Managing Data Through Band Order Dependencies

by

Yaksh Joshi

Undergraduate Honours Thesis
Faculty of Science (Computer Science)
Ontario Tech University
Oshawa, Ontario, Canada

Supervisor(s): Dr. Jaroslaw Szlichta

Abstract

Ordered Series are prone to multiple inaccuracies within data due to multiple external factors. It is important to identify such inaccuracies within the series. We propose a demonstration plan for the users to rank the series, identify the inaccuracies and repair them using our three different repair algorithm. This demonstration will also help users to identify which series outweigh other series in terms of overall quality of data through our measure of interestingness algorithm.

Keywords: Ranking Series; Repairing Series; Measure of Interestingness

Dedication

This thesis is dedicated to my family, friends and professors who have supported throughout my education.

Acknowledgements

I would like to thank my supervisor and mentor, Dr. Jaroslaw Szlichta for the support and the opportunity that is given to me to do research under his supervision. Jessica for code base and directions to setup the project repository and Naida Tania for helping me on front-end of the project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

High quality data is required for meaning analysis so that meaningful decisions can be made. Higher quality of data is must, as the data is processed using AI technologies which are all automated. It is crucial to understand the semantics of the data.

Data dependencies are the key to capture such semantics. Previous work focused on Functional Dependencies (FDs) [2]. Order Dependencies (ODs) [3, 4, 5], one of the several extension of the FD have been studied and used to express semantics involving ordered data. To understand better, Table 1.1 shows sample releases of the *Music* dataset (Reprise records) from Discogs[1] that are integrated from various sources. It is a common practice in music companies to assign catalog number (cat#) to each release of a particular label for tracking.

To demonstrate ODs, the attribute cat# is ordered in the series, there is likely to be another attribute/s that are ordered. For this case, the release date (combination of year and month) is approximately ordered over a subset of the data called the series i.e., $\{t_1\text{-}t_9\}$ and $\{t_{10}\text{-}t_{14}\}$ which are ordered in ascending order and $\{t_{15}\text{-}t_{22}\}$ which is ordered

[1] www.discogs.com

Table 1.1: Reprise records.

id	release	country	year	month	cat#
t_1	Unplugged	Canada	1992	Aug	CDW45024
t_2	Mirror Ball	Canada	**2012**	Jun	CDW45934
t_3	Ether	Canada	1996	Feb	CDW46012
t_4	Insomniac	Canada	1995	Oct	CDW46046
t_5	Summerteeth	Canada	1999	Mar	CDW47282
t_6	Sonic Jihad	Canada	2000	Jul	CDW47383
t_7	Title of...	Canada	1999	Jul	CDW47388
t_8	Reptile	Canada	2001	Mar	CDW47966
t_9	Always...	Canada	2002	Feb	CDW48016
t_{10}	Take A Picture	US	2000	Nov	9 16889-4
t_{11}	One Week	US	1998	Sep	9 17174-2
t_{12}	Only If...	US	1997	Nov	9 17266-2
t_{13}	Never...	US	1996	Nov	9 17503-2
t_{14}	We Run ...	US	1994	Dec	9 18069-2
t_{15}	The Jimi...	US	1982	Aug	9 22306-1
t_{16}	Never...	US	1987	Jan	9 25619-1
t_{17}	Vonda Shepard	US	1989	Mar	9 25718-2
t_{18}	Ancient Heart	US	**Null**	Jul	9 25839-2
t_{19}	Twenty 1	US	1991	May	9 26391-2
t_{20}	Stress	US	1990	Apr	9 26519-1
t_{21}	Play	US	1991	Mar	9 26644-2
t_{22}	Handels...	US	1992	Apr	9 26980-2

in descending order.

