Abstract: Feature selection is an important for high dimensional dataset. The best subset contains the least number of dimensions that highly contributes to the accuracy and so the remaining unimportant dimensions are ignored. Selecting relevant features from unlabelled data is a challenging task due to the absence of label information by which the feature relevance can be assessed. The unique characteristics of IT log further complicates the challenging problem of unsupervised feature selection, (e.g., part of IT log data is linked, which makes invalid the independent and identically distributed assumption), bringing about new challenges to traditional unsupervised feature selection algorithms. In this paper we compare the performance of Linked Unsupervised feature selection algorithm [1] and feature selection using feature similarity [2]. We perform experiments with IT log dataset to evaluate the effectiveness of the both the frameworks.

Keywords: IT log analysis, high dimensional, unlabelled data, attribute-value.

I. INTRODUCTION

IT solutions and IT departments generate an enormous quantity of logs and trace data. Trends in these databases can be identified using data mining practices, which sort and model the data in order to arrive at a conclusion. The data mining applications present the data in the form of data marts. In IT logs, however, the lack of standard vocabulary has hindered the process of data mining to a certain extent.

This could lead to unnecessary problems, during the process of data mining. The increase in the use of standardized terms will reduce the percentage of errors in the data mining process. The huge data poses new challenges to data mining tasks such as classification and clustering.

The effective approach to handle high-dimensional data is feature selection. According to whether the training data is labelled or unlabelled, feature selection algorithms can be roughly divided into supervised and unsupervised feature selection.

It is time-consuming and costly to obtain labelled data. Given the scale of IT log, we propose to study unsupervised feature selection.

Unsupervised feature selection is particularly difficult due to the absence of class labels for feature relevance assessment.

Most existing feature selection algorithms work with “flat” attribute-value data which is typically assumed to be independent and identically distributed. (i.i.d.).

However, the i.i.d. assumption does not hold for IT log since it is inherently linked.
among data instances, and (2) how to take advantage of these relations for feature selection using unlabelled data. In supervised learning, label information plays the role of constraint. Without labels, other alternative constraints are proposed, such as data variance and separability.

IT industries will not have the manpower or resource to churn through all the information by hand, let alone in real time. Linked unsupervised feature selection algorithm and feature selection based on feature similarity is done to select best feature subset out of the log files for analysis. LUFS-Framework.

By introducing the concept of pseudo-class labels, constraints from both link information and unlabelled attribute value data are ready for unsupervised feature selection. The framework of linked unsupervised feature selection, LUFS, with our solutions to the two challenges (need to take into account of linked data and lack of labels): extracting constraints from both linked and attribute-value data, and then constructing pseudo-class labels through social dimension extraction and spectral analysis.

LUFS framework

Algorithm 1 Correlated Unsupervised Feature Selection

Input: \( \{X, R, \alpha, \beta, \lambda, c, K, k\} \)

Output: \( k \) most relevant features

1. Obtain the social dimension indicator matrix \( H \)
2. Set \( F = H(H^TH)^{-\frac{1}{2}} \)
3. Construct \( S \) through Eq. (11)
4. Set \( L = D - S \)
5. Set \( A = XLX^\top + \alpha X(I_n - FF^\top)X^\top \)
6. Set \( B = XX^\top + \lambda I \)
7. Set \( t = 0 \) and initialize \( D_0 \) as an identity matrix
8. while Not convergent do
9. Set \( C_t = B^{-1}(A + \beta D_t) \)
10. Set \( W_t = \{q_1, \ldots, q_c\} \) when \( q_1, \ldots, q_c \) are the eigenvectors of \( C_t \) corresponding to the first \( c \) smallest eigenvalues
11. Update the diagonal matrix \( D_{t+1} \)
12. Set \( t = t + 1 \)
13. end while
14. Sort each feature according to \( W \) in descending order and select the top \( k \) ranked ones;

In LUFS algorithm, dimension extraction and weighted dimension indicator construction are from line 1 to line 2. The iterative algorithm to optimize Eq. (19) is presented from line 8 to line 13.

