Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 05-024

Summarization - Compressing Data into an Informative Representation Report

Varun Chandola and Vipin Kumar

June 08, 2005

Summarization - Compressing Data into an Informative Representation

Varun Chandola
Department of Computer Science
University Of Minnesota
Minneapolis, MN 55455
chandola@cs.umn.edu

Vipin Kumar
Department of Computer Science
University Of Minnesota
Minneapolis, MN 55455
kumar@cs.umn.edu

Abstract

Summarization is an important problem in many domains involving large datasets. Summarization can be essentially viewed as transformation of data into a concise yet meaningful representation which could be used for efficient storage or manual inspection. In this paper, we formulate the problem of summarization of a large dataset of transactions as an optimization problem involving two objective functions - compaction gain *and* information loss. *We propose metrics to characterize the output of any summarization algorithm. We propose data mining techniques to obtain a summary for a given set of transactions while optimizing these two objective functions. We illustrate one application of summarization in the field of network data where we show how our technique can be effectively used to summarize network traffic into a meaningful representation. We first present a direct application of a standard clustering scheme to generate summaries. We then show how this could be significantly improved by using a multi-step approach which involves generating candidate summaries for a dataset using association analysis and then choosing a subset of these candidates as the summary with the desired compaction and information content. We present results of experiments conducted on real and artificial datasets to demonstrate the effectiveness of our techniques.*

1 Introduction

Summarization is a key data mining concept which involves methods for finding a compact description for a dataset. A simple example would be tabulating the mean and standard deviations for all features. Such summarization techniques are often applied for exploratory data analysis, data visualization and automated report generation where the actual data is huge and cannot be handled manually.

The above mentioned techniques reduce the data to a few numbers which are used to characterize the entire data. But they will not work for richer datasets which have multiple attributes (both continous and categorical). Such datasets include network traffic, census data and other forms of transaction data. The size of the data is typically huge. The data exhibits highly frequent patterns as well as low frequent patterns and outliers. These low frequent patterns and outliers are also important and cannot be discarded as noise.

An interesting application for summarization can be found in the domain of network intrusion detection. Network data is a set of records, where each record represents a network communication entity - *flows,packets or sessions*. Each record has different features such as the IPs and ports involved, packets and bytes transferred. Table 1 lists the typical features and their types for network flows. The important characteristic of network data is that it has a mix of categorical and continuous features. Some features might be more important than others. Another characteristic of network data is that there are lot of communications which are outliers but are of equal or more interest to an analyst.

Feature	Type	Possible Values
Source IP	Categorical	2^{32}
Source Port	Categorical	2^{16}
Destination IP	Categorical	2^{32}
Destination Port	Categorical	2^{16}
Protocol	Categorical	≤ 10
Number of Packets	Continuous	1 - ∞
Number of Bytes	Continuous	1 - ∞
TCP Flags	Categorical	≤ 10

Table 1: Different features for network data

The size of network data which a network analyst has to monitor is huge. On a typical day at the University of Minnesota, one million flows are collected for every 10 minutes. Manual monitoring of this data is impossible and motivates the need for obtaining a compact representation which contains information about every individual data item. Look-

ing at such a summary would enable an analyst to do two things - first, detect the frequent behaviors in the network and second, detect any interesting but less frequent patterns (possibly single communications) in the network.

Figure 2 shows how a set of network communications might be summarized to provide a compact representation of the data. The dataset has size 8 while the summary is of size 3. We see that every data item has a representation in the summary. The first individual summary represents 4 items, the second represents 3 items while the last one is actually a data item itself. The first data item has 6 of its 8 features retained in the summary while the last data item has all of its features retained. Thus different data items lose different amount of information in the summary. As stated earlier, different features might have different weights, so the summary should represent these features better than the others. For example an analyst might want to summary such that the IP and destination port information for a connection is retained while not caring about the source port.

We view summarization as a tranformation from a given dataset to a smaller set of high-level patterns (individual summaries) with an objective of retaining the maximum information content. A fundamental requirement is that *every data item should be represented in the summary*. We associate two objective functions with a summary - *compaction gain* and *information loss*. The compaction gain signifies the amount of reduction done in the transformation from the actual data to a summary. The information content is defined by the amount of information retained for the different data items.