Note that t_3 has smaller cat# than t_4, but it is released a few months later. It is common in music industry to assign cat# to a music label before it is released at the production stage. To address small variations in the series, band ODs is introduced, where band is a permissible range to accommodate these small variations. Note t_2 in series $\{t_1\text{-}t_9\}$ and t_18 in series $\{t_{15}\text{-}t_{22}\}$ severely breaks the OD between cat# and (year and month).Those tuples cannot be categorized as inaccuracies but instead as errors. To discover ODs, abcOD - a type of data dependency is used, which generalizes band ODs to hold approximately with some exceptions (abODs [6]).

After identifying series obeying abcODs, it is important to rank them so to measure the statistical significance of each series. In this paper, we rank the series using *measure of interestingness [5], [1].* By understanding the significance of cleaning the data, we also included three different approaches namely

1. Min/Max LMB Average
2. Winsorization.
3. Machine Learning (Linear Regression)

to repair the series.

Chapter 2

System Overview

2.1 Definitions

Few important terminologies used in paper are explained below.

2.1.1 Bidirectional Band OD

Given a band-width Δ, a list of attributes $\mathbf{X}$, a marked list of attributes $\overline{\mathbf{Y}}$ and a bidirectional band order dependency denoted by $\mathbf{X} \mapsto_{\Delta} \overline{\mathbf{Y}}$ holds over a table r, if $t \preceq_{\mathbf{X}} s$ implies $t \preceq_{\Delta,\overline{\mathbf{Y}}} s$ for every tuple pair $t, s \in r$. Bidirectional Band ODs specify that if the tuples are ordered increasingly on the left hand side (i.e., cat#) then the right hand side (i.e., year) should be ordered non-decreasingly or non-increasingly within the specified band-width.

2.1.2 Band OD

A bidirectional band order dependency is called a band OD when a list of attributes within it is all marked as ascending or all as descending.

2.1.3 LMB

LMB is defined as a sequence of tuples $T = \{t_1, t_2, \cdots, t_n\}$, a marked list of attributes $\overline{\mathbf{Y}}$ and band-width Δ, a *monotonic band* (MB) is a subsequence of tuples $M = \{t_i, \cdots, t_j\}$ over T, such that $\forall_{k_1,k_2 \in \{i,\cdots,j\}, k_1<k_2}\ t_{k_1} \preceq_{\Delta,\overline{\mathbf{Y}}} t_{k_2}$. The longest subsequence M satisfying this condition is called a longest monotonic band (LMB).

2.1.4 Coverage

For a approximate band OD $\mathbf{X} \mapsto \mathbf{Y}$, in a given table r, coverage is defined as $coverage(\varphi) = \frac{|\{t_i,t_j\}|t_i \neq t_j \in r, \{t_i,t_j\} \models \varphi|}{n*(n-1)/2}$ where t_i and t_j are the pair of tuples satisfying this dependency and n is the total number of tuples in table r. The numerator of coverage counts total number of different evidences (abOD) which is divided by total number of possible evidences.

2.2 Discovery of abOD

In real world data, band OD does not hold due to errors in data. But with inclusion of approximation band OD for the series can be satisfied, thus abOD. The computation of LMBs are used to calculate abOD. Given a band OD $\mathbf{X} \mapsto_\Delta \overline{\mathbf{Y}}$, the goal is to verify whether a band OD holds, such that inconsistent tuples that severely violate a band OD are few. Given a band OD φ: $\mathbf{X} \mapsto_\Delta \overline{\mathbf{Y}}$ and table the approximate band OD discovery problem is to identify the minimal set of tuples that violate a band OD with an error ratio $e(\varphi) = min\{|t|\ :\ t \subseteq r, r \setminus t \models \varphi\}/|r|$.

2.3 Discovery of abcOD

To make the band OD relevant to the real world application, the goal is to segment the sequence into multiple sequences known as series, which are contiguous and non-overlapping sub-sequences of tuples. By doing so, the large proportion of tuples in each series will satisfy the band OD and there will be very few instances of anomaly tuples in the series. Let S be a non-overlapping segmentation that splits T into m segments where $\mathbf{X} \mapsto_{\Delta} \mathbf{Y}$ holds in eachi.e., $S = \{T[1,i], T[i+1, i+k], \cdots, T[i+j, n]\}$, $1 \leq i \leq n, 1 < k \leq j \leq n$. The description complexity of segment S_i with length k is $\ln(k+1)$. The description complexity $g(S)$ of segmentation S is $\sum_{i=1}^{m} \ln(|S_i|)$, where $|S_i|$ is the length of segment S_i in T.