Feature selection using feature similarity

The task of feature selection involves two steps, namely, partitioning the original feature set into a number of homogeneous subsets and selecting a representative feature from each such cluster. Partitioning of the features is done based on the k-NN principle using one of the feature similarity measures. In doing so, we first compute the \( k \) nearest features of each feature. Among them the feature having the most compact subset is selected and its \( k \) neighbouring features are discarded. The process is repeated for the remaining features until all of them are either selected or discarded.

II. ALGORITHM

Let the original number of features be \( D \), and the original feature set be \( O = \{F_i | i = 1, \ldots, D\} \). Represent the dissimilarity between features \( F_i \) and \( F_j \) by \( S(F_i, F_j) \). Higher the value of \( S \), the more dissimilar are the features. Let \( k^r \) represent the dissimilarity between feature \( F_i \) and its \( k \)th nearest neighbour feature in \( R \).

Then

Step 1: Choose an initial value of \( k \leq D - 1 \). Initialise the reduced feature subset \( R \) to the original feature set \( O \).

Step 2: For each feature \( F_j \in R \), compute \( k^r \).

Step 3: Find feature \( F_i \) for which \( r^r_k \) is minimum. Retain this feature in \( R \) and discard \( k \) nearest features of \( F_i \).

Step 4: If \( k > \text{cardinality}(R) - 1: \) \( k = \text{cardinality}(R) - 1 \).

Step 5: If \( k = 1: \) Go to step 8.

Step 6: While \( r^r_k > \epsilon \) do:

(a) \( k = k - 1 \)

(b) \( r^r_k = \inf_{i \in R} \{F_i \} \) (for the “kth nearest neighbour” of at least one of the features in \( R \) is less than \( \epsilon \)-dissimilar with the feature)

End While

Step 7: Go to Step 2.

Step 8: Return feature set \( R \) as the reduced feature set.

The algorithm has low computational complexity with respect to both number of features and number of samples of the original data.

III. EXPERIMENTS AND DISCUSSION

In this section, we present experiment details to verify the effectiveness of both the framework, LUFS and feature similarity. After introducing real-world IT log, we first
evaluate the quality of selected features in terms of clustering performance, then study the effects of parameters on performance and finally further verify the constraint extracted from link information by all dimensions. We have collected two sample dataset from IT log files. Some statistics of the datasets are shown in Table 1.

Table 1: Statistics of the Datasets

<table>
<thead>
<tr>
<th></th>
<th>Dataset1</th>
<th>Dataset2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>6,816</td>
<td>5,553</td>
</tr>
<tr>
<td># of Features</td>
<td>10,081</td>
<td>9,765</td>
</tr>
<tr>
<td># of Classes</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td># of Links</td>
<td>20,567</td>
<td>34,441</td>
</tr>
</tbody>
</table>

LUFS selects features in batch mode by simultaneously exploiting linking information residing in the log files. Feature selection based on similarity measure works by evaluating the data set with maximum information compression index.

Following the existing evaluation practice for unsupervised feature selection, we assess LUFS in terms of clustering performance. We vary the numbers of selected features as [200, 300, 400, 500, 600, 700, 800, 900, and 1000]. Each feature selection algorithm is first performed to select features, then K-means clustering is performed based on the selected features.

The quality of features selected by both the algorithms using performance metrics is tabulated. We observe the performance change with the numbers of selected features: it increases, reaches the peak, and then decreases. For example, LUFS achieves its peak values when the number of selected features are 500 and 300 in both the log files respectively. The clustering performance with as few as 200 features is better than that with all features. For instances, LUFS obtains 10.51% and 18.68% relative improvement in terms of accuracy for both the log files, respectively. These results demonstrate that the number of features can be significantly reduced without performance deterioration.

