
Independent Data Marts	DSs→DMs→UI
Bus	DSs→DMs→UI
Three Layer	DSs→ODS→DW→UI
Drill-through	DSs→ODS→DW→DM→UI ↓ UI
Hub and Spoke	DSs→ODS→DM→UI
Centralized	DSs→DSA→DW→UI
Federated	DSs→DMs→IL[1]→UI
Kim-mon (Generic)	DSs→DSA→ODS→DW→ DM→UI

Regardless of the architecture of the DW project, data passes through a long way of Extract, Transform, and Load (ETL) processes from its origin in the DSs till it is transformed into information by the UI applications. During this journey data is wrapped, integrated, aggregated, cleansed, loaded, and accumulated which could highly affect the quality of information delivered to the decision makers. Therefore, DW testing is a critical stage in the DW development life cycle which gained multiple researchers' attention to propose a testing technique that is suitable for use in DW projects and provide implementation mechanisms for testing technique to speed the process of testing.

This paper tackles the DW testing from a different perspective. Instead of proposing a testing technique that is suitable for use with a specific DW architecture, this paper proposes a generic DW testing approach and provides an accomodation mechanism that adapts the proposed DW testing approach according to the DW used architecture.

The remainder of this paper will be organized as follows; Section 2 presents a survey on DW testing showing the influence of architectural variations on the DW testing process. Section 3 introduces the generic testing approach that is adequate for use with the Kim-mon architecture. Section 4 describes the technique used to accommodate the proposed testing approach to be adequate for use with other architectures. Section 5 briefly states the implementation details of the accommodation technique. Section 6 discusses findings from applying the proposed technique on several case studies and presents an overall evaluation of the proposed technique. Finally, we conclude our work in section 7.

2. Related Work

Testing DW systems had been studied in literature from various perspectives. Some attempts customized the Software testing strategies to be adequate for use in DW testing [2, 3, 5, 18, 24] while others concentrated on addressing the ETL testing since most of the work is done in the ETL process [7, 35, 22, 26]. A broader view of the DW testing process was studied to address the problem from various perspectives and present to the DW field an integrated solution for DW testing [13, 16, 15, 30, 29, 31, 32, 36, 37, 39, 38]. These approaches were previously studied from the test routine coverage point of view in [11, 10] and it was concluded that none of the existing approaches fully cover the DW testing process. In this paper, we are more concerned with the architectural diversities between the existing approaches and the possibility of generalizing any of the existing approaches to suite several architectures.

By exploring the various DW testing approaches mentioned above, we uncovered considerable diversities between approaches with respect to the architectures that these testing approaches target. From the other perspective, there are many DW architectures defined in literature that needs a DW testing technique to be used in conjunction with them when they are put into operation and yet none of the existing approaches supported these architectures.

[1] IL refers to Integration Layer either Physical or Logical

Table 2 presents a brief comparison allocating existing DW testing approaches to their specified or inferred architectures. The first column displays different architectural patterns, discussed in Table 1, and the second column presents the DW testing approaches along with their architectures that each approach used while describing the DW testing process.

Table 2. Architectural Coverage of DW Testing Approaches

Architecture Name	DW Testing Approaches and Their Used Architectures
One Layer	N/A
Two Layer	[2, 7, 35] DS→DW [24] DS→DW→DM→UI [26] DS→DW→UI
Independent Data Marts	N/A
Bus	N/A
Three Layer	[3] DS→ODS→DW→DM2→ UI [5] DS →ODS→DW→UI
Drill-through	N/A
Hub and Spoke	[15] DS→DSA→DM→UI
Centralized	[22] DS→DSA→DW→DM→UI [29] DS→DSA→DW/DM→DDB3→UI
Federated	N/A
Kim-mon (Generic)	[36] DS→DSA→ODS→DW→ UI

By comparing the architectural components in Table 1 and the architectures in Table 2, we noticed that, few architectures proposed in literature were addressed by the DW testing approaches without further modifications. All the proposed approaches used variations of the defined architectural patterns and customized their DW testing approach based on these variation. It is also shown that some architectures were not addressed by any testing approach like Independent Data Marts, Bus and Federated architectures

What could be concluded from the comparison matrix in Table 2 is that each DW testing approach was defined targeting specific architecture and is therefore adequate for use on DW projects that use the same architecture. If a different architecture is used, then this testing approach will not be adequate for use as it is. Some sort of customization should take place to extend this testing approach to fit the new architecture. This customization could not take place by a testing expert nor a DW expert alone. It is a joint process that should take place benefiting from both experts' knowledge, which is not possible in most cases due to time and budget constraints.

To overcome the weaknesses in the existing approaches, we considered defining a testing approach that is generic enough to be used in multiple DW projects and provide a customization mechanism that is able to accommodate the proposed testing approach to different DW architectures.

Each DW testing approach consists of a group of test routines that describe how this approach tests the DW to improve the quality of the output product. The next section will discuss the group of test routines of the proposed generic DW testing approach.

3. A generic DW testing approach

Test routines defined for DWs are diverse and on different levels of detail, as previously discussed in [10]. To develop a generic DW testing approach that works with different DW architectures, we need to comprehensively determine and describe the complete set of test routines that cover all DW components and transformations. It should also be taken into consideration that these descriptions should be done on a low level of detail to allow later customization for different architectures. For this reason, the Kim-mon architecture presented in Figure 1 will be used for defining DW test routines as it contains all DW components that are most commonly used in all DW architectures.

3.1 Test routine list

Our proposed set of test routines, presented in Table 3, is a refinement of the set of test routines previously presented in [10] when it was used to evaluate and compare the available DW testing approaches. This set of test routines was categorized according to the layer that each test routine targets, the level of detail that this test involves, and when this test takes place. It is worth mentioning that the User Interface layer (DM→UI) will not be part of our proposed solution. The refinements took place to come up with a uniform and consistent set of test routines. These refinements are as follows:

1. Unifying synonymous test routines like Field mapping, Data type compatibility, and Data Layout Format.
2. Removing the *Overall* test routines and define them redundantly on each layer.

As shown in Table 3, the rows represent the layers of the Kim-mon architecture, the columns represent the level of detail that each test routine involves, and test routine periodicity is represented by *italicizing* test routines that are conducted after system development. The underlined test routines are the ones that are redundant on several layers. Introducing *redundant test routine* came from the need to support multiple architectures. When a different architecture is under test the proposed approach will customize the Table 3 to fit the new architecture.

After identifying the set of test routines that are suitable for use in DW projects, it is mandatory to provide the tested with descriptions of these test routines to assist him/her during the testing process. The next section

[2] *Italic* data warehouse components in Table 2 refer to components in the data warehouse architecture that are defined in the proposing approach but not tested.

[3] DDB refers to Dimensional Database

presents the description scheme that we introduced and used to provide descriptions for all test routines inclosed in the test routine list.

Table 3. Proposed DW Test Routines

	Schema	Data	Operation
DS→DSA	1. User requirements 2. Field mapping a. Field naming, b. Data types match, c. Field size match, 3. Correct data selection	*1. Record counts* 2. Threshold test 3. Data boundaries 4. Data profiling *5. Random record comparison* *6. Field to field comparison*	*1. Rejected records* 2. Data access 3. Security
DSA→ODS	1. Schema Design 2. Field mapping a. Field naming, b. Data types match, c. Field size match, d. Data type constraints 3. Aspects of transformation rules a. Captured b. Formula syntax c. Transformation Logic	*1. Record counts* 2. Data integrity a. Identity integrity b. Referential integrity c. Cardinal integrity d. Inheritance integrity e. Domain integrity f. Relationship dependency integrity g. Attribute dependency integrity 3. Parent-child relationship *4. Duplicate detection* 5. Threshold test 6. Data boundaries *7. Data profiling* *8. Random record comparison* *9. Field to Field Comparison* *10. Surrogate keys* *a. Correctness* *b. Integrity*	1. Review job procedures 2. Error logging 3. Performance *4. Rejected record* 5. Data access 6. Forced Error test 7. Stress test 8. Security
ODS→DW	1. User requirements coverage 2. DW conceptual schema: a. Conformed hierarchy b. Understandability c. Usability d. Mapping to logical model 3. DW logical model: a. Mapping to physical model b. Functionality c. Performance (comply with MDNF) 4. Integrity constraints 5. Hierarchy level integrity 6. Granularity 7. Derived attributes checking 8. Field mapping a. Field naming, b. Data types match, c. Field size match, d. Data type constraints	*1. Record counts* 2. Data integrity a. Identity integrity b. Referential integrity c. Cardinal integrity d. Inheritance integrity e. Domain integrity f. Relationship dependency integrity g. Attribute dependency integrity 3. Parent-child relationship *4. Duplicate detection* 5. Threshold Test 6. Data boundaries 7. Data profiling *8. Random Record Comparison* *9. Field to field comparison* 10. No constants loaded 11. No Null records loaded 12. Simulate data loading 13. Data aggregation 14. Reversibility of data from DW to DS *15. Confirm all fields loaded* 16. *Data freshness*	1. Review ETL documentation 2. ETL test a. ETL activity ordering b. ETL recoverability c. Job sequence d. Error propagation through jobs e. Job resetting f. Batch failure propagation g. Batch reset in case of failure 3. Scalability 4. Initial load *5. Incremental load* 6. Data access *7. Rejected record* 8. Performance 9. Error logging 10. Forced error test 11. Stress test 12. Security 13. HW and SW configuration
DW→DM	1. Schema Design 2. Calculated members 3. Irregular hierarchies 4. Correct data filters 5. Additivity guards	1. Measure Aggregation	1. Security 2. HW and SW configuration

3.2 Test routine description scheme

Test routines listed in Table 3 are a group of test routines that are refined and few of them were introduced to the DW testing process. To be able to use these test routines they need to be fully described. The full description will not appear in this section. However, The description scheme that we introduced to broadly describe all test routines is as follows:

1. Name: The common name used in testing field for this test routine.
2. Layer: Which layer of the Kim-mon architecture does this test routine take place?
3. Level: What is being tested in this test routine? (Schema/Data/Operation)
4. Objective(s): a textual description of the test routine showing its objective(s).
5. Type: The type of the test routine: (Verification/Validation)
6. Severity: The importance of this test routine (Mandatory, Recommended, Optional)
7. Periodicity: How often does this test routine take place? (Schema Change/Data Load)
8. Part Under Test: which part of the component under test is being tested (ex: Schema, Table, Attribute, etc…)
9. Input(s): Required documents that need to be available to conduct this test routine (if any).
10. Testing Scenario: The detailed description of how this test routine is conducted.
11. Automation: the possibility of automating this test routine and the type of automatic assistance required, (Testing Tool, Data Generation Tool, Test Case Generation Tool)

Each test routine was described using the above scheme. Required information about each test routine was gathered from existing testing approaches and few of them were defined from scratch. For example; the test routine named *Duplicate Detection* is described in Table 4 using the aforementioned scheme.

Table 4. Duplicate Detection Test Routine Description

Name: Duplicate Detection
Layer: DSA→ODS
Level: Data
Objective(s): Confirm that no duplicate records exist in the ODS
Type: Verification
Severity: Mandatory
Periodicity: Data Loads
Part Under Test: Every Table in the Destination
Input(s): None
Testing Scenario:

There are two types of duplicated that needs to be detected and resolved:

I. Duplicates resulting from incorrect data transformation procedure.
II. Duplicates resulting from integrating data from different data sources.

This first type of duplicates could be detected as follows:

1. Run a query on each destination table to retrieve duplicates. An example of this query could be as follows:

```
Select *
From <TableName>
Group by <AllAtributes>
Having count(*) >1.
```

2. If this query returns any results, it means that there are duplicate records in this table and these duplicates are a result of an incorrect transformation process

The second type of duplicates could be detected by applying one of the duplicate detection techniques that have been severely studied in science to solve the problem of duplicate detection and resolution in integrated data. [12]

Automation: Testing Tool is required

All test routines displayed in Table 3 were described using the same scheme, used in Table 4, to provide the testers with some sort of instruction manual for DW

testers, available at [8, 9], supplying them with any required information regarding the process of DW testing. However, this test routine description is adequate for use with the Kim-mon architecture only. If another architecture is used in a DW project, these test routines need to be adapted to be adequate for use with the used architecture. This paper proposes a customization mechanism in the next section.

4. Multiple architectural accomodation

As it was previously discussed in section 1, DW components are the interrelated parts of the DW architecture that are connected together to transform data in DSs into information. Moreover, by studying the available architectural patterns discussed in literature, it was notable that all the architectural components are always used in the same order (DS, DSA, ODS, DW, DM) if they are part of the selected architecture.

Since we are concerned with testing DWs with different architectures, then the above mentioned set of test routines that are defined on the Kim-mon architecture need to be customized in a way to fit different architectures.

Each test routine stated in Table 3 is mapped to a specific DW layer. Each layer consequently relates this test routine to two DW components (source and destination components). For example, the test routine "Duplicate Detection" presented in Table 4 is defined on the DSA→ODS layer, hence, relates this test routine to both layers the DSA and ODS. However, this test is concerned with detecting duplicates that exist between the ODS records and it is not concerned with the DSA by any means. On the Contrary, the test routine "Record Counts", defined on the DS→DSA layer in Table 3, requires the participation of both the DS and the DSA in the test routine in order to compare record counts and confirm that they are matching.

So, relating test routines to DW layers only will not help in the process of test routine customization to multiple architectural patterns because each test is not related explicitly to each DW component. For this reason, adding more descriptive attributes to the test routine description scheme discussed in the previous section is needed to include in the test routine description the prerequisite components that this test routine involves or requires.

Two prerequisite attributes need be specified for each test routine; one is a source prerequisite and the other is a destination prerequisite. Depending on the objective of each test routine and the role of the DW component with respect to the test routine. Whether it is the source of the data being transformed, or the component that receives the data. These two attributes are not mandatory to all test routines. Some test routines might require both attributes to be specified like the Record Counts test routine discussed above because its objective is to compare results between the source and destination. While another test routine requires only one prerequisite, either the source or the destination, because they check one component's consistency or its validity with respect to some other parameter like user requirements or business rules. Example for these test routines is the "Duplicate Detection" and "User Requirements". The two attribute templates are as follows:

Prerequisite(s):

- Source: <ComponentName>
- Destination: <ComponentName>

From another perspective, Test routines stated in Table 3 could be clustered according to the purpose of each. Some of them are concerned with the successful transformation of data from the data sources through all the DW system data storages till it reaches the user. Examples of these test routines are Record Counts, Duplicate Detection, Data Boundaries, Error Logging, etc. While others are concerned to experiment a specific functionality that is served by a specific DW component. For example, DW conceptual schema, DW Logical Schema, and Measure Aggregation. This type of clustering needs to be taken into consideration while defining each test routine, because when a DW component is not part of the architecture used, then these test routines need not be considered in the proposed test routine list. For this reason, an extra attribute named "Single Layer Test (SLT)" is assigned the value 1 for test routines that experiment specific functionalities to be able to differentiate between the purpose of each test routine. The template of the SLT attribute is as follows:

SLT: <Binary>

Till this point, all test routines have been mapped to its proper components and all desired information about each test routine is available in the test routine description over the Kim-mon architecture. But, when the architecture under test is a different architecture, a mapping technique needs to be defined to re-direct the test routines that refer to a DW component that is not part of the architecture under test to another component that is present in the given architecture. This technique is defined as follows;

Using the extra two attributes that define the prerequisites, each test routine is related to one or more DW component each of which acts as a source or destination prerequisite. When the test routine is related to a prerequisite DW component that is not part of the architecture under test, alternative component needs to take over the place of the absent component and this test routine will be conducted on the alternative component instead to guarantee a proper transformation of data between participating DW components. This rule does not apply to test routines whose attribute SLT takes the value 1 since these test routines are testing a specific functionality of the prerequisite component that is currently not part of the architecture used. Therefore, no

alternative component will perform this specific functionality and consequently the test routines testing it need not be taken into consideration.

Choosing the suitable alternative component is a decision that is taken based on the type of the absent component, whether it was a source or destination prerequisite with respect to the test routine. If it was a source prerequisite, the preceding alternative needs to be chosen as the absent component's replacement, and if it was a destination prerequisite, the succeeding alternative will be chosen. Figure 2 presents the succeeding and preceding alternatives for all possible DW components.

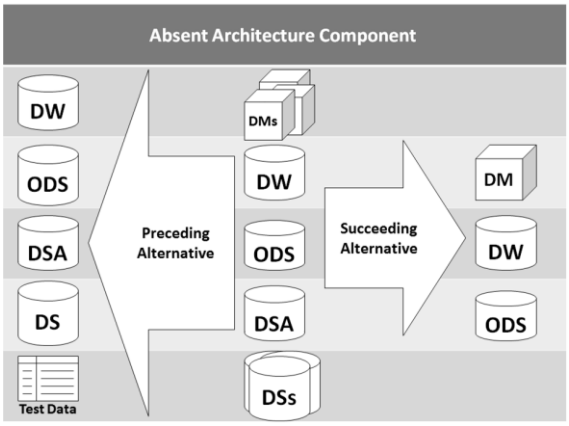


Figure 2. DW Component Alternatives

It is quite common in the architectures discussed earlier in section 1 that more than one consecutive component could be absent from the DW architecture with respect to the generic (Kim-mon) architecture presented in Figure 1. To find a suitable alternative for test routine's prerequisites, transitivity is applied on the alternative relationship defined above. The relationship is transitive in the sense that if an absent component is a preceding or a succeeding alternative of another absent component, then find the alternative component's alternative to replace it and assign it to the specified test routine.

To determine the alternatives for each absent prerequisite the following steps will take place:

1. For each test routine that is not a Single Layer Test, determine whether its prerequisites are absent or present components. Since Single Layer Tests need not be mapped on any other layers because they target functionalities that are specific to their prerequisites and when these prerequisites are not part of the architecture under test, these test routines will not be part of the customized test routine list.
2. For each absent prerequisite, determine its preceding or succeeding alternative components, depending on the type of the prerequisite relation whether this component is a source or a destination prerequisite to this test routine.
3. For each preceding alternative component, if it is also an absent component, then get its preceding alternative. Repeat this sequence until the transitivity rule leads to an alternative that is not an absent component.
4. For each succeeding alternative component, if it is also an absent component, then get its succeeding alternative. Repeat this sequence until the transitivity rule leads to an alternative that is not an absent component.

This technique will assign test routines to DW components that are part of the architecture under test to be able to test it properly.

4.1 Example

If the architecture used for the DW is the Two Layer architecture, presented in Table 1, which consists of DSs, DW, and DMs. Then the two components DSA and ODS are considered absent components with respect to the generic (Kim-mon) architecture presented in Figure 1.

By applying the test routine customization on the test routines presented in Table 3 and re-directing test routines to their suitable prerequisite alternatives as discussed previously the outcome of this process will result in the set of test routines presented in Table 5 that was customized to the Two Layer architecture not by removing the two layers DS→DSA and DSA→ODS but redirecting their test routines to the appropriate alternative layers.

Table 5. Test Routines for the Two-Layer Architecture

	Schema	Data	Operation
DS→DW	1. User requirements 2. Field mapping a. Field naming, b. Data types match, c. Field size match, d. Data type constraints 3. Correct data selection 4. Schema Design 5. Aspects of transformation rules a. Captured b. Formula syntax c. Transformation Logic 6. User requirements coverage 7. DW conceptual schema: a. Conformed hierarchy b. Understandability c. Usability d. Mapping to logical model 8. DW logical model: a. Mapping to physical model b. Functionality c. Performance (comply with MDNF) 9. Integrity constraints 10. Hierarchy level integrity 11. Granularity 12. Derived attributes checking	1. Record counts 2. Threshold test 3. Data boundaries 4. Data profiling 5. Field to field comparison 6. Random record comparison 7. Parent-child relationship 8. Duplicate detection 9. Surrogate keys a. Correctness b. Integrity 10. Data integrity a. Identity integrity b. Referential integrity c. Cardinal integrity d. Inheritance integrity e. Domain integrity f. Relationship dependency integrity g. Attribute dependency integrity 11. No constants loaded 12. No null records loaded 13. Simulate data loading 14. Data aggregation 15. Reversibility of data from DW to DS 16. Confirm all fields loaded 17. Data freshness	1. Rejected records 2. Data Access 3. Security 4. Review job procedures 5. Error logging 6. Performance 7. Forced Error test 8. Stress test 9. Review ETL documentation 10. ETL test a. ETL activity ordering b. ETL recoverability (Robustness) c. Job sequence d. Error propagation through jobs e. Job resetting f. Batch failure propagation g. Batch reset in case of failure 11. Scalability 12. Initial load 13. Incremental load 14. HW and SW configuration
DW→DM	1. Schema Design 2. Calculated members 3. Irregular hierarchies 4. Correct data filters 5. Additivity guards	1. Data Aggregation	1. Security 2. HW and SW configuration

5. Implementation

To structure and keep track of this large amount of information required for test routines and the relationships between test routines and their prerequisite components, It was mandatory to store these details in a structured yet flexible format to accommodate any future changes that might take place; like adding test routines, modifying test routine descriptions, and/or deleting unnecessary test routines.

The proposed test routine description and customization mechanisms were implemented using the graph database Neo4j [25]. We chose the graph database because of its flexible structure that could evolve through time without affecting stored information. It is also distinguished to have a very simple yet powerful querying language called *Cypher* that is used as both data definition and data manipulation language.

Using Cypher query, we have prepared a data definition script that fully defines and fills the graph database with all test routine definitions and relationships. Another Cypher query template was defined to accommodate the graph database according to the architecture under test. Finally, a third Cypher query was defined to generate from the customized graph database a detailed test routine list clustered according to levels and layers as the one displayed in tables Table 3 or Table 5 when the architecture under test is the Kim-mon architecture or the Two-Layer Architecture, respectively.

6. Case studies and evaluation

The proposed approach claims to provide a testing technique that is adjustable according to the architecture of the DW system under test. For this reason, it was mandatory to experiment with it using different case studies with different DW architectures.

The experimentation mechanism we used to apply the proposed customization technique was getting access to abstract information about the DW under test, proposing the set of test routines adequate for the given architecture, and getting the feedback from the DW testers regarding the proposed test routine list. On the other hand, we considered studying the testing technique used in the DW under test (if available) and compared it with our proposed test routine list.

During the selection of the case studies, we were keen to find case studies with different architectures and to choose companies in different sizes. The proposed approach was applied to three case studies from three different sized companies;

1. CentriVision: a small sized Egyptian company, founded in 2003, whose development services involves business intelligence solutions.[6]
2. SMSMT: a medium sized Australian company, founded in 1986, that provides testing as one of its services.[28]
3. Teradata: a large sized American company, founded in 1979, that sells analytic data platforms, applications and related services. [34]

Each of these companies was using a different DW architecture in the DW project they supplied us with, except for Teradata since it uses a generic architecture for all its projects. The architectures of the three case studies are displayed in Figure 3.

The results concluded from applying the proposed accommodation mechanism to customize a test routine list for each case study is presented separately in the following three sections. Each section will present a comment on the adequacy of the proposed approach when applied to one of the case studies.

6.1 CentriVision

According to the DW development/testing team at CentriVision, testing in this DW project was conducted using team members' experiences. There was no standard testing technique used. However, a set of tests takes place at different levels of detail to guarantee the quality of the DW under development. The categorization of CentriVision's undocumented testing activities is displayed in Table 6 categorized by layers and levels they apply to with respect to the DW architecture used.

Table 6: CentriVision Test Routine Categorization

Level / Layer	Schema	Data	Operation
DS→DM	• User Requirements Coverage • DM Schema Design • Field mapping in Transformation rules	• Record Counts • Data Aggregation • Calculated members • No null measures exists • Duplicate detection • Simulate data loading • Data freshness	• HW and SW configuration • Scalability of data • Performance Test • Review Job Procedures
DM→UI	User requirements coverage	-	-

By communicating the proposed test routine list with a project manager at CentriVision the feedback was as follows:

1. Most recommended test routines that were proposed in the test routine list are usually conducted either directly by snooping for mismatches in the migrated data or indirectly during the process of defect tracking.
2. Supplying them with the customized test routine list is of great help to give them ideas about what needs to be tested and how these tests could be conducted rather than depending on tester's experience and jeopardize the DW quality.
3. Regarding the test routines they did not support, their feedback was that it would highly increase the quality of their output products if taken into consideration during the testing process.

From our point of view, depending on the tester's experience in testing the system is not a reliable way of finding errors. Having a consistent and well documented testing strategy that could be used as an instruction manual for the tester to follow during the testing process could highly assist the tester in knowing the possible vulnerabilities that could take place in the DW and testing the system to prevent it against possible threats. Applying the testing technique that is defect oriented; where the tester follows defects to fix the errors, is not the best way to find and fix errors. It could be possible that errors exist in the system, but the right test was not conducted to reveal them.

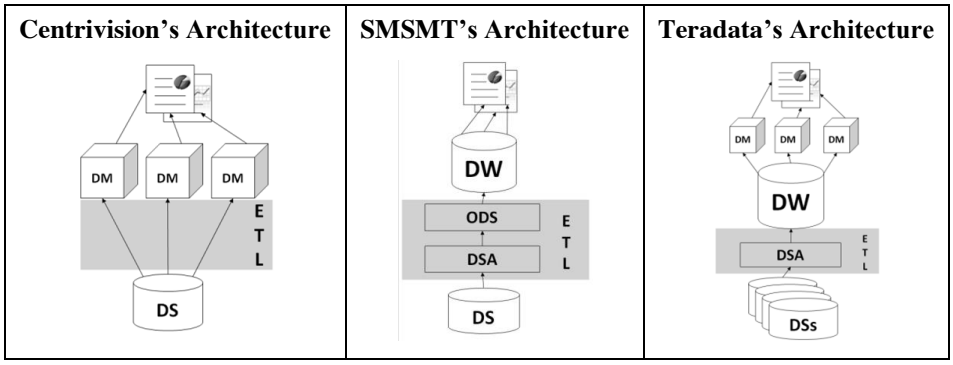


Figure 3. Architectures of the Case Studies

6.2 SMSMT

According to a Testing and Quality Assurance consultant at SMSMT, the company agreed to use a set of test routines in their current project of developing a DW for a Legacy system. Table 7 presents the test routines used in this case study. The set of test routines used in this case study are mostly data tests which involve simulating and comparing data transformations between different data storages.

Table 7. SMSMT Test Routine Categorization

Level / Layer	Schema	Data	Operation
DS→DSA	Field Mapping	• Not nullable fields • Record counts • Field to field comparison • Simulate data loading	Delta load test
DSA→ODS	• Field mapping • Transformation rules test	• Record counts • Field to field comparison • Simulate data loading • Domain Key Metric test	Delta load test
ODS→DW	• Field mapping • Transformation rules test	• Record counts • Field to field comparison • Simulate data loading • Domain Key Metric test	• Initial load test • Incremental load test
DW→UI	-	• Result comparison across data sources	-

By communicating the proposed set of test routines customized for the SMSMT case study, the feedback of the test and quality assurance consultant was as follows:

1. The proposed set of test routines is quite comprehensive for a one-time load, however, it misses the tests for delta loads on several layers.
2. Some test routines that we proposed needs to be conducted on different layers like the performance and stress tests needs to take place between DS and DW not only ODS and DW.
3. The proposed set of test routines lacks the consideration of timeliness of test routines to guard against the execution of test routines that might cause conflicts in the data if run at the same time.

The comparison between the proposed and the used set of test routines showed that 90% of the test routines used in SMSMT case study are included in the proposed set of test routines which confirms the soundness of the proposed approach and proves its adjustability to a different architecture.

6.3 Teradata

According to the interview conducted with the test manager at Teradata Egypt, there exists a documented, well defined, and formulated testing strategy that is used in all DW projects in Teradata, yet it is still in the editing phase [33]. The basic modules of tests were interpreted from the testing strategy and was categorized according to layers and levels of the DW architecture. Table 8 presents the test routines introduced in the Teradata testing strategy.

As it is shown in Table 8, all test routines used by Teradata are tests that take place before system delivery. Any post delivery tests are conducted by the quality assurance team based on the customer's reporting of errors or inconsistencies on the output data. The Teradata testing strategy is also notable for its coverage on all three testing levels (Data, Schema, and Operation), which highly enriches the quality of their DW products.

Table 8. Teradata Test Routine Categorization

Level / Layer	Schema	Data	Operation
DS→ DSA	• User Requirements • Data types • Columns order	• Record counts • *Pattern counts* • Data profiling • Random record comparison	• *All sources are loaded* • Rejected records
DSA→DW	• Field mapping • Review Logical schema for customizations • *Naming Conventions*	• Record Counts • Field to field comparison • Random record comparison • Primary key index integrity • Surrogate keys integrity and correctness • Referential integrity • Domain integrity • *History integrity* o *Reverse History* o *History Overlap* o *Null History* o *History Gap* o *Open end History*	• Rejected records • Performance test • *ETL Scheduling*
DW→DM	• Schema Design	• Record counts • Field to field comparison • Data aggregation • *Business Rules Integrity*	• Security • HW and SW configuration
DM→UI	• User Requirements Coverage	• Business logic testing	

A comparison took place between the Teradata testing strategy and the proposed test routine list customized on the Teradata architecture. Table 9 presents a numerical reference, for the number of test routines that Teradata testing strategy takes into account from the proposed test routine list. For example, On the DSA→DW layer at the Data level, the Teradata testing strategy considers 8 of the proposed 17 test routines and applies an extra 5 test routines to test this layer of the DW.

This comparison revealed that, in general, the proposed test routine list agrees with the Teradata testing strategy in 40% of the proposed test routines. This is because the proposed set of test routines contains all possible test routines that could be used to test any DW, however; the proposed test routines should not be all conducted on any system. Some of them are alternative to each other which leaves to the tester the option of choosing among them.

Table 9. Teradata Vs Proposed Testing Strategy

Level / Layer	Schema	Data	Operation
DS→ DSA	1/4	3/6 +1	1/3 +1
DSA→DW	2/9 +1	8/17 +5	7/14 +1
DW→DM	1/5	1/1 +1	1/2

According to the analysis abstracted in Table 9, it is clear that the Teradata testing strategy agrees with the proposed test plan to a great extent on the *data* level comparisons. On the *schema* level, however, the Teradata testing strategy lacks the support of most of these types of tests. This was due to their reliance on the data tests that will reveal any possible schema or structural problems or inconsistencies.

What Table 9 revealed as well was that the tests on the *operational* level like scalability, security, or stress were not conducted on a project basis. This is because they use the Teradata Database Management System which is extensively tested from these perspectives and results are guaranteed if proper HW and SW configurations took place.

On the other hand, to fairly compare the two testing strategies, Table 9 shows that on certain levels and layers the Teradata testing strategy supported some test routines that were not part of the proposed test routine list. These test routines are *Italicized* in Table 8. By studying these test routines we see that including these test routines in our proposed testing technique would be a good addition to it.

From a different angle, after communicating the proposed detailed test routine list with the Test Lead and a Testing Developer at Teradata Egypt, we were able to get their feedback regarding the proposed set of test routines. Their comments were as follows:

1. The proposed set of test routines could be considered as the standard suite of tests required for system test a DW solution.
2. It is satisfactory for the technical requirements of testing a DW solution.

From their point of view, what lacks the proposed testing strategy is reference to commercial tools that could be used to automate each test routine, showing stakeholders involvement in each test routine and absence of the test routines supported by Teradata and not included in the proposed testing strategy as discussed previously.

In spite of the above drawbacks, due to the flexibility of the graph database, they could be easily overcome by introducing the required modifications on the graph database with minimal change in the Cypher query graph creator script.

6.4 Overall evaluation

As discussed previously in section 02 the main drawback of existing testing approaches was their rigidity with respect to the architecture of the DW. Each approach assumed that the architecture used in their approach is the DW architecture mostly used and proposed a testing strategy for it. What distinguishes our proposed approach is its flexibility of adapting the test routines according to the architecture under test unlike other approaches.

Table 10 presents possible architectures that the proposed test routine list could be customized for. The proposed approach was the first approach that covers testing DWs with different architectures and not only supported well-known architecture types discussed in the literature, but also supported other DW architectures that are sometimes used but it was not named in the literature.