2.4 Measure of Interestingness

After discovering the series satisfying abcODs, they are ranked and scored using Measure of Interestingness. Measure of Interestingness is calculated with the help of coverage(φ) formula. By ranking each series, it can help data analyst and data scientist to use the series with more statistical significance.

Chapter 3

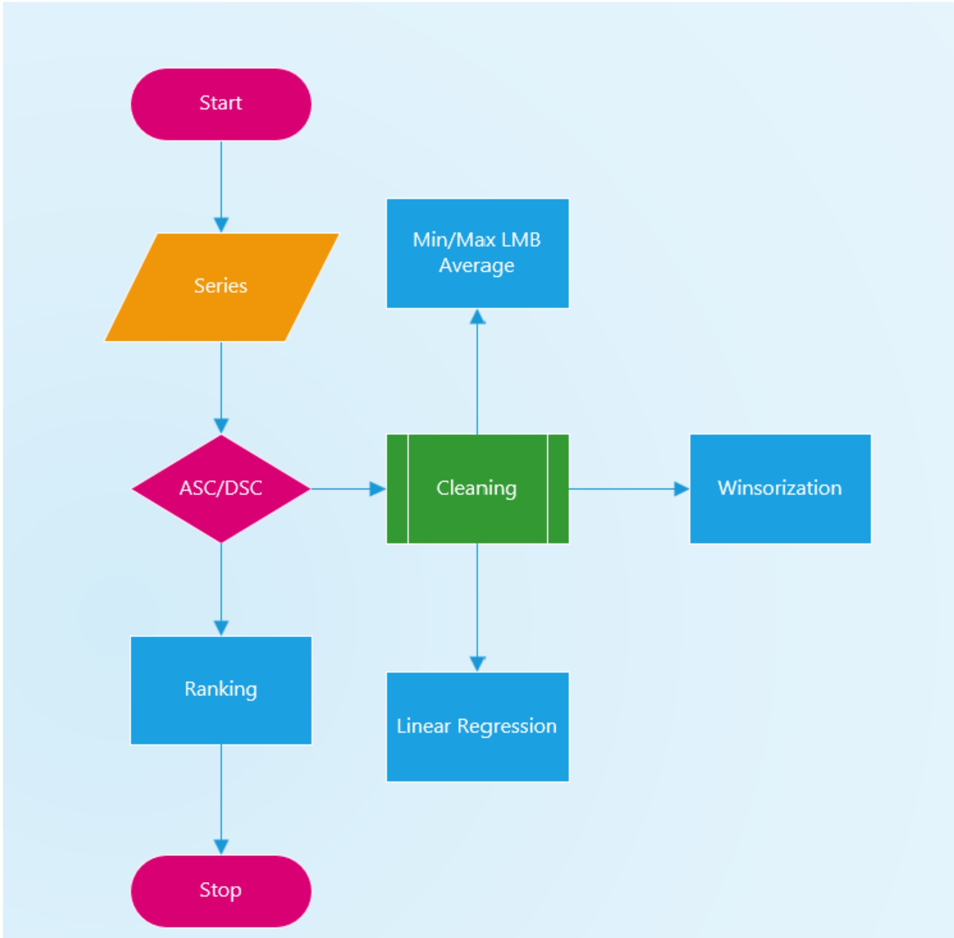
Ranking and Cleaning

The series used for computation are extracted from the previous work done on finding abcODs. The development of algorithm is done using Java language on Eclipse IDE.