A trivial summary for a set of transactions would be itself. The information loss here would be zero but there would be no compaction. Another trivial summary would be the empty set ϵ which would represent all transactions. In this case the gain in compaction would be maximum but the summary would have no information content.

We propose a two-step technique to summarize data which takes into account the loss-gain aspect of the problem. The basic idea behind summarization is to select an appropriate mix of highly frequent patterns, less frequent patterns and individual data items such that every item belonging to the dataset is well-represented in the summary. We show that getting the best possible summary in terms of maximum compaction and least information loss is a combinatorial optimization problem involving exploratory search in exponential space. We then propose two algorithms which heuristically find an approximately good summary with $O(nk)$ complexity for a dataset of size n from a set of candidate frequent patterns of size k.

It is important to note that summarization addresses different issues than clustering. Clustering also in a way transforms data to cluster centroids but is more focussed at detecting frequent modes of behavior in the data. Its biggest drawback lies in the fact that it does not allow the outliers or less frequent patterns to have a meaningful representation.

The rest of the paper is organized as follows. We present the related work in section 2. We formulate the problem of summarization in section 3 as a dual-optimization problem and characterize a good summary. We also define metrics to evaluate a summary based on these characteristics. In section 4, we propose a direct application of clustering to obtain a summary for a given dataset. We argue how this approach does not optimize the loss and gain parameters for summarization. In section 5 we propose a two-step framework to obtain a representative summary from a set of candidate patterns which could either be a set of clusters or frequent itemsets from association analysis domain [2]. We present two heuristic algorithms based on this framework in section 5.1 and section **??**. We provide empirical results of our two schemes on 1998 DARPA Off-line Intrusion Detection Evaluation data [9] and artificial network data generated by SKAION Corp [13] in section **??**. We also show how the two-step approach performs better than the single-step clustering based approach. We also illustrate the results of our algorithms on a real life network traffic at the University of Minnesota where the connections are ordered based on their anomalous nature.

2 Related Work

Summarization for transaction type datasets with different kinds of attributes like network data has not been explored much in the research community. As mentioned earlier, clustering [7] can be thought of as a transformation of data to cluster centroids does not directly optimize the loss-gain characteristics of a summary. Afrati et al propose an algorithm in [1] to approximate a collection of frequent itemsets. But their work aims at finding a subset of frequent itemsets which approximately covers the frequent itemsets of a dataset. Thus it does not address the issues of summarization of the actual dataset. Similar works on condensed frequent itemsets [12], closed frequent itemsets [11][4] and top-k frequent itemsets [5] also attempt to compress the frequent itemsets while not focussing on individual data items.

Text summarization[10] is a widely-researched topic in the research community. Text summarization has been so far dominated by statistical techniques [3] which are suitable only for text data and cannot be extended to richer trdatasets which have different kinds of attributes. In [6], the authors aggregate customer reviews in a simplistic way focussing on condensing each review by itself and not focussing on the overall compaction gain and information loss for the summary.

src IP	sPort	dst IP	dPort	protocol	flags	packets	bytes
12.190.84.122	32178	100.10.20.4	80	tcp	—APRS-	9	504
88.34.224.2	51989	100.10.20.4	80	tcp	—APRS-	15	2543
12.190.19.23	2234	100.10.20.4	80	tcp	—APRS-	13	2298
98.198.66.23	27643	100.10.20.4	80	tcp	—APRS-	18	18702
192.168.22.4	5002	100.10.20.3	21	tcp	—A-RSF	6	310
192.168.22.4	5001	100.10.20.3	21	tcp	—A-RS-	44	2898
67.118.25.23	44532	100.10.20.3	21	tcp	—A-RSF	2	130
192.168.22.4	2765	100.10.20.4	113	tcp	—APRSF	232	1223504

src IP	sPort	dst IP	dPort	protocol	flags	packets	bytes
..*.*	****	100.10.20.4	80	tcp	—APRS-	[9,18)	[504,18702)
..*.*	****	100.10.20.3	21	tcp	****	[2,44)	[130,2898)
192.168.22.4	2765	100.10.20.4	113	tcp	—APRSF	232	1223504