Table 10. Proposed Framework Architectural Coverage

Architectural Pattern Name	Proposed Approach's Coverage
One Layer	X
Two Layer	√
Independent Data Marts	√
Bus	√
Three Layer	√
Drill-through	√
Hub and Spoke	√
Centralized	√
Federated	X
Kim-mon (Generic)	√
DS→DSA→DW→DM	√
DS→DSA→DM	√

Architectures not supported in the proposed approach are the ones that involve a special nature layer which is not commonly used in DWs. For example, the Single Layer architecture integrates data virtually through a group of on-the-fly transformation rules. Neither integrated data are stored nor are historic data available for any decision making processes. This type of DWs requires a custom made test plan that could not be provided given the proposed test routine list. The same rule applies for the Federated

architecture, where the Integration Layer is a special layer that needs a custom made testing technique. However, our proposed test routine list is adequate for testing the transformation of data from the DSs to the federated DMs smoothly.

What needs to be clarified regarding the architectural coverage is that the proposed approach does not present test routines that targets the layer of the UI which is a drawback that affects all supported architectures. However, The UI Layer testing were beyond our scope and excluding it was based on expert's recommendation because there are different means of UIs like reports, charts, DSS tools, and Analytical tools; where, each of which required a different testing technique for their verification and validation.

7. Conclusion and future work

What distinguishes this study from any other proposed solution for the issue of DW testing is the comprehensiveness of the description scheme that was used to describe all test routines, studying DW architectures to interpret means of relating architectural components in order to fulfill the aim of providing the DW testers with a generic solution for DW testing adequate for use in multiple projects with different architectures, benefiting from other people's work by integrating the proposed solution with their work to come up with a detailed technique for DW testing with minimum effort and maximum gain, and finally, considering future system modifications by using a graph database to store the test routine descriptions to be easily extended to contain various system additions or changes.

Future work in this area could be categorizing test routines according to the well known software testing phases, namely; Unit testing, Integration testing, System testing, and User Acceptance testing. Since it has been widely agreed upon that the testing phase is categorized into the aforementioned test phases, it would be of great help to the testing team to suggest for them test routines given in the categorization scheme they are familiar with.

Another possible extension to the proposed work could be including the group of test routines targeting the layer of User Interfaces since it has been skipped, though our research.

Test routines suggested in this paper are all possible test routines. Not all of them need to be conducted on any DW to test it. Consequently, testers are given the opportunity to choose the proper set of test routines to conduct on their DW. Hence, providing the testers with enough information about relationships between test routines and possible dependencies that might take place between them might help them choose the proper set of test routines without sacrificing their system's quality.

In the end, we would like to conclude that the proposed architecture-oriented testing approach may not be the ultimate solution for DW testing, however, it has gained multiple user's trust. It is powerful for its flexibility and might be adequate for use by small to medium sized DW development companies that do not have standardized or comprehensive DW Testing frameworks.

8. Acknowledgements

We would like to thank the companies who supported us with time and resources to be able to experiment with the proposed architecture-oriented DW testing approach (Teradata Egypt, Centrivision, and SMSMT).

9. REFERENCES

[1] R. Abellera, *Data Warehouse Architectures: Overview of the Corporate Information Factory and the Dimensional Modeling*, The Data Warehouse Institute, 2010.

[2] C. Bateman, *Where are the Articles on Data Warehouse Testing and Validation Strategy?*, *Information Management*, 2002.

[3] S. Bhat, *Data Warehouse Testing - Practical*, *Stick Minds*, 2007.

[4] B. Boehm, *A Spiral Model of Software Development and Enhancement*, 21 (1988), pp. 61-72.

[5] K. Brahmkshatriya, *Data Warehouse Testing*, *Stick Minds*, 2007.

[6] CentriVision, *www.Cenrivision.com*, 2003.

[7] R. Cooper and S. Arbuckle, *How to Thoroughly Test a Data Warehouse*, *Software Testing Analysis and Review (STAREAST'02)*, Orlando, Florida, 2002.

[8] N. ElGamal, *Data Warehouse Test Routine Descriptions*, *Technical Report*, www.researchgate.net/publication/289729988_Data_Warehouse_Test_Routine_Descriptions, 2016.

[9] N. ElGamal, *Data Warehouse Testing*, *PhD. Thesis, Faculty of Computers and Information*, Cairo University, 2015, Appendix B, pp. 193-228.

[10] N. ElGamal, A. ElBastawissy and G. Galal-Edeen, *Data Warehouse Testing*, *Proceedings of the Joint EDBT/ICDT PhD Workshop*, ACM, Genoa, Italy, 2013.

[11] N. ElGamal, A. ElBastawissy and G. Galal-Edeen, *Towards a Data Warehouse Testing Framework*, *Proceedings of the 9th International Conference on ICT and Knowledge Engineering (ICT&KE'11)*, IEEE, Bangkok, Thailand, 2011, pp. 67-71.

[12] A. K. Elmagarmid, P. G. Ipeirotis and V. S. Verykios, *Duplicate Record Detection: A Survey*, IEEE Transactions on Knowledge and Data Engineering, 19 (2007), pp. 1-16.

[13] M. Golfarelli and S. Rizzi, *A Comprehensive Approach to Data Warehouse Testing*, *Proceedings of the ACM 12th international workshop on Data warehousing and OLAP (DOLAP'09)*, ACM, Hong Kong, China, 2009, pp. 17-24.

[14] M. Golfarelli and S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies*, McGraw Hill, 2009.

[15] M. Golfarelli and S. Rizzi, *Data Warehouse Testing*, International Journal of Data Warehousing and Mining, 7 (2011), pp. 26-43.

[16] M. Golfarelli and S. Rizzi, *Data Warehouse Testing: A prototype-based methodology*, Information and Software Technology, 53 (2011), pp. 1183-1198.

[17] J. Guerra and D. Andrews, *Why You Need a Data Warehouse*, www.rapiddecisions.net, Copyright Andrews Consulting Group, Inc., 2011.

[18] S. L. Gupta, P. Pahwa and S. Mathur, *Classification of Data Warehouse Testing Approaches*, International Journal of Computers and Technology, 3 (2012), pp. 381-386.

[19] W. H. Inmon, *Building the Data Warehouse*, Wiley Comp., 1996.

[20] M. Jarke, M. Lenzerini, Y. Vassiliou and P. Vassiliadis, *Fundamentals of Data Warehouses*, Springer-Verlag New York, Inc., 2001.

[21] R. Kimball, L. Reeves, W. Thornthwaite and M. Ross, *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*, John Wiley \& Sons, Inc., 1998.

[22] M. P. Mathen, *Data Warehouse Testing*, *Infosys Technologies Limited*, 2010.

[23] R. Mattison, *The Data Warehousing Handbook*, XiT Press, Oakwood Hills, Illinois-USA, 2006.

[24] A. Munshi, *Testing a Data Warehouse Application*, *Wipro Technologies*, 2003.

[25] Neo4j, *Neo4j Graph Database*, 2013.

[26] V. Rainardi, *Testing your Data Warehouse*, *Building a Data Warehouse with Examples in SQL Server*, Apress, 2008.

[27] W. W. Royce, *Managing the development of large software systems: concepts and Techniques*, *Proceedings of the 9th International Conference on Software Engineering (ICSE'87)*, Monterey, California, USA, 1987, pp. 328-338.

[28] SMSMT, *SMS Management and Technology* www.smsmt.com, 1986.

[29] P. Tanuška, O. Moravčík, P. Važan and F. Miksa, *The Proposal of Data Warehouse Testing Activities*, *Proceedings of 20th Central European conference on Information and Intelligent Systems*, Varaždin, Croatia, 2009, pp. 7-11.

[30] P. Tanuška, O. Moravčík, P. Važan and F. Miksa, *The Proposal of the Essential Strategies of Data Warehouse Testing*, *Proceedings of 19th Central European Conference on Information and Intelligent Systems (CECIIS'08)*, 2008, pp. 63-67.

[31] P. Tanuška, P. Schreiber and J. Zeman, *The Realization of Data Warehouse Testing Scenario*, *proizvodstvo obrazovanii*. *(Infokit-3) Part II: 3 meždunarodnaja nature-techničeskaja konferencija*., Stavropol, Russia, 2008.

[32] P. Tanuška, W. Verschelde and M. Kopček, *The proposal of Data Warehouse Test Scenario*, *Proceedings of European conference on the use of Modern Information and Communication Technologies (ECUMICT'08)*, Gent, Belgium, 2008.

[33] Teradata, *Teradata Testing Strategy*, 2014.

[34] Teradata, www.teradata.com, 1980.

[35] J. Theobald, *Strategies for Testing Data Warehouse Applications*, *Information Management*, 2007.

[36] D. Vucevic and W. Yaddow, *Testing the Data Warehouse Practicum - Assuring Data Content, Data Structures and Quality*, Trafford, 2012.

[37] D. Vucevic and M. J. Zhang, *Testing Data Warehouse Applications*, Trafford Publishing, 2011.

[38] W. Yaddow, *Conducting end-to-end testing and quality assurance for data warehouses*, *IBM Data Magazine*, 2013.

[39] W. Yaddow, *Enriching data warehouse testing with Checklists*, *IBM Data Magazine*, 2013.

Supervised Learning in Matrix Completion Framework for Recommender System Design

Anupriya Gogna, Angshul Majumdar

IIIT-Delhi, Delhi
INDIA
anupriyag@iiit.ac.in, angshul@iiitd.ac.in

Abstract

Recommender systems primarily utilize the, highly sparse, explicit rating information to make relevant predictions. This data scarcity places a limit on the accuracy of prediction. In this work we attempt to alleviate the problem of data sparsity by using secondary information. Most existing works incorporate auxiliary information in a (bi-linear) matrix factorization setup; whilst our model is based on a (convex) matrix completion framework. In this work, we use auxiliary information about users and items to impose additional constraints on the recovered rating values; adopting ideas from supervised learning. Alongside, we also propose a method to utilize the information map extracted from supervised learning approach to handle the cold start problem. Most works that address the cold start problem are focused on users with very few ratings - this is not the pure cold-start problem. However, in this work we target new users and items which have no ratings available for them; and only has the associated metadata. We propose an algorithm using split Bregman technique for solving our formulations. Comparison of our design with existing state of the art methods for RS design on the movie recommender systems clearly indicate the superiority of our formulation over existing methods.

1. Introduction

Today recommender systems (RS) [1,2] are the workhorse behind all Business-to-Client eCommerce portals. To facilitate the user, a recommender system predicts the user's choices and suggests a handful of items; if the prediction is good the user buys it. The importance of accurate recommendation and hence the focus on building efficient RS is very clear - better the prediction, more is the revenue for the portal.

RS largely rely on some form of feedback provided by users on a subset of items, such as purchase information, like/dislike options or explicit rating data, to predict the ratings on yet unrated items. Gathering this information involves a user's active participation, either by means of purchase or some form of interviewing process (like seeking user's rating on a selected set of items), which is not always a plausible scenario. Lack of this preference information, especially in case of new users registering on the system can be major bottleneck in improving customer satisfaction. It is essential for RS to provide satisfactory suggestions to such (new) users as well, failing in which can cause potential loss of customers and revenue.

In absence of any explicit predilection information, the rating prediction for new users (user cold start problem) can be based on available secondary data like user's demographics. Consider for example distribution based on age grouping; children in age group of 1-10 will most likely have affinity for animation movies; similarly, young adults (say 20-30 years) can have affinity for action/thriller. Similarly, women may have in general affinity for rom-com or family genres whereas males might be more inclined towards action. On similar lines, metadata for new items (such as their category information) can be used to gauge user's interest in them; thereby solving the item cold start problem. For example, a user who liked comedies in past will most likely enjoy comic recommendations. Thus, auxiliary data can prove to be a valuable source of information in RS; idea being exploited in several works [3,4]. Despite the difficulty in

garnering collaborative information for new users or items, most existing works [5,6] handling the cold start problem work with users and/or items which have small number of ratings available for them i.e. solve the partial cold start problem. Several works rely on building interviewing process [7,8] to collect (new) user's ratings on few selected items, which might not be convenient in all scenarios outside the academia. For example, e-retailers such as amazon or alibaba does not gather such information from new users and sites garnering such information are becoming increasingly rare.

The metadata used for prediction to solve the cold start problem can also be used to augment the rating dataset for warm start users. The explicit rating data (from users) is much more reliable than implicitly gathered information but suffer from extreme data sparsity. To alleviate this data sparsity, user/item auxiliary information can be exploited.

Traditionally, collaborative filtering (CF) [9,10] techniques have been used as de-facto approach to harness the explicit rating data for rating prediction. In recent past, researchers have proposed models based on CF schemes to assimilate user/ item metadata as well. This additional data has been used to augment the explicit rating data in either a memory based setup [4,11] or in latent factor framework [12,13].

Neighborhood based models [14] although easy to implement, do not always yield the best of results [15]; latent factor models [16] being more powerful. These models assume that user's choices on items are determined by very few factors. The user has an affinity towards these factors whereas the items possesses these factors to a lesser or greater extent. Thus both the users and items can be characterized as vectors of latent factors; user's rating on an item expressed as an inner product between the user and item latent factor vectors.

In light of the arguments presented above, in this work, we aim to use auxiliary data in a latent factor framework to improve prediction accuracy for both warm start and pure cold start scenarios. The highlight of our approach is that the proposed method to solve the cold start problem for new users/items is a direct increment of the model proposed for improving prediction accuracy for existing users; thereby handling both the major problems without increased resource requirement or complexity. Such a comprehensive model has not been proposed.

Another highlight of our model is use of matrix completion formulation, instead of more commonly employed matrix factorization (MF) [17], which attempts to recover the rating matrix as a product of two matrices – user's latent factor and item's latent factor matrix. Matrix factorization is computationally fast, but unfortunately it is bilinear and hence non-convex. Recently, researchers in signal processing showed that, instead of formulating the latent factor model as a matrix factorization problem, it can be recast as a low-rank matrix completion problem (LRMC) - a convex formulation [18,19]. As discussed above, latent factor model assumes that an item's rating is a function of a handful of features (latent factors). As, the entire rating matrix is a result of interaction amongst the latent factor vectors of users and items, its structure is governed by the small number of factors only. This results in the low rank nature of the rating matrix enabling use of LRMC techniques for rating prediction. We formulate our proposition as an augmented matrix completion problem (with additional regularization terms) – which enjoys the benefit of a convex formulation, deriving ideas from supervised learning.

In addition, unlike most recent works, which use user's social profile or trust network as additional data source [13], we use user's demography and item category information to supplement the rating database. It is difficult for RS to acquire social relation data for user; limiting the applicability of models using the same. Most RS maintain a database of item genre/category (for example an online book store will always have books categorized as per genre). Also, usually users are required to fill up some basic information (like age, gender etc.) while registering on an online portal. Hence, this information is readily available and at no extra cost, enabling a wider applicability of our model.

The novelty of our approach lies in the use of easily and widely available data (user demography and item genres) in a supervised learning environment to generate effective recommendations. We group together (label) users based on their demographic information – age, gender and occupation. The rating prediction is done under the additional constraint of maintaining label consistency. Similar strategy is adopted for items as well by using the genres as classification labels. Use of additional information (as constraints) to augment the matrix completion model reduces the problem search (solution) space, making the problem less underdetermined. There are few works [4,12] that incorporate demographic data of users, however none of them follow the principles of supervised learning followed in our work. Also, as indicated in results section; ours is a far superior formulation.

We extend our supervised learning based model to mitigate the cold start problem as well. The label consistency model is used to derive a relation between a user's and/or item's class labeling and their rating pattern. This information is used to predict the ratings and make effective recommendations for new users (or new items) for which secondary information is available. When a new user enters a system, their record is updated and so is the case with a new item (say movie) made available at an online portal. Thus, in absence of any rating data for such cold start conditions, this information makes effective recommendations plausible. We also design an algorithm based on split Bregman technique for our formulations.

2. Related Work

2.1 Matrix Factorization Framework for Latent Factor Model

The explicit rating provided by a user ($R_{i,j}$, user *i* on item *j*) can be viewed as a combination of two factors – baseline estimate and interaction component. The baseline constitutes user and item biases. There are some users who are overtly critical and tend to rate everything on the lower side of the scale – they have negative bias; similarly, there are some movies which are always rated on the higher side – they have positive bias. The 'interaction' part models a user's affinity for an item.

Usually baseline is computed offline by solving (2) via stochastic gradient descent algorithm [17].

$$\min_{b_i,b_j} \sum_{i,j\in avail\_rat} \left\|R_{i,j}-b_j-b_i-\mu\right\|_2^2+\delta\left(\left\|b_j\right\|_2^2+\left\|b_i\right\|_2^2\right) \qquad (2)$$

where, μ is the global mean; b_i is i^{th} the user bias and b_j is the item bias of j^{th} item; $\left(\mu+b_i+b_j\right)$ is the baseline component; δ is the regularization parameter.

The interaction (*Y*) between the user and the item $\left(Y_{i,j}=R_{i,j}-b_j-b_i-\mu\right)$ is modelled in terms of latent factors. Consider the case of movie ratings; choice of a movie is determined by very few factors - genre, director, cast, music etc. Each movie possesses these factors to a certain extent, and each user has affinity towards these factors. Based on this model one can represent a user (*i*) by a vector U_i and an item (*j*) by a vector V_j corresponding to latent factors. The 'interaction' can hence be expressed as inner product of two $\left\langle U_i,V_j\right\rangle$.

The problem in CF is that all the user ratings are not available; a typical user will only rate a small percentage of all the items. Thus, if we consider the interaction matrix (Y), it is incomplete. The problem in CF is to predict all the missing ratings - i.e. fill in the rating matrix. This can be expressed as an inverse problem [14]; $Y=M\odot(UV)$, where M is a binary mask having 1's in place of available ratings and 0 elsewhere.

This problem is solved via the following optimization:

$$\min_{U,V}\left\|Y-M\odot(UV)\right\|_F^2+\lambda\left(\left\|U\right\|_F^2+\left\|V\right\|_F^2\right) \qquad (3)$$

This problem is non-convex in *U* and *V*, owing to the bi-linearity. Thus there is no convergence guarantee.

2.2 Matrix Completion

If, we consider all the users and the items, the interaction matrix will be represented as $Z=UV$; Z $\left(Z\in\mathbb{R}^{K\times N}\right)$ is complete interaction matrix with *K* users and *N* items.

Traditionally latent factor models formulated the interaction component as a matrix factorization problem. However, if we concentrate on rating prediction (only the interaction *Z*), we do not need to solve for the user (*U*) and the item (*V*) factor matrices separately, as long as we can estimate Z. Recent studies proposed estimating Z directly, by solving the inverse problem $Y=M\odot Z$.

This is an under-determined inverse problem with infinitely many solutions. In order to find a reasonable solution, one needs some prior assumption regarding *Z*. Even though *Z* is a very large matrix (hundreds of thousands of users and items), it has a very low-rank; the rank being the same as the number of latent factors. Thus predicting the missing interactions turns out to be a Matrix Completion problem (4)

$$\min_Z\left\|Y-M\odot Z\right\|_F^2+\lambda\left\|Z\right\|_* \qquad (4)$$

The nuclear norm penalty promotes a low-rank solution [20]. In this section, we review few LRMC algorithms briefly.

Toh, & Yun [21] proposed Accelerated Proximal Gradient (APG) algorithm for LRMC. It employs Proximal Gradient (PG) [22] method with an appropriate step size and an extra interpolation step to achieve faster convergence. The iterative algorithm can be summarized as follows

$$\begin{aligned}
&W^k=X^k+\frac{t^{k-1}-1}{t^k}\left(X^k-X^{k-1}\right)\\
&G^k=W^k-\left(\tau^k\right)^{-1}A^T\left(A\left(W^k\right)-b\right)\\
&X^{k+1}=S_{\tau^k}\left(G^k\right);t^{k+1}=\frac{1+\sqrt{1+4\left(\tau^k\right)^2}}{2}\\
&s.t.\ b=vec\left(Y\right);A:block\ diag\ \ form\ of M
\end{aligned} \qquad (5)$$

Authors in [23] proposed a method for low-rank matrix recovery using the Iterative Least Square (IRLS) technique. It aims at minimizing the weighted Frobenius norm, $\left\|W_p^{(1/2)}X\right\|_F^2$ of matrix, *X*. A low rank matrix (*X*) results if weighting matrix W_p is chosen appropriately. IRLS algorithm for nuclear norm minimization consists of following iterates

$$\begin{aligned}
&X^k=\arg\min\left\{Tr\left(W_p{}^{k-1}X^TX\right):A\left(X\right)=b\right\}\\
&W_p{}^k=\left(\left(X^k\right)^T\left(X^k\right)+\gamma^kI\right)^{p/2-1}
\end{aligned} \qquad (6)$$

Most of the existing methods for LRMC require large number of iterations for convergence on large datasets. We propose an algorithm for our augmented matrix completion formulation based on split Bregman technique [24]. Use of split Bregman helps achieve faster convergence and improved recovery accuracy.

2.3 Use of Auxiliary Information

To augment the (sparse) explicit rating dataset several researchers have utilized available secondary data. In this section we review some of the techniques for the same.

Authors in [25] proposed a similarity measure (sim_{mod}) to determine nearest neighbours based on both rating data and demographic information (7).

$$sim_{mod} = sim_{dem} \times sim_{rat} + sim_{rat} \quad (7)$$

where, sim_{dem} is similarity computed using demographics and sim_{rat} is computed using explicit rating data.

Rating data is augmented with geo-spatial information, in a neighbourhood based model, for photograph recommendation in [26]. They used geographical tag data to group photographs into clusters and propagate ratings amongst the members of the same cluster. Thus, a dense rating matrix is obtained which is used as input to neighbourhood based CF algorithm.

Authors in [12] used graph regularization to augment the matrix factorization model. User and item graphs were constructed by utilizing user's demographic and social profile data and item's genre classification.

$$\min_{U,V} \|Y - M \odot (UV)\|_F^2 + \lambda \left(Tr\left(U^T G_{lu} U\right)\right) + \gamma \left(Tr\left(V^T G_{lv} V\right)\right) \quad (8)$$

where, G_{lu} and G_{lv} are the graph Laplacians for user and item graphs respectively.

In [27] social network information and ratings are used in a PMF (Probabilistic Matrix Factorization) framework. Standard PMF models latent factor vectors as independent Gaussian priors. In [12] PMF is modified to allow for correlation between these Gaussian priors, incorporating similarity amongst items/users.

Most Existing works, as discussed above, augment the conventional matrix factorization framework with secondary data. Also, they mainly rely on grouping of users and/or items and promoting similarity amongst latent factor vector of similar (grouped) users and/or items. Though, authors in [28] augmented matrix completion model, their model is also based on grouping together similar users. They minimized the rating variation amongst similar users (9). They do not exploit item metadata in their framework

$$\min_{Z} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* + \sum_{G \in Groups} \left(\mu_G \sum_{g \in G} \text{var}_g (R) \right) \quad (9)$$

In this work, we build up on the (convex) matrix completion model incorporating user/item metadata (demographic information and item categorization) in a label consistent (supervised learning) framework. Also, our formulation can exploit both item and user metadata. Use of label consistency model helps us derive linear maps from rating space to item/user label domain. This assists in solving the rating prediction problem for new users and new items (cold start). None of prior art targets both warm and cold start users together.

2.4 Cold Start Poblem

The problem of providing effective suggestions to new users or recommending new items to existing users – the cold start problem, is a big challenge in RS design. We review some of prior art in the area.

In [29] used a trust based measure to determine similar users instead of rating based similarity for cases where very few ratings are available. They argued that because trust propagates, there can be many more similar users than if (very few) ratings are considered, making predictions better. Authors in [6] used social tags as a means of relating users to items. The predictions are based on the frequency of tags and the semantic relationships between tags and items.

Works like [30] use small amount of rating information alone to target partial cold start problem. They based their predictions on a new similarity measure that also consider the frequency and count of co-rated items to remove disparity between users with highly varied rating patterns.

Authors in [31] used user's demographics to model an alpha-community space model. Once a new user's communities are defined, one recommendation list per community is generated based on adhoc level of agreement recommendation process.

Most works, as highlighted above, solve the cold start problem for cases where some rating information is available. We, in this work attempt to solve the pure cold start problem. Also, unlike existing methods which attempt to separately solve the cold start problem, our framework is a cohesive model aiming for improvements in accuracy for existing users and mitigating the cold start problem.

3. Proposed Formulation

In this section, we describe our proposed formulation for design of a RS incorporating user-item metadata to improve prediction accuracy. The design is also extended to solve the pure cold start problem. The novelty of our work lies in formulating a matrix completion based model for exploiting user (demographic profile) metadata and item categories along with the ratings. We augment the LRMC model with label consistent constraints, derived from user/item metadata, imposed on the rating matrix. Also, the highlight of our design is that we put forth a comprehensive model to handle two major problems afflicting the RS – improving quality of prediction and the cold start problem.

3.1 Problem Formulation

3.1.1 Low Rank nature of rating matrix

As discussed above, we perform offline baseline estimation and work with interaction component alone. Once the complete interaction matrix (Z) is recovered (using proposed formulation) the baseline estimates are

added back.

Latent factor model states that the interaction between users and items is governed by a small number of factors – the latent factors; say, for books the latent factors may be author and genre; for movies director, genre, cast etc. As the interaction matrix is a function of very few variables (~40-50) as compared to matrix dimensions (hundreds of thousands of users and items), the matrix is fairly low rank. The low-rank property of Z can be used to predict the missing ratings using LRMC framework. Thus predicting the missing interactions turns out to be a Matrix Completion problem (10).

$$\min_{Z} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* \qquad (10)$$

where, A is a binary mask, which is 1's in place of available rating values and *0* otherwise; Z is the completely filled matrix of interaction component; Y interaction component of available ratings.

3.1.2 Incorporating Metadata

Nuclear norm minimization (10) requires that for a rank r matrix of size $n \times n$, at least $(6n-5r)r$ samples be available [20]. For the case of RS design, size of matrix is at least 1000×1000, thereby requiring around 23% of the ratings to be available for reasonable reconstruction accuracy (assuming rank to be 40). However, in real world datasets, the available information is less than 10%, in some cases even as low as 1%.

Hence, there is considerable need for additional information, which can alleviate data sparsity to improve prediction accuracy. In this paper, we make use of user's demographic data and item genre information to augment the rating data for a movie recommender system. Often, during the process of sign up users are required to enter their basic demographic data. Also, all portals maintain a database of their item categories. Thus, collecting this information invites no additional cost. Even for new users and new items, this metadata is readily available; even if collaborative information is missing.

Our model utilizes a label information data (matrix) defining relations between users and/or items and the class they belong to. For users, classes are defined on the basis of age, gender and their occupation; for items, multiple genres form the distinct classes. Our framework can make use of any additional available information as well for classification purpose. We incorporate label data into the matrix completion framework by modifying (10) to include additional label consistent regularization terms.

Considering user metadata, we define multiple classes based on gender, age brackets and different occupational profiles; user can simultaneously belong to multiple classes. Using this label information a user-class label matrix (L_u) is defined, such that $L_u(i,c) = 1$ if user i belongs to class c else 0. Let us consider an example wherein we form 2 distinct gender (M/F) groups, P distinct non-overlapping age groups (say 1-17, 18-24 and so on) and Q distinct occupational categories. The label matrix (L_u) will have a row corresponding to each user and columns corresponding to $(2+P+Q)$ classes as shown in fig. 1. Let us consider a user (User 1), who is a male in age group of 18-24 and a lawyer by profession. The classification information of this user can be used to fill up first row of L_u. Similarly, for a female in age group of 60+ and an artist by profession, corresponding row will be as shown in row 2 and so on.

This class label matrix provides additional data to help predict the missing values in the rating matrix. The ratings are predicted under the add-on constraint of maintaining label consistency (appended as a regularization term) as

$$\min_{Z,W_u} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* + \lambda_u \|L_u - ZW_u\|_F^2 \qquad (11)$$

where, W_u is the linear map from user-item rating space to user-class space. It defines relation between a user's class and their ratings; λ_u is the regularization parameter governing the relative importance given to rating data and the demographic information.

Similar model is built for items as well; establishing a relation between the item genre and the ratings given to them by users. Each item (movie, in this case) may belong to several classes (genres). A class-item label matrix (L_v) is constructed such that $L_v(c,j) = 1$ if item j belongs to class c else 0. Similar to formulation discussed in (12) we propose item metadata based framework (12)

$$\min_{Z,W_v} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* + \lambda_v \|L_v - W_v Z\|_F^2 \qquad (12)$$

where, W_v is the linear map (to be estimated) from user-item rating space to class-item space and λ_v is the

	Gender 1 (M)	Gender 2 (F)	Age 1 (1-17)	Age 2 (18-24)	….	Age P (60+)	Occ. 1 (Tech.)	Occ. 2 (Artist)	….	Occ. Q (Lawyer)
User 1	1	0	0	1	0	0	0	0	0	1
User 2	0	1	0	0	0	1	0	1	0	0
..	..	..	..	..	..	..	..	..	..	..
User |U|	1	0	0	1	0	0	1	0	0	0

Figure 1. Construction of label matrix

regularization parameter.

We also club together both formulations to exploit both item and user metadata simultaneously as shown in (13).

$$\min_{Z,W_v,W_u} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* + \lambda_v \|L_v - W_v Z\|_F^2 + \lambda_u \|L_u - Z W_u\|_F^2 \quad (13)$$

Equation (13) illustrates our final formulation for supplementing the matrix completion model with item and user metadata. Use of additional information helps improve the robustness and accuracy of our recommender system by making the problem less underdetermined.

3.1.3 Alleviating Cold Start Problem

For new users or items there are no ratings; making rating prediction a challenge. We propose to use the information map (W_v and W_u) extracted from solving (13), almost as a by-product, to solve the pure cold start problem.

First let us consider, the information map W_u i.e. one generated using user metadata. It is a map from rating information to user label (classification) space. The map primarily correlates the ratings or user's choice with the demographic profile of a user. Consider a new user U_{new} entering a system. As he/she signs up on the portal, their demographic information is captured. Thus, a vector $(U_{coldstart})$ defining class labelling of the said user $\left(\left[U_{new-c_1} \quad U_{new-c_2} \quad . \quad U_{new-c_{cu}}\right]\right)$ can be constructed, where *cu* is the number of classes considered for users. From solution top (13), we have the deciphered map W_u. The new user's demographic information (label vector) and the deciphered map can be related as

$$\left[U_{new-c_1} \quad . \quad . \quad U_{new-c_{cu}}\right] = \left[Z_{new-i_1} \quad . \quad . \quad Z_{new-i_N}\right] \times \begin{bmatrix} W_{u(11)} & W_{u(12)} & W_{u(13)} & . \\ W_{u(21)} & W_{u(22)} & . & . \\ . & . & . & . \\ . & . & . & W_{u(i_N c_{cu})} \end{bmatrix} \quad (14)$$

where, $\left(Z_{coldstart} = \left[Z_{new-i_1} \quad . \quad . \quad Z_{new-i_N}\right]\right)$ is the vector defining the new user's rating (interaction components) for each item in the database (total number of items, *N*). Equation (14) can be written as set of linear equation (15)

$$U_{coldstart} = Z_{coldstart} W_u \quad (15)$$

Predicted interaction part for new user, $Z_{coldstart}$, can be obtained by solving (15) using any conjugate gradient type algorithm.

Similar approach can be followed for item cold start problem as well by utilizing the genre information of new item (V_{new}) and the information map W_v. As a new item (say movie in our case) is added to the system, its genre information is easily available. The information map, W_v, establishes a relation between the rating data and the genre of items i.e. it captures information relating user's choice of an item to its genre content. This information map is used to determine user's preference for a new item.