3.1 Flow of Computation

The first step of the computation is identifying the semantics of the series. As the sequence for a dataset is broken into smaller series, it is easier to work on each series to identify its behaviour (ASC or DSC). After identifying, the series goes through the Ranking Algorithm developed using Measure of Interestingness. Simultaneously, the cleaning algorithm identifies the anomaly tuple/s (if exist) in the series. Then the cleaning algorithm will suggest three different repair for detected error/s in the series based on three different techniques. The flow of the computation is illustrated in the given figure 3.1.

Figure 3.1: Flow of Computation



3.2 Ranking Series

3.2.1 Description

To rank a series, it is evident that the series need to be obeying abcOD. After the series is obeying the dependency, it is passed through a conditional statement to identify if the series is ascending/descending order. After which, using for loop, we are comparing all the tuples with each other to check if the semantics is continuous throughout the series. We will compare all the tuples with each other such that the tuple which is compared to the subsequent tuples should always be less than or equal to that tuple. If a tuple is greater than the comparing tuples, we discard that instance, Unless the difference between those tuple is less than or equal to the band, for which we include them as they will be regarded as inaccuracy. We have a placeholder integer to calculate the instances in which all the pair of tuples passed the dependency. After completing calculation for all tuples, we divide our total instance count by the total number of tuples to calculate the coverage(i.e.,Rank).

The series will be assigned integer $\in$ [0,1]. The series without a single anomaly tuple will score 1 (ideal) and series with multiple anomaly will score close to 0. For the real world, the ideal series with perfect score will produce more significant analysis than the one with a low score.

3.2.2 Pseudo Code

Algorithm 1 Ranking Series

```
Require: SeriesS
Ensure: Rank(φ) ∈ [0, 1]
  denom = len(S) ∗ len(S − 1)/n                ▷ n is the total number of tuples
  count = 0
  if S in ascending then
      for i in S do
          for j in S do
              if S[i] ≤ S[j] then
                  count + +
              else
                  if i − j ≤ Δ then
                      count + +
  Rank(φ) = count/denom
  return Rank(φ)
```

3.2.3 Example

For the series $\{t_1$-$t_9\}$, we see that t_2 breaks the OD so can be counted as error. For the first iteration, t_1 pairing with all the other tuples will be counted as a positive instance. So count will be 8. In the same manner, if we calculate for each iteration we get coverage(φ)

for this series to be $\frac{8+6+5+4+3+2+1}{36}$ 0.805.

3.3 Cleaning Series

In the real world, ordered series are prone to many errors. It would be a bad practice, if we discard the series because of its ranking. The series containing more tuples but having low rank than the series with the perfect score but negligible amount of tuples, can be considered a better option for the analysis. To address this issue, we developed three different techniques to repair the series.

3.3.1 Min/Max LMB Average

Previous repair techniques suggested the repairs using the standard Average method, which is the average of adjacent tuples but it did not align with the real world scenarios. We are introducing the Min/Max LMB Average Technique, this technique include the computation of LMB for a series. While calculating the rank, we made a list of errors, if encountered. We also have the LMB as a seperate list for the calculation. We comapre both the list, to create two lists recursively. The newly created list will have the tuples from the left and right of the anomaly tuple. We than take the maximum value tuple from the left list add the tolerance to it (Δ). Take that value and add the minimum value tuple from the right list and divide the product by 2 to get the repair value for the tuple.

Pseudo Code

Algorithm 2 Repairing Using Min/Max LMB Average

Require: $Series, S$
Ensure: $SuggestedRepair/s$
$error - list = [errors]$ ▷ Calculated simultaneously in Ranking
$l - m - b = [t_1, .., t_9]$ ▷ Calculated simultaneously in Ranking
for i in $error - list$ **do**
  for j in $l - m - b$ **do**
    if $j \leq i$ **then**
      $left.append(l - m - b[j])$
    else
      $right.append(l - m - b[j])$
$Repair = (Max(left) + \Delta + Min(Right))/2$
return $Repair$

3.3.2 Winsorization

Winsorization technique follows the similar pattern like Min/Max LMB Average but it is more robust as it compares the Minimum and Maximum tuple from left and right list with the erroneous value, and this technique selects the closest acceptable value to the error.