Table 2: A synthetic dataset of network flows(*left*) and one possible summary for the dataset(*right*). Note that each transaction is represented in some form in the summary

3 Characterizing a Summary

Summarization can be viewed as compressing a given set of transactions to a smaller set of patterns while retaining the maximum possible information. A good summary is the one which is of small size but still retains enough information about the data as a whole and also for each transaction. This is important for network data because a network analyst does not want to lose some communications completely or to a great degree because they did not contain a frequent pattern.

Formally, summarization can defined as following:

Input A set of n features,F and an associated weight vector,w such that, each $w_i \in W$ represents the weight of the feature $f_i \in F$ and a set of M transactions, T, such that each $T_i \in T$ is a set of n features.

DEFINITION 1.(**Candidate Set**) *A candidate set, C is a set of patterns* $\{C_1, C_2, ...C_k\}$ *such that for each* $C_i \in C$, $C_i \subset F$

DEFINITION 2.(**Summary**) *A summary S, is a set* $\{S_1, S_2, ...S_m\}$ *such that for each* $T_i \in T$, *there exists at least one* $S_j \in S$, *such that* $S_j \in C$ *or* $S_j \in T$ *and* $S_j \subseteq T_i$. Each S_j is called an individual summary.

DEFINITION 3.(**Summarization**) *Summarization is a transformation of a given set of transactions,* T *to a summary,* S

For a summary S for a set of transactions T, we define following terms -

DEFINITION 4.(**Compaction Gain for a Summary**) Compaction Gain = $\frac{M}{m}$

DEFINITION 5.(**Information Loss for a transaction**) For a given transaction $T_i \in T$ and an individual summary $S_j \in S$, $loss_{ij}$ = $\sum_{q=1}^{n} w_q * b_q$
where, $b_q = 0$ if $T_{iq} \notin S_j$ and 1 otherwise

DEFINITION 6.(**Best Summary for a transaction**) For a given transaction $T_i \in T$, a best individual summary $S_j \in S$ is the one for which $loss_{ij}$ is minimum.

DEFINITION 7.**Information Loss for a Summary** For a summary S of a set of transactions T, Information Loss = $\sum_{i=1}^{M} loss_{ibest}$, where $S_{best} \in S$ is the best individual summary for T_i.

A summary is characterized by two factors - *compaction gain* and *information loss*. Information Loss characterizes the loss of information due to summarization.

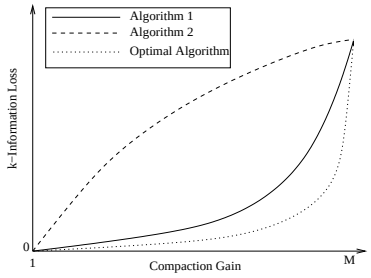

Figure 1: ICC Curve for summarization algorithms

A trivial summary for a set of transactions would be itself. The information loss here would be zero but there would be no compaction. Another trivial summary would be the empty set ϵ which would represent all transactions. In this case the gain in compaction would be maximum but the summary would have no information content.

It is to be noted that the above two characteristics follow a pareto-optimality curve as shown in Figure 1 such that increasing the compaction results in increase of information loss. We denote this curve as ICC (*Information-loss Compression-gain Characteristic*) curve.

The ICC curve is a good indicator of the performance of a summarization algorithm. The beginning and the end of the curve are fixed by the two trivial solutions discussed earlier. For any summarization algorithm the area under its ICC curve should be minimum. It can be observed that getting an optimal curve as shown in Figure 1 involves searching for a solution in exponential space and hence not feasible. But a good algorithm should be close enough to the optimal curve like 1 and not like 2 in the figure shown. As the ICC curve indicates, there is no global maxima for this dual-optimization problem since it involves two orthogonal objective functions. So any summarization algorithm needs to fix one characteristic and find a summary which has the optimal value for the other function. So a typical objective would be - *for a given level of compaction find a summary with the lowest possible information loss.*

Single-step Algorithm
Input: T: a transaction data set.
W: a set of weights for each feature.
m: size of final summary
Output S: the final summary.