Similar to equation constructed above for users, we can formulate item cold start problem as

$$\begin{bmatrix} V_{new-c_1} \\ V_{new-c_2} \\ . \\ V_{new-c_{cv}} \end{bmatrix} = \begin{bmatrix} W_{v(11)} & W_{v(12)} & . & . \\ W_{v(21)} & . & . & . \\ . & . & . & . \\ . & . & . & W_{v(c_{cv}u_K)} \end{bmatrix} \begin{bmatrix} Z_{new-u_1} \\ Z_{new-u_2} \\ . \\ Z_{new-u_K} \end{bmatrix} \quad (16)$$

where, $V_{coldstart} = \left[V_{new-c_1} \quad . \quad . \quad V_{new-c_{cv}}\right]^T$ is the class label vector for the new item (*cv*: number of distinct classes); $Z_{coldstart} = \left[Z_{new-u_1} \quad . \quad . \quad Z_{new-u_K}\right]^T$ defines interaction component of ratings by all existing users for new item.

Equation (16) can be compactly written as in (17) and solved using a conjugate gradient solver.

$$V_{coldstart} = W_v Z_{coldstart} \quad (17)$$

Hence, our model can be used to mitigate both user and item end (pure) cold start problem, as an extension of our label consistent model, without significant computational burden.

3.2 Algorithm Design

In this section, we present the algorithm for our proposed formulation (13) using split Bregman technique.

Use of split Bregman technique [24] aids in faster convergence and lower recovery errors, as no cooling of regularization parameter is required and thus optimal values of regularization parameters for each of the sub problem can be set.

Firstly, in order to enable splitting of multiple norm terms, we introduce proxy variables (*P* and *Q*) in our formulation (13) as in (18).

$$\min_{Z,W_v,W_v,P,Q} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* + \lambda_v \|L_v - W_v P\|_F^2 + \lambda_u \|L_u - Q W_u\|_F^2 + \mu_v \|P - Z - B1\|_F^2 + \mu_u \|Q - Z - B2\|_F^2 \quad (18)$$

where, *B*1 and *B*2 are the Bregman variables.

Use of Bregman variables ensures that the equality between original and proxy variables need not be strictly enforced from the start. Updation of Bregman variables helps add back the error thus making the algorithm self-correcting and also helps in faster convergence.

We split our formulation into simpler sub problems using Alternating Direction method of Multipliers.

Sub Problem 1

$$\min_Z \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* + \mu_v \|P - Z - B1\|_F^2 + \mu_u \|Q - Z - B2\|_F^2 \quad (19)$$

Sub Problem 2

$$\min_P \lambda_v \|L_v - W_v P\|_F^2 + \mu_v \|P - X - B1\|_F^2 \quad (20)$$

Sub Problem 3

$$\min_{Q} \lambda_u \left\| L_u - QW_u \right\|_F^2 + \mu_u \left\| Q - X - B2 \right\|_F^2 \quad (21)$$

Sub Problem 4

$$\min_{W_v} \left\| L_v - W_v P \right\|_F^2 \quad (22)$$

Sub Problem 5

$$\min_{W_u} \left\| L_u - QW_u \right\|_F^2 \quad (23)$$

Now, focusing on sub problem 1, it can be recast as

$$\min_{Z} \left\| \begin{pmatrix} Y \\ \sqrt{\mu_v}\left(P - B1\right) \\ \sqrt{\mu_u}\left(Q - B2\right) \end{pmatrix} - \begin{pmatrix} A \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix} Z \right\|_F^2 + \lambda \left\| Z \right\|_* \quad (24)$$

Equation (24) can be solved by soft thresholding of singular values [32] as follows

$$Z \leftarrow Soft\left(Singular\ value\left(T\right), \frac{\lambda}{2\alpha} \right)$$

$$T = Z + \frac{1}{\alpha} \begin{pmatrix} A \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix}^T \left(\begin{pmatrix} Y \\ \sqrt{\mu_v}\left(P - B1\right) \\ \sqrt{\mu_u}\left(Q - B2\right) \end{pmatrix} - \begin{pmatrix} A \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix} Z \right) \quad (25$$

where, $Soft\left(t,u\right) = sign\left(t\right)\ \max\left(0, \left|t\right| - u\right)$ and

$$\alpha \geq \max\left(eig \begin{pmatrix} A \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix}^T \begin{pmatrix} A \\ \sqrt{\mu_v} I \\ \sqrt{\mu_u} I \end{pmatrix} \right).$$

2nd subproblem can be cast as a least square expression as

$$\min_{P} \left\| \begin{pmatrix} \sqrt{\lambda_v} L_v \\ \sqrt{\mu_v}\left(X + B1\right) \end{pmatrix} - \begin{pmatrix} \sqrt{\lambda_v} W_v \\ \sqrt{\mu_v} I \end{pmatrix} P \right\|_F^2 \quad (26)$$

Similarly, sub problem 3 can be recast as follows

$$\min_{Q} \left\| \begin{pmatrix} \sqrt{\lambda_u} L_u \\ \sqrt{\mu_u}\left(X + B2\right) \end{pmatrix} - Q \begin{pmatrix} \sqrt{\lambda_v} W_u \\ \sqrt{\mu_u} I \end{pmatrix} \right\|_F^2 \quad (27)$$

Equation (22), (23), (26) and (27), are simple least square expressions which can be efficiently solved using any conjugate gradient type solver. In each iteration Bregman variables are updated as follows

$$B2 = B2 + Z - Q \quad (28)$$

$$B1 = B1 + Z - P \quad (29)$$

The iterations continue till convergence. The complete algorithm (LCMC-Label consistent matrix Completion) is given in fig 2.

Initialize variables,
Set regularization parameters; *max_iter*
while not convergence
$//\ Solve\ for\ \mathrm{Z};\ Z \leftarrow Soft\left(Singular\ value\left(T\right), \frac{\lambda}{2\alpha} \right)$
$//\ Solve\ for\ \mathrm{P};\ Solve \begin{pmatrix} \sqrt{\lambda_v} L_v \\ \sqrt{\mu_v}\left(X + B1\right) \end{pmatrix} = \begin{pmatrix} \sqrt{\lambda_v} W_v \\ \sqrt{\mu_v} I \end{pmatrix} P$
$//\ Solve\ for\ \mathrm{Q};\ Solve \begin{pmatrix} \sqrt{\lambda_u} L_u \\ \sqrt{\mu_u}\left(X + B2\right) \end{pmatrix} = Q \begin{pmatrix} \sqrt{\lambda_v} W_u \\ \sqrt{\mu_u} I \end{pmatrix}$
$//\ Solve\ for\ W_v;\ Solve\ \ \min_{W_v} \left\| L_v - W_v P \right\|_F^2$
$//\ Solve\ for\ W_u;\ Solve\ \ \min_{W_u} \left\| L_u - QW_u \right\|_F^2$
$//$ *Update Bregman Variable*;
$B2 = B2 + Z - Q;\ B1 = B1 + Z - P$
end while

Figure 2. Algorithm - LCMC

4. Experiment and Results

We demonstrate the performance of our algorithm for a movie recommender system. We conducted experiments on 100K and 1M Movielens datasets (http://grouplens.org/datasets/movielens/). To the best of our knowledge, these are the only public datasets which provide relevant user and item metadata with ratings.

4.1 Description of Datasets

Both the datasets contain ratings on a scale of 1-5. 100K dataset contains 100K ratings given by 943 users on 1682 movies and 1M dataset has 1M ratings on around 3952 movies given by 6040 users. Both datasets have less than 5% of the ratings available and hence the improvement achieved by using metadata can be adequately gauged.

For users 30 groups are constructed – 2 for gender (M/F), 7 for multiple age-brackets (1-17, 18-24, 25-34, 35-44, 45-49, 50-55 and 56+) and 21 for various occupations. For items, 19 groups are formed, each representing a different genre. This information is used to construct label matrices $\left(L_u, L_v\right)$ as discussed in section 3.

4.2 Experimental Setup and Evaluation Criteria

We conducted 5-fold cross validation on both the datasets; 80% of the ratings forming the train set and remaining 20% used for testing. The simulations are carried out on system with i7-3770S CPU @3.10GHz with 8GB RAM. For cold start testing, 80% of users (items) were kept as part of training data and test done on remaining 20% users (items).

For offline baseline estimation, value of δ in (2) is set as $1e-3$. The value of regularization parameters for our formulation (18) is selected using greedy L-curve technique [33]. The values for both 100K and 1M dataset are $\lambda = 1e+1$, $\lambda_u = 1e-1$, $\lambda_v = 1e-1$, $\mu_u = 1$, $\mu_v = 1$. The overall accuracy of our model is evaluated using MAE (Mean absolute error) (30) and RMSE (root mean

square error) (31).

$$MAE = \frac{\sum_{i,j} R_{i,j} - \hat{R}_{i,j}}{|R|} \quad (30)$$

$$RMSE = \sqrt{\frac{\sum_{i,j} \left(R_{i,j} - \hat{R}_{i,j}\right)^2}{|R|}} \quad (31)$$

where, R and $\hat{R}$ are the actual and predicted ratings and $|R|$ is the cardinality of the rating matrix R.

The relevance of recommendations for each user is measured in terms of precision (32) and recall (33) [34] for top-N recommendations. The values depicted in the results are the average of values computed for each user. Precision and recall curves are plotted for varying number of recommendations.

$$Precision = \frac{\#t_p}{\#t_p + \#f_p} \quad (32)$$

$$Recall = \frac{\#t_p}{\#t_p + \#f_n} \quad (33)$$

Here, t_p denotes true positive (item relevant and recommended), f_p denotes false positive (item irrelevant and recommended) and f_n denotes false negative (item relevant and not recommended). An item is marked relevant if it's rated as above 3 else irrelevant.

4.3 Analyzing impact of metadata

In this section we present the results of our proposed formulations – Label consistent matrix completion with user metadata (LCMC-U) (11), Label consistent matrix completion with item metadata (LCMC-I) (12), Label consistent matrix completion with user and item metadata (LCMC-UI) (13).

We compare the result of our work with state of the art matrix completion and matrix factorization algorithms – Accelerated Proximal gradient (APG) [21], Block Co-ordinate descent based Non negative matrix factorization (BCD-NMF) [35], Factored item similarity model (FISM) [36] and Probabilistic matrix factorization (PMF) [37].

To further highlight the contribution of user/item metadata in improving recommendation accuracy, we also show the results for following two (sub) formulations:

1. MC: Formulation exploiting just the rating data in a nuclear norm minimization framework i.e. user/item metadata is not utilized (34).

$$\min_{Z} \|Y - A(Z)\|_F^2 + \lambda \|Z\|_* \quad (34)$$

For solving (34) we adopt split Bregman technique, similar to one used for our formulation, to maintain consistency of algorithm efficiency and highlight the contribution of our model (13).

2. LC: Formulation exploiting only the label consistency constraints i.e. without the low rank nature of rating matrix being taken into consideration (35).

$$\min_{Z,W_v,W_v} \|Y - A(Z)\|_F^2 + \lambda_v \|L_v - W_v Z\|_F^2 + \lambda_u \|L_u - Z W_u\|_F^2 \quad (35)$$

Equation (35) is a least squares formulation which can be easily solved.

TABLE 1. ERROR MEASURES

	100K Dataset		1M Dataset	
Algorithm	**MAE**	**RMSE**	**MAE**	**RMSE**
LCMC-U	**0.7230**	**0.9207**	**0.6767**	**0.8634**
LCMC-I	**0.7224**	**0.9216**	**0.6766**	**0.8612**
LCMC-UI	**0.7193**	**0.9145**	**0.6731**	**0.8559**
MC	0.7351	0.9319	0.6813	0.8711
LC	0.7481	0.9473	0.7186	0.9094
APG	0.8847	3.7076	0.9782	3.8109
PMF	0.7564	0.9639	0.7241	0.9127
BCD-NMF	0.7582	0.9816	0.6863	0.8790
FISM	0.7432	0.9439	0.7196	0.9102

Table 1 illustrates the MAE and RMSE values for the 100K and 1M datasets for various algorithms. The results obtained for nuclear norm minimization algorithm using split Bregman technique (MC) indicate that it gives around 3% lower MAE and 3.5% lower RMSE value than the next best latent factor model based MF algorithm i.e. PMF. Also, MC is superior than the neighborhood inspired factor model (FISM) and achieves a 1.5% lower MAE than the latter. This demonstrates the efficiency of our algorithm using split Bregman technique over other methods.

Also, our formulation using only the metadata (label consistency) constraints also yields fairly good results. We are able to outperform the existing matrix factorization algorithm as well (i.e. PMF and BCD-NMF) by around 1.7%. It gives results quite close (MAE 0.7481) to those obtained using FISM (MAE 0.7432). Thus, both our individual formulations, one including rating information and other involving metadata give good results. Then the obvious next step is to combine both information sources to get improved prediction accuracy, as in our combined formulation LCMC.

Comparison of our formulations incorporating user/item metadata (LCMC) with one using just the rating data (MC) corroborate our claim that use of secondary information indeed improves recovery accuracy. Our proposed formulations are able to better the MAE and RMSE values by around 2% over the MC algorithm. Using both user and item metadata yields slightly better result than each of them individually.

For 1M dataset also MC formulation outperforms existing MC/MF algorithms. Use of secondary data is

able to achieve a reduction of around 1.5% in error measures over formulations just exploiting rating data.

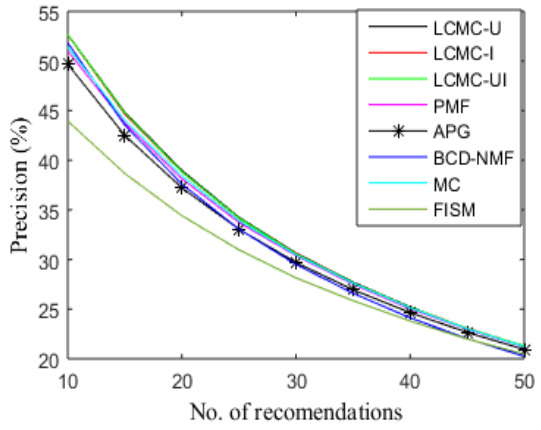


Figure 3. Precision Curve (100K dataset)

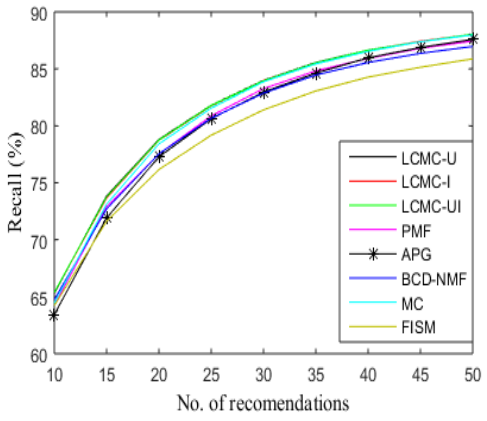


Figure 4. Recall Curve (100K dataset)

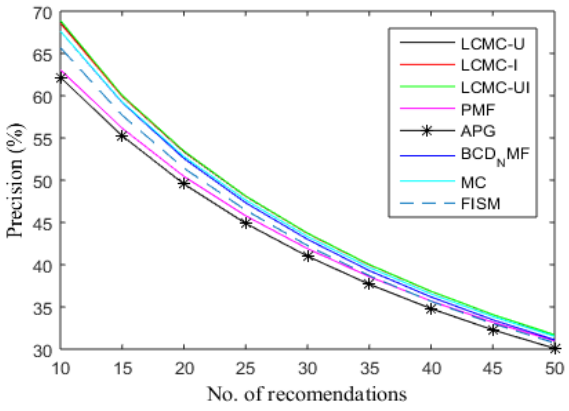


Figure 5. Precision Curve (1M dataset)

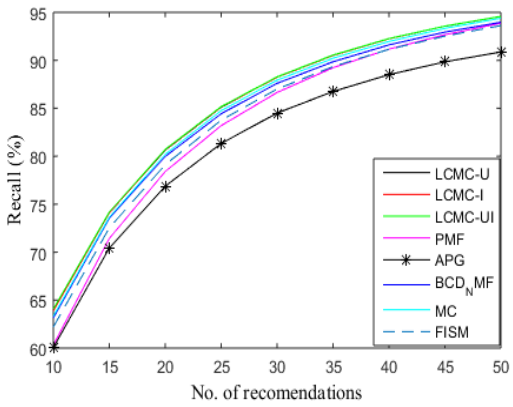


Figure 6. Recall Curve (1M dataset)

The precision and recall curves for all the algorithms for 100K and 1M dataset are given in figures 3-6. Here also, our formulations (LCMC) show better performance than the algorithms compared against. However, there isn't much difference between the precision and recall values for LCMC formulation using either individual user or item metadata or a combination of both. Also, the improvement using our algorithm is more pronounced for the 1M dataset.

4.4 Comparison with existing techniques

In this section we showcase the superiority of our supervised learning based approach for assimilating user/item metadata over other methods utilizing similar information. We compare the performance of our formulation against a neighbourhood based method (KNN) proposed in [25] and against a latent factor MF based formulation (Graph Reg) using graph regularization [12]. We also compared our work against two other works - a semi supervised learning based non negative matrix factorization (SSNMF) technique proposed in [38] and matrix completion framework with user metadata (MCAI) proposed in [28].

TABLE 2. ERROR MEASURES

	100K Dataset		1M Dataset	
Algorithm	**MAE**	**RMSE**	**MAE**	**RMSE**
LCMC-UI	**0.7193**	**0.9145**	**0.6739**	**0.8559**
KNN	0.8302	1.0467	0.8198	0.9989
SSNMF	0.7723	1.0112	0.7285	0.9401
Graph Reg	0.7577	0.9616	0.7233	0.9139
MCAI	0.7206	0.9187	0.6749	0.8622

Table 2 shows the comparison of error measures for 100K and 1M datasets. Amongst all the algorithms for both the datasets KNN gives the poorest results. This is owing to the fact that neighbourhood based methods are simple heuristic measures which perform worse than latent factor models. On comparison to latent factor formulation – Graph Reg – our method yields more than

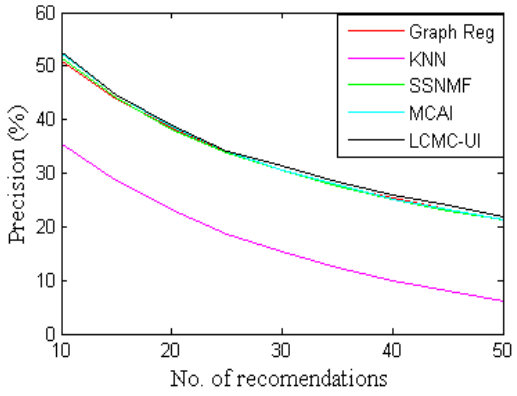


Figure 7. Precision Curve (100K dataset)

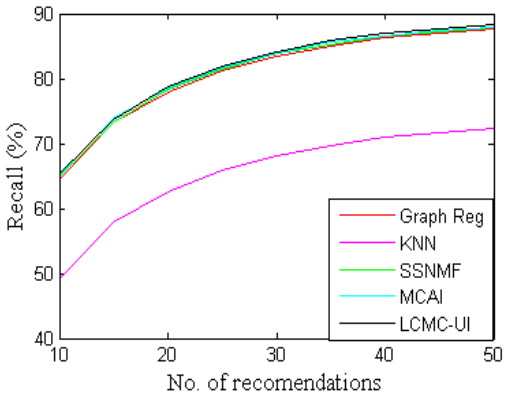


Figure 8. Recall Curve (100K dataset)

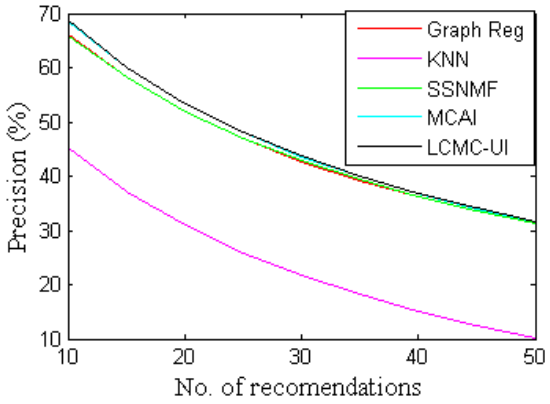


Figure 9. Precision Curve (1M dataset)

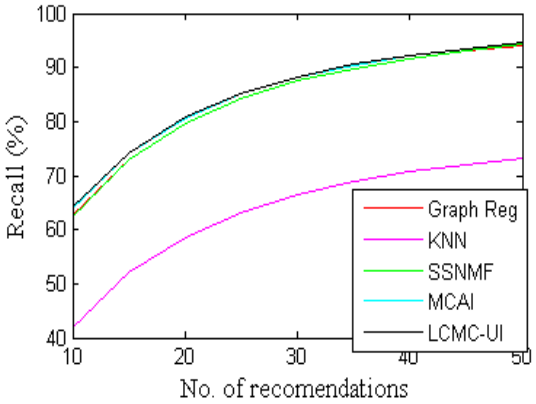


Figure 10. Recall Curve (1M dataset)

5% lower MAE and RMSE values. Also, as compared to semi-supervised learning approach adopted in [38] our label consistent formulation is much better at capturing the metadata information. We are able to get ~8% reduction in MAE and RMSE values. It is also partly contributed by use of our algorithm designed using split Bregman approach. Comparison to another matrix completion based approach (MCAI) also indicates that our formulation is able to achieve a reduction in both MAE and RMSE. It can be contributed to use of our novel label consistent formulation that enables use of both user and item metadata. Thus, it validates our claim that our label consistent formulation is able to better capture the correlation amongst users and items based on their associated metadata.

The precision and recall curves for these methods are given in figure 7-10. On this measure also, it's clear that our method performs better or at least comparable than the other two compared against. The improvement is more significant for 1M dataset, owing to higher sparsity of the rating dataset.

4.5 Cold Start Problem

In this section we present our results for both the user and item (pure) cold start problem (U-CS and I-CS). For evaluation of our algorithm, we compute MAE and RMSE values. None of the existing works report results on both (user and item) cold start problems and hence we compare against different works. For comparison, we report the results indicated in the recent works. Table 3 gives results for our algorithm for item and user cold start condition for 100K and 1M datasets.

TABLE 3. ERROR MEASURES FOR COLD START

Algorithm	MAE	RMSE
User Cold Start - 100K	0.7275	0.9217
Item Cold Start - 100K	0.7271	0.9214
User Cold Start - 100K	0.7100	0.8984
Item Cold Start - 100K	0.7099	0.8983

From the above data it can be observed that our design methodology for solving the cold start problem gives fairly good results. The MAE and RMSE values for cold start (users or items) is sufficiently close to those obtained for existing (warm start) users and items; as shown in results discussed in section 4.4.

Results shown in previous works are very limited with most of the works solving the user end cold start problem. In [39] authors solved new user cold start problem by proposing a hybrid system based on SCOAL. They

segregate users into groups based on available information and design separate prediction model for each group. The new user, based on this demographic profile, is assigned to closest group and his ratings predicted accordingly. They reported a MAE of 0.93 for the 100K dataset, 29% higher than our MAE (0.73).

In [11] authors used known classification algorithms in combination with similarity techniques (similarity computed based on demographic information) and prediction mechanisms to retrieve recommendations. They conducted experiment on Movielens 1M dataset and reported an MAE of 0.75 and RMSE of 0.95. Our corresponding values for 1M dataset are 0.71 and 0.89.

Thus our algorithm significantly outperforms existing, state of the art, works for mitigating the cold start problem.

5. Conclusion

In this work, we propose a formulation to incorporate user-item metadata in a supervised learning augmented matrix completion framework. Our design targets accuracy improvement for new users and rating prediction for new users and items. Most existing works incorporate secondary information in a matrix factorization framework. However, MF being bi-linear and hence non-convex formulation does not provide convergence guarantees. We augment the convex matrix completion framework to include available metadata.

We defined multiple classes for both users and items based on available secondary information. Using this information, label matrices were constructed and used as additional information source. The rating values were predicted under the additional constraint of maintaining this label consistency. Use of add-on constraint helps reduce solution search space; in effect reducing the underdetermined nature of the problem. We also propose an algorithm using split Bregman technique for our proposed formulation.

Our design for cold start problem also uses information generated using the proposed label consistent model and hence proves efficient in terms of computational load. Most existing works focus on cases where a few ratings are available, whereas in this paper we solve a more challenging, pure cold start problem.

We illustrated the efficiency of our algorithm structure by comparing a basic matrix completion framework using split Bregman with existing MF/MC methods. We find that we are able to achieve better results than state of the art techniques in low-rank matrix completion. Secondly, we demonstrate the improvement obtained using the base MC formulation by augmenting it with label consistent information. Comparison with existing methods using metadata also shows the superiority of our design. In case of cold start problem, our framework is able to generate far superior results than the existing state of the art methods for both new user and new items. In the future, we would like to extend our design for other recommender system as well as for simultaneous new user-new item cold start problem.

References

[1] Bobadilla, Jesús, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. "Recommender systems survey." Knowledge-Based Systems 46 (2013): 109-132.

[2] Park, Deuk Hee, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. "A literature review and classification of recommender systems research." Expert Systems with Applications 39, no. 11 (2012): 10059-10072

[3] Melville, P., Mooney, R. J., & Nagarajan, R. (2002, July). Content-boosted collaborative filtering for improved recommendations. In AAAI/IAAI (pp. 187-192).

[4] Vozalis, M. G., & Margaritis, K. G. (2007). Using SVD and demographic data for the enhancement of generalized collaborative filtering. Information Sciences, 177(15), 3017-3037.

[5] Victor, Patricia, Martine De Cock, Chris Cornelis, and A. Teredesai. "Getting cold start users connected in a recommender system's trust network." Computational Intelligence in Decision and Control 1 (2008): 877-882.

[6] Zhang, Zi-Ke, Chuang Liu, Yi-Cheng Zhang, and Tao Zhou. "Solving the cold-start problem in recommender systems with social tags." EPL (Europhysics Letters) 92, no. 2 (2010): 28002

[7] Zhou, K., Yang, S. H., & Zha, H. (2011, July). Functional matrix factorizations for cold-start recommendation. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (pp. 315-324). ACM.

[8] Houlsby, N., Hernandez-lobato, J. M., & Ghahramani, Z. (2014). Cold-start active learning with robust ordinal matrix factorization. In Proceedings of the 31st International Conference on Machine Learning (ICML-14) (pp. 766-774).

[9] Schafer, J. Ben, Dan Frankowski, Jon Herlocker, and Shilad Sen. "Collaborative filtering recommender systems." In The adaptive web, pp. 291-324. Springer Berlin Heidelberg, 2007

[10] Cacheda, Fidel, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems." ACM Transactions on the Web (TWEB) 5, no. 1 (2011): 2

[11] Lika, Blerina, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. "Facing the cold start problem in recommender systems." Expert Systems with Applications 41, no. 4 (2014): 2065-2073.

[12] Gu, Quanquan, Jie Zhou, and Chris HQ Ding. "Collaborative Filtering: Weighted Nonnegative Matrix Factorization Incorporating User and Item Graphs." In SDM, pp. 199-210. 2010.
[13] Zhang, Y., Chen, W., & Yin, Z. (2013). Collaborative filtering with social regularization for TV program recommendation. Knowledge-Based Systems,54, 310-317
[14] Wang, Jun, Arjen P. De Vries, and Marcel JT Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion." In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 501-508. ACM, 2006.
[15] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." Knowledge and Data Engineering, IEEE Transactions on 17, no. 6 (2005): 734-749
[16] Hofmann, Thomas. "Latent semantic models for collaborative filtering." ACM Transactions on Information Systems (TOIS) 22, no. 1 (2004): 89-115
[17] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37
[18] Jaggi, Martin, and Marek Sulovsk. "A simple algorithm for nuclear norm regularized problems." In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 471-478. 2010
[19] Shamir, Ohad, and Shai Shalev-Shwartz. "Collaborative filtering with the trace norm: Learning, bounding, and transducing." (2011
[20] Candès, Emmanuel J., and Benjamin Recht. "Exact matrix completion via convex optimization." Foundations of Computational mathematics 9, no. 6 (2009): 717-772.
[21] Toh, K. C., & Yun, S. (2010). An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. Pacific Journal of Optimization, 6(615-640), 15.
[22] Nesterov, Y., & Nesterov, I. E. (2004). Introductory lectures on convex optimization: A basic course (Vol. 87). Springer.
[23] Mohan, K., & Fazel, M. (2012). Iterative reweighted algorithms for matrix rank minimization. The Journal of Machine Learning Research, 13(1), 3441-3473.
[24] Goldstein, T., & Osher, S. (2009). The split Bregman method for L1-regularized problems. SIAM Journal on Imaging Sciences, 2(2), 323-343.
[25] Vozalis, M., & Margaritis, K. G. (2004, August). Collaborative filtering enhanced by demographic correlation. In AIAI Symposium on Professional Practice in AI, of the 18th World Computer Congress
[26] Ostrikov, Alexander, Lior Rokach, and Bracha Shapira. "Using geospatial metadata to boost collaborative filtering." In Proceedings of the 7th ACM conference on Recommender systems, pp. 423-426. ACM, 2013
[27] Zhou, Tinghui, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. "Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information." In SDM, vol. 12, pp. 403-414. 2012
[28] Gogna, Anupriya, and Angshul Majumdar. "Matrix completion incorporating auxiliary information for recommender system design." Expert Systems with Applications 42, no. 14 (2015): 5789-5799.
[29] Massa, Paolo, and Paolo Avesani. "Trust-aware recommender systems." In Proceedings of the 2007 ACM conference on Recommender systems, pp. 17-24. ACM, 2007.
[30] Bobadilla, JesúS, Fernando Ortega, Antonio Hernando, and JesúS Bernal. "A collaborative filtering approach to mitigate the new user cold start problem." Knowledge-Based Systems 26 (2012): 225-238
[31] Nguyen, An-Te, Nathalie Denos, and Catherine Berrut. "Improving new user recommendations with rule-based induction on cold user data." In Proceedings of the 2007 ACM conference on Recommender systems, pp. 121-128. ACM, 2007
[32] Mohan, K., & Fazel, M. (2010, June). Reweighted nuclear norm minimization with application to system identification. In American Control Conference (ACC), 2010 (pp. 2953-2959). IEEE.
[33] Lawson, C. L., & Hanson, R. J. (1974). Solving least squares problems (Vol. 161). Englewood Cliffs, NJ: Prentice-hall
[34] Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In Recommender systems handbook (pp. 257-297). Springer US.
[35] Xu, Y., & Yin, W. (2013). A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM Journal on Imaging Sciences, 6(3), 1758-1789
[36] Kabbur, S., Ning, X., & Karypis, G. (2013, August). Fism: factored item similarity models for top-n recommender systems. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 659-667). ACM
[37] Mnih, A., & Salakhutdinov, R. (2007). Probabilistic matrix factorization. InAdvances in neural information processing systems (pp. 1257-1264).
[38] Lee, Hyekyoung, Jiho Yoo, and Seungjin Choi. "Semi-supervised nonnegative matrix factorization." Signal Processing Letters, IEEE 17, no. 1 (2010): 4-7.
[39] Pereira, Andre Luiz Vizine, and Eduardo Raul Hruschka. "Simultaneous co-clustering and learning to address the cold start problem in recommender systems." Knowledge-Based Systems 82 (2015): 11-19

BaSE(Byte addressable Storage Engine) Transaction Manager

Sathyanarayanan Manamohan
Hewlett-Packard Enterprise
sathya@hpe.com

Krishnaprasad Shastry
Hewlett-Packard Enterprise
krishnaprasad.shastry@hpe.com

Shine Mathew
Hewlett Packard Enterprise
shine.mathew@hpe.com

Ravi Sarveswara
Hewlett-Packard Enterprise
ravi.s@hpe.com

Kirk Bresniker
Hewlett-Packard Laboratories
kirk.bresniker@hpe.com

Goetz Graefe
Hewlett-Packard Laboratories
goetz.graefe@hpe.com

Abstract

Non-Volatile Memory (NVM) is an emerging memory technology that combines the best properties of current hard disks and main memories by providing non-volatility, high density, high speed, and byte addressability. This provides an opportunity to redesign systems and their software stacks to improve performance and to reduce the complexity. Present-day database systems are designed and optimized for traditional disks and memory hierarchies. They are very complex because they handle varying levels of storage latencies, from CPU caches to hard disks. Our intention is to build a prototype storage engine that is optimized for NVM and which takes advantage of the collapsed memory hierarchy. We are developing this storage engine in an incremental way. In this paper, we describe a novel approach to optimize write-ahead logging (WAL) for NVM based systems.