Pseudo Code

Algorithm 3 Repairing Using Min/Max LMB Average

Require: $Series, S$
Ensure: $SuggestedRepair/s$
$error - list = [errors]$ ▷ Calculated simultaneously in Ranking
$l - m - b = [t_1, .., t_9]$ ▷ Calculated simultaneously in Ranking
for i in $error - list$ **do**
  for j in $l - m - b$ **do**
    if $j \leq i$ **then**
      $left.append(l - m - b[j])$
    else
      $right.append(l - m - b[j])$
if $d(Max(left), error)d(Min(right), error)$ **then**
  $Repair = Min(right) + \Delta$
else
  $Repair = Max(left) + \Delta$
return $Repair$

3.3.3 Linear Regression

Our last technique uses Machine Learning technique called Linear Regression. As our series are ordered, it seemed to apply Linear regression technique for repairing the value. For this case, we trained our model with lots of series, the erroneous values were gradually removed and then for the test case we passed the LMB list with null tuples as a placeholder for erroneous values. We are using simple Linear Regression model, in which we have the single input (year).

We use, Y = a + bx model, where Y is the dependent variable , X is an independent variable, b is the slope of the line and a is the y-intercept. For our case, X is $index of the tuple$ and Y is year.

Chapter 4

Web Demonstration

4.0.1 Demonstration Plan

As our preliminary work included working with Java. For the demonstration plan, as we already exported the series to the .csv format for calculation. My plan was to have a two different webpages showcasing ranking and cleaning of the series. I developed two webpages, one which showcase the ranking of the series and the other in which you can repair series with different methods. My focus was to demonstarte the ranking and also user friendly cleaning strategy.

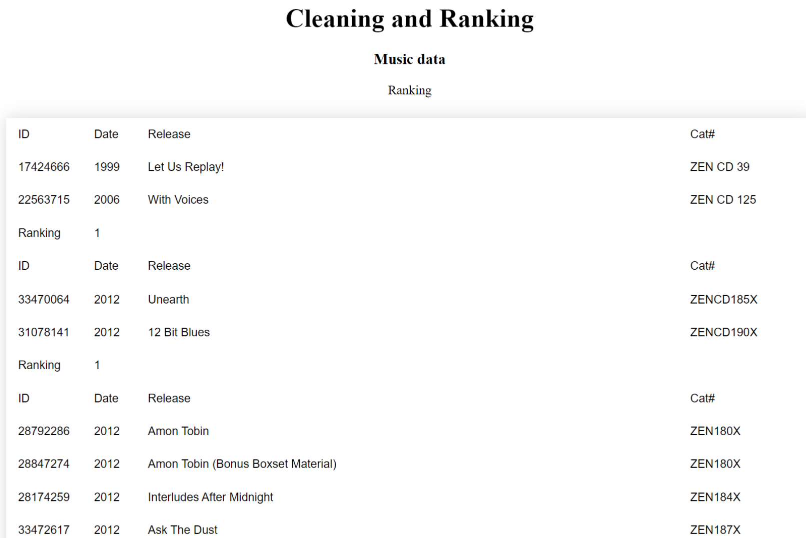
4.0.2 Technology

For Front-end of the project, I used php and javaScript and for the backend, I used Java. I utilized sublime and Eclipse IDE for the development and used xampp server to run on my local machine. The data was in .csv format.

4.0.3 Ranking Demo

In the figure 4.1 and 4.2, you can notice that we have multiple series and all of them are assigned rank to them. With the help of this web interface, it is easier to recognize the series with the highest rank. It can save much time and it also showcases the effectiveness of our ranking algorithm.