Variables: $\bar{C}$: candidate set.

Method:
1. Initialize $S = \{\}$
2. Run clustering T to obtain a set of m clusters, $\bar{C}$.
3. **for** each $\bar{C}_i \in \bar{C}$
4. $S_i = \bigcap_{j=1}^{m} C_{ij}$
5. **endfor**
6. **End**

Figure 2: The Single-step Algorithm

4 A Simple Application of Clustering for Summarization

In this section, we propose a direct application of clustering to obtain a summary for a given set of transactions. Typical output of a clustering algorithm are the centroids of the clusters which can be possibly considered as representative of the members of the cluster. But using cluster centroid as a representative summary is meaningless if we have categorical attributes. We need to have a representation of a cluster that can be used as a summary as defined in the earlier section. One such representation is to take all the attributes which have the same value for every cluster member. In all the proposed algorithms we handle continuous attributes by using *equal-width binning* to convert them into a fixed number of categorical attributes.

Figure 2 gives the single-step algorithm using clusters. Thus, if $\bar{C}$ is a set of clusters obtained from a set of transactions T from clustering, then each cluster produces an individual summary which is essentially the features which are present in all transactions covered by that cluster. The number of clusters here fix the compaction gain for the summary. The weights W are used to calculate the distance between two data transactions.

In the related works section, we have discussed several other works that aim at finding the frequent patterns in the data. These can be similarily adapted to obtain summaries but all of these approaches suffer from the drawback that they are not specifically designed to optimize the balance between compaction gain and information loss but rather try to group similar objects together. One specific drawback is that these approaches do not know how to handle outliers. They are either ignored or forced to belong to a group which does not represent it adequately.

5 A Two-step Approach to Summarization

In this section we propose a two-step methodology to address the problem of summarization. The basic idea is to start with a set of candidate summaries, C (see definition 1). Each of these candidates are a group of transactions which are of varying sizes and depending on their size incur different information loss. Note that C also contains the individual transactions themselves to ensure that outliers can be represented as such. In this context, summarization problem can be viewed as selecting a subset of C of a particular size such that it incurs minimum information loss as defined in the previous section.

We first show that selecting the optimal subset from C is an exponential problem. This is true because we need to search the $2^{|C|}$ space for the solution. In the case of network data, the number of values which a categorical attribute can take is 2^{32}(see table 1). So C is typically very huge and hence obtaining the optimal summary is not feasible.

An important observation is that the global quality of a summary would depend on what C is used for selection. A trivial candidate set would be the set of transactions themselves in which case our approach would do no better than removing the duplicates. The exhaustive collection of candidates would be a set of all frequent itemsets (from association analysis domain) obtained from the data by keeping a support threshold of 2. Another possible collection of candidates is a set of clusters obtained from a clustering algorithm. Note that the candidates need not be non-overlapping. Through our results we show that using an exhaustive set of frequent itemsets or a set of clusters of varying sizes perform equally well for network data.

In the next two subsections we propose two algorithms which aim at obtaining an approximately good subset of C for a given set of transactions T with $O(n^2)$ complexity. We first define the score of a candidate summary as,

DEFINITION 8. (**Information loss for a Candidate Summary**) For a given candidate summary $C_i \in C$,the set of features, F and the associated weight vector w, the loss associated with C_i is given by,
$loss_i = \sum_{j=1} w_j * b_j$, where b_j = 1 if $f_j \in C_i$ and 0 otherwise.

It is obvious that if a candidate summary is a single transaction, it will have a 0 information loss. Similarily, a candidate summary which covers everything - ϵ has the maximum information loss. Another thing to note is that if we add more transactions to a candidate (thereby increasing its size), the information loss will remain the same or increase.

DEFINITION 9. (**Score of a Candidate Summary**) For a given candidate summary $C_i \in C$ its score is given by, $score_i = size(C_i) * (1 - k_s * loss_i)$. Score of a single transaction would be 1. $size(C_i)$ refers to the size of the candidate summary.