Most database systems use ARIES-style write-ahead logging to implement transactions. ARIES techniques are optimized for disk based systems and tuned for the sequential write performance of disks. We leverage the high speed, byte-addressable random access of NVM to design a high-performance logging mechanism. We discuss the bottlenecks of sequential logging, identify the challenges of distributed logging and propose a novel solution. We show that NVM-optimized logging improves performance 8-15 times over default MariaDB/XtraDB for log-intensive workloads.

1. Introduction

Transactions are an essential part of OLTP data management systems. Strong transactional support is crucial for supporting the operational activities of all businesses. A significant amount of research effort is dedicated to the design of efficient, reliable and scalable transactions. A key research focus area in transaction processing systems is the support of ACID properties. The transaction systems use write-ahead logging or shadow copy and concurrency control techniques to support ACID properties. Traditional database systems, which support strong ACID compliance most commonly use write-ahead logging. The non-relational data management systems, also popularly called as NoSQL, support eventual consistency properties based on CAP theorem [15] and commonly use shadow copy.

Traditional database storage engines can be divided into four important modules based on functionality. They are Access methods, Log manager, Lock manager and Buffer pool. These four modules account for about 80% of CPU cycles when the database system runs entirely in memory [1].

Most of the transaction processing system implementations rely on DRAM for performance and disks for persistent storage. The data structures and the algorithms are optimized for this type of memory hierarchy. The advent of NVM provides opportunity to redesign and optimize the data structures and algorithms to use a single layer of flat memory. We evaluate various properties of NVM and the opportunity it provides to redesign the transaction management systems that are used in relational databases.

The reminder of the paper is organized as follows. The next few sub-sections provide the background on NVM, transaction management system components and a specific implementation in an open source RDBMS,

MariaDB [13]. Section 2 captures some of the related work to optimize the transaction systems. Section 3 discusses the implementation of write ahead logging on NVM and its benefits. Section 4 mentions the prototype implementation and the results. The final section contains our summary and conclusions.

1.1. Non-volatile memory

Non-Volatile Memory (NVM) is an emerging memory technology that combines best properties of today's hard disks and main memories. It offers non-volatility, high speed and byte addressability [19]. There are many different forms of NVM technologies such as Phase Change Memory (PCM) [16], Spin-Transfer Torque RAM (STT-RAM) [17] and Memristor [18] that are being developed actively. Each of these uses a different underlying technology. They exhibit different characteristics in terms of read/write latency, endurance, energy efficiency etc.

However all these NVM technologies offer byte-addressability, high speed read/write access that is comparable to DRAM and storage capacities that is comparable to HDD or SSD.

We design our solution based on these generic properties. We are not dependent on any specific NVM implementation. It is a conscious attempt to make the solution NVM-technology neutral.

Byte addressability provides an interesting programming model as it allows programs to persist data objects directly on NVM without converting them into disk format. We leverage this aspect in our solution to avoid maintaining multiple formats of data and having to convert between them.

There are many different areas in which NVM can be used in a system architecture. The straight-forward option is to use NVM as a disk replacement, which maintains the programming semantics and thus involves the least amount of changes to the software layer. But, in general, this approach does not work because it upsets the optimization and fine tuning for hard disks. The next option is to use NVM as fast cache in between disk and DRAM. The third option is to use NVM alongside DRAM in same address space. This is the most interesting one because the CPU can use load and store operations to access NVM directly. We develop our solution with the assumption of a direct load/store model.

1.2. Transactional logging in relational and non-relational systems

Transaction support is essential in all data management systems for handling failures and reducing the impact of failures on the system's overall behavior. We consider everything from file systems to complex RDBMS in the scope of transactions for data management. Several methods have been adopted to handle failures ranging from protecting the metadata for single operations, like it is done in file systems, to complex multi-operation, multiple data entities in RDBMS. There are also complex combination of techniques that attempt to provide ACID properties to file system operations [7][8].

Irrespective of the use case, the systems have to be engineered from the ground up to support solid transaction semantics and robust failure recovery. RDMBS are built ground up with transaction semantics, but these design choices made them very complex and rigid. These, to some extent contributed to the development of NonSQL solutions that are more flexible. But they make other design choices that, in turn, make it very difficult to support ACID properties. Hence, flexibility and performance of the transaction management system are essential design parameters for any high-performance transaction manager. The emergence of non-volatile memory gives us an opportunity to re-design the transaction managers to achieve these goals and make them suitable for the data management systems of the future.

1.3. Limitations of WAL

Most database systems use ARIES [2] style write-ahead logging (WAL) to implement transactions. ARIES design decisions are made to get optimal performance for disk based systems. ARIES adopts a centralized logging and optimizes it to leverage the sequential write performance of disks. To hide the performance difference between DRAM and disks the log records are cached in DRAM and forced to disk at the time of commit. This creates a two-layered logging system.

The centralized logging with two-layered design causes several bottlenecks. Aether [3] identifies four types of delays that impact the logging performance: (a) IO related delay; (b) excessive context switching; (c) log induced lock contention; and (d) log buffer contention.

Several techniques have been developed to address these problems on the traditional assumptions of slow block oriented disk and byte-oriented DRAM. The NVM technologies open up a new opportunity to optimize WAL for byte-addressable persistent memory.

1.4. Logging in XtraDB

XtraDB [14] is a transactional storage engine for MariaDB. A transactional storage engine, in MariaDB's context, is a pluggable software module that performs various data management operations, such as create, insert, update and delete, on the data that it manages in a transactionally consistent manner. It uses the concept of write ahead logging to manage transactions. XtraDB maintains the transaction logs in DRAM as a circular buffer and a persistent copy of the same content in a flat file on disk. All database operations that are performed by the storage engine to manipulate the database's pages are logged in a Redo log prior to the actual execution. The

contents of the redo logs are flushed to disk before transaction commit, in line with WAL semantics. The redo log is used during system recovery. In XtraDB the system recovery starts off by replaying the redo log on to the buffer pool until all the database pages are recovered. This replay starts from the last successful checkpoint. Once this replay is over, the undo log is used to rollback all partially complete transactions.

XtraDB maintains the undo log in memory and on disk. The undo logs contain the before images of database records that were modified by a transaction. The undo logs in XtraDB are used to support transaction rollback, database flush of dirty pages into disk and the concept of MVCC to optimize read performance. The data itself is maintained as a B-tree on disk and a hash table in an in-memory buffer pool. The redo log and buffer pool are flushed periodically flushed into the disk. We use XtraDB as the vehicle to demonstrate the effect of design changes as it is extensively used commercially and it is open source. This allows us to run relevant benchmarks on the solution and prove the solution on real-world applications.

2. Related work

Aether [3] identifies a set of challenges with WAL – (a) I/O related delays, (b) log induced lock contention, (c) excessive context switching and (d) log buffer contention. The paper recommends a set of optimizations using a combination of early lock release and flush pipelining. Early lock release allows transactions to release their locks as soon as the commit records have been made durable. Flush pipelining helps to reduce the I/O delay and log induced lock contention. The authors also recommend redesigning of log buffer to enable better parallelism.

Fung R. et al. [4] describe an approach to implement WAL on storage class memory (SCM). They allocate log records directly on SCM to reduce I/O related delays. They avoid the techniques like group-commit. However, they still write the log records sequentially.

MARS [5] moves most of the logging functionality to hardware and eliminates the Log Sequence Number (LSN) and log checkpointing. It accomplishes this by allowing the storage array to maintain the ordering of write at commit time instead of maintaining the LSN at a software level. MARS also relies on hardware writes to eliminate the need for log checkpointing.

There have been proposals to revisit the design of logging in Flash and PCM based storage [9][10]. Sang-won Lee et.al. propose a technique called in-page logging (IPL) as a new storage model for flash based database systems. To overcome the erase-before-write limitation of flash memory, they propose the IPL technique to co-locate the log and data pages. The IPL-P paper is an extension of the in-page logging method and proposes move the log storage to PCM based storage for better performance.

Tianzheng Wang et al. [20] describe an approach for distributed, NVM-backed logging. They evaluate the performances of two log distribution schemes – one that is distributed on a transaction level and another that is distributed on a page level. They recommend distributing on a transaction level. We show using a transaction level distribution for the undo log and a page level distribution for the redo log performs better.

Jian Huang et al. [21] show that they get the best transactions per dollar rate by moving only the logging subsystem to NVRAM, rather than replacing all disks with NVRAM. In their implementation, they use a circular log on NVRAM and chain the log entries using pointers to enable recovery. However, this slows the recovery process because the entire log chain must be traversed to build the set of dirty pages. This approach to page-level recovery is not efficient.

3. Our solution

We have designed a NVM-optimized logging system that writes log records directly to NVM and have implemented a prototype based on MariaDB/XtraDB.

In the context of our solution, we assume the NVM is directly accessible by the database process using load/store model. We treat it as a byte-addressable persistent memory alongside DRAM memory.

Our design depends on system primitives and programming APIs to read and write the NVM. These APIs shall: (a) support namespaces for NVM; (b) support dynamic memory management; and (c) support variable length read/write operations in an atomic and durable way. There are challenges in implementing variable length atomic read/write operations. The implementation has to take care of write ordering, cache flushes, fault zones, fault containment in NVM etc. Our implementation assumes these challenges will be abstracted and handled by the NVM programming APIs. Atlas [19] is an example of such an API that provides the necessary atomicity guarantees and also handles the memory management. We did not have access to this library at the time of this work and so, we modeled NVM access using mmap files.

There are two main types of log records that are used in traditional ARIES-style WAL: (a) undo log records, which store information about how to undo a change; and (b) redo log records, which store information about how to reproduce a change. In traditional database systems, the log records are cached in memory and persisted on disk as a large sequential file. The log files are flushed to disk before the transaction commits. Database systems employ several optimizations to improve the performance of log writing, such as group-commit [6], which aggregates multiple log write requests into a single IO, and asynchronous commit, which allows transactions to commit without waiting for log IO requests to complete.

We write log records directly on NVM. This eliminates the IO related to log records and also simplifies

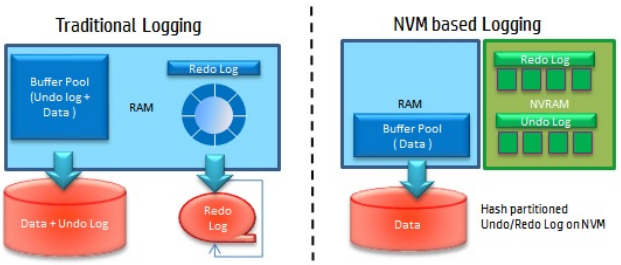


Figure 1: Logging on NVM

log buffer management. XtraDB has separate memory space in DRAM for redo and undo log records. The undo log records are stored along with page data in the buffer pool. The redo log records are stored in a separate circular buffer. In our solution, we do not maintain any of the log records on DRAM. We maintain two separate hash lists in NVM for the redo and undo records. Figure 1 shows log files representation on default MariaDB/XtraDB and our NVM optimized log manager.

As writes to NVM are very fast, we write the log records synchronously. Persisting of the redo data is synchronous with the commit. We do not wait for a separate thread to flush to disk before finishing the commit operation. This reduces the extensive context switching that occurs due to log-record related IO operations in traditional systems.

Traditionally, log records are cached in DRAM using an in-memory format and stored on disks in a block-oriented format [22]. The log records are converted from the memory format to the disk one while persisting them. The disk format is converted to the memory format when the log records are read for recovery. We avoid the multiple formats and implement a single unified format of log records on NVM. Thus we avoid the extra memory copy and log record conversion complexity.

Synchronous write also allows our solution to eliminate the group commit. This in turn reduces the log induced lock contention.

To overcome the log-related contention, we have re-designed the redo and undo logs. The operations on these log files exhibit different degrees of parallelism. We have designed them with different parallelism schemes after taking their usage into account. The undo operations are applicable for a transaction and hence can be parallelized at transaction level. The redo operations are applicable for a page and hence can be parallelized at page level. We observe these parallelism needs of undo and redo operations and design a customized distribution scheme. We distribute undo log records based on transaction ID and implement it as a linked list of undo records belonging to a transaction. We implement a hash based distribution of transaction IDs.

Figure 2 shows the structure of undo log records. The information about transaction state and pointer to undo log records are maintained in a hash table. The undo log records, that contain undo number, page number, LSN

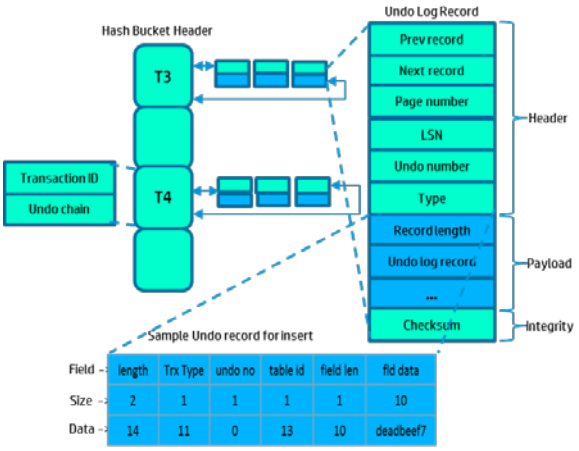


Figure 2: Undo log record structure

and undo operation details (called as payload) is implemented as a linked list. This eliminates the contention for writing undo log records from multiple transactions. Only during the beginning of a transaction we need to acquire a lock to get the corresponding hash slot. Thus we improve the concurrency of undo log operations.

Similarly we distribute the redo log records based on page id. Figure 3 shows the structure of redo log records. We store the details of page number, start and end LSN of the page and pointer to redo chain in hash bucket header. We store redo records that consists of transaction ID, record type, LSN, record payload etc. as a linked list. This reduces the redo log write contention across the pages. Only the parallel transactions that operate on the same page contend for the redo log chain. With this customized distribution of redo and undo log records we can implement more granular latches and increase the parallelism in logging operations. This reduces the log buffer contention and improves the performance.

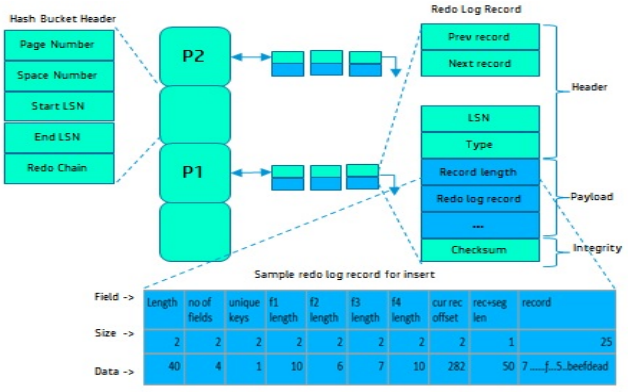


Figure 3: Redo log structure

ARIES recommends periodic checkpointing to accelerate the recovery. The checkpointing flushes the log records and dirty pages in buffer pool to disks. The checkpoint log record holds information about active transactions, its state and flushed LSN. Since the active transactions and their states are directly written to NVM

along with undo logging, the checkpoint log record has to just write the flushed LSN. This improves the checkpointing performance.

Our parallel hash based distribution of the redo and undo log records opens up the opportunity to parallelize recovery operations. ARIES recommends the recovery in 3 phases: (i) analysis phase – during which the algorithm reads the flushed LSN information from checkpoint and scans the log records sequentially to gather the required redo and undo operation information, (ii) redo phase – during which the redo operations are applied to bring the database back to the state before the crash; and (iii) undo phase – during which undo operations are applied to reverse the effect of inflight transactions. Our distributed redo log records enable parallelism in building and applying redo operations. Distributed undo log records enable parallelism in rolling back the inflight transactions. Thus we improve the recovery performance. The benefits of partitioning log records are explained in the following sections.

3.1. Partitioning log structures

Partitioning of log structures addresses several bottlenecks that are seen with sequential logs.

Centralized log structure is well suited for disk based systems. In such systems, log records from multiple transactions are logged into a centralized log structure. These log records are persisted on disk using the sequential IO. Optimizations like group-commit are done to further improve the IO performance. But a centralized log creates synchronization problem. Multiple transactions that run in parallel will contend to get the lock at the head of the log structure to write log records. The contention increases with number of parallel transactions and causes a concurrency limitation. Since the threads have to wait for the lock they get context switched. This also increases the number of context switches and impacts the performance.

The centralized logging will also impact the parallelism of recovery operation. To parallelize the recovery operation, the recovery system has to process the sequential log to extract the undo and redo information into some partitioned structure. Typically the recovery system does this during the initialization phase. Otherwise, the system would be scanning and applying the sequential logs one record at a time, which will significantly increase the recovery time.

Our design avoids these problems by partitioning the undo and redo logs in NVM. We use hash based partitioning scheme. By hashing we break up the single centralized log head into several streams equal to the bucket size of hash table. This eliminates the bottleneck on global log by parallelizing access to the logs. This will enable several threads to write to the log records simultaneously. Consequently, this reduces log induced contention and context switching.

Also having to separate logs structures for undo and redo logs allows us to partition the log in the most optimal way based on the use case. Undo logs are closely associated with a transaction hence we partition it based on transaction Id. This allows us to create sufficiently large hash buckets to the extent where we can completely eliminate the need of synchronization constructs and make the undo logging practically lock free. As an example, if we optimize the system to handle 2K concurrent transactions, we can create an undo log hash table having more than 2K (closest prime number) bucket. In this way every transactions will get its own exclusive hash bucket there by eliminating the need to have a synchronization construct to manage the undo log. This also simplifies the search of undo logs during recovery operation, as all the undo logs pertaining to individual transaction that needs to be rolled back during recovery are found grouped in the same hash bucket.

Similarly redo logs are partitioned on page ID that allows several database threads to operate in parallel as long as they don't try to append redo log records on the same page. Due this approach, optimizations like grouping redo logs prior to append into the global logs are no longer needed. Transactions can directly append the logs into the log structure directly as and when they are generated. This also simplifies the recovery operation. During recovery multiple threads can be created to recover the pages in parallel.

3.2. Benefits of undo and redo log optimizations

In this section, we explore the implications of the design optimizations that we explained in the previous section. The first implication is the simplification in the process of releasing locks that were held by the transaction. Transactions acquire locks to protect the data that it is using, against possible corruption from concurrent access. This ensures the isolation guarantee of the database is maintained. Lock release happens at the end of a transaction when the commit status of the transaction is flushed into durable media. In traditional systems, we have to wait for the flushing of commit records to complete. This creates an I/O bottleneck. Transactions are made to wait [3] [6] for the grouping of log data to be sufficient enough to overcome the cost of a doing a serial I/O to disk. Our solution eliminates this completely because the logs are directly written to persistent media in their native form. Our solution does not maintain two distinct data structures, one where we buffer the logs and the other that is used to do bulk I/O to the disk. This approach also simplifies the code. Due to this design attribute of our solution, locks can be released as soon as the commit record is posted into the data structure.

The second implication of our design is the reduced context switching. Most modern databases are multi-threaded to take advantage of the abundance of compute

cores available in state-of-the-art CPUs. However, even though this is largely beneficial, due to I/O and synchronization bottlenecks, much of the compute cycle is wasted in context switching and spin locks. Our solution addresses the context switching part of the problem. The synchronization problem is tougher to handle as it requires a complex redesign of XtraDB to resolve synchronization bottlenecks. In our solution, the persistence of log records is now reduced to a write operation to a NVM resident data structure. This now can be done in the same thread, without having to wait for I/O operations, which are usually done by other I/O threads [3]. This optimization allows us to drive the cores to do more user work rather than waiting for I/O operations to complete.

The third implication of the design is the elimination of log multiplexing, which attempts to combine logs records from various transactions to achieve optimal volume to make flush operations efficient. On disk based logging systems [2] [3] [6] it is used as a method to reduce the I/O overhead for writing to the disk. In flash based systems [9] [10] it is used as a method to reduce erase-unit overheads. In both the systems it is quite possible that the system writes more data than what was actually updated. This arises due the block oriented nature of writes on these systems. On disk based systems the block is usually 4-16 KB and on flash based systems the block size (erase unit size) is 128KB. Since our solution relies on byte-addressability we write variable log data without having to worry about block boundaries. This results in faster commit time and better utilization of cores to do more useful work.

3.3. Implications on checkpointing operation

In data management systems, a checkpoint can be treated as a marker that indicates the extent to which state information has been transferred to the secure persistent storage. Modifications to data pages are not necessarily flushed to disk in a synchronous manner for performance reasons. Checkpointing is a costly operation and has serious impact on the throughput of the system. Checkpoints are classified into two categories: full checkpoint and fuzzy checkpoint. In a full checkpoint, the data management system writes all the dirty information to the disk. The fuzzy checkpoint, which is commonly used for performance reasons, writes only a certain number of dirty pages. Fuzzy checkpoints are used in XtraDB.

Checkpointing in NoSQL solutions, like HBase, are more like full checkpoints where the entire of the old version metadata and the logs are combined synchronously on a standby node to a create the newer version of the metadata that is then gradually transmitted to all the active node servers in the system. What ever the method used, the system will experience a drop in performance for the duration of the checkpointing operation. In our solution, due to NVM, the flush of the redo logs is completely eliminated and the fuzzy checkpoint needs to maintain only the state of the pages that were flushed from the buffer pool. This also simplifies page stealing because the logs are already persisted and hence the buffer pool manager has the freedom to pick up dirty pages on demand. This makes the checkpoint process and page stealing simpler and faster.

3.4. Implications on crash recovery operation

Recovery or crash recovery operation rebuilds the internal data structures of the data management system to a consistent state from which the storage engine can start processing transactions again. The recovery process also ensures that the overall consistency of the data is maintained. In XtraDB, the recovery happens in several steps. These steps are semantically similar for any data management system which supports crash recovery. The first task is to bring the data pages that were present in the buffer pool without being flushed to persistent media, back to a consistent state. This is done by applying the redo log from the last checkpoint forward until all of the redo log is exhausted. This process brings the buffer pool up to the state just prior to the point of failure.

However the buffer pool also contains dirty data pages that are part of incomplete transactions. These transactions need to be rolled back. The undo logs are used to perform this operation. There are several variants in the recovery process based on the richness of the redo and undo information stored by the system prior to crash. In the case of file systems, the recovery is usually limited to metadata. In more complex systems, higher levels, that might include actual file data, are supported [7][8]. In NoSQL systems, the recovery is only in the forward direction as they typically do not store undo information. These systems rely on replication and some variation of voting to get the data to a consistent state eventually. RDBMSes have richer information in their logs and hence, can restore the database to the closest possible state prior to the crash, when compared with all other data management systems.

In our solution we support both undo and redo logs. Hence our transaction manager can be used to perform RDBMS-like recovery. The primary performance bottleneck with recovery operations is the time spent in doing lots of random IOs to get the buffer pool back to a state where undo information can be used. Also, processing and converting the block based redo logs on disk to a format that is usable in DRAM impacts recovery performance. It should also be noted that the system is not available for transactions until the buffer pool is restored by the redo log.

In our solution, hash partitioning the redo log on PageID enables parallel recovery of pages. Instead of reading a serial redo log, recovery threads are assigned to

process several hash buckets in parallel. Our solution also has a single format in which the redo logs are stored hence the cost of converting disk based structure to a DRAM based structure is completely avoided. Also, undo of in-flight transactions can be parallelized because the undo log records are hash partitioned on TranasactionID. These improvements can significantly reduce recovery time of the system that uses our transaction manager.

4. Performance evaluation

We implemented the techniques described in section 3 on MariaDB/XtraDB storage engine. We used a simulated NVM environment for the validation and performance measurement.

Our prototype was built on a Linux machine with 16 GB of RAM running 8 CPU Xeon Processors with 2 cores. The system had separate disks for the host operating system, and the data and log files used by XtraDB. All disks were standard 7200 RPM 200 GB IDE disks.

4.1. Storage engine development

We prototyped our log manager and integrated it with MariaDB/XtraDB. MariaDB is a popular open source fork of MySQL. We wanted to test our solution on a commercial database software to get a better understanding of the implications of a new log manager in the real world operation environment.

We followed a modular design approach to develop the new log manager. The interactions with the NVM device and memory management functionalities were developed as a pluggable component. This provides an easy option to plug in different NVM technologies. We isolated the creation, modification and management of redo and undo log records into a separate module. We defined a set of APIs to interact with the log manager. The parallelization of undo and redo log records, maintenance of hash structures are all contained in the log manager code. The log manager is integrated with XtraDB using the APIs. And the XtraDB code is modified to write log records using the log manager interfaces. The recovery module of XtraDB reads the log files from disk and prepares an in-memory structure to parallelize the recovery option. We bypassed this layer as the log records are already partitioned according to their usage pattern.

We configured XtraDB data files and log files on separate volumes. This helped us to understand the I/O characteristics of the workload. Separation of the log volume from the data volume helped us to understand the characteristics of logging and make a good performance comparison.

4.2. NVM simulation

At the time of prototyping, we did not have access to a NVM device. We simulated the NVM using the Linux file system. We used this NVM simulation to demonstrate both the functionality and performance gains.

To demonstrate the parallel logging and recovery, we implemented the log manager using mmap files. The redo and undo log records are persisted on mmap-ed files. We manually crashed the system to force the recovery operation. During the recovery process, XtraDB reads the log records from the mmap files to reconstruct the undo and redo operations.

We implemented log records on mmap files from tmpfs to demonstrate the performance gain with NVM. Memory mapping tmpfs files avoids the IO operation. This simulates the implementation of log records with read/write access latencies of memory which is an idealistic NVM environment. In reality there might be different write/read speeds for NVM systems.

4.3. Performance test bed

We used the TPC-C [12] benchmark and a custom built insert workload to evaluate performance of our solution. The TPC-C benchmark is an online transaction processing benchmark. It is based on an order-entry use case with several complex transactions that simulate real-life operations on a typical order-entry system.

We developed a custom built parallel insert program to simulate logging intensive workloads. The program inserts 2 million random records of 80 bytes length using tunable number of concurrent threads.

We measured 3 dimensions of performance: a) Elapsed time to complete a fixed number of inserts; b) the CPU and I/O utilization characteristics for the inserts; and c) End-to-End transactions per second on TPCC workload. The results are discussed in the next section.

4.4. Results

We ran the insert program with 1, 2, 4, 8 and 16 parallel threads. We measured the overall time taken to ingest 2 million records by the standard XtraDB program and our modified XtraDB that has our log manager (henceforth referred to as BTM.)

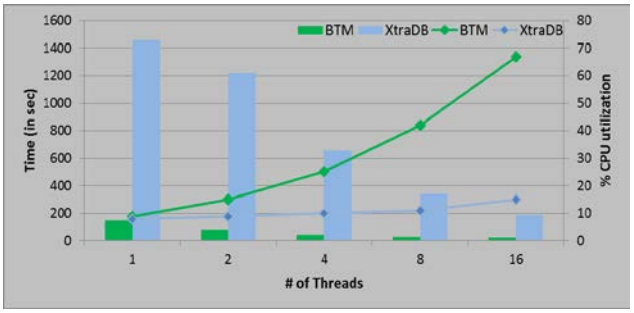


Figure 4: Performance comparison of BTM and default XtraDB

Figure 4 shows the execution time comparison for this insert program on the standard XtraDB and our BTM. We demonstrate 8-15 times improvement in performance.

We also measured the CPU utilization. We get good CPU utilization with BTM. We enable more processing by eliminating IO and lock bottlenecks. This reduces the time required to finish the work and results in higher throughput. In the case of the default XtraDB code, the CPU utilization does not go above 15%. The threads spend most of the time waiting on either IO or on locks for log records.

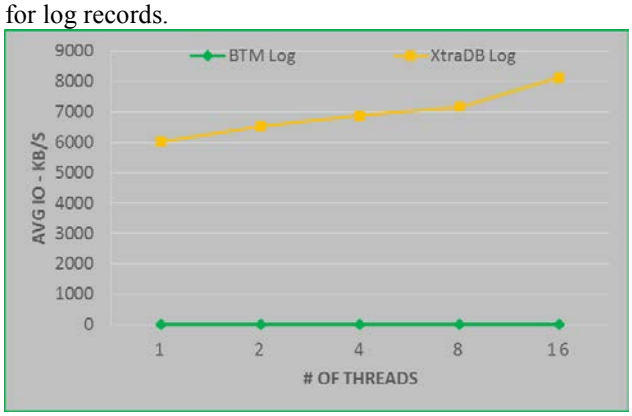


Figure 5a: Log IO comparison of BTM and default XtraDB

Figure 5a and Figure 5b show the IO operations on both data and log volumes. Figure 5a indicates the elimination of log IO in BTM. By eliminating wait times and lock contentions, BTM is able deliver better throughput and process more data. This is seen in Figure 5b.

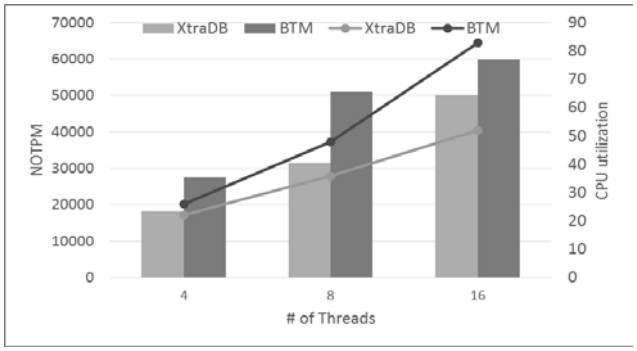


Figure 6: TPCC performance comparison

Figure 6 shows the performance comparison for TPC-C benchmark and the corresponding CPU and IO utilization. Our implementation eliminates the log IO and improves the CPU utilization. We get around 1.2-1.6 times improvement in the throughput. The improvement is limited by the lock contention in other modules of MariaDB. Similar to insert benchmark, we see good reduction in CPU utilization with BTM log manager. This is again attributed to the reduction in lock contention for log records.

5. Conclusions

In conclusion, our design improves the performance of transaction manger by eliminating disk I/O that is needed for performing log operations. In the process, it simplifies the code and reduces the path length due to elimination of log I/O and synchronization code. It improves throughput of the system by parallelizing access to log data structures and eliminating single entry point bottlenecks. It improves core utilization as now most of the time is spent in doing useful work than waiting for I/O

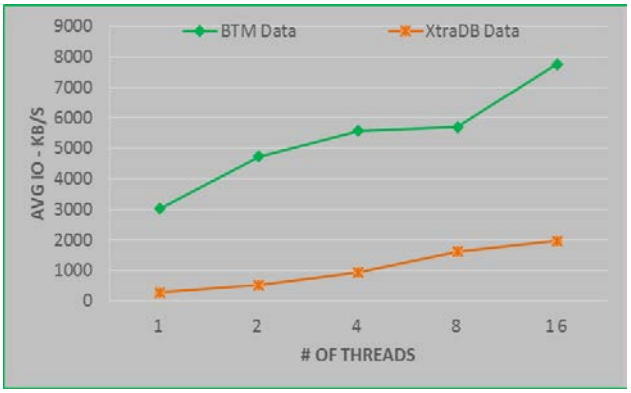


Figure 5b: Data IO comparison of BTM and default XtraDB

completion or latches. It also eliminates the need for techniques like flush pipelining, group commits and early lock release. We have also shown that these optimizations results in 8-15 times improvement in performance and higher CPU utilization for doing useful work.

6. References

[1] Stavros Harizopoulos, Daniel J. Abadi, Samuel Madden, Michael Stonebraker "OLTP Through the Looking Glass, and What We Found There" SIGMOD 2008

[2] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz. "Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging." ACM Trans. Database Syst., 17(1):94–162, 1992.

[3] R. Johnson, I. Pandis, R. Stoica, M. Athanassoulis, and A. Ailamaki. "Aether: a scalable approach to logging." Proc. VLDB Endow, 3:681–692, September 2010.