Figure 4.1: Ranking Demo


Cleaning and Ranking

Music data

Ranking

ID	Date	Release	Cat#
17424666	1999	Let Us Replay!	ZEN CD 39
22563715	2006	With Voices	ZEN CD 125
Ranking	1		
ID	Date	Release	Cat#
33470064	2012	Unearth	ZENCD185X
31078141	2012	12 Bit Blues	ZENCD190X
Ranking	1		
ID	Date	Release	Cat#
28792286	2012	Amon Tobin	ZEN180X
28847274	2012	Amon Tobin (Bonus Boxset Material)	ZEN180X
28174259	2012	Interludes After Midnight	ZEN184X
33472617	2012	Ask The Dust	ZEN187X

4.0.4 Cleaning Demo

For the cleaning demonstration, my focus was to make it more interactive with the user. To do so, the tuples with erroneous values were marked as red. The user can freely tap on the erroneous tuple and can see our three techniques to work. After tapping on the tuple, a pop-up window transitions to the top with the suggested repair values for the tuple selected. In below Figures 4.3, 4.4 and 4.5 you can see the examples.

Figure 4.2: Ranking Demo II

4022238	2006	Sleep / Untitled Dialogue	ZEN 10156
160457	2001	Receiver	ZEN 12100
200154	2001	Ataride / Tomorrow Acid	ZEN 12101
186890	2001	The Great Drive By	ZEN 12102
25124788	2001	The Great Drive By	ZEN 12102
354477	2002	All That You Give	ZEN 12103
226778	2001	Get A Move On / Ug	ZEN 12104
20195421	2001	Inner Spacesuit	ZEN 12105
226769	2002	Pneumonia	ZEN 12106
1335036	2002	Something Wicked / Mr Holmes	ZEN 12111
22197022	2002	Verbal	ZEN 12118
5034686	2002	Hello / One Session	ZEN 12120
2729132	2002	Time To Build / Distinguished Jamaican English	ZEN 12122
21309849	2002	Sweetsmoke	ZEN 12124
Ranking	0.96		

Figure 4.3: Cleaning Demo

Ranking and Cleaning Example

ID	Release	Country	Year	Month	Cat#
t1	Unplugged	Canada	1992	Aug	CDW45024
t2	Mirror Ball	Canada	2012	Jun	CDW45934
t3	Ether	Canada	1996	Feb	CDW46012
t4	Insomniac	Canada	1995	Oct	CDW46046
t5	Summerteeth	Canada	1999	Mar	CDW47282
t6	Sonic Jihad	Canada	2000	Jul	CDW47383
t7	Title of...	Canada	1999	Jul	CDW47388
t8	Reptile	Canada	2001	Mar	CDW47966
t9	Always...	Canada	2002	Feb	CDW48016