The parameter k_s is a user-defined trade-off parameter which allows a summary to have higher compaction gain or lower information loss. We have discussed before that these two entities act orthogonally to each other. A low value of k_s favors very specific candidate summaries which have low information loss. Similarily, a higher value for k_s favors general candidate summaries which have lower information loss.

The role of k_s can also be viewed from the ICC curve perspective. Choosing a very low value of k_s would choose candidate summaries with minimal information loss leading to very less (potentially 0) compaction. By increasing the value of k_s we control how far should the ICC curve move in one step.

5.1 TDS - A Top-down Summarization Algorithm

In this section we propose an top-down algorithm - *TDS* to generate a summary for a given set of transactions. **TDS** algorithm (see figure 3) also has a refinement module (see figure 4) which further improves the quality of the summary by minimizing overlaps among individual summaries.

In this algorithm, each transaction selects a candidate as its best representative summary based on the score of the summary as given in definition 9. Each transaction itself is also considered as a candidate. The rationale behind this algorithm is that all transactions which are similar will select the same candidate as their representative. The transactions which are outliers will select themselves as their representative.

The inputs to this algorithm are the dataset of transactions, T, the set of candidates, C and the tradeoff parameter, k_s. The value of k_s determines the desired compaction. If the value of k_s is 0, the score of candidates which cover one transaction (and hence lose no information) is the maximum. Thus every transaction would select itself as its representative in the summary resulting in no compaction gain and no information loss. As k_s increases, the score of the candidates with larger size increases and they will be selected over the individual transactions.

This "greedy" selection of candidates might result in situation as described in the following example. Consider a summary S as the output of **TDS**. Let a transaction T_1 chooses a candidate C_1 as the best representative for it in S. Let us suppose that all other transactions covered by C_1 choose some other candidates as their best representative. Then we can replace C_1 in S by T_1 to obtain a new summary, S'. Now size of S and S' is same hence they have the same compaction gain. The information loss for T_1 in S will be higher because T_1 is covered by C_1 in S while it is covered by itself in S'. Hence the information loss for S will be higher than S'. Thus S' is not an optimal global summary. This motivates the **TDS**-*ref* algorithm which is a refining module for the main algorithm. This algorithm - **TDS-***ref* aims at minimizing the overlap between the individual summaries.

TDS Algorithm
Input: T: a transaction data set.
W: a set of weights for the features.
k_s: value for the trade-off parameter.
k_{iter}: number of iterations for refinement.
C: candidate summaries
Output S: the final summary.
Method:
1. Score each candidate in C
2. **For** each $T_i \in T$
3. Find $C_j \in C$, s.t. forall $C_j \subseteq T_i$, C_j has max score
4. **If** (score of $C_j >$ score of T_i) **and** $C_j \ni S$
5. Insert C_j in S
6. **Endif**
7. **Endfor**
8. S = TDS-ref(S,k_{iter},C)
9. **End**

Figure 3: The **TDS** Algorithm

The refining algorithm **TDS-***ref* takes a summary as an input. For each of the individual summaries belonging to the summary, it counts the number of transactions that are best represented by that individual summary. It then counts the number of transactions that consider it as their best representative. The size of the corresponding candidate is set to this number. In the next iteration the candidates are scored again using new sizes.

The logic behind **TDS-***ref* can be explained using an example. Consider two candidates C_1, C_2 which belong to the summary, S. Let only one transaction covered by C_1 consider it as its best match. Let all transactions covered by C_2 consider it as their best match. After one iteration of **TDS-***ref*, the size of C_1 will be 1 while size of C_2 will be unchanged. This would reduce the score of C_1 and hence will have lesser chance of being picked by any transaction.

Thus this refinement helps eliminate candidates which cover a lot of transactions but only a few transactions consider it as their representative. This solves the problem discussed in the previous example.

5.2 BUS - A Bottom-up Summarization Algorithm

In this section we propose an increment bottom-up algorithm - **BUS** (see figure 5). While **TDS** first obtains a global summary and then iteratively refines it, **BUS** builds the solution incrementally.