[4] Ru Fang, Hui-I Hsiao, Bin He, C. Mohan, Yun Wang. "High performance database logging using storage class memory." ICDE, pp. 1221–1231, 2011.

[5] Joel Coburn, Trevor Bunker, Rajesh K. Gupta, Steven Swanson. "From ARIES to MARS: Transaction support for next-generation, solid-state drives." SOSP, pp. 197–212, 2013.

[6] P. Helland, H. Sammer, J. Lyon, R. Carr, and P. Garrett. "Group Commit Timers and High-Volume Transaction Systems." In Proc. HPTS, 1987

[7] Wright, Charles P.; Spillane, Richard; Sivathanu, Gopalan; Zadok, Erez; 2007; "Extending ACID

Semantics to the File System; ACM Transactions on Storage

[8] Spillane, Richard; Gaikwad, Sachin; Chinni, Manjunath; Zadok, Erez and Wright, Charles P.; 2009; "Enabling transactional file access via lightweight kernel extensions"; Seventh USENIX SIGMOD international conference on Management of data Pages 55-66

[10] Kang-Nyeon Kim, Sang-Won Lee, Bongki Moon, Chanik Park, Joo-Young Hwang "IPL-P: In-Page Logging with PCRAM" VLDB 2011

[11] Jim Gray and Andreas Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993.

[12] Transaction Processing Performance Council. "TPC - C v5.5:On-Line Transaction Processing (OLTP) Benchmark."

[13] MariaDB. URL: https://mariadb.org/

[14] XtraDB. URL: https://www.percona.com/software/mysql-database/percona-server/xtradb

[15] Seth Gilbert and Nancy Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59

[16] M. J. Breitwisch. Phase change memory. Interconnect Technology Conference, 2008. IITC 2008. International, pages 219–221, June 2008.

[17] B. Dieny, R. Sousa, G. Prenat, and U. Ebels. Spindependent phenomena and their implementation in spintronic devices. VLSI Technology, Systems and

Conference on File and Storage Technologies (FAST 2009)

[9] Sang-Won Lee, Bongki Moon "Design of Flash-Based DBMS: An In-Page Logging Approach" SIGMOD '07 Proceedings of the 2007 ACM

Applications, 2008. VLSI-TSA 2008. International Symposium on, pages 70–71, April 2008.

[18] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. Nature, (7191):80–83, 2008.

[19] Dhruva R. Chakrabarti, Hans-J. Boehm, Kumud Bhandari. "Atlas: leveraging locks for non-volatile memory consistency", Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, pages 433-452.

[20] Tianzheng Wang, Ryan Johnson. "Scalable Logging through Emerging Non-Volatile Memory," Proceedigns of the VLDB Endowment, Vol. 7, No. 10, pages 865-876.

[21] Jian Huang, Karsten Schwan, Moinuddin K. Qureshi. "NVRAM-aware Logging in Transaction Systems," Proceedings of the VLDB Endowment, Vol. 8, No. 4, pages 389-400.

[22] Jay Janssen. "The relationship between Innodb Log checkpointing and dirty Buffer pool pages." URL: https://www.percona.com/blog/2012/02/17/the-relationship-between-innodb-log-checkpointing-and-dirty-buffer-pool-pages/

Top-K High Utility Episode Mining from a Complex Event Sequence

Sonam Rathore
IIIT-Delhi, India
sonam13108@iiitd.ac.in

Siddharth Dawar
IIIT-Delhi, India
siddharthd@iiitd.ac.in

Vikram Goyal
IIIT-Delhi, India
vikram@iiitd.ac.in

Dhaval Patel
IIT Roorkee, India
patelfec@iitr.ac.in

ABSTRACT

Utility episode mining has emerged as an interesting and challenging research topic in data mining. It finds applications in anomaly detection, biomedical data analysis, predicting stock trends etc. The number of high-utility episodes that can be extracted from a sequence depends upon the value of minimum utility threshold. It is often difficult for a user to find a suitable threshold value which fits their purpose. The sequence can generate many high-utility episodes at low threshold value and very few episodes at higher threshold values. In order to relieve the user from this tedious task, we propose an algorithm for mining top-k high utility episodes from a complex event sequence. The parameter k can be set by the user according to his/her needs. We also propose effective strategies for raising the threshold value in order to prune the search space effectively. We conduct extensive experiments on real and synthetic datasets and the experimental results demonstrate the effectiveness of our proposed strategies in terms of total execution time and the number of candidate episodes generated.

1. INTRODUCTION

Frequent pattern mining finds patterns from a database, which have frequency no less than a given minimum support threshold. Frequent pattern mining finds applications in market-basket analysis, mining association rules [23], plagiarism detection and biomedical data analysis [30]. As types of data vary from one application to another, researchers have developed various frequent pattern mining algorithms. For example, frequent itemset mining [16] deals with transaction data, sequential pattern mining [27] operates on sequence data, frequent episode mining [24] works on long event sequence, and frequent pattern mining in data streams [19]. The algorithms developed for mining frequent patterns have mostly employed the monotone/anti-monotone property to prune the exponential search space effectively. The monotone property states that the subsets of a frequent pattern are also frequent and the anti-monotone property states that the supersets of an infrequent pattern are also infrequent.

However, the frequent patterns extracted can be of low-profit value. The concept of *high-utility pattern mining* was introduced to capture the notion of utility, which has been observed in real life. *High-utility pattern mining* finds patterns from a database which have their utility value no less than a given minimum utility-threshold. The utility function measures the importance of a pattern and varies according to the application. For example, in a retail store domain, a utility function can measure the profit made by the store by selling the items in the itemset together over a period of time. High-utility pattern mining has wide range of applications in cross-marketing in retail stores [8, 20], web-click stream analysis [18], medicine [25] etc. High-utility mining has also been applied with other mining techniques like high-utility sequential pattern mining [34, 31], high-utility pattern mining from a transaction database [5, 7, 10], mining high-utility patterns from a data stream [3], mining utility-frequency skyline pattern [11] and high-utility episode-pattern mining [32].

Mining high-utility episodes from a customer shopping sequence may discover patterns which may help in discovering the purchasing behaviour of customers. In this scenario, the items purchased by a customer in a transaction can be represented as events occurring at a time point. High-utility episode mining can also be used for stock prediction or investment [21, 22]. Some algorithms [26, 14] have been proposed to mine high-utility episodes from simple event sequences, where only one event occurs at a point in time. However, complex event sequences often occur in real life and has many applications as mentioned by Wu et al. [32]. Recently, Wu et al. [32] proposed an algorithm UP-Span to mine high-utility episodes from a complex event sequence. The number of high-utility episodes which can be extracted depends upon the value of the chosen utility threshold and the characteristics of the database. It is possible to extract many episodes at lower threshold values and very few episodes at higher threshold values. The user must analyze the distribution of items, utility value and density of the database in order to choose an appropriate utility value. In summary, a user's engagement while mining high-utility episodes is not a desirable solution.

In this paper, we propose a solution for mining top-k high-utility episodes from a complex event sequence. Our aim is

to relieve a user from the task of analyzing the database and choosing a utility threshold, which is often a difficult task for any user. The parameter k is the number of high-utility episodes to be extracted from the database. A naive approach for extracting top-k high-utility episodes can be to set the minimum utility threshold to zero and apply any high-utility episode mining algorithm to find the complete set of high-utility episodes. Top-k episodes can be then chosen from the result set. However, this approach is computationally very inefficient as the search space is exponential in the number of different items. In order to improve the efficiency, we propose effective strategies to raise the minimum utility threshold from zero as quickly as possible.

Our research contributions can be summarized as follows:

- We propose an algorithm to mine top-k high-utility episodes from a complex event sequence.
- We develop effective strategies to raise the minimum utility threshold quickly during the mining process in order to reduce the search space effectively.
- We conduct extensive experiments on real as well as synthetic datasets and the experiment results demonstrate the effectiveness of our approach.

The paper is organized as follows. Section 2 reviews the related work and background knowledge is explained in Section 3. We propose our algorithm and our strategies for improving the performance in Section 4. The experimental results are presented in Section 5 and Section 6 concludes the paper.

2. RELATED WORK

In this section, we briefly discuss some related work done in this field. We divide the work into two lines of research:

2.1 Frequent Episode Mining

Frequent Episode Mining (FEM) was first introduced by Mannila et al. [24]. Two algorithms WINEPI and MINEPI were proposed in this paper. In WINEPI, the events were sampled regularly over a sequence of events. An episode was considered interesting if it fits into a window width which is defined by the user. The support was computed by counting the number of sliding windows in which episode appeared. However, the algorithm could not avoid the double counting of occurrence of an episode. In order to resolve the issue, the concept of minimal occurrence was introduced. The minimal occurrence of an episode α is a time interval $[t_s,\ t_e]$ where the episode α occurs and it does not occur in any proper subinterval of $[t_s,\ t_e]$. In order to find the support, the algorithm counted the number of minimal occurrences of an episode.

However, the algorithms generate a large of candidate episodes. Several methods have been proposed to improve the performance of existing *FEM* algorithms. However, a majority of the studies are devoted towards mining frequent episodes in simple event sequences [1, 2, 4, 6, 15]. Mining frequent episodes from a complex event sequence were only considered by Huang et al. [17].

2.2 Utility Episode Mining

The *FEM* framework assumes that all the events are of same importance. Therefore, it may report many episodes of low revenue and miss high revenue but low-frequency episodes. For solving this issue, the concept of utility was introduced in episode mining by Guo et al. [13]. However, this paper only considered the external utility of an episode and mining was performed in a simple event sequence. Wu et al. [32] addressed this problem and proposed an algorithm UP-Span to discover high utility episodes in the complex event sequence. The events were associated with internal utility (quantity) and external utility (profit). The authors also proposed two strategies, namely Discarding Global unpromising Events (DGE) and Discarding Local unpromising Events (DLE), to discard unpromising events and reduce the search space. To speed up the UP-Span algorithm, Guo et al. [12] presented a prefix tree structure and tighter upper bounds for candidate episodes utility.

From the above-related work, we can conclude that only preliminary work is done in mining high utility episodes. Many algorithms have been proposed for top-k high-utility pattern mining from transaction [33, 29] and sequential databases [35]. However, the existing top-k utility mining algorithms cannot be directly applied to top-k high utility episode mining on a very large complex event sequence. If we try to transform a complex event sequence into a set of transactions or a set of sequences to make a sequential database, it is not straightforward. Further, this makes the existing algorithms inefficient for top-k utility episode mining from a complex event sequence.

3. BACKGROUND

In this section, we present some definitions given in the earlier works [32] and describe the problem statement formally.

3.1 Episode Mining

Definition 1. ***(Event)*** *An event is defined by the pair (e, t) where e is the event type and $t \in N^+$ is the time at which event occurs.*

Definition 2. ***(Complex event sequence)*** *A complex event sequence $CES = <(SE_1, t_1), (SE_2, t_2), \ldots, (SE_n, t_n)>$ is an ordered sequence of simultaneous event sets, where each simultaneous event set is associated with a time point $t \in N^+$ and $t_i<t_j$, for all $1 \leq i<j \leq n$.*

Let us consider the complex event sequence as shown in Figure 1. $(\langle(D)\rangle, 5)$ is an event which occurs at t_5.

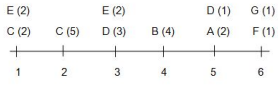


Figure 1: A Complex Event Sequence

A	B	C	D	E	F	G
1	4	2	1	5	3	2

Table 1: External Utility

Definition 3. ***(Simultaneous event set)*** *A simultaneous event set is composed of a set of events, where each event occurs at the same time point t.*

For example, $(\langle(DA)\rangle, 5)$ is a simultaneous event set which occurs at t_5.

Definition 4. ***(Episode containing simultaneous event sets)*** *An episode* $\alpha = <(SE_1), (SE_2), \ldots, (SE_n)>$ *is a non-empty totally ordered set of simultaneous events, where* SE_i *appears before* SE_j *for all* $1 \leq i < j \leq n$.

For example, $\langle (EC), (D) \rangle$ is an episode.

Definition 5. ***(Occurrence)*** *For an episode* $\alpha = <(SE_1, t_1), (SE_2, t_2), \ldots, (SE_n, t_n)>$, *the time interval* $[T_s, T_e]$ *is called the occurrence of episode if* α *occurs in* $[T_s, T_e]$ *and the first simultaneous event* SE_1 *of* α *occurs at time* T_s, *while the last simultaneous event set* SE_k *of* α *occurs at time* T_e.

For example, the occurrence set of episode $\langle (E), (D) \rangle$ is $occ(\langle (E), (D) \rangle) = [1, 3], [3, 3], [1, 5], [3, 5]$.

Definition 6. ***(Minimal Occurrence)*** *The occurrence time interval* $[T_s, T_e]$ *of an episode is called a minimal occurrence if there exists no sub-interval of* $[T_s, T_e]$ *in occurrence of* α. *Consider two time intervals* $[T_s, T_e]$ *and* $[T_s\,', T_e\,']$, $[T_s\,', T_e\,']$*is the sub-interval of* $[T_s, T_e]$ *if* $T_s \leq T_s\,'$ *and* $T_e\,' \leq T_e$.

For example, the minimal occurrence set of episode $\langle (E), (D) \rangle$ is
$minOcc(\langle (E), (D) \rangle) = [1, 3], [3, 3], [3, 5]$.

Definition 7. ***(Support of an episode)*** *The support of an episode is defined as the number of minimal occurrences of an episode.*

For example, the support of episode $\langle (E), (D) \rangle$ is 3.

Definition 8. ***(Internal and External Utility)*** *In mining high utility episodes, each event* e_i *is associated with a positive number* $p(e_i)$, *called as the external utility. Each event* e_i *in a simultaneous event set* SE_j *at the time point* t_j *is associated with a positive number* $q(e_i, t_j)$, *called as internal utility.*

For example, the external utility of events is shown in Table 1.

Definition 9. ***(Utility of an event at a time point)*** *The utility of an event* e_i *at a time point* t_j *is* $u(e_i, t_j) = p(e_i, CES) \times q(e_i, t_j)$.

For example, the utility of event $\langle (F) \rangle$ at time t_6 is $1 \times 3 = 3$.

Definition 10. ***(Utility of a simultaneous event set at a time point)*** *The utility of a simultaneous event set* $SE = \langle (e_1, t_1), (e_2, t_2), (e_3, t_3), \ldots, (e_k, t_k) \rangle$ *at a time point* t_i *is defined as* $u(SE, t_i) = \Sigma_{i=1}^{k} u(e_i, t_i)$.

For example, utility of simultaneous event $\langle (GF) \rangle = (2 \times 1) + (3 \times 1) = 5$.

Definition 11. ***(Utility value of an episode w.r.t. its minimal occurrence)*** *Let* $mo(\alpha) = [T_s, T_e]$ *be a minimal occurrence of the episode* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_n) \rangle$, *where each simultaneous event set* $SE_i \in \alpha$ *is associated with a time point* T_i. *The utility of the episode* α *w.r.t* $mo(\alpha)$ *is defined as* $u(\alpha, mo(\alpha)) = \Sigma_{i=1}^{k} u(SE_j, t_i)$.

Definition 12. ***(Utility of an episode in a complex event sequence)*** *Let* $moSet(\alpha) = [TI_1, TI_2, \ldots, TI_k]$ *be the set of all minimal occurrences of the episode* α, *where* TI_i *is a minimal occurrence of* α *for* $1 \leq i \leq k$. *The utility value of the episode* α *in a complex event sequence* CS *is defined as* $uv(\alpha, CS) = \Sigma_{i=1}^{k} u(\alpha, TI_k)$.

For example, $u(\langle (E), (D) \rangle, [1, 3]) = 10 + 3 = 13$, and $u(\langle (E), (D) \rangle, CES) = u(\langle (E), (D) \rangle, [1, 3]) + u(\langle (E), (D) \rangle, [3, 3]) + u(\langle (E), (D) \rangle, [3, 5]) = 13 + 13 + 11 = 37$.

Definition 13. ***(High Utility Episode(HUE))*** *If the utility of an episode is not less than the minimum utility threshold, then it is called as a high utility episode.*

Definition 14. ***(Maximum Time Duration)*** *Maximum Time Duration (abbreviated as MTD) is a user specified window. A minimal occurrence* $mo(\alpha) = [T_s, T_e]$ *satisfies the maximum time duration constraint iff,* $(T_e - T_s + 1) \leq MTD$.

Definition 15. ***(Simultaneous and Serial concatenations)*** *Let* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_x) \rangle$ *and* $\beta = \langle (SE_1'), (SE_2'), \ldots, (SE_y') \rangle$ *be episodes. The simultaneous concatenation of* α *and* β *is defined as:* $simulconcat(\alpha, \beta) = \langle (SE_1), (SE_2), \ldots, (SE_x \bigcup SE_1'), (SE_2'), \ldots, (SE_y') \rangle$.
The serial concatenation of α *and* β *is defined as:* $serialconcat(\alpha, \beta) = \langle (SE_1), (SE_2), \ldots, (SE_x), (SE_1'), (SE_2'), \ldots, (SE_y') \rangle$.

For example, Let $\alpha = (\langle (B) \rangle, [4, 4])$ and $\beta = (\langle (D) \rangle, [5, 5])$. The new episode formed by serial concatenation of α and β is $\gamma = \langle (B), (D) \rangle, [4, 5]$. Let now $\alpha = (\langle (A) \rangle, [5, 5])$. The simultaneous concatenation of γ and α is $\langle (B), (DA) \rangle, [4, 5]$.

Since downward closure property doesn't hold in utility mining, the authors [32] proposed the concept of Episode-weighted Utilization (EWU). The EWU satisfies the downward closure property i.e. if the EWU of an episode is less than the minimum utility threshold, all its super episodes will have low utility.

Definition 16. ***(Episode-Weighted Utilization of an episode w.r.t a minimal occurrence)*** *Let* $mo_j(\alpha) = [T_s, T_e]$ *be a minimal occurrence of the episode* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_x) \rangle$, *where each simultaneous event set* $SE_i \in \alpha$ *is associated with a time point* $T_i (1 \leq i \leq k)$ *and* $mo_j(\alpha)$ *satisfies MTD. The episode-weighted utilization of* α *w.r.t* $mo(\alpha)$ *is defined as* $EWU(\alpha, moj(\alpha)) = \Sigma_{i=1}^{k-1} u(SE_i, t_i) + \Sigma_{i=e}^{s+MTD-1} u(tSE_i, t_i)$ *where* tSE_i *is the simultaneous event set occurring at time point* T_i *in* CES.

Definition 17. ***(Episode-Weighted Utilization of an episode)*** *Let* $moSet(\alpha) = [TI_1, TI_2, \ldots, TI_k]$ *be a set of minimal occurrences of the episode* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_x) \rangle$, *where each simultaneous event set* $SE_i \in \alpha$ *is associated with a time point* $T_i (1 \leq i \leq k)$ *and* $mo_j(\alpha)$ *satisfies MTD. The episode weighted utilization of* α *in complex event sequence* CES *is defined as* $EWU(\alpha) = \Sigma_{i=1}^{k} u(\alpha, TI_i)$.

For example, let the MTD set by the user be 3.The EWU of $\langle (E), (D) \rangle$ w.r.t a minimal occurrence is $EWU(\langle (E), (D) \rangle, [1, 3]) = 10 + 13 = 23$.

The EWU of $\langle(E),(D)\rangle$ in the CES is $EWU(\langle(E),(D)\rangle) = [EWU(\langle(E),(D)\rangle),[1,3]]+[EWU(\langle(E),(D)\rangle),[3,5]] = 23+13 = 36$.

Definition 18. ***(High Weighted Utilization Episode (HWUE))*** *An episode is called High Weighted Utilization Episode its EWU is no less than the minimum utility threshold $min\_utility$.*

4. EFFICIENT MINING OF TOP-K HIGH UTILITY EPISODES

In this section, we present our proposed algorithm, TUP-Basic (Top-k Utility episode mining), for mining top-k high utility episode mining in a complex event sequence. First, we propose a basic algorithm to find the top-k high utility episodes. Then, we propose some strategies to effectively raise the minimum utility threshold during the mining process.

4.1 TUP-Basic

In this subsection, we present a basic algorithm for discovering top-k high-utility episodes from a complex event sequence (See Algorithm 1). The algorithm takes as input a complex event sequence and a parameter k. The algorithm maintains a sorted list of size K dynamically, which contains the top-k high-utility episodes. The minimum utility threshold ($min\_utility$) stores the utility of the current k^{th} episode. The minimum utility threshold is initialized to zero before the start of mining process as the top-k buffer is empty.

First, the database is scanned once to find all 1-length episodes (Line 1). The minimal occurrence, utility and EWU of 1-length episodes are calculated. If the EWU of an episode is greater than the minimum threshold (Line 4), the algorithm explores the search space of high utility episodes with the current episode as the prefix. The algorithm follows a depth-first search strategy for finding new episodes.

To explore the search space, an episode is concatenated simultaneously (Line 5) and serially (Line 7) to events. Let the occurrence of an episode α be $[T_s, T_e]$. The events occurring at T_e are simultaneously concatenated to the episode α. While the events occurring between T_e+1 to $T_s+MTD-1$ are serially concatenated to episode α according to the definition 15.

Each new episode, β, formed by concatenation, calls Insert_Episode(.) which updates the list of Top-k episodes (See Algorithm 2). The input episode is added to the list if the size of the list is less than user-defined k or its EWU is no less than the minimum utility threshold. Once k candidates are discovered the minimum utility is raised to k^{th} highest utility in the list i.e. to the least utility in the list. Further, only episodes satisfying minimum utility is inserted in the list and the $(k+1)^{th}$ episode is removed from the buffer. After the algorithm terminates, top-K list contains the desired output of top-k high utility episode mining in the complex event sequence.

The algorithm returns the correct result as if the EWU of an episode is less than the utility of the current kth episode, it is guaranteed that no super-set of that episode can be in the top-k buffer due to downward closure property. The top-k episodes for k=3 and MTD=2 is: $\{\langle(ED),(B)\rangle : 29, \langle(E),(B)\rangle : 26, \langle(EC),(C)\rangle : 24\}$

Algorithm 1 TUP-Basic(CES, MTD, k)

Input: A complex event sequence CES,desired number of episodes k
Output: The complete set of top-k high utility episodes

```
1: Scan CES to find all one length episodes (oneLengthEpiSet).
2: min_utility=0.
3: for episode epi in oneLengthEpiSet do
4:     if EWU(epi) ≥ min_utility then
5:         Simultaneous Concatenation(epi, minOcc(epi),
6:         min_utility, k).
7:         Serial Concatenation(epi, minOcc(epi)
8:         , min_utility, k).
9:     end if
10: end for
```

Algorithm 2 Insert_Episode($epi, min\_utility, k$)

Input: an episode epi,minimum utility $min\_utility$, desired number of episodes k
Output: Top-K List: Top-k high utility episodes among the candidates

```
1: if size(Top − KList)<k then
2:     Add epi to Top-k List.
3: else
4:     if utility(epi)>min_utility then
5:         Remove k^th high utility episode i.e. episode
6:         having least utility.
7:         Add epi to the List.
8:         Sort the list in decreasing order of utility values
   of episodes.
9:         min_utility = least utility in Top-K List.
10:    end if
11: end if
```

4.2 Pre-insertion Strategy

The TUP-Basic algorithm generates many candidates since the minimum threshold start from zero. We try to raise the minimum threshold before mining high-utility episodes from a complex event sequence. The idea is to pre-insert the simultaneous event sets, i.e. the event sets occurring at the same time point, in the top-K list after the initial scan of the database. We call this strategy as the pre-insertion strategy. We will illustrate the effectiveness of this approach with an example.

Let us consider the example sequence as shown in Figure 1 and utility of events as per Table 1. Let the value of k be 3. First, the event set, $\langle(EC)\rangle : 14$, occurring at time point 1 is inserted. Similarly, the utilities of other simultaneous event sets are calculated and the event sets are inserted in the top-k list.After the simultaneous event sets are inserted, the top-k list is $\{\langle(B)\rangle : 16, \langle(EC)\rangle : 14, \langle(ED)\rangle : 13\}$. As seen from the example, the pre-insertion strategy increases the minimum utility threshold from zero to 13 before starting the mining process.

4.3 EWU Strategy

The EWU associated with every episode captures the frequency as well as utility aspects nicely. The probability of an episode to be of high-utility increases with its EWU value. The idea is to start exploring those episodes first which have higher EWU compared to others. We know that the min-

imum utility threshold in high-utility mining remains fixed and the order in which paths are explored does not affect the efficiency. But, in Top-k the order of path taken does matter because the minimum utility threshold is dependent on the candidates list. So in this strategy, we process those episodes first which have a higher EWU compared to other episodes. we sort the episodes w.r.t their EWU's before concatenating them serially or simultaneously with other episodes. Our hypothesis is that the EWU strategy will work better on dense datasets compared to sparse datasets as dense datasets generate a lot of high EWU episodes.

We illustrate the working of EWU strategy with an example. The EWU of single episodes is shown in Table 2. Since episode (E) has the highest EWU, it is processed first. Let the value of MTD and k be 2 and 3 respectively. The

E	C	D	B	A	G	F
53	47	37	19	8	5	5

Table 2: EWU of single episodes

simultaneous episode $\langle(EC)\rangle : 14$ is generated and inserted into the top-k buffer. Since no events can be simultaneously concatenated with $\langle(EC)\rangle$, the method for serial concatenation is called. The serial episode $\langle(EC),(C)\rangle : 24$ with utility 24 is generated and added to the top-k buffer. Since, no episodes can be serially or simultaneously concatenated with $\langle(EC),(C)\rangle$, the execution returns to episode $\langle(E)\rangle$ and the simultaneous episode $\langle(ED)\rangle : 13$ is generated and added to the buffer. The minimum utility threshold is set to 13. If we randomly select a path or use lexicographical order the minimum utility threshold may not increase so fast. For example, if we first process the episodes of path starting with episode (A) then the minimum threshold raises to value 3 as the Top-k list consists of episodes $\{\langle(AD)\rangle : 3, \langle(A,G)\rangle : 4, \langle(A,F)\rangle : 5\}$ after traversing this path. So, the strategy 2 helps in raising the minimum threshold efficiently and hence it prunes the search space faster.

5. EXPERIMENTS AND RESULTS

In this section, we evaluate the effectiveness of our proposed strategies. We also study the performance of combining pre-insertion and EWU strategies, which we refer as TUP-Combined. We conduct experiments on various real and synthetic datasets. The description of the real datasets is shown in Table 3. A transaction database can be considered as a complex event sequence by considering each item as an event and the items in a transaction as a simultaneous event. We implemented all the algorithms in Java with JDK 1.7 on a Windows 8 platform.

Table 3: *Characteristics of Real Datasets*

Dataset	#T_x	Avg. length	#Items	Type
ChainStore Small	10,000	7.2	46086	Sparse
Retail Small	10,000	5.2	16470	Sparse
Accidents Small	10,000	10	468	Dense
Mushroom	8,124	10	119	Dense

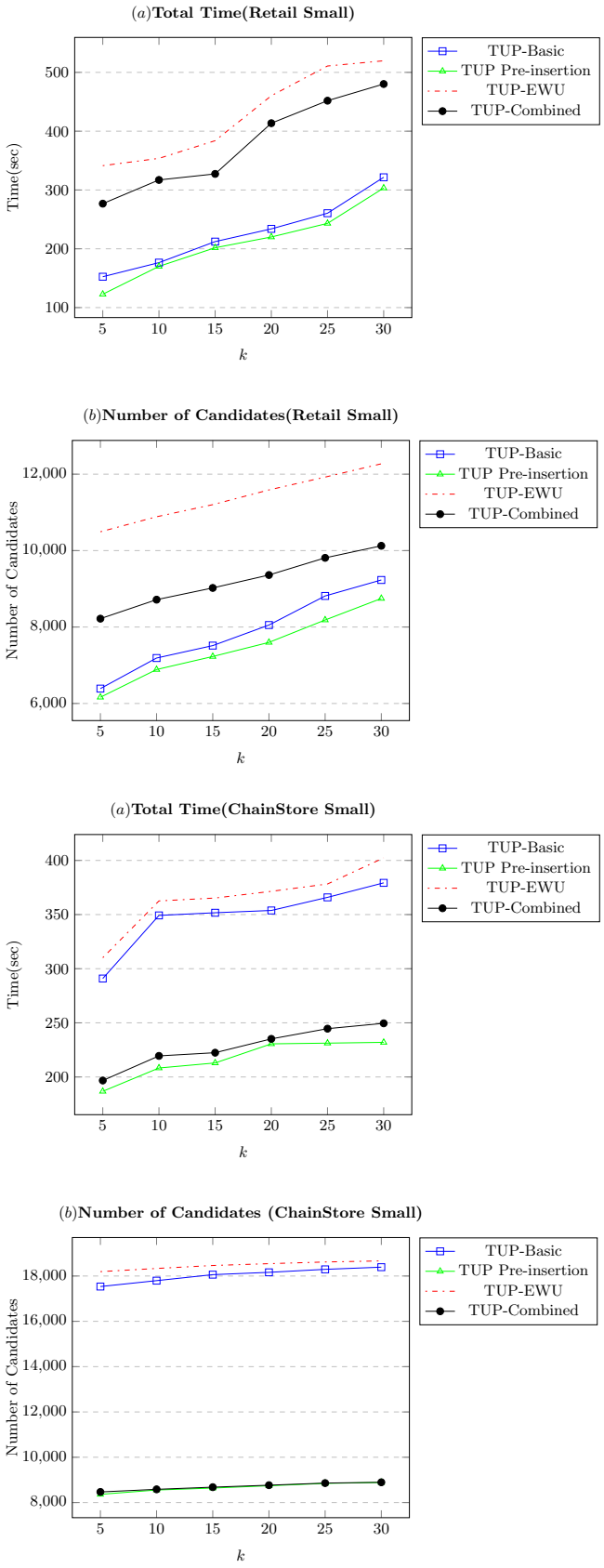
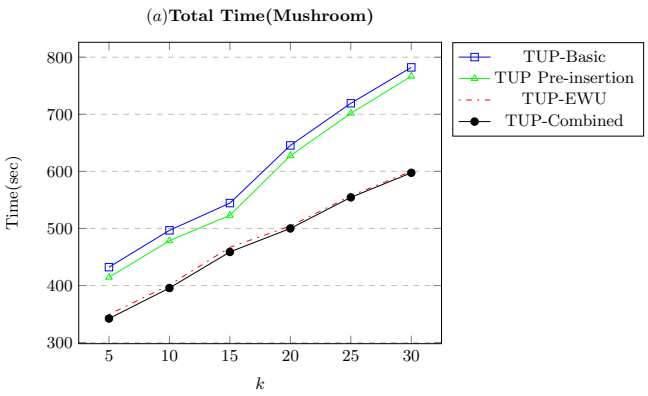
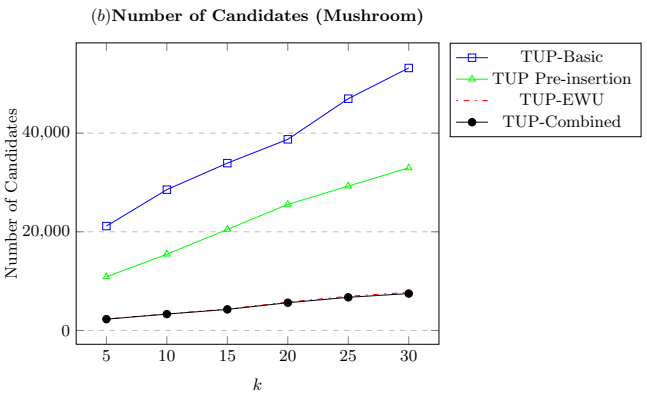
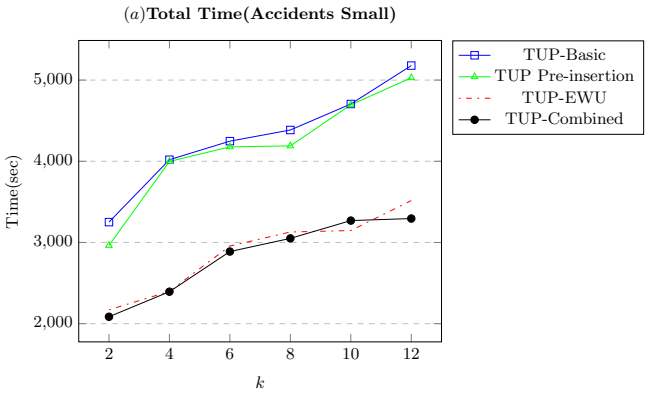