Figure 4.4: Cleaning Demo II

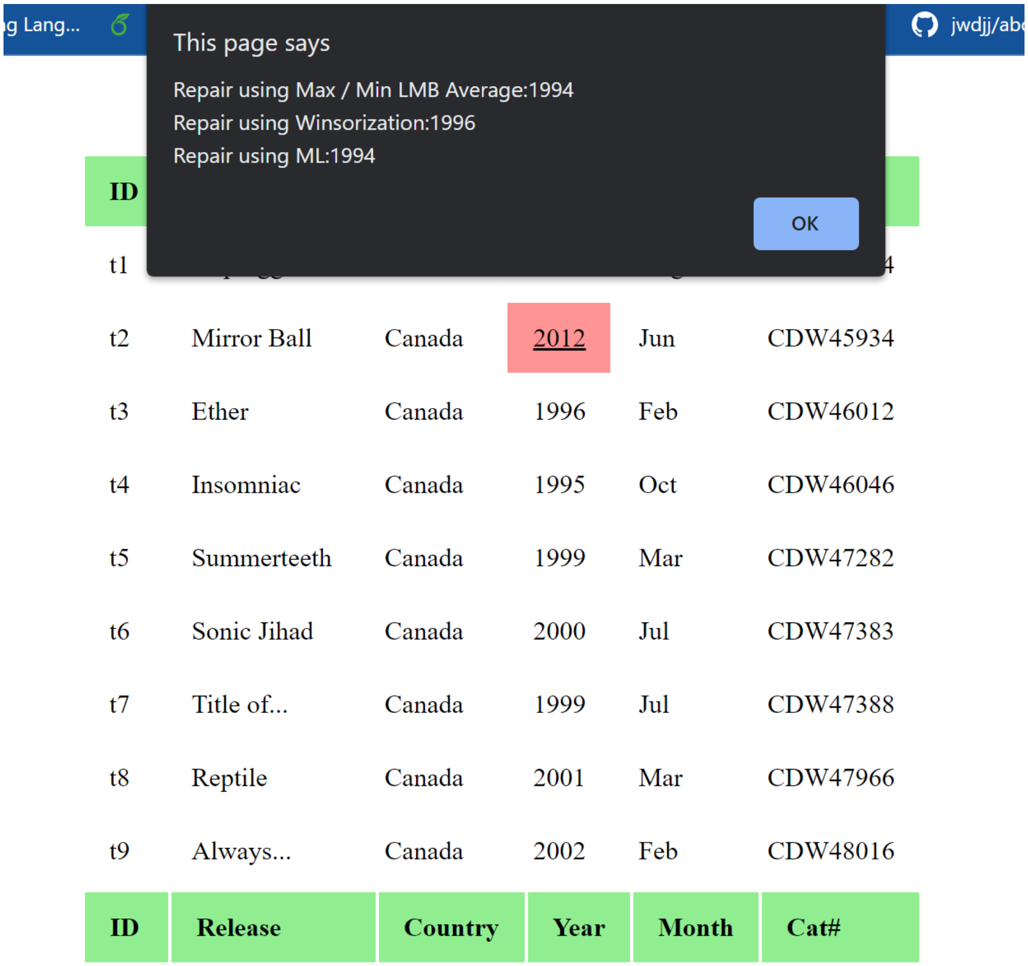


Figure 4.5: Cleaning Demo III

t9	Always...	Canada	2002	Feb	CDW48016
ID	**Release**	**Country**	**Year**	**Month**	**Cat#**
t10	Take A Picture	US	2000	Nov	9 16889-4
t11	One Week	US	1998	Sep	9 17174-2
t12	Only If...	US	1997	Nov	9 17266-2
t13	Never...	US	1996	Nov	9 17503-2
t14	We Run ...	US	1994	Dev	9 18069-2

Chapter 5

Contribution

5.0.1 Repair Algorithm

Together with the help of professor, we came up with the two repair strategies. The Min/Max LMB Average and Winsorization, which included the tolerance of the ordered series and provided more robust repair suggestions. Both the techniques are briefly explained in Chapter 3.

5.0.2 Web Implementation

Web system implementation by integrating the Ranking and Repair features based on the original paper including ranking algorithm covered in the paper. The system is currently in a single local machine but all the code and system deployment information can be provided upon request.

Chapter 6

Conclusion

6.1 Conclusion

To conclude my thesis, I contributed on two different repair algorithms, which took tolerance into the account while calculating the suggested repair. I developed the Web Interface for the demonstration of Ranking and Cleaning strategies. The ultimate goal was to rank the series and repair them, which showcased how the data quality can be significantly improved for the ordered series using these developed techniques.

Bibliography

[1] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: Theory, practice & open challenges. *PVLDB*, 5(12):2018–2019, 2012.

[2] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: an efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[3] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. Effective and complete discovery of bidirectional order dependencies via set-based axiomatization. *VLDB Journal*, 24(7):573–591, 2018.

[4] P. Langer and F. Naumann. Efficient order dependency detection. *The VLDB Journal*, 25(2):223–241, 2016.

[5] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava. Effective and complete discovery of order dependencies via set-based axiomatization. *PVLDB*, 10(7):721–732, 2017.

[6] Ling Ling Yan and M. Tamer Özsu. Conflict tolerant queries in aurora. In *COOPIS*, pages 279–, 1999.