TDS-ref Algorithm
Input: S: an initial summary.
k_{iter}: number of iterations for refinement.
C: candidate summaries

Output S_{ref}: the refined summary.

Variables : S_c: the current summary.

Method:
1. Initialize $S_c = S$
2. **For** count:1 to k_{iter}
3. **For** each $S_i \in S_c$
4. **If** $S_i \in C$
5. Count $S_j \in S$ that consider S_i as best summary
6. Update count as the new size of S_i in C
7. **Endif**
8. **Endfor**
9. Score each candidate in C using updated sizes
10. $S_c = \{\}$
11. **For** each $S_i \in S$
12. Find $C_j \in C$ s.t. for all $C_j \subseteq S_i$, C_j has max score
13. **If** (score of $C_j >$ score of S_i) **and** $C_j \ni S_c$
14. Insert C_j in S_c
15. **Endif**
16. **Endfor**
17. **Endfor**
18. $S_{ref} = S_c$
19. **End**

Figure 4: The **TDS-*ref*** Algorithm

BUS Algorithm
Input: T: a transaction data set.
W: a set of weights for the features.
k_s^{init}: initial value for the trade-off parameter.
δ_k: increment value for k_s.
$iter$: number of iterations to run.
C: candidate summaries

Output S: the final summary.

Variables : S_c: the current summary.
k_s: current value for the trade-off parameter.
Method:
1. Initialize S,$S_c = T$ and $k_s = k_s^{init}$
2. **For** $count$:1 to $iter$
3. $S_c = S$
4. Score each candidate in C
5. Select C_{max} from C with maximum score
6. **If** $C_{max} \notin S$
7. Add C_{max} to S
8. **For** each $S_i \in S$
9. **If** $C_{max} \subset S_i$
10. Remove S_i from S
11. **Endif**
12. **Endfor**
13. **For** each $C_j \in C$
14. Revise size of C_j based on S
15. **Endfor**
16. **Endif**
17. **If** $|S_c| == |S|$
18. $k_s = k_s + \delta_{iter}$
19. **Endif**
20. **Endfor**
21. **End**

Figure 5: The **BUS** Algorithm

The inputs to this algorithm are the dataset of transactions, T, the set of candidates, C, the initial value of the tradeoff parameter, k_s^{init} and the increment, δ_k for the tradeoff parameter. Another parameter $iter$ specifies the number of iterations for which the algorithm is run. This determines the compaction for the final summary.

The initial summary is the set of transactions itself. At every step the algorithm selects a best candidate from the candidate summaries based on the scoring function and replaces all existing individual summaries contained in this candidate with this candidate. The initial value k_s^{init} is typically chosen very low, hence only those candidates are added which have very low information loss. As the value of k_s is incremented, candidates with larger sizes are favored to be added to the summary. This results in more compaction.

Consider a transaction in T which is an outlier. Thus it will be covered only by very general patterns (candidates). In the initial iterations we select candidates with low information loss hence the outliers will be preserved in the summary.

6 Experimental Evaluation And Results

In this section we present the performance of our proposed algorithms on network data. We compare the performance of **TDS** and **BUS** with the single-step clustering based approach to show that they perform better in terms of achieving lower information loss for a given degree of compaction. We also illustrate the summaries obtained for different algorithms to give a qualitative comparison between

them. The algorithms were implemented in GNU-C++ and were run on Linux platform on *intel*-i686 machine.

6.1 Input Data

We ran our experiments on two different artificial datasets generated for evaluation of intrusion detection systems generated by DARPA [9] and SKAION corporation [13]. These datasets have a mixture of normal traffic (which fall into frequent patterns) and attack traffic (which are either outliers or less frequent patterns).