Figure 2: Performance Evaluation on Sparse Datasets







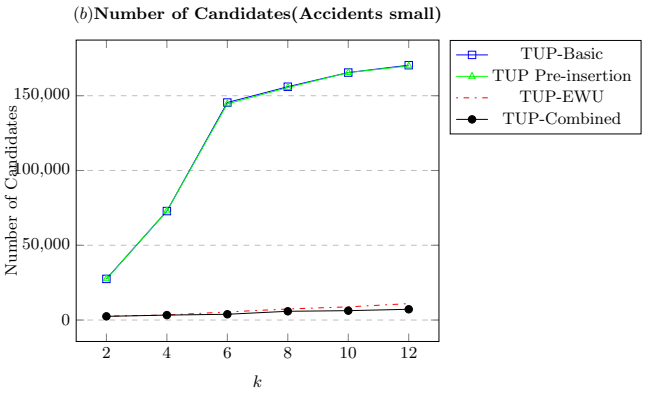
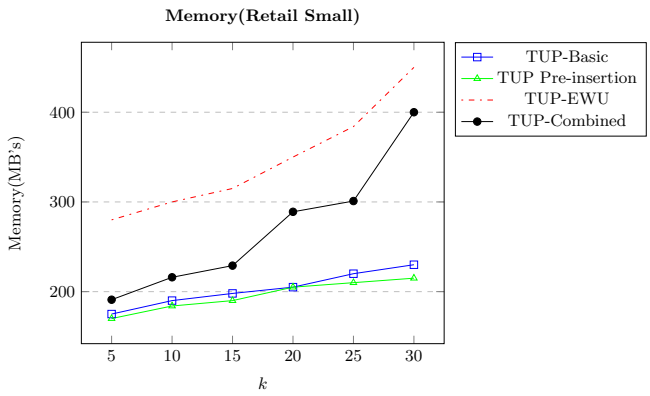
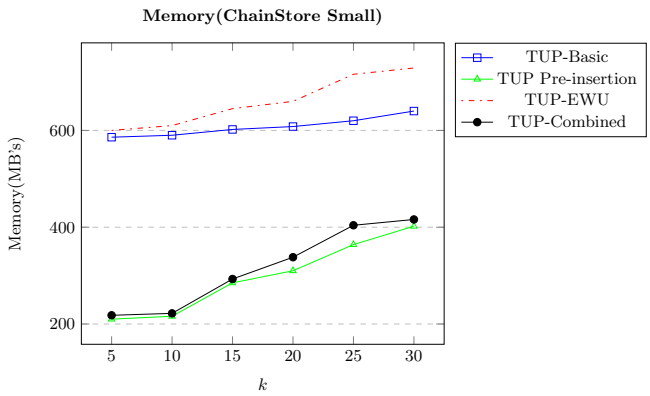
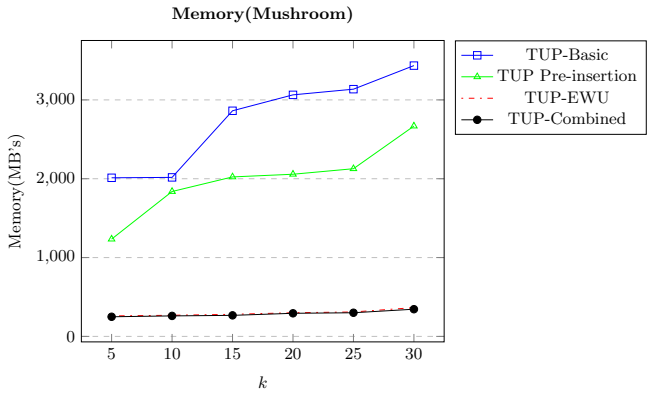


Figure 3: Performance Evaluation on Dense Datasets







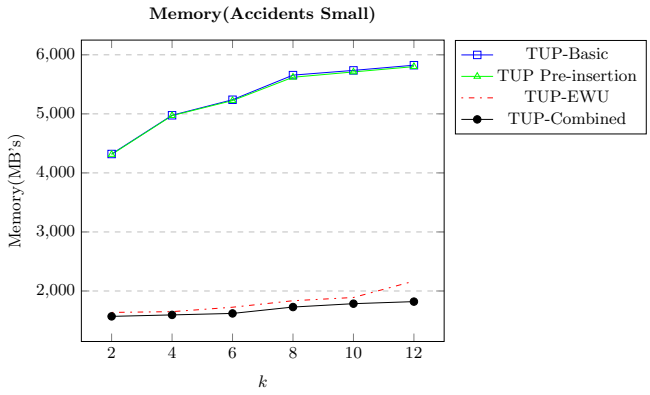


Figure 4: Memory Consumption on Real Datasets

The experiments were performed on an Intel Xeon(R) CPU=26500@2.00 GHz with 64 GB RAM. All real datasets except ChainStore were obtained from FIMI Repository [9]. The ChainStore dataset was obtained from Nu-MineBench 2.0 repository [28]. The quantity information for items was chosen randomly from 1 to 5. The external utility values were generated between 1 to 1000 using log-normal distribution.

We compared the performance of the algorithms on the basis of total execution time as well as the number of candidates. For datasets except Mushroom, we only take the first 10,000 transactions as it takes a lot of time to run experiments on a bigger sequence. Only ChainStore dataset has utility values associated with each item in the database. For other datasets, the utility values are generated between 1 to 5 using log-normal distribution. The quantity values are generated randomly between 1 to 5. We fix the maximum time duration(MTD) parameter to 2 for our experiments.

The results on sparse datasets are shown in Figure 2. The graphs show that pre-insertion strategy beats the other strategies in terms of total execution time and number of candidate episodes generated. The *EWU* strategy performs the worst as sparse datasets usually generate few short high-utility episodes.

The results on dense datasets are shown in Figure 3. The result shows that the performance of TUP-EWU and TUP-Combined strategy outperforms the performance of other strategies. Processing those episodes first which have higher *EWU* is effective in dense datasets as the high-utility episodes usually have high support value and are of longer length compared to sparse datasets.

We further analyzed the memory consumption of the algorithms. The results are shown in Figure 4. The results show that TUP−EWU and TUP−Combined consume less memory on dense datasets as they generate less number of candidate episodes.

6. CONCLUSION

High-utility Episode mining in a complex event sequence is an emerging topic in data mining and has many applications in real world. In this paper, we propose an efficient algorithm for mining top-k high utility episodes in a complex event sequence. We further proposed effective strategies to prune the search space effectively by raising the minimum utility threshold. We conducted extensive experiments on various datasets and the results demonstrate the effectiveness of our proposed strategies.

7. REFERENCES

[1] A. Achar, I. A, and P. S. Sastry. Editorial: Pattern-growth based frequent serial episode discovery. *Data Knowl. Eng.*, 87:91–108, Sept. 2013.

[2] A. Achar, S. Laxman, and P. S. Sastry. A unified view of the apriori-based algorithms for frequent episode discovery. *Knowl. Inf. Syst.*, 31(2):223–250, May 2012.

[3] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and H.-J. Choi. Interactive mining of high utility patterns over data streams. *Expert Systems with Applications*, 39(15):11979 – 11991, 2012.

[4] M. Atallah, W. Szpankowski, and R. Gwadera. Detection of significant sets of episodes in event sequences. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 3–10, Nov 2004.

[5] R. Bansal, S. Dawar, and V. Goyal. An efficient algorithm for mining high-utility itemsets with discount notion. In N. Kumar and V. Bhatnagar, editors, *Big Data Analytics*, volume 9498 of *Lecture Notes in Computer Science*, pages 84–98. Springer International Publishing, 2015.

[6] G. Casas-Garriga. Discovering unbounded episodes in sequential data. In N. Lavra, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Knowledge Discovery in Databases: PKDD 2003*, volume 2838 of *Lecture Notes in Computer Science*, pages 83–94. 2003.

[7] S. Dawar and V. Goyal. Up-hist tree: An efficient data structure for mining high utility patterns from transaction databases. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, IDEAS '15, pages 56–61, 2014.

[8] A. Erwin, R. Gopalan, and N. Achuthan. Efficient mining of high utility itemsets from large datasets. In T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5012 of *Lecture Notes in Computer Science*, pages 554–561. 2008.

[9] B. Goethals and M. Zaki. the fimi repository, 2012.

[10] V. Goyal, S. Dawar, and A. Sureka. High utility rare itemset mining over transaction databases. In W. Chu, S. Kikuchi, and S. Bhalla, editors, *Databases in Networked Information Systems*, volume 8999 of *Lecture Notes in Computer Science*, pages 27–40. 2015.

[11] V. Goyal, A. Sureka, and D. Patel. Efficient skyline itemsets mining. In *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, C3S2E '15, pages 119–124, 2008.

[12] G. Guo, L. Zhang, Q. Liu, E. Chen, F. Zhu, and C. Guan. High utility episode mining made practical and fast. In X. Luo, J. Yu, and Z. Li, editors, *Advanced Data Mining and Applications*, volume 8933 of *Lecture Notes in Computer Science*, pages 71–84. 2014.

[13] T. Guo, S. Lin, Y. Wang, and J. Qiao. A new framework for detecting high-utility episodes in event sequence. In *IEEE International Conference on Oxide Materials for Electronic Engineering (OMEE)*, pages 370–373, 2012.

[14] R. Gwadera, M. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 67–74, Nov 2003.

[15] R. Gwadera, M. J. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. *Knowl. Inf. Syst.*, 7(4):415–437, May 2005.

[16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[17] K.-Y. Huang and C.-H. Chang. Efficient mining of frequent episodes from complex sequences. *Information Systems*, 33(1):96 – 114, 2008.

[18] H.-F. Li, H.-Y. Huang, and S.-Y. Lee. Fast and memory efficient mining of high-utility itemsets from data streams: with and without negative item profits.

Knowledge and Information Systems, 28(3):495–522, 2011.

[19] H.-F. Li and S.-Y. Lee. Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert Systems with Applications*, 36(2, Part 1):1466 – 1477, 2009.

[20] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. *Data and Knowledge Engineering*, 64(1):198 – 217, 2008.

[21] Y.-F. Lin, C.-F. Huang, and V. S. Tseng. A novel episode mining methodology for stock investment. *Journal of Information Science and Engineering*, 30(3):571–585, 2014.

[22] Y.-F. Lin, C.-W. Wu, C.-F. Huang, and V. S. Tseng. Discovering utility-based episode rules in complex event sequences. *Expert Systems with Applications*, 42(12):5303 – 5314, 2015.

[23] B. L. W. H. Y. Ma. Integrating classification and association rule mining. In *international Conference on Knowledge Discovery and Data Mining*, 1998.

[24] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences (extended abstract). In *In 1st Conference on Knowledge Discovery and Data Mining*, pages 210–215, 1995.

[25] F. Medici, M. I. Hawa, A. Giorgini, A. Panelo, C. M. Solfelix, R. Leslie, and P. Pozzilli. Antibodies to gad65 and a tyrosine phosphatase-like molecule ia-2ic in filipino type 1 diabetic patients. *Diabetes Care*, 22(9):1458–1461, 1999.

[26] A. Ng and A.-c. Fu. Mining frequent episodes for relating financial events and stock trends. In K.-Y. Whang, J. Jeon, K. Shim, and J. Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, volume 2637 of *Lecture Notes in Computer Science*, pages 27–39. 2003.

[27] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, pages 215–224, 2001.

[28] J. Pisharath, Y. Liu, W.-k. Liao, A. Choudhary, G. Memik, and J. Parhi. Nu-minebench 2.0. *Department of Electrical and Computer Engineering, Northwestern University, Tech. Rep*, 2005.

[29] H. Ryang and U. Yun. Top-k high utility pattern mining with effective threshold raising strategies. *Knowledge-Based Systems*, 76:109 – 126, 2015.

[30] W. Shi, F. K. Ngok, and D. R. Zusman. Cell density regulates cellular reversal frequency in myxococcus xanthus. *Proceedings of the National Academy of Sciences*, 93(9):4142–4146, 1996.

[31] B.-E. Shie, H.-F. Hsiao, V. Tseng, and P. Yu. Mining high utility mobile sequential patterns in mobile commerce environments. In J. Yu, M. Kim, and R. Unland, editors, *Database Systems for Advanced Applications*, volume 6587 of *Lecture Notes in Computer Science*, pages 224–238. 2011.

[32] C.-W. Wu, Y.-F. Lin, P. S. Yu, and V. S. Tseng. Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 536–544, 2013.

[33] C. W. Wu, B.-E. Shie, V. S. Tseng, and P. S. Yu. Mining top-k high utility itemsets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 78–86, New York, NY, USA, 2012.

[34] J. Yin, Z. Zheng, and L. Cao. Uspan: An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 660–668, 2012.

[35] J. Yin, Z. Zheng, L. Cao, Y. Song, and W. Wei. Efficiently mining top-k high utility sequential patterns. In *IEEE 13th International Conference on Data Mining*, pages 1259–1264, Dec 2013.

Soft Monotonic Constraint Support Vector Regression

Sapan Shah, Avadhut Sardeshmukh,
Shuaib Ahmed, Sreedhar Reddy

Tata Research Development and Design Center,
Tata Consultancy Services Limited,
Pune 411013
India
{sapan.hs, avadhut.sardeshmukh, s.ahmed2, sreedhar.reddy}@tcs.com

Abstract

This paper proposes a model for learning soft-monotonic regression functions in the presence of imperfect domain knowledge. It proposes an extension to support vector regression (SVR) wherein a new hardness parameter is introduced to configure the degree of monotonicity. The model supports multiple monotonicity constraints over multiple input dimensions simultaneously. The proposed model has been validated on synthetic datasets as well as on benchmark datasets obtained from real world problems. The results show that our model has better extrapolation capabilities than SVR. The results also demonstrate the ability of the model to generalize over multiple input dimensions.

1. Introduction

In machine learning, prior domain knowledge improves the quality of the model learnt, especially when data is scant or noisy. A common class of such prior knowledge is monotonicity constraints. For example, in credit rating, the chances of getting a loan increase with income, provided other variables are same [1]. The price of houses increases with better characteristics of houses such as the number of rooms [2]. It is reasonable to expect a machine learning algorithm to capture such knowledge in the model learnt, but that depends on the quantity and quality of the data available. Data inadequacy is a common problem in many domains. For instance, in materials engineering, process-structure-property relations play a key role in the design process. However, this data is hard to come by. In such cases, incorporating prior domain knowledge into a machine learning algorithm can improve the quality of the model learnt. For instance, in a carburization process, an increase in carbon potential increases the hardness of the material. This information can be leveraged by a learning algorithm to improve the model for a heat-treatment process.

Monotonic regression on a data set of n 1-dimensional pairs $\{(x_i, y_i)\}_{i=1}^{n}$ can be defined as follows:

$$\hat{f}(x) = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(x_i))^2$$
$$subject\ to \quad x_i \succcurlyeq x_j \Rightarrow f(x_i) \geq f(x_j); \quad \forall i, j; 1 \leq i, j \leq n \qquad (1)$$

where $f(x_i)$ is the approximation of y_i; and $\succcurlyeq$ is the preorder relation that is reflexive and transitive on input space X. Monotonically decreasing approximation can be defined by replacing the constraint as $x_i \succcurlyeq x_j \Rightarrow f(x_i) \leq f(x_j)$ in (1).

Several algorithms exist for solving the problem defined in (1). One of the earliest algorithms is the pool adjacent violator algorithm (PAVA), based on which a lot of research has been reported [3]. These are well summarized in [4,5]. However, these methods are not applicable when the input space is d dimensional i.e. $X \subseteq \mathbb{R}^d$, because they assume that the input is a sequence i.e. $x_i < x_{i+1}$, which cannot be generalized to d dimensions.

Also, domain knowledge is not always perfect. In the credit rating example, the chances of getting a loan may decrease with past record of defaulting despite higher income. In house pricing example, a flat with one room in a posh area may be costlier than a house with three rooms in a rural area. To address this, we use the notion of soft monotonic regression. Soft monotonicity balances between the best fit $(y = f(x))$ and the monotonic fit $(y = \hat{f}(x))$. This can be formally represented as,

International Conference on Management of Data
COMAD 2016, Pune, India, March 11-13, 2016

$$\hat{f}(x)_\lambda = \underset{f}{\operatorname{argmin}} \left(\begin{array}{l} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda * \\ \sum_{i=1}^{n} \sum_{j=i}^{n} \mathcal{K}\left(x_i, x_j, f(x_i), f(x_j)\right) \end{array} \right) \quad (2)$$

$$where\ \mathcal{K}\left(x_i, x_j, f(x_i), f(x_j)\right) = \begin{cases} 0 & \begin{array}{l} if\ monotonically\ increasing\ and \\ \begin{cases} x_i \geq x_j\ and\ f(x_i) \geq f(x_j)\ Or \\ x_j \geq x_i\ and\ f(x_j) \geq f(x_i) \end{cases} \end{array} \\ 0 & \begin{array}{l} if\ montonically\ decreasing\ and \\ \begin{cases} x_i \geq x_j\ and\ f(x_i) \leq f(x_j)\ Or \\ x_j \geq x_i\ and\ f(x_j) \leq f(x_i) \end{cases} \end{array} \\ 1 & otherwise \end{cases}$$

Here, λ is the penalty parameter which penalizes the pair that violates the monotonicity constraint. Setting λ to 0 yields the best fit solution. As $\lambda \to \infty$, the solution approaches strictly monotonic for the problem defined in (1). This kind of problem has recently been addressed in [6] using a modified version of PAVA. However, in line with the original version, it too focuses on one dimensional input data represented as a sequence.

In this paper, we propose a solution for soft monotonic regression for multidimensional input i.e. $\mathrm{X} \subseteq \mathbb{R}^d$. More precisely, we propose an extension to SVR called monotonic constraint support vector regression (MC-SVR) and present its formulation. The proposed method has been validated with numerical experiments. The capability of MC-SVR to impose different monotonic constraints (increasing, decreasing, or none) at different input variables is studied. The effect of softness of the constraint is discussed. Results pertaining to the extrapolation capability of MC-SVR, and generalization capability over multiple input dimensions are also presented.

2. Monotonic Constraint Support Vector Regression

Support vector regression (SVR) basically takes the form [7,8]

$$\begin{array}{l} f(\mathrm{x}) = \langle \mathrm{w}, \mathrm{x} \rangle + b \\ with \quad \mathrm{w}, \mathrm{x} \in \mathbb{R}^d\ and\ b \in \mathbb{R} \end{array} \quad (3)$$

where $\langle .,. \rangle$ denotes the dot product. w is determined by solving the following convex optimization problem,

$$\min_{w,b,\xi_i,\widehat{\xi_\iota}} \left(\frac{1}{2}\|\mathrm{w}\|^2 + C \sum_{i=1}^{n} \left(\xi_i + \widehat{\xi_\iota}\right) \right)$$
$$sub.to \begin{cases} (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) - y_i & \leq & \epsilon + \xi_i;\ \forall i \\ y_i - (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) & \leq & \epsilon + \widehat{\xi_\iota};\ \forall i \\ \xi_i, \widehat{\xi_\iota} & \geq & 0;\ \forall i \end{cases} \quad (4)$$

where, $\xi_i, \widehat{\xi_\iota}$ are slack variables for *soft margin* to accommodate infeasible optimization that may arise due to noisy input variables.

2.1 Monotonicity Constraints

To incorporate monotonicity in SVR, additional constraints have to be formulated. The monotonicity constraint proposed in this paper is based on the following assumption,

$$\begin{array}{lll} f(\mathrm{x} + \delta\mathrm{x}) + \zeta & \geq f(\mathrm{x}); & \text{which simplifies to,} \\ \langle \mathrm{w}, \mathrm{s} \rangle - \langle \mathrm{w}, \mathrm{x} \rangle & \geq -\zeta & where\ \ \mathrm{s} = \mathrm{x} + \delta\mathrm{x} \end{array} \quad (5)$$

The above equation can be interpreted as: a fraction of addition to the input should lead to a solution that is either increasing or stays the same. It should be noted that δ is a vector $\{\delta\}_{i=1}^{d}$ where each component can be set differently for different input dimensions. If $\delta^{(i)}$ is positive, the function is monotonically increasing in i^{th} dimension; if $\delta^{(i)}$ is negative, it is monotonically decreasing; and if $\delta^{(i)} = 0$, no monotonic constraint is imposed. Since the prior domain knowledge is imperfect, some violation in the constraint is allowed. Variable ζ introduced in (5) accounts for this violation. When ζ is zero, the function is strictly monotonic. The new variable $\mathrm{s} = \mathrm{x} + \delta\mathrm{x}$ ensures that the function f is monotonic around x. s is computed once for entire input data x. It should be noted that we only need to compute the value of s and not $f(s)$.

2.2 Incorporating monotonicity constraints in SVR

The variable ζ in (5) can be incorporated in MC-SVR similar to the slack variables ξ_i in standard SVR optimization as given in (4). Considering the above formulation of constraint to implement the MC-SVR, the optimization problem of SVR given in (4) becomes,

$$\min_{w,b,\xi_i,\widehat{\xi_\iota},\zeta_i} \frac{1}{2}\|\mathrm{w}\|^2 + C \sum_{i=1}^{n} \left(\xi_i + \widehat{\xi_\iota}\right) + D \sum_{i=1}^{n} \zeta_i$$
$$\begin{array}{lll} sub.\ to & & \\ (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) - y_i & \leq & \epsilon + \xi_i, \forall i; \\ y_i - (\langle \mathrm{w}, \mathrm{x}_i \rangle + b) & \leq & \epsilon + \widehat{\xi_\iota}, \forall i; \\ \langle \mathrm{w}, \mathrm{x}_i \rangle - \langle \mathrm{w}, \mathrm{s}_i \rangle & \leq & \zeta_i, \forall i: \mathrm{s}_i = \mathrm{x}_i + \delta\mathrm{x}_i \\ \xi_i, \widehat{\xi_\iota}, \zeta_i & \geq & 0 \end{array} \quad (6)$$

where D is the hardness parameter. It represents the amount of margin allowed to violate the monotonicity constraint. When D is large, the penalty for violating the constraint is high. This results in a strict monotonic solution. Conversely, when $D = 0$, monotonicity is not guaranteed.

The quadratic optimization problem in (6) can be solved by the method of Lagrangian multipliers

similar to the standard SVR [7,8]. The primal Lagrangian for (6) is as given in (7) below:

$$\begin{aligned}\mathcal{L}_p = {} & \frac{1}{2}\|\mathrm{w}\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \widehat{\xi}_\iota\right) + D\sum_{i=1}^{n}\zeta_i \\ & - \sum_{i=1}^{n}\alpha_i(-(\langle \mathrm{w}, \mathrm{x}_i\rangle + b) + y_i + \epsilon + \xi_i) \\ & - \sum_{i=1}^{n}\widehat{\alpha_\iota}\left((\langle \mathrm{w}, \mathrm{x}_i\rangle + b) - y_i + \epsilon + \widehat{\xi}_\iota\right) \\ & - \sum_{i=1}^{n}\beta_i\left(\zeta_i - (\langle \mathrm{w}, \mathrm{x}_i\rangle - \langle \mathrm{w}, \mathrm{s}_i\rangle)\right) \\ & - \sum_{i=1}^{n}\mu_i\xi_i - \sum_{i=1}^{n}\widehat{\mu_\iota}\widehat{\xi}_\iota - \sum_{i=1}^{n}\lambda_i\zeta_i \\ sub.to:\ & \alpha_i, \widehat{\alpha_\iota}, \beta_i, \mu_i, \widehat{\mu_\iota}, \lambda_i \geq 0\end{aligned} \quad (7)$$

Following the saddle point condition, the partial derivatives of $\mathcal{L}_p$ with respect to primal variables $(\mathrm{w}, \mathrm{b}, \xi_i, \widehat{\xi}_\iota, \zeta_i)$ should vanish for optimality. The partial derivatives are,

$$\begin{aligned}\frac{\partial \mathcal{L}_p}{\partial w} &= w - \sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)x_i + \sum_{i=1}^{n}\beta_i(x_i - s_i) = 0 \\ \frac{\partial \mathcal{L}_p}{\partial b} &= \sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i) = 0 \\ \frac{\partial \mathcal{L}_p}{\partial \xi_i} &= C - \alpha_i - \mu_i = 0 \\ \frac{\partial \mathcal{L}_p}{\partial \widehat{\xi}_\iota} &= C - \widehat{\alpha_\iota} - \widehat{\mu_\iota} = 0 \\ \frac{\partial \mathcal{L}_p}{\partial \zeta_i} &= D - \beta_i - \lambda_i = 0\end{aligned} \quad (8)$$

Substituting (8) in (7) yields the dual optimization problem as follows.

$$\begin{aligned}\mathcal{L}_D = \max_{\alpha,\widehat{\alpha},\beta}\Bigg\{ & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i, \mathrm{x}_j\rangle \\ & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ & +\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\beta_j\langle \mathrm{x}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ & +\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j\rangle \\ & +\sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)y_i - \epsilon\sum_{i=1}^{n}(\widehat{\alpha_\iota} + \alpha_i)\Bigg\} \\ subject\ to: & \sum_{i=1}^{n}(\widehat{\alpha_\iota} - \alpha_i) = 0; \\ & 0 \leq \widehat{\alpha_\iota}, \alpha_i \leq C\ \ \forall i; \\ & 0 \leq \beta_i \leq D\ \ \forall i\end{aligned} \quad (9)$$

The problem in (9) is convex and hence, it forms a standard quadratic optimization problem. This fact is proved in theorem 1.

In order to represent the dual in standard quadratic form, the dot products in (9) can be rewritten as,

$$\left.\begin{aligned} X_iX_j = \langle x_i, x_j\rangle;\quad S_iS_j = \langle s_i, s_j\rangle \\ X_iS_j = \langle x_i, s_j\rangle;\quad S_iX_j = \langle s_i, x_j\rangle = X_jS_i^T\end{aligned}\right\}\forall i, j \quad (10)$$

The quadratic term is no longer a $2n \times 2n$ matrix as in conventional linear ε-insensitive loss SVR. Instead, it is a $3n \times 3n$ matrix composed of variables $\hat{\alpha}, \alpha\ and\ \beta$ as shown below,

$$\begin{aligned}&\min_{\alpha,\widehat{\alpha},\beta}\ \frac{1}{2}[\hat{\alpha}\ \ \alpha\ \ \beta]\ H\ \begin{bmatrix}\hat{\alpha}\\ \alpha\\ \beta\end{bmatrix} + C^T\begin{bmatrix}\hat{\alpha}\\ \alpha\\ \beta\end{bmatrix} \\ &where, \\ &H = \begin{bmatrix} XX & -XX & -(XX - XS) \\ -XX & XX & (XX - XS) \\ -(XX - XS)^T & (XX - XS)^T & (XX - SS - (XS + XS^T)) \end{bmatrix} \\ &C = \left(\epsilon\begin{bmatrix}1\\1\\0\end{bmatrix} - \begin{bmatrix}y\\-y\\0\end{bmatrix}\right) \\ &subject\ to, \begin{bmatrix}-1\\1\\0\end{bmatrix}^T\begin{bmatrix}\hat{\alpha}\\ \alpha\\ \beta\end{bmatrix} \leq 0; \\ &\qquad 0 \leq \hat{\alpha}, \alpha \leq C; 0 \leq \beta \leq D;\end{aligned} \quad (11)$$

Theorem 1: *The problem of MC-SVR defined in (9) is convex.*

Proof: For the problem in (9) to be convex, the quadratic term should be necessarily positive semi-definite. The quadratic term in $\mathcal{L}_D$ while minimizing is,

$$\begin{aligned}&\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i, \mathrm{x}_j\rangle \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ &\quad - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\alpha_\iota} - \alpha_i)\beta_j\langle \mathrm{x}_i, \mathrm{x}_j - \mathrm{s}_j\rangle \\ &\quad - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\left(\widehat{\alpha_\jmath} - \alpha_j\right)\langle \mathrm{x}_i - \mathrm{s}_i, \mathrm{x}_j\rangle \\ &= \frac{1}{2}\Big\langle\sum_i(\widehat{\alpha_\iota} - \alpha_i)\mathrm{x}_i - \sum_i\beta_i(\mathrm{x}_i - \mathrm{s}_i)\ , \\ &\qquad \sum_j\left(\widehat{\alpha_\jmath} - \alpha_j\right)\mathrm{x}_j - \sum_j\beta_j\left(\mathrm{x}_j - \mathrm{s}_j\right)\Big\rangle \\ &= \frac{1}{2}\left\|\sum_i(\widehat{\alpha_\iota} - \alpha_i)\mathrm{x}_i - \sum_j\beta_j\left(\mathrm{x}_j - \mathrm{s}_j\right)\right\|^2 \\ &\geq 0\end{aligned}$$

Thus, the quadratic term is positive semi-definite. Hence, (9) is convex.

In order to solve the MC-SVR problem, it is important to find the parameters of interest i.e. w and b. Using the partial derivatives of the primal Lagrangian, variable w can be written as,

$$\mathrm{w} = \sum_{i=1}^{n} (\hat{\alpha}_i - \alpha_i)\mathrm{x}_i - \sum_{i=1}^{n} \beta_i(\mathrm{x}_i - \mathrm{s}_i) \quad (12)$$

It is possible to describe w completely in terms of training data x_i (since, s_i is derived from x_i) even for MC-SVR. Substituting (12) in (3) yields,

$$f(x) = \sum_{i=1}^{n} (\hat{\alpha}_i - \alpha_i)\langle x_i, x\rangle - \sum_{i=1}^{n} \beta_i(\langle x_i, x\rangle - \langle s_i, x\rangle) - b \quad (13)$$

The variable b can be found following the Karush-Kuhn-Tucker (KKT) conditions [8] which state that at optimality, the product of dual variable and the constraint should vanish. Thus, b can be identified as,

$$b = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left(\sum_{j=1}^{n}(\hat{\alpha}_j - \alpha_j)\langle \mathrm{x}_j, \mathrm{x}_i\rangle - \sum_{j=1}^{n}\beta_j(\langle \mathrm{x}_j, \mathrm{x}_i\rangle - \langle \mathrm{s}_j, \mathrm{x}_i\rangle)\right) - \epsilon\right) \quad (14)$$

It is important to note that in (9) to (14), instead of dot product, a kernel expansion can also be applied, which will lead to non-linear MC-SVR. Standard kernels for SVR (such as, Linear, Polynomial, Gaussian, and so on) can be effectively applied.

The hardness parameter, D, bounds the extra variable β in (9). Hence, when D is sufficiently small, then $\beta \ll \alpha$, and there will not be any significant effect of monotonicity constraint. Conversely when D is sufficiently large, then $\beta \gg \alpha$, and the effect of monotonicity will be dominant. For the sake of brevity, the MC-SVR formulation in (6) uses the same parameter D for all monotonicity constraints. Similarly, it uses the same slack variable ζ_i for all monotonicity constraints. The formulation in (6) can be easily extended to support separate D and ζ_i for each monotonicity constraint. In this case, the parameter D will be a vector. The components of D will specify hardness for each constraint. Similarly, there will be a separate set of slack variables $\zeta_i{}^C$ for each constraint.