Table 1: Performance in logfile1

<table>
<thead>
<tr>
<th></th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUFS</td>
<td>29.19</td>
<td>29.51</td>
<td>29.90</td>
<td>32.70</td>
<td>30.41</td>
<td>31.17</td>
<td>30.48</td>
<td>31.79</td>
<td>31.17</td>
</tr>
<tr>
<td>Similarity Measure</td>
<td>26.16</td>
<td>25.41</td>
<td>27.76</td>
<td>28.14</td>
<td>29.19</td>
<td>30.04</td>
<td>29.78</td>
<td>21.76</td>
<td>30.09</td>
</tr>
</tbody>
</table>

Table 2: Performance in logfile1

<table>
<thead>
<tr>
<th></th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUFS</td>
<td>0.0951</td>
<td>0.1026</td>
<td>0.1489</td>
<td>0.1601</td>
<td>0.1582</td>
<td>0.1701</td>
<td>0.1614</td>
<td>0.1681</td>
<td>0.1596</td>
</tr>
<tr>
<td>Similarity Measure</td>
<td>0.1011</td>
<td>0.1223</td>
<td>0.1431</td>
<td>0.1566</td>
<td>0.1648</td>
<td>0.1309</td>
<td>0.1406</td>
<td>0.1346</td>
<td>0.1455</td>
</tr>
</tbody>
</table>

Table 1: Performance in logfile2

<table>
<thead>
<tr>
<th></th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUFS</td>
<td>26.24</td>
<td>24.56</td>
<td>25.81</td>
<td>29.65</td>
<td>28.36</td>
<td>29.21</td>
<td>29.41</td>
<td>29.69</td>
<td>29.16</td>
</tr>
<tr>
<td>Similarity Measure</td>
<td>27.21</td>
<td>27.38</td>
<td>24.64</td>
<td>27.22</td>
<td>26.31</td>
<td>29.54</td>
<td>28.78</td>
<td>25.37</td>
<td>29.06</td>
</tr>
</tbody>
</table>

Table 2: Performance in logfile2

<table>
<thead>
<tr>
<th></th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUFS</td>
<td>0.0843</td>
<td>0.1131</td>
<td>0.1245</td>
<td>0.1503</td>
<td>0.1673</td>
<td>0.1553</td>
<td>0.1460</td>
<td>0.1672</td>
<td>0.1358</td>
</tr>
<tr>
<td>Similarity Measure</td>
<td>0.1530</td>
<td>0.1241</td>
<td>0.1320</td>
<td>0.1441</td>
<td>0.1490</td>
<td>0.1176</td>
<td>0.1354</td>
<td>0.1230</td>
<td>0.1128</td>
</tr>
</tbody>
</table>
IV. CONCLUSION

This survey provides a comprehensive overview of two unsupervised feature selection algorithms for IT log analytics. The feature selection of audit data has adopted three main methods; wrapper, filter, and hybrid method. The hybrid approaches have been proposed to improve both filter and wrapper method. However, in some recent applications of feature selection, the dimensionality can be tens or hundreds of thousands. Such high dimensionality causes two major problems for feature selection. One is the so called “curse of dimensionality”. As most existing feature selection algorithms have quadratic or higher time complexity about N, it is difficult to scale up with high dimensionality. Since algorithms in the filter model use evaluation criteria that are less computationally expensive than those of the wrapper model, the filter model is often preferred to the wrapper model in dealing with large dimensionality.

The quality of features selected by both the algorithms using performance metrics is tabulated. We observe the performance change with the numbers of selected features: it increases, reaches the peak, and then decreases. For example, LUFS achieves its peak values when the number of selected features are 500 and 300 in both the log files respectively. The clustering performance with as few as 200 features is better than that with all features. For instances, LUFS obtains 10.51% and 18.68% relative improvement in terms of accuracy for both the log files, respectively. These results demonstrate that the number of features can be significantly reduced without performance deterioration.

REFERENCES