6.2 Comparison of ICC Curves between single-step technique, TDS and BUS

We ran the single-step clustering based algorithm by first generating clusters of different sizes using a hierarchical clustering software *CLUTO* [8]. We weighted the features when finding the similarity between different transactions to match the weighting scheme used for evaluating information loss. We then summarized the clusters as explained in section 4. For **TDS** and **BUS**, we present the results from using clusters and frequent itemsets as the candidates. For clusters, we used *CLUTO* to generate clusters of different sizes from 2 to half the size of the data. We used frequent itemsets generated by *apriori* algorithm using a support threshold of 2. Figures 7 and 6 show the results for each of

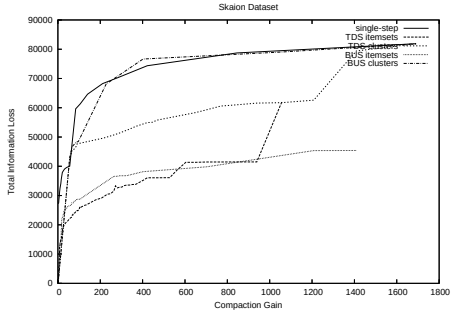

Figure 6: Results of different algorithms on SKAION dataset - *size = 8459 flows*

the five approaches on the DARPA data and the SKAION data in the form of their ICC curves. For the DARPA data we selected a subset of the data corresponing to only *udp* traffic. The size of the SKAION dataset was 8459 flows while size of DARPA dataset was 2677 flows. The **TDS** algorithm was executed with the value for k_{iter} = 20. The value of k_s was increased from 0 onwards to achieve higher compaction. The **BUS** algorithm was executed with initial value of k_s = 0 and the increment, δ_k = 0.1 . We can observe that the **BUS** algorithm using frequent itemsets gives

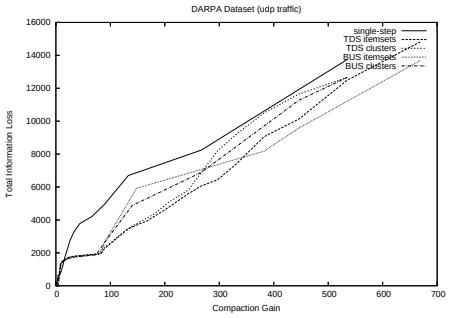

Figure 7: Results of different algorithms on the DARPA dataset (only *udp* traffic)- *size = 2677 flows*

the best results in terms of information loss for a given compaction gain. The **TDS** algorithm performs worse than **BUS** but still outperforms the single-step approach. Both **TDS** and **BUS** perform as well as the single-step approach when using clusters. This is because clusters provide a limited candidate set and do not cover all frequent patterns in the dataset of varying sizes.

7 Conclusion

Summarization is a very useful concept in areas dealing with large amount of data. Network Intrusion Detection is one such field where summarization can be applied. A variant of our proposed **TDS**-algorithm at the University of Minnesota as a part of MINDS (Minnesota Intrusion Detection System) where it enables security administrator of the University to visualize the anomalous traffic in a concise manner. The intrusion detection data is typically divided into normal traffic and anomalous traffic. The analysts are more interested in the anomalous part. A logical enhancement to summarization would be to make use of the normal traffic to generate discrimintative summaries for the anomalous traffic such that these summaries represent only the anomalous behavior.

References

[1] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 12–19, New York, NY, USA, 2004. ACM Press.

[2] R. Agrawal, T. Imieliski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216. ACM Press, 1993.

[3] K. Ahmad, B. Vrusias, and P. C. F. de Oliveira. Summary evaluation and text categorization. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 443–444, New York, NY, USA, 2003. ACM Press.

[4] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 74–85, London, UK, 2002. Springer-Verlag.

[5] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top.k frequent closed patterns without minimum support. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 211, Washington, DC, USA, 2002. IEEE Computer Society.

[6] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM Press.

[7] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[8] G. Karypis. Cluto 2.1.1 software for clustering high-dimensional datasets.

[9] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Comput. Networks*, 34(4):579–595, 2000.

[10] I. Mani. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, MA, USA, 1999.

[11] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT '99: Proceeding of the 7th International Conference on Database Theory*, pages 398–416, London, UK, 1999. Springer-Verlag.

[12] J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed frequent pattern bases. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 378, Washington, DC, USA, 2002. IEEE Computer Society.

[13] SKAIONCorporation. Skaion intrusion detection system evaluation data.