2.3 Degree of Monotonicity

To assess the degree of monotonicity, we use Kendall correlation metric [9]. Let $(x_1, y_1),\ (x_2, y_2), \ldots, (x_n, y_n)$ be a set of observations of variables X and Y respectively. Any pair of observations (x_i, y_i) and (x_j, y_j) are said to be *concordant* if the ranks for both elements agree (i.e. $x_i > x_j\ and\ y_i > y_j\ OR\ x_i < x_j\ and\ y_i < y_j$). They are *discordant* if $x_i > x_j\ and\ y_i < y_j\ OR\ x_i < x_j\ and\ y_i > y_j$. The Kendall correlation metric is then defined as,

$$Kendall(X,Y) = \frac{\#concordant\ pairs - \#discordant\ pairs}{\frac{1}{2}n(n-1)} \quad (15)$$

If the target variable is a monotonically increasing function of input then Kendall correlation is 1; while for a monotonically decreasing function, it is -1.

We define *degree of monotonicity* as follows,

$$degree\ of\ monotonicity(X,Y) = \frac{(Kendall(X,Y) * sign + 1)}{2}$$
$$where,\quad sign = \begin{cases} 1 & ;\ Y\ is\ a\ monotonically\ increasing\ \ function\ of\ X \\ -1 & ;\ Y\ is\ a\ monotonically\ decreasing\ \ function\ of\ X \end{cases} \quad (16)$$

This measure is in the range of 0 to 1 and it normalizes Kendall correlation such that for a perfect monotonic function (increasing or decreasing), it reaches the value of 1. We define the *degree of monotonicity of a model* with respect to a pair of variables as the degree of monotonicity that would be expected to be observed between the variables when data is generated randomly from the model. It is related to the hardness parameter D in the following sense:

$D_1 > D_2$ => degree of monotonicity of a model learnt with D_1 is greater than the degree of monotonicity of a model learnt with D_2.

Thus we can increase or decrease the degree of monotonicity of a model by increasing or decreasing D.

Degree of monotonicity is a more intuitive parameter for a domain expert to specify. A value of 1 specifies that the model is completely monotonic; a value of 0.8 specifies that 80% of the data is expected to be monotonic, while 20% may be contrarian, and so on. It is possible to convert degree of monotonicity into an appropriate value of D by writing a wrapper function that internally searches for a value of D that gives the desired degree of monotonicity.

3. Experimental Results

We have carried out two sets of experiments to validate MC-SVR: one set with synthetic models and the other with real-life datasets.

3.1 Experiments with synthetic models

Our objective in using synthetic models is to study how the MC-SVR behaves under various controlled experimental conditions, such as dimensionality of input data, degree of the true function, effect of noise in data, etc.

These experiments use the following scheme: A true model of a chosen degree and dimensionality is constructed. Data is generated from this model. To simulate the effect of noise in observed data, we add random sinusoidal noise (with amplitude drawn from Gaussian distribution) to the generated data. To study the effect of degree of monotonicity, we add an additional term to the true model to introduce regions where monotonicity is violated. We then compare the learnt models with the true model. The output of MC-SVR for constraint violation region is analysed. Softness in MC-SVR is controlled by the *degree* parameter. The effect of softness on the learnt model is also analysed.

In all experiments we first learn the SVR model using a Gaussian kernel and optimize the hyper parameters γ and C using 10-fold cross validation. We then use the same hyper parameters for MC-SVR and set the monotonicity parameters (i.e. monotonicity constraint and degree of monotonicity) as required. Both SVR and MC-SVR are implemented in octave 3.8 using PR_LOQO [10] as the QP solver.

3.1.1 Comparison between SVR and hard MC-SVR (degree of monotonicity =1)

In this experiment we choose a simple linear function with one input. The true model and the data generation model with noise term are as given in (17) below,

$$\begin{aligned} &\text{True model: } y = \mathrm{x} \\ &\text{Data model: } y = x + \mathcal{N}(0.08, 0.04) * \sin(25 * x) \end{aligned} \quad (17)$$

The amplitude of the noise is drawn from a Gaussian distribution with an arbitrary mean 0.08 and standard deviation 0.04. We generated 200 data points in the range 0 to 1. Figure 1 shows the true model as well as the observed data points. As mentioned in section 2.1, the sign $(\delta^{(i)} > 0 \; or \; \delta^{(i)} < 0)$ determines whether the constraint is monotonically increasing or decreasing. As the true function is monotonically increasing in x, we set the monotonicity constraint as $\delta = \{0.1\}$. This corresponds to 10% of the normalized data range, specifying that the monotonicity constraint should hold with data points that are within this range of an input data point.

An SVR with Gaussian kernel is optimized for minimizing the root mean squared error (RMSE). The best parameters found are $\gamma = 2^{-4} \; and \; C = 10$. MC-SVR is learnt using the same hyper-parameters. As the true model is monotonic in the full input range, we set the degree of monotonicity to 1 (hard MC-SVR). The results are shown in Figure 1. It can be observed from the figure that MC-SVR learns a better fit to the true model than SVR. The RMSE of MC-SVR and SVR are 0.0430 and 0.0545 respectively, showing that MC-SVR performs significantly better than SVR (approximately 20% improvement).

Next we study the behaviour of hard MC-SVR for a non-linear function. The true model in this case is a simple quadratic function *viz.* $y = x^2$. Similar to the linear case, the data generation model adds noise terms with amplitude drawn from a Gaussian distribution (with mean = 0.08 and standard deviation = 0.04). 200 data points are generated in the range 0 to 1. Figure 2 shows the true model and the observed data points. As the function is monotonically increasing in x, the monotonicity constraint is set as $\delta = \{0.1\}$. An SVR with Gaussian kernel is optimized to find the best parameter setting ($\gamma = 2^{-3} \; and \; C = 10$). MC-SVR is learnt with the same hyper-parameters. The degree of monotonicity is set to 1 as the true model is monotonic in the full input range. Again, as can be observed from Figure 2, MC-SVR learns a better model compared to SVR. The RMSE of MC-SVR and SVR are 0.0387 and 0.0523 (approximately 26% improvement over SVR).

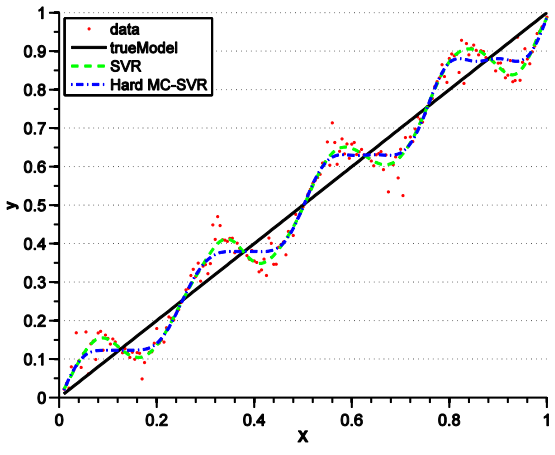


Figure 1: Comparison of SVR and Hard MC-SVR (degree = 1) for 1 dimensional linear function

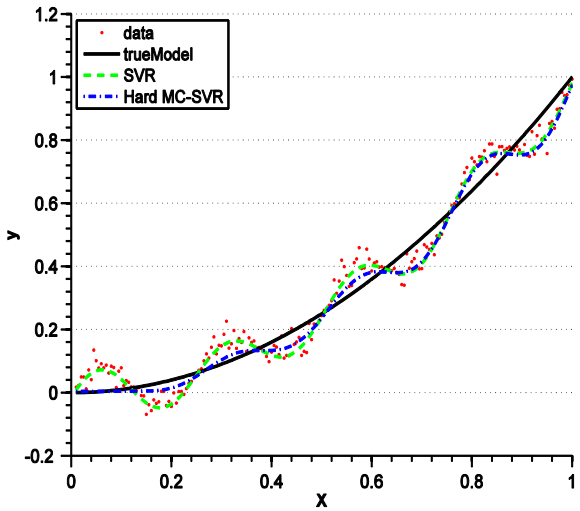


Figure 2: Comparison of SVR and Hard MC-SVR (degree = 1) for 1 dimensional quadratic function

We have also studied the behaviour of hard MC-SVR for multidimensional input. Similar to the one dimensional case, we set up a two dimensional experiment using an arbitrary linear function with two inputs. The true model and the data generation model with noise are as given in (18) below,

$$\begin{aligned}\text{True model: } & z = 0.7 * x - 0.5 * y \\ \text{Data model: } & z = 0.7 * x - 0.5 * y \\ & + \mathcal{N}(0.05,0.025) * \sin(25 * x) \\ & + \mathcal{N}(0.05,0.025) * \sin(25 * y)\end{aligned} \quad (18)$$

The data model adds noise to both the input dimensions. We choose 300 data points randomly from a set of 1156 generated data points (a two dimensional grid with spacing 0.03) in the range 0 to 1. The true model in (18) is monotonically increasing in x and decreasing in y. Hence we set the constraint as $\delta = \{0.1, -0.1\}$.

Similar to the one-dimensional case, an SVR with Gaussian kernel is used to minimize the root mean squared error. The best parameters found are $\gamma = 2^{-2}$ and $C = 32768$. MC-SVR is learnt using the same hyper-parameters. The degree of monotonicity is set to 1 as the true model has perfect monotonic behaviour in both dimensions (increasing in the first and decreasing in the second). Once again, it has been observed that MC-SVR produces a better fit to the true model than SVR. MC-SVR achieves a significant improvement in RMSE over SVR – from 0.0607 for SVR to 0.0428 for MC-SVR, an improvement of around 30%. This experiment shows that MC-SVR can learn different monotonicity relations in different dimensions, and the model incorporating these relations performs better than SVR.

The experiments in this section suggests that the difference between MC-SVR and SVR becomes more significant as the complexity (degree or dimensionality) of the problem increases.

3.1.2 Softness capability of MC-SVR

In order to evaluate the softness capability of MC-SVR, we modify the true model of (17) by imposing monotonically decreasing behaviour in input x for approximately 10% of the data. If we think of this data as coming from a real-life process, the monotonically decreasing part represents exceptions to the generally expected behaviour. We show that MC-SVR with an appropriately defined degree of monotonicity (which can be set by a domain expert) has the capacity to model this behaviour and is capable of achieving better accuracy than both SVR and hard MC-SVR.

The true model and the data generation model for this experiment are as follows,

$$\begin{aligned}\text{True model: } & y = \mathrm{x} - 0.4 * e^{-\left(\frac{x-0.5}{0.07}\right)^2} \\ \text{Data model: } & y = \mathrm{x} - 0.4 * e^{-\left(\frac{x-0.5}{0.07}\right)^2} \\ & + \mathcal{N}(0.08,0.04) * \sin(25 * x)\end{aligned} \quad (19)$$

The exponential term adds monotonically decreasing behaviour around 0.5 to an otherwise monotonically increasing true model. As before, the data model adds noise to the data points before they are observed. We generated 200 data points in the range 0 to 1. Figure 3 shows the true model as well as the observed data points. This function is studied with monotonically increasing constraint i.e. $\delta = \{0.1\}$.

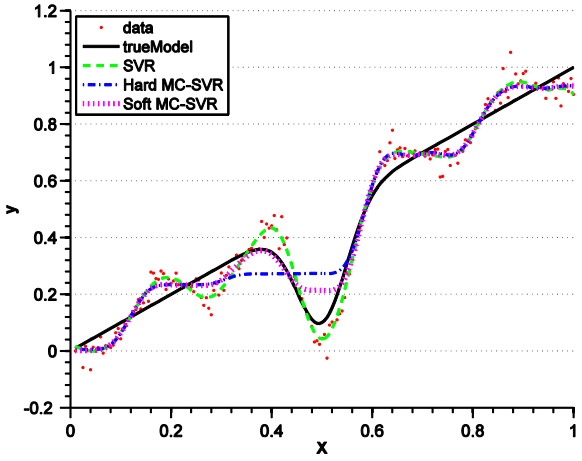


Figure 3: Comparison of Soft MC-SVR with SVR and Hard MC-SVR on linear soft monotonic function

Table 1: RMSE for synthetic 1-dimensional linear data – monotonically increasing function with opposite behavior in approximately 10% data

Function	RMSE
SVR	0.0522
$soft\ MC-SVR\ (degree = 0.9)$	0.0441
$hard\ MC-SVR\ (degree = 1)$	0.0595

Again, an SVR with Gaussian kernel is optimized ($\gamma = 2^{-4}$ and $C = 10$) for minimizing the root mean squared error. MC-SVR is learnt using the same hyper-parameters. The target variable in (19) is a monotonically increasing function of input. However, approximately 10% data in the true model has opposite behaviour. To account for this, we incorporate softness in MC-SVR by specifying the degree of monotonicity as 0.9. We also learn hard MC-SVR (i.e. degree of monotonicity = 1). As observed from Figure 3, soft MC-SVR produces a better fit compared to hard MC-SVR. This is due to the fact that hard MC-SVR does not use the information that 10% of data has opposite behaviour and builds a model that is monotonic in the full input range. The model leant using SVR captures this behaviour. However, it also learns the noise present in the observed data. The model learnt using soft MC-SVR provides a balance between SVR and hard MC-SVR. Table 1 shows the RMSE values obtained for these three models. Soft MC-SVR achieves significantly better RMSE compared to both SVR and hard MC-SVR.

Next we study the softness capability of MC-SVR on a non-linear function. The true model and the data generation model with noise are as given below,

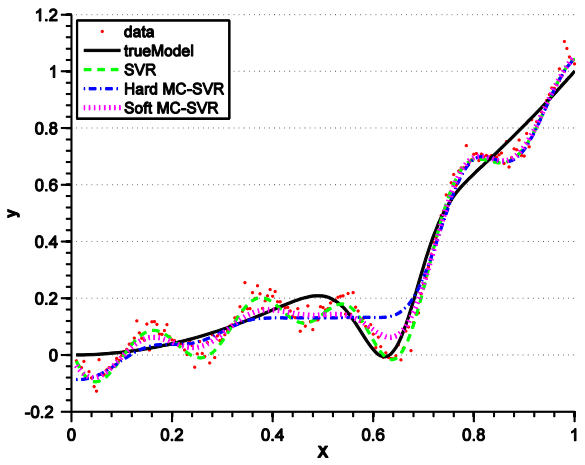


Figure 4: Comparison of Soft MC-SVR with SVR and Hard MC-SVR on quadratic soft monotonic function

Table 2: RMSE for synthetic 1-dimensional quadratic data – monotonically increasing function with opposite behavior in approximately 10% data

Function	RMSE
SVR	0.0535
$soft\ MC-SVR\ (degree = 0.9)$	0.0445
$hard\ MC-SVR\ (degree = 1)$	0.0543

Table 3: RMSE for synthetic 2-dimensional data – monotonically increasing function with opposite behavior in approximately 10% data

Function	RMSE
SVR	0.1746
$soft\ MC-SVR\ (degree = 0.9)$	0.0486
$hard\ MC-SVR\ (degree = 1)$	0.0810

$$\text{True model: } y = \mathrm{x}^2 - 0.4 * e^{-\left(\frac{x^2-0.4}{0.1}\right)^2}$$
$$\text{Data model: } y = \mathrm{x}^2 - 0.4 * e^{-\left(\frac{x^2-0.4}{0.1}\right)^2} + \mathcal{N}(0.08,0.04) * \sin(30 * x) \quad (20)$$

The model adds contrarian behaviour around 0.4 to an otherwise monotonically increasing function. 200 data points are generated in the range 0 to 1. Figure 4 shows the true model and the observed data points. As the function is monotonically increasing in x for the most part, the constraint is set as $\delta = \{0.1\}$.

As before, an SVR with Gaussian kernel is optimized to find the best parameter setting ($\gamma = 2^{-3}\ and\ C = 250$). MC-SVR is also learnt with the same hyper-parameters. The degree of monotonicity is set to 0.9 for soft MC-SVR as approximately 10% of data exhibits the opposite behaviour. Hard MC-SVR is also learnt to compare the effect of softness. Figure 4 shows the results. As can be seen, here also soft MC-SVR produces a better fit than SVR and hard MC-SVR. Table 2 reports the RMSE for the three models. As expected, soft MC-SVR performs significantly better than both SVR and hard MC-SVR.

We also studied the effect of softness on the learnt models for multidimensional input. The true model and the data generation model are as given below,

$$\text{True model: } \mathrm{z} = 0.7 * \mathrm{x} - 0.5 * \mathrm{y} - 0.5 * e^{-\left(\frac{x-0.5}{0.09}\right)^2}$$
$$\text{Data Model: } z = 0.7 * x - 0.5 * y - 0.5 * e^{-\left(\frac{x-0.5}{0.09}\right)^2} + \mathcal{N}(0.05,0.025) * \sin(25 * x) + \mathcal{N}(0.05,0.025) * \sin(25 * y) \quad (21)$$

The model is the same as that used in (18), except that, here, the true model has a monotonically decreasing behaviour in x for approximately 10% of the data.

Here again, an SVR with Gaussian kernel is optimized for minimizing root mean squared error ($\gamma = 2^{-3}\ and\ C = 10$). The same hyper-parameters are then used for MC-SVR. Since approximately 10% of the data exhibits opposite behaviour, we set the degree of monotonicity to 0.9. We also learn hard MC-SVR (degree of monotonicity = 1). Once again, as expected, soft MC-SVR produces a better fit to the true model than SVR and hard MC-SVR. A comparison of RMSE obtained with all these three approaches is shown in Table 3. It can clearly be observed that MC-SVR, with the additional knowledge about degree of monotonicity (as opposed to just the knowledge about monotonicity) outperforms both SVR and hard MC-SVR.

We have also studied the effect of softness on multidimensional non-linear functions using $x^2 - y^2$ as the true model with 10% contrarian behaviour. Once again soft MC-SVR performed significantly better than both SVR and hard MC-SVR.

3.2 Experiments on Real world Datasets

3.2.1 Extrapolation capability of MC-SVR

In this section, global warming dataset has been considered which is first studied in [11]. Recently, Tibshirani *et. al.* developed a nearly monotonic regression using modified PAVA for this dataset [6]. The dataset contains annual temperature anomalies from 1856 to 1999, relative to the 1961-1990 mean. It has 150 data points. It can be observed from Figure 5 that the actual data is monotonically increasing with respect to year, with possible decrease around 1900.

An SVR with Gaussian kernel is optimized ($\gamma = 2^{-2}\ and\ C = 250$) for minimizing the root mean squared error. Figure 5 shows the models produced by SVR, hard MC-SVR (degree of monotonicity = 1) and soft MC-SVR (degree of monotonicity = 0.83). It can be observed that SVR misses the monotonically increasing characteristic and learns a model that has decreasing behaviour at multiple places. On the other hand, hard MC-SVR learns a model that is increasing in the full input range. Soft MC-SVR produces a fit

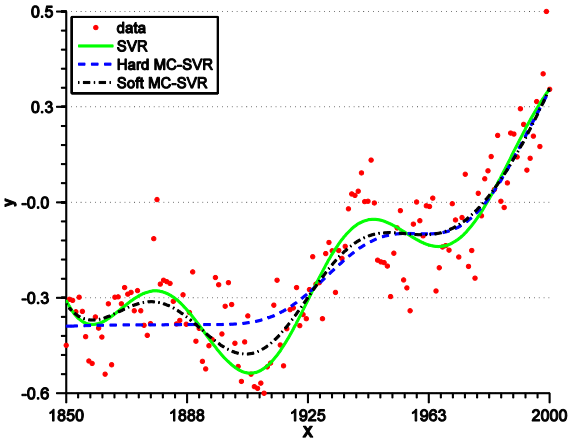


Figure 5: Results of SVR, Soft MC-SVR and Hard MC-SVR on Global Warming dataset

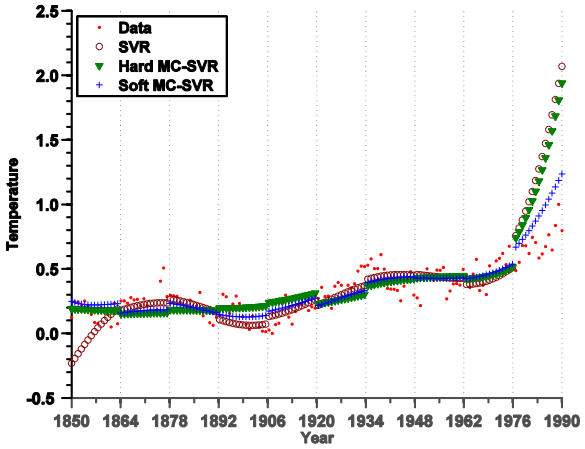


Figure 6: Results on extrapolation capability of SVR, Soft MC-SVR and Hard MC-SVR on Global Warming dataset

that captures the decreasing behaviour around year 1900 but is monotonically increasing otherwise.

In order to study the extrapolation capability of MC-SVR, the dataset is partitioned into 10 equally spaced bins. At a time, 9 bins have been used for training and the remaining for evaluation. This process has been continued for all the 10 bins. It has been observed that soft MC-SVR has produced a significantly better RMSE than standard SVR and full monotonic approximation as given in Table 4. Considering that the RMSE is reported on normalized data, soft MC-SVR improves SVR and hard MC-SVR by 17% and 10% respectively. The results are also portrayed in Figure 6. It can be observed that the extrapolation at the beginning as well as at the end is better for soft MC-SVR than for SVR and hard MC-SVR. Similar results have been observed by changing the number of bins to 5 and 20.

Table 4: RMSE for Global Warming dataset

Function	RMSE
SVR	0.1193
$Soft\ MC-SVR\ (degree = 0.83)$	0.0988
$Hard\ MC-SVR\ (degree = 1)$	0.1101

3.2.2 Generalization capability of MC-SVR over multiple input dimensions

In order to study monotonicity for multidimensional input, Cars dataset has been considered. This dataset has been studied in [12,13]. It contains 4 input attributes of cars viz. displacement, engine output in horsepower, weight and time to accelerate from 0 to 60 mph (acceleration time); the output is the prediction of fuel efficiency in miles per gallon. The first 3 input attributes have monotonically decreasing relation with the output attribute while the last attribute, acceleration time, has monotonically increasing relation. The dataset contains 392 instances.

The quantitative performance on RMSE has been assessed by performing 10-fold cross validation on normalized data. The constraint has been set as monotonically decreasing for *displacement*, *engine output* and *weight* while monotonically increasing for *acceleration time*. Grid search is performed to find the hyper- parameters (γ *and* C) that minimize root mean squared error for SVR. The same hyper-parameters are then used for MC-SVR. The degree of monotonicity is varied to incorporate softness in MC-SVR.

Table 5 reports RMSE for various experiments. The first experiment uses all four input attributes and accordingly the constraint is set as $\delta = \{-0.1, -0.1, -0.1, 0.1\}$. As can be noticed from Table 5, MC-SVR (soft as well as hard constraints) gives better results than SVR (approximately 10% improvement). The second experiment uses a 2-dimensional input – displacement and acceleration time. Once again,

Table 5: Cars dataset: RMSE for SVR and Soft MC-SVR and Hard MC-SVR for multiple input dimensions

Input Variables	SVR	Hard MC-SVR	Soft MC-SVR
Displacement, Engine Output, Weight, Acceleration Time	0.1037	0.0930	0.0929
Displacement, Acceleration time	0.1290	0.1229	0.1228

MC-SVR performs better than SVR (approximately 5% improvement).

The results show that MC-SVR leverages the monotonicity information present in multiple input dimensions. It can also be noticed that the difference between MC-SVR and SVR seems to become more significant as the dimensionality of the problem increases.

5. Related Work

Monotonic function learning has been extensively studied in literature for both classification as well as regression problems. Classification of ordered classes is often assumed to be monotonic with respect to input features. Hence, incorporating monotonicity for ordinal classification was investigated by many researchers. Neural networks is one of the well-studied algorithms in this direction. [14,15] approached monotonicity in neural networks by enforcing constraints on the weights and architecture of the network. An additional error term called monotonicity error, was implemented in neural network to ensure monotonicity in [16]. Makino *et. al.* [17] proposed classification trees with monotonicity constraint for binary classification. This was extended for multiclass decision trees in [2] as quasi-monotone decision trees. An instance based method for ordinal classification called ordinal stochastic dominance learner was proposed in [18]. Decision rules and ensemble of decision rules were also proposed for ordinal classification using monotonicity constraint [19]. Apart from modifying the classical learning algorithms, a method of altering the training data such that they are consistent with monotonicity definition was also proposed in [20]. Many real world applications were investigated using ordinal classification. These include liver disorder diagnosis [16], house pricing [2], internet content filtering [21] and breast cancer diagnosis [22]. The methods discussed above have been reported to perform well for ordinal classification. However, they cannot be directly extended to problems with continuous variables.

In order to implement monotonic regression, a smoothing method based on constraints was proposed in [12]. The capability of smoothing method was demonstrated on a few real world datasets such as cars and onion. In [23], monotonic regression was modelled using Gaussian process. Monotonicity was incorporated by virtual training examples that are generated from derivatives of actual data. This method was shown to improve performance on a synthetic dataset. Incorporating prior knowledge in the form of equality and inequality constraints was extensively discussed in [24,25], where linear programming formulation was used for support vector regression. Quadratic programming based learning with monotonicity of sequential data was studied in [26]. In this method, the input data are one dimensional and are assumed to be in sequence and cannot be used for multidimensional dataset. All the methods discussed above assume perfect prior domain knowledge about monotonicity. They also do not support monotonicity across multiple input dimensions. The solution proposed in this paper addresses these short comings. It provides a way of specifying partial a priori knowledge in the form of degree of monotonicity. The solution can be applied to multiple input dimensions where the monotonicity constraints can be increasing or decreasing in each individual dimension.

6. Conclusion

This paper addresses the problem of learning soft-monotonic regression functions in the presence of imperfect domain knowledge. A novel monotonicity constraint based support vector regression has been proposed. A new hardness parameter, D, is introduced in order to configure the degree of monotonicity required. The working of MC-SVR has been validated using datasets obtained from linear and non-linear synthetic functions. Our experiments on global warming dataset show that MC-SVR with soft constraint has better extrapolation capability than standard SVR. The experiment on Cars dataset shows the generalization capability of MC-SVR over multiple input dimensions. The formulation presented can also be extended to support soft convex constraints [6] and soft positive constraint SVR [13]. Though the results seem encouraging, more theoretical investigations are required to study its generalization properties.

References

1 Chen, Chih-Chuan and Li, Sheng-Tun. Credit Rating with a Monotonicity-Constrained Support Vector Machine Model. *Expert Syst. Appl.*, 41, 16 (2014), 7235-7247.

2 Potharst, Rob and Feelders, Adrianus Johannes. Classification Trees for Problems with Monotonicity Constraints. *ACM SIGKDD Explorations Newsletter*, 4, 1 (2002), 1-10.

3 Brunk, H. D. Maximum Likelihood Estimates of Monotone Parameters. *Ann. Math. Statist.*, 26, 4 (1955), 607-616.

4 Barlow, R. E.and Bartholomew, D., Bremner, J. M., and Brunk, H. D. *Statistical Inference under Order Restrictions; The Theory and Application of Isotonic Regression*. Wiley, New York, 1972.

5 Jan, de Leeuw, Mair, Patrick, and Hornik, Kurt. Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods. *J stat soft*, 32, 5 (2009), 1-24.

6 Tibshirani, J., Ryan, Hoefling, Holger, and Tibshirani, Robert. Nearly--Isotonic Regression. *Technometrics*, 53, 1 (2011), 54-61.

7 Smola, Alex J and Sch. A Tutorial on Support Vector Regression. *Stat. comput.*, 14, 3 (2004), 199-222.

8 Vapnik, Vladimir Naumovich and Vapnik, Vlamimir. *Statistical Learning Theory*. Wiley New York, 1998.

9 Kendall, Maurice G. A New Measure of Rank Correlation. *Biometrika* (1938), 81-93.

10 Vanderbai, Robert J. LOQO: an interior point code for quadratic programming. *Optimization Methods and Software*, 11, 1 (1999), 451-484.

11 Wu, Wei Biao, Woodroofe, Michael, and Mentz, Graciela. Isotonic Regression: Another Look at the Changepoint Problem. *Biometrika*, 88, 3 (2001), 793-804.

12 Mammen, E., Marron, J. S., Turlach, B. A., and Wand, M. P. A General Projection Framework for Constrained Smoothing. *Statistical Science: a Review Journal*, 16, 3 (2001), 232-248.

13 Ichiro, Takeuchi, Quoc, V. Le, Timothy, D. Sears, and Alexander, J. Smola. Nonparametric Quantile Estimation. *JMLR*, 1231-1264.

14 Wang, S. Neural Network Techniques for Monotonic Nonlinear Models. *Computers & OR*, 21 (1994), 143-154.

15 Daniels, H. and Kamp, B. Application of MLP Networks to Bond Rating and House Pricing. *Neural Comput. Appl.*, 8 (1999), 226-234.

16 Sill, Joseph and Abu-Mostafa, Yaser S. Monotonicity Hints. *Adv. Neural Inf. Process. Syst.* (1997), 634-640.

17 K. Makino, T. Suda, H. Ono and Ibaraki, T. Data Analysis by Positive Decision Trees. *IEICE Transations on Information and Systems*, E82-D (1999), 76-88.

18 Cao-Van, Kim and De Baets, Bernard. Growing decision trees in an ordinal setting. *Int. J..Intell. Syst.*, 18, 7 (2003), 733-750.

19 Dembczy, Kot, and S. Ensemble of Decision Rules for Ordinal Classification with Monotonicity Constraints. In *Rough Sets and Knowledge Technology*. Springer, 2008.

20 Horv, Eckhardt, Alan, Buza, Kriszti, Vojtas, P, and Schmidt-Thieme, Lars. Value-Transformation for Monotone Prediction by Approximating Fuzzy Membership Functions. In *IEEE CINTI 2011* (2011), 367-372.

21 Jacob, Varghese S, Krishnan, Ramayya, and Ryu, Young U. Internet Content Filtering using Isotonic Separation on Content Category Ratings. *ACM TOIT*, 7, 1 (2007), 1.

22 Ryu, Young U, Chandrasekaran, Ramaswamy, and Jacob, Varghese S. Breast Cancer Prediction using the Isotonic Separation Technique. *Eur. J. Oper. Res.*, 181, 2 (2007), 842-854.

23 Riihim and Vehtari, Aki. Gaussian Processes with Monotonicity Information. In *International Conference on Artificial Intelligence and Statistics* (2010), 645-652.

24 Fabien, Lauer and Gerard, Bloch. Incorporating Prior Knowledge in Support Vector Regression. *Mach. Learn.*, 70, 1 (2008), 89-118.

25 Olvi, L. Mangasarian, Jude, W. Shavlik, and Edward, W. Wild. Knowledge--Based Kernel Approximation. *JMLR*, 5 (2004), 1127-1141.

26 Gamarnik, David. Efficient Learning of Monotone Concepts via Quadratic Optimization. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (1998.), 134-143.

UCliDSS : *An Unsupervised Clinical Decision Support System for Text (Demo Paper)*

Tahir Dar
International Institute of Information Technology
Bangalore, 560100
Karnataka, India
tahir.dar@iiitb.org

Sumant Kulkarni
International Institute of Information Technology
Bangalore, 560100
Karnataka, India
sumant.k@iiitb.org

Srinath Srinivasa
International Institute of Information Technology
Bangalore, 560100
Karnataka, India
sri@iiitb.ac.in

Ullas Nambiar
EMC Corporation
Bangalore, 560048
Karnataka, India
Ullas.Nambiar@emc.com

ABSTRACT

We present our tool UCliDSS, an Unsupervised Clinical Decision Support System for textual data. UCliDSS is aimed at building a retrieval engine to retrieve documents from a bio-medical data-set to provide decision support for medical professionals. In this work, we use a term co-occurrence graph (TCG) based approach augmented with Solr to build the document retrieval engine (UCliDSS). The TCG is built from the documents of a given bio-medical data-set and is used for query expansion using a variant of random walk algorithm. The expanded query is given to Solr as input to retrieve relevant documents.

Keywords : Unsupervised Clinical Decision Support, Bio-medical document retrieval, Information retrieval

1. INTRODUCTION

Medical professionals (predominantly doctors) usually look for decision support systems to help them in different task of their daily routine like diagnosis, treatment, prescription of relevant tests and prescription of relevant effective medication. There can be different types of text documents like earlier diagnosis reports, discharge summaries, conference and journal papers which help the physician and specialist doctors make decisions about the case at hand. However, due to the sheer number of these documents, it has become almost impossible for a medical professional to manually search the relevant documents for a given problem. It becomes even more difficult problem to filter the document based on the aspects like diagnosis, treatment, test and so on. This puts forth an immediate need for building a information retrieval system for medical data. *UCliDSS* is such a system developed to serve the purpose of clinical decision support.

The motive of UCliDSS is to retrieve relevant medical documents from a large dataset, which help in answering generic clinical questions about medical conditions of patients. UCliDSS also filters the retrieved medical documents according to particular aspect (such as diagnosis, treatment, symptoms). We assume that UCliDSS works on a large corpus of medical text documents that contains information related to various medical and clinical cases. The input query to UCliDSS is a medical text scenario about the case in hand. The input query may narrate medical condition of a patients, may describe the symptoms. Additionally, it also mentions the aspect at which the user is interested. The medical documents to be retrieved from the corpus should be relevant to the input query in that aspect of the case.

For the information needs of physicians, the input queries can be put according to the many common generic clinical questions type. Some input query types are as in Table 1. The type determines the kind of question we would ask about the given case at hand.

Table 1: Some types of input queries.

Type	Generic Clinical Question
Symptoms	Symptoms of the disease ?
Diagnosis	Diagnosis of a disease ?
Prescription	Medicines prescribed to a disease ?

The input query is expected to be free text. It may consists of a current case report, summary investigation, or history of patients condition. It's *type* (aspect) is one of the generic clinical question (like mentioned in table 1). We expect the relevant text documents retrieved from the corpus.

In this work, we describe our approach in section 3. The section 4 describes the tool *UCliDSS*. Further, in section 5, we discuss an use case of the tool, where we participated in

a clinical data challenge using UCliDSS. Finally the future scope of work for UCliDSS is discussed in section 6.

2. RELATED WORK

There have been multiple approaches to information retrieval in medical domain. The most predominate ones are listed below for the sake of completeness. Collen and Flagle [1] proposed a primitive command line based medical information system which is based on a comprehensive database. However, the most predominant systems appeared post 2000. Mao and Chu [7] used phrase based vector space model to index and retrieve the medical documents. This problem suffers with the same issue of scale and issues with dimensionality reduction. Liu and Chu [6] proposed a medical IR system which used domain knowledge based query expansion. Even though it looked similar to our approach, they used human constructed thesaurus. It is a tedious task to build such domain knowledge and thesaurus. Zuccon et.al [12] came up with approach which exploited medical hierarchies for information retrieval. However, this approach relies on subsumption hierarchies which are very difficult to build. Holzinger et.al [2] presented a survey on the biomedical text mining approaches.

Even though our approach also requires background knowledge for medical information retrieval, the process of building the background knowledge is completely unsupervised. Our approach depends on the co-occurrence graph based text mining approaches [9, 5, 3, 8, 4].

3. THE METHOD

All the nouns are extracted from the given biomedical text document and a term co-occurrence graph (TCG) is built from these terms. The term co-occurrence graph represents the knowledge of the system. The TCG is treated as the background knowledge of the systems and is used for query expansion of the input query.

The text in the biomedical documents is used to create the term co-occurrence graph. We extract nouns from each text document using statistical noun phrase extraction techniques. For each paragraph having k unique nouns, we create a clique of size k with the nouns as nodes and the edge weight being one. These cliques are merged with each other to create a single large TCG. The detailed explanation of the TCG creation are in [9, 5]. We then convert it into a generatability graph (as discussed in [9, 5]).

We use the TCG for *query expansion* of the input query. Here, we expand the terms in the given input query to get more relevant terms. From the text of the input query, the nouns are extracted which are called *input query terms*. To incorporate the aspect (like diagnosis, test, symptoms) of the input query, we treat the term representing the *type* also as a part of the input query terms.

For each of the *input query term* we get their degree in TCG. We assume that a term with more neighbors is a common term and hence has lesser importance. Hence we calculated the importance score for term t as $I(t)$ as the inverse of the number of neighbours of t. The TCG is queried for the *semantic context of closure* (SCC) of the given *input query term*. The SCC of a terms on TCG returns the induced sub-graph of all the first hop neighbors of the given set of input terms. We run a variant of random walk algorithm [10] on SCC. The random walk algorithm is explained in algorithm 1. This algorithm, on stationary distribution, leaves each node in SCC with some amount of cash. Once we finish running the random-walk for all the terms in *input query terms*, we add the cash for each unique term and choose the top 20% nodes with most cash sum as the query expansion. We assume that top twenty percent of the terms represent eighty percent of the content importance. This selection of smaller number of query terms also helps in reducing noisy, unwanted query terms. To make sure that the keywords in the *aspect* are not missed out due to lesser cash sum, we append all the nouns extracted from the aspect to this list of top 20% terms. We call these terms as *expanded query* set.

Data: Generatability graph G_l, seed terms t, its importance score s, a bound on the maximum cash difference between two consecutive iterations in the random walk max_cash_dif
Result: Cash history H of all the terms in G_l
for *all nodes* $i \in G_l$ **do**
  $H[i] \longleftarrow 0$; $c[i] \longleftarrow 0$;
end
$c[t] \longleftarrow s$;
while *there exists a term* $\boldsymbol{m}$ *for which* $abs(P[m] - H[m]) \geq max\_cash\_dif$ **do**
  $historysum \longleftarrow 0$;
  $P \longleftarrow H$;
  for *each node i picked at random from* G_l **do**
    $H[i] \longleftarrow H[i] + c[i]$;
    for *each node j in N (i) and* G_l **do**
      $c[j] \longleftarrow c[j] + (\Gamma_{i \to j} \times c[i])$;
    end
    $historysum \longleftarrow historysum + H[i]$;
    $c[i] \longleftarrow 0$;
  end
  for *each node i in* G_l **do**
    $H[i] \longleftarrow \frac{H[i]}{historysum}$;
  end
end
// Stationary distribution reached
return H;

Algorithm 1: The Random walk algorithm runs random walk starting from the given node t with seed cash s on the given graph and produces the cash history H as the output.

To retrieve files from the dataset using the terms retrieved from term co-occurrence graph after query expansion, the approach is augmented with Solr a search engine[1]. Here, the given text corpus is parsed and indexed into Solr. The *expanded query* is given to Solr and relevant documents are retrieved. The topmost documents are found to be more relevant to the input query terms.

4. THE TOOL

UCliDSS is developed in ruby language and is a desktop application. The tool contains a parser which can extract text from html, xml or single column pdf file. The terms(nouns) are extracted from the unstructured text documents using an algorithm which has extended features from the *rb-brill-tagger*[2]. UCliDSS is integrated with Solr as dis-

[1]http://lucene.apache.org/solr
[2]https://github.com/taf2/rb-brill-tagger.git

cussed in section 3.

The term co-occurrence graph is stored in Agama a graph database[3] where each term is a node and each directed edge has a weight of generatability of the target term from source term.

We have a command line interface where we can perform basic operations. The co-occurrence graph can be created from a given corpus as below.

```
$uclidss --tcg <folder-having-documents>
```

UCliDSS also takes the input query and the aspect to retrieve the set of relevant documents.

```
$uclidss --query <query> --aspect <aspect>
```

5. TREC CLINICAL DATA CHALLENGE WITH UCLIDSS

We found that this framework was suitable to be applied on TREC Clinical Decision Support Track (Trec-cds-2015 challenge[4]). Here the task was to retrieve documents which can aid the medical experts in their daily routine tasks.

The dataset in Trec-cds contained seven lakh thirty three thousand documents (journal articles) in nxml format. Each document(journal article) describes a medical outcome. Many of these articles are relevant to the medical experts in certain aspects like *diagnosis*, *test prescription* and *treatment*. The task was to retrieve the most relevant of these journal articles for a given input query (with aspect).

There were thirty input queries in an xml file. Each input query (also known as topic) in Trec-cds challenge is of one of three types (aspects) – Diagnosis, Treatment and Test.

As the document corpus was huge, it was practically impossible to create a TCG of all the journal articles. We however assumed that a TCG created using a significant number of randomly sampled journal articles would be a good representative of the complete corpus. We choose around 13000 random documents and generated the term co-occurrence graph from it. This was the *background knowledge* for the UCliDSS.

UCliDSS is augmented with Solr for retrieving the journal articles. All journal articles were parsed and indexed into Solr a search engine. Solr is used after performing query expansion using random walk algorithm.

The experiments of UCliDSS were based on several factors so that we can retrieve and compare between multiple results. The experiments were done on an i7 4th generation intel machine with 30 gb ram. The execution time to get results for given 30 input queries (topics) was some 1.5 hours. The following experiments were performed.

1. The Topic as shown below consists of topic Number, type, description and summary.

   ```
   <topic number="12" type="test">
   <description>
   A 44-year-old man was recently in an
   automobile accident where he sustained
   a skull fracture.  In the emergency room,
   he noted clear fluid dripping from his
   nose.  The following day he started
   complaining of severe headache and fever.
   Nuchal rigidity was found on physical
   examination.
   </description>
   <summary>
   A 44-year-old man complains of severe
   headache and fever.  Nuchal rigidity was
   found on physical examination.
   </summary>
   </topic>
   ```

   In the first experiment textual information in both "description" and "summary" and extracted input query terms from the same and also appended the type of the topic . The sementic context of closure (SCC) for the input query terms was itself a large induced sub graph and it was heavy in terms of execution time. Hence, we decided to choose only certain percentage C of neighboring nodes with higher generatability (from the given *input query term)* to build the SCC. Even though there was no rational way of identifying what would be the right percentage of nodes C to generate SCC, we experimented with three values for C, 1%, 5% and 10%. Through the manual evaluation we realized that results with $C = 5\%$ and $C = 10\%$ were almost the same and were better than the results of $C = 1\%$. Hence, we chose $C = 5\%$ as the cutoff to generate SCC.

2. In the second experiment we extracted terms only from "description" only (and not the "summary") and performed the similar retrieval of documents as explained in the first experiment. We chose $C = 1\%$, $C = 5\%$ and $C = 10\%$ to genrate the sementic context of closures (SSC). Once again we found out that $C = 5\%$ was better due to its better quality of results even with lower number of nodes in SCC.

3. In third experiment the similar experiment as second was repeated extracted terms only from "summary" only (and not the "description") with value $C = 5\%$.

4. The above was the task A of the Trec-CDS 2015. For, Task B we were given an extra field called diagnosis. It looked like,

   ```
   <diagnosis>Bacterial Meningitis</diagnosis>
   ```

   Here, In Fourth experiment in addition to the summary filed, we also appended the data in the diagnosis filed and then preformed the query expansion. The rest of the execution remained the same.

To perform the following experiments, we had to write a small wrapper which understood the different fields in the test cases given. The results of the experiment 3 and 4 were submitted for the competition. The results have been evaluated by medical experts.

5.1 Results of TREC-CDS Challenge

The evaluation results of TREC-CDS is done based on the paper [11] which describes two methods of large scale retrieval evaluation, infAP(inferred Average Precision) and infAPNDCG(inferred Average Precision Normalised Discounted Cumulative Gain). In addition for our results R-Precision

[3]https://github.com/arrac/agama.git
[4]http://www.trec-cds.org/2015.html

Table 2: tab: Evaluation Results of Trec-cds

Experiment Number	Input Query	Mean infAP	Mean infNDCG	Mean R-precision	Mean Precision@10
1	Summary + Description	0.0402	0.1929	0.1592	0.3000
2	Description	0.0322	0.1772	0.1445	0.2733
3	Summary	0.0277	0.1534	0.1169	0.2500
4	Summary with diagnosis	0.0450	0.2145	0.1577	0.3233

and Precision@10 is also calculated which are known measures of information retrieval evalvation. The table 2 shows the measures calculated for our results on TREC-CDS data and input queries. All the above measures for the information retrieval are based on a ranked relevancy file which contains ranked relevant document ids for each input query. The relavancy file for TREC-CDS data has been provided by NIST. Based on the relavancy file the measures in table 2 have been calculated.

6. FUTURE WORK

The future work is to develop an optimized algorithm in terms of execution time to create a term co-occurrence of all the documents in a large corpora. The document-id of the document will be made to co-occur with all the terms/concepts of the document in the co-occurrence graph. The cash leaking random walk on the induced sub-graph of the neighbours of the input terms will be applied. Once the stationary distribution is achieved, system will identify the journal-ids which have accumulated the most cash history and declares them as the relevant documents for input query. Thus the role of Solr a search engine will be reduced. We also intend to implement a distributed version of the algorithm.

7. REFERENCES

[1] M. F. Collen and C. D. Flagle. Full-text medical literature retrieval by computer: a pilot test. *JAMA*, 254(19):2768–2774, 1985.

[2] A. Holzinger, J. Schantl, M. Schroettner, C. Seifert, and K. Verspoor. Biomedical text mining: State-of-the-art, open problems and future challenges. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, pages 271–300. Springer, 2014.

[3] S. Kulkarni and S. Srinivasa. Sortinghat: a deep matching framework to match labeled concepts. In *Proceedings of the 20th International Conference on Management of Data*, pages 134–137. Computer Society of India, 2014.

[4] S. Kulkarni, S. Srinivasa, and R. Arora. Topic expansion using a term co-occurrence graph. Technical report.

[5] S. Kulkarni, S. Srinivasa, J. N. Khasnabish, K. Nagal, and S. G. Kurdagi. Sortinghat: A framework for deep matching between classes of entities. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 90–93. IEEE, 2014.

[6] Z. Liu and W. W. Chu. Knowledge-based query expansion to support scenario-specific retrieval of medical free text. *Information Retrieval*, 10(2):173–202, 2007.

[7] W. Mao and W. W. Chu. The phrase-based vector space model for automatic retrieval of free-text medical documents. *Data & Knowledge Engineering*, 61(1):76–92, 2007.

[8] A. R. Rachakonda, S. Srinivasa, S. Kulkarni, and M. Srinivasan. Mining analytic semantics from unstructured text. Technical report.

[9] A. R. Rachakonda, S. Srinivasa, S. Kulkarni, and M. Srinivasan. A generic framework and methodology for extracting semantics from co-occurrences. *Data & Knowledge Engineering*, 2014.

[10] M. Yazdani and A. Popescu-Belis. A random walk framework to compute textual semantic similarity: a unified model for three benchmark tasks. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 424–429. IEEE, 2010.

[11] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 603–610, New York, NY, USA, 2008. ACM.

[12] G. Zuccon, B. Koopman, A. Nguyen, D. Vickers, and L. Butt. Exploiting medical hierarchies for concept-based information retrieval. In *Proceedings of the Seventeenth Australasian Document Computing Symposium*, pages 111–114. ACM, 2012.

Towards a General Framework for Data-Driven City Comparison and Ranking

Vishalaksh Aggarwal
IBM Research - India
vishalaksh@in.ibm.com

Biplav Srivastava
IBM Research - India
sbiplav@in.ibm.com

Srikanth Tamilselvam
IBM Research - India
srikanth.tamilselvam@in.ibm.com

ABSTRACT

Knowing about cities that one lives in or wants to visit is of much interest to citizens, tourists, businesses, investors and governments. Open government data provides us this opportunity since data about various domains like crime, traffic and health are being made available by the government. In this paper, we present our approach of using open data from multiple agencies and domains in comparing and ranking cities in a developing country. The framework relies on vocabulary based data normalisation to overcome data collection noise and easily scales with new domains.

Keywords: City Comparison, Inconsistent Data, Data Preparation, Open Data, Clustering, Visualization
Category of Submission: Demonstration
Demo URL: http://city-explorer.mybluemix.net/
Demo Status: Prototype ready

1. INTRODUCTION

A valuable piece of information for citizens, tourists, businesses, investors and governments is to know how good a city is in itself and in comparison with others can be valuable. The traditional way to know this is by surveys. However, such results have many problems like limitations of sample size and possibility of survey bias.

This is where open data can help. Open data is the practice by organizations and governments to make their data amenable to reuse. For cities, governments around the world are making data available about various domains like crime, accident and health[1].

A number of data-driven approaches for exploring cities are coming up. The approach of Global City Indicators (GCI) [2] is to define a set of indicators grouped around themes like *governance*, *people* and *safety*. The indicators are expressed using terms from an ontology[3] and its quantitative values are calculated to help compare cities. City Data[4] is a new initiative where data is collected from mutliple sources and then organized for rapid discovery. Unfortunately, it works only for US cities and technical details are not public.

In this paper, we present the City Explorer app which generates insights about cities and their comparison using open data. What sets it apart from other approaches is that it works directly with noisy data prevalent in developing countries when multiple decentralized agencies produce data. The city data comes from multiple domains (like crime, accidents and health), is organized by districts and spans multiple years. We uses data cleaning and vocabulary based normalization to prepare grounds for city exploration across domains and cities.

Our contributions are that we:

1. formalize characteristics of Indian open data, a first in literature.
2. demonstrate multi-dimensional exploration of city performance based on open data across domains and time
3. perform vocabulary-based city data normalization across domains
4. demonstrate unprecedented multi-dimensional comparison of all cities where data allows.
5. provide a scalable framework which will work for more domains and data from more countries

The rest of the paper is organized as follows: we begin with a background and formalization of Indian open city data and then present our solution approach. We then discuss the salient points with examples and give pointers to future work.

2. DATA CHARACTERISTICS

India has over half a million villages and hundreds of towns and cities. However, the recognized unit of territory is a *district*. The full list of districts in India along with their standard names and unique identifier can be computed from the official controlled vocabulary [5]. (The vocabulary itself has list of states and list of districts for each state). We will refer to it as D^* and call it Normalized District Names (NDN). Its size is 721 (i.e., $\mid D^* \mid$).

We use data about Indian districts from India's open data portal[6]. They correspond to the domains of crime[7], accidents[8] and health services[9]. The data is about districts and each city can have 1 or more districts. We use the terms *city* and *district* interchangeably.

2.1 A Formalization of Data Used

The input data is a set S of 3-dimensional vectors Vs. Each vector V_i represents data of a domain i. We use the domains of crime (V_c), accidents (V_a) and health (V_h). The dimensions (x,y,z) of V_i represent districts, domain attributes and year, respectively.

A reported data for a domain V_i is (x_j, y_k, z_l). Here, $x_j \in D^{V_i}$. We will use D^{V_i} to refer to the set of districts in a V_i and call them

Domain	$\mid D^{V_i} \mid$	$\mid A^{V_i} \mid$	$\mid Y^{V_i} \mid$
Crime (V_c)	807	30	12
Accident (V_a)	50	4	1
Health (V_h)	629	5	1

Table 1: Statistics about data used.

Domain	$\mid V_i \mid$	$\mid V'_i \mid$
Crime (V_c)	8597	6843
Accident (V_a)	50	40
Health (V_h)	637	539

Table 2: Statistics about data filtered based on district name analysis.

Data District Names (DDN). The data for the domains come from different agencies and do not use the normalized district names. y_k refers to a domain attribute from the set of attributes A^{V_i} which can vary from domain to domain. z_l refers to a year. The datasets may have different range of years and we use Y^{V_i} to refer to years in V_i.

2.2 Data Challenges

The challenges with the data are:

1. Inconsistent naming of districts across all datasets. We found that $D^{V_i} \nsubseteq D^*$, for V_c, V_a and V_h, as one would have expected, indicating district names are not in the master list.
2. Inconsistent availability of data for different districts across domains.
3. Inconsistent availability of data for years. Even in the same domain, data for different districts need not be for the same years.
4. Inconsistent labeling of missing data with NA, - or blanks.

We overcome the inconsistent naming problem by using a standard district name vocabulary[5]. Since the data came from multiple sources, there were disparity in districts,state names like 'Kolkata', 'Calcutta' and 'New Delhi', 'Delhi' to name a few. To resolve them across datasets, we used heuristics[10] to arrive at potential matches which were then manually verified. If any of the following happens, it is a potential match: (1) If a district's name was contained in another's name, (2) the similarity in names between two districts is above 0.4 as measured by cosine similarity, (3) The rest were then manually verified to resolve disparity. To handle inconsistent data availability, we restrict comparison to only the cases where districts and years are common.To tackle inconsistent labeling of missing data, we simply assumed all the data which could not be parsed into numbers as missing data.

3. SOLUTION APPROACH

Our solution is a two step process where in the first step input data V_i is normalized and filtered based on D^*. This offline process is then followed by the online process where selected districts X^{V_i} for the years Y^{V_i} are pair-wise compared and displayed. We also demonstrate that when data exists, we can also do all-pair comparison to gain meaningful insights about cities in India.

The resulting City Explorer system is shown in Figure 1. It is cloud-based and available online[11].

3.1 Data Preparation

In this offline process, the controlled vocabulary services[4] is accessed by REST API to get the latest state information, namely unique state code and state name. Currently it has information on all 36 Indian states and union territories. It is available in both XML and JSON format. For our solution, we rely on JSON format. It also exposes APIs to get district and taluk (another administrative terriotrial unit) level information for each of the state. Our experiments are based upon state and district level information alone. District level information are merged with its state details and this we refer to as Normalized District Names (NDD) D^*.

Each district name j in V_i (in S) is matched with D^*. Table 3 shows result of exact match comparison while Table 4 shows substring match. The second approach increased matching values substantially, but had errors that needed human intervention. For example, substring match equated 'PATNA' city with 'VISAKHAPATNAM' and 'ASANSOL DURGAPUR' with 'DURG' though they are not the same in reality. Such errors accounted for roughly 1% of the districts. Each matched records were analyzed by two annotators to remove such wrong matches. We also found synonymous places with different spellings like 'Bengaluru', 'Bglr', 'Bangalore' which we did not include in our current matching. Only districts matching D^{V_i} are retained for further processing while the rest are discarded. Table 2 shows details of domain wise retained data D^{V_i}. (V_h originally had many synonymous districts).

3.2 Pair-wise City Comparison

Comparing a pair of cities (districts) c_1 and c_2 means one wants to compare two corresponding vectors $v_1^i(c_1, i, j)$ and $v_2^i(c_2, i, j)$ for each domain i and year j. The notation can be suitably modified if a city is being compared to its own performance in a domain but in a different year.

For each of the city chosen for comparison, its relative ranking for each of attributes A^{V_i} of vector V_i is computed for the same year Y^{V_i}. The left side of Figure 1 shows barometer ranking of both the compared districts which represents relative ranking of the selected districts with respect to other districts on the two extremes for each of the attribute, in this case highest and lowest number for attribute murder. The positioning on the linear scale is based on the relative number of crimes. Likewise, the color coding is also based on relative number of crimes. The district with lowest number of crime is marked in green color (lowest rank), while the one with highest is marked in red (highest rank) and the one with the average of two numbers in yellow is positioned in the middle. The number of crimes are shown below the name of the district. The selected district is written below the scale while the extremes are written above the scale. The Figure in brackets signifies its relative rank out of the total number of districts in scope for that year and domain. (A minor note is that if a category, like health facilities, is of reverse semantics (where more is better), the signs of values are reversed before processing.) In the right hand side of Figure 1, the line series chart displays the distribution of an attribute for the years Y^{V_i}.

To get a composite view of the two cities across all attributes in a domain, we calculate a dominance score between the cities as defined below with δ_t^i. The score is asymmetric and measures the percentage of times a city dominates the other with ϵ accounting for the cases where their ranks are the same. Figure 2 shows the scores calculated for two cities in the system for each domain.

$$\delta_t^i(c_1, c_2) = \frac{\#(rank(c_1) \prec rank(c_2)) * 100}{\mid A^{V_i} - \epsilon \mid} \quad (1)$$

We now calculate the overall dominance score across all domains as defined below with δ_t. For the experiments, w_i was 1, thus

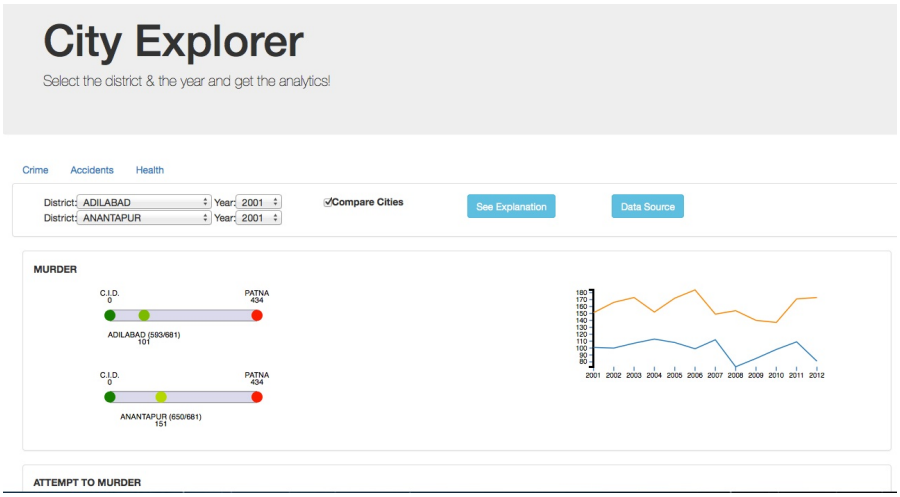


Figure 1: A City Explorer View.

DatasetNamel	# Total Records	# Matches	# Non Matches
Crime	807	485	302
Accident	50	35	15
Health	629	396	133

Table 3: Exact matches between DDNs and NDNs

DatasetNamel	# Total Records	# Matches	# Non Matches
Crime	807	621	186
Accident	50	40	10
Health	629	544	85

Table 4: Partial matches between DDNs and NDNs

Figure 2: Comparing a pair of cities.

weighing all attributes equally across all the domains.

$$\delta_t(c_1, c_2) = \frac{\sum_i w_i.\delta_t^i(c_1, c_2)}{\sum_i w_i} \quad (2)$$

We now define the dominance relation $\succ_t$ between two cities for year t iff:

$$(c_1 \succ_t c_2) = \begin{cases} true, & \text{if } \delta_t(c_1, c_2) \succ 50\% \\ false, & \text{otherwise} \end{cases}$$

In Figure 2, Chandigarh $\succ_t$ Patna and conveys that Chandigarh broadly dominates Patna across the considered categories and their attributes.

3.3 Comparing All Cities

Once we can compare a pair of cities, we also try to compare all cities for which data is available. A problem we faced was that not all cities (districts) release data for all the domains. Hence, the analysis was restricted to only 17 districts that the data was available and only for 2012.

In Figure 3, dominance relationship for all districts are shown, where the district had published data for all the domains under consideration, i.e., crime, accident and health. Here, an edge from district A to district B represents the dominance of A over B. We notice that Indore dominates all cities (source node) and Dhanbad is dominated by all (sink node). Hence, they correspond to the best and worst cities based on available data. We are not aware of any prior work giving this insight.

4. DISCUSSION AND FUTURE WORK

We presented a general data-driven approach of comparing cities using open data in a developing country. The data mining contributions relate to:

1. Data preparation - handling missing data and district normalization
2. Clustering - coming up with a general city comparison framework across categories and attributes
3. Interactive visualization - that can handle missing data and time

The data is noisy and non-uniform. As a result, the insights presented are preliminary and likely to improve / change with better data publishing practices. Specifically, we note a data-based limitation of the current results. The tier-1 cities in India are: Delhi, Mumbai, Kolkatta, Chennai, Bangalore and Hyderabad. However, none of them figure in the list of 17 cities for all-city comparison. The reason is that an official mapping of cities to districts is not available and we must create one manually.

The work can be extended in many ways: (a) We presently looked at simple weighted aggregation function for comparing cities but many others can be considered. (b) The controlled vocabulary for India has information for only districts. One can extend it to cities and villages, and then report results at these level of granularities. (c) For India, one can build a map between cities and the districts they contain to support drill-down, roll-up of comparisons and results. (d) One can normalize city data based on their population to

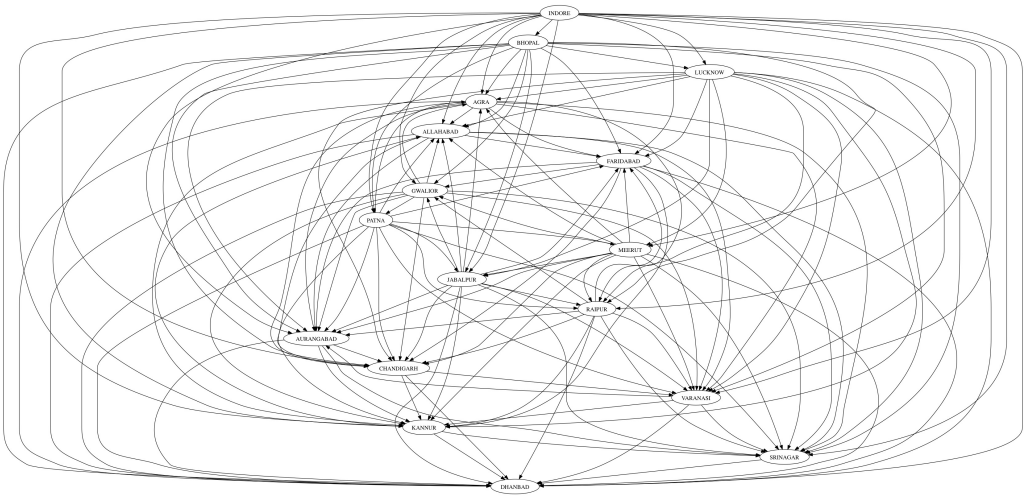


Figure 3: Dominance relationship among districts based on 2012 crime, accident and health data.

provide a more balanced comparison. (e) One can extend to data from more countries [12].

Apart from the general methods developed, we addressed a few UI challenges in building the online system. We believe they are important for any useful system in this space.

1. Making the selectable input options intuitive: We had to fetch a unique list of districts from dataset and arrange them alphabetically so that the user can easily locate the desired district in the dropdown. Also, once the district was selected, we had to first find out the list of years for which data was available for the selected district from the dataset and display it in the second dropdown.

2. Responsiveness to user selections: we have to continuously watch for any new selection of district or year that user makes from the dropdown so that the charts are re-rendered upon the selection.

3. Make the visualizations responsive: The visualizations should not overlap each other upon the change of size of the browser window. Upon decrease of width, the 2nd visualization comes below the 1st one so that both of them are still visible and the user only has to scroll downwards.

5. CONCLUSION

In this paper, we presented a general approach for comparing and ranking cities using open data. We took India as the case study and considered data from different domains from multiple agencies and domains. The framework is general-purpose and can give novel insights about a city with respect to its past, relative to its peers and as a whole group.

6. REFERENCES

[1] "Open data barometer report, second edition 2015," http://barometer.opendataresearch.org/, accessed 6 May 2015.

[2] GCI, "Global city indicators," in *At http://www.cityindicators.org/Deliverables/ListAccessed 6 May 2015*, 2015.

[3] M. Fox, "Foundation ontologies requirements for global city indicators," in *The AAAI 2014 Workshop on Semantic Cities: Beyond Open Data to Models, Standards and Reasoning, Quebec City, Canada*, 2014.

[4] "City data," http://www.city-data.com/, accessed 6 May 2015.

[5] "Controlled vocabulary services," http://vocab.nic.in, accessed 6 May 2015.

[6] "Indian open data," http://data.gov.in, accessed 6 May 2015.

[7] "District-wise crime under various sections of indian penal code (ipc) crimes," https://data.gov.in/catalog/district-wise-crimes-under-various-sections-indian-penal-code-ipc-crimes, accessed 6 May 2015.

[8] "Road accidents profile of selected cities — open government data (ogd) platform india," https://data.gov.in/catalog/road-accidents-profile-selected-cities, accessed 6 May 2015.

[9] "District-wise availability of health centres in india," https://data.gov.in/catalog/district-wise-availability-health-centres-india, accessed 6 May 2015.

[10] A. Singhal, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[11] "City explorer," http://city-explorer.mybluemix.net/, accessed 9 July 2015.

[12] "Us open data," http://data.gov, accessed 6 May 2015.

COMAD 2016 Sponsors

In association with

acm India
ACM Pune Professional Chapter

Platinum Sponsors

ORACLE

Gold Sponsors

Flipkart

Microsoft Research

TATA CONSULTANCY SERVICES

Facility Sponsors

PERSISTENT