Integrated Intelligent Research (IIR)
International Journal of Data Mining Techniques and Applications
Volume: 06 Issue: 01 June 2017 Page No.17-25
ISSN: 2278-2419

# Silhouette Threshold Based Text Clustering for Log Analysis

Jayadeep J
Data Scientist, Tata Consultancy Services Ltd, Chennai, India
Email: jayadeepj@gmail.com

**Abstract -** Automated log analysis has been a dominant subject area of interest to both industry and academics alike. The heterogeneous nature of system logs, the disparate sources of logs (Infrastructure, Networks, Databases and Applications) and their underlying structure & formats makes the challenge harder. In this paper I present the less frequently used document clustering techniques to dynamically organize real time log events (e.g. Errors, warnings) to specific categories that are pre-built from a corpus of log archives. This kind of syntactic log categorization can be exploited for automatic log monitoring, priority flagging and dynamic solution recommendation systems. I propose practical strategies to cluster and correlate high volume log archives and high velocity real time log events; both in terms of solution quality and computational efficiency. First I compare two traditional partitional document clustering approaches to categorize high dimensional log corpus. In order to select a suitable model for our problem, Entropy, Purity and Silhouette Index are used to evaluate these different learning approaches. Then I propose computationally efficient approaches to generate vector space model for the real time log events. Then to dynamically relate them to the categories from the corpus, I suggest the use of a combination of critical distance measure and least distance approach. In addition, I introduce and evaluate three different critical distance measures to ascertain if the real time event belongs to a totally new category that is unobserved in the corpus.

**Keywords -** Clustering methods, Text mining, System Log Analysis, Spherical K-Means, Silhouette Index

## I. INTRODUCTION

With the growing complexity of IT systems, it has now become ardous to continue the traditional approaches to system monitoring and management. Just like the infrastructure, the logs have also grown in complexity both in terms of scale and heterogeneity. The log files generated by the systems contain wealth of information related to states of the system, component interactions, faults & failures, and trends in their operation. The traditional approach to solve problems in IT systems has been to rely on the expertise of domain specialists to sift through these log files, correlate the log events with similar past occurrences (if any) or investigate them from the root. However, it is often infeasible to do this kind of manual log analysis. Hence it is quite common to see terra bytes of unused logs generated from databases, infrastructure monitors, networks, and of course user applications even in mid-sized enterprises. This is where machine learning and text analytics techniques can become handy. However automated log file analysis varies considerably from scenarios where document clustering techniques is typically used. First of all system log data is high dimensional and sparse. Secondly the log events are often semantically ineligible and considerably disparate in nature. A log event could be a process failure from the operating system, a standard warning message from a user application, a network failure message from an infrastructure component or even a simple disk usage error. Third the log event statements are usually shorter though varied in format. Finally traditional document clustering approaches rarely consider the high velocity streaming nature of real time logs. In this paper, I try to experimentally evaluate and sometimes adapt the classical document clustering techniques to log event analysis. However, in this paper I do not perform any semantic understanding or determination of grammatical relationships between words of the log events in order to find out the meaning of the log data.

The remainder of this paper is organized as described as below. Section 2 describes the problem of real time log categorization in detail and its associated challenges. Section 3 presents the vector space model. Section 4 gives an overview of each of the two clustering techniques used, similarity measures and cluster evaluation techniques along with their underlying mathematical descriptions. Section 5 describes the datasets used in the experiments. Section 6 introduces the experimental approaches and implementation to categorize the log archive corpus. Similarly Section 7 introduces the experimental approaches and implementation to dynamically cluster real time logs through computationally efficient means. In section 7 I also introduce the concept of Critical Distance measures. Both section 6&7 also goes over the results of each technique, followed by a comparison of the results. A brief conclusion is presented in Section 8.

## II. OVERVIEW OF LOG CATEGORIZATION

### A. *Real Time Logs Vs Log Archive (Corpus)*

Before describing the problem of log categorization, it is essential to distinguish real time log events from an archive of log events. As mentioned above, a log event in general is any report of an abnormality detected in the system .Typical log events could be a process failure from the operating system, a standard warning message from a user application, a network failure message from an infrastructure component or a simple disk overflow error message Modern IT systems spawn & feed log events continuously to monitoring systems. These real time log events are monitored by IT monitoring staff to identify anomalies that require alerts and investigation. By and large there are 3 modes of actions a support staff takes on an incoming real time log event

Table I: Comparison of Real Time Logs vs. Log Corpus

| Attribute/Log Type | Real Time Log | Log Corpus |
|---|---|---|
| Volume | Low | High |
| Typical Size | 2 or 3 Sentences | Terabytes |
| Incoming Velocity | High | Low |
| Analysis Process | Dynamic | Batch |
| Performance Need | High | Low |

1) No Response - Skip the log event if he knows from experience that the event is relatively insignificant
2) Immediate Response - If he knows from experience that the event is significant. He may correlate it with occurrences in the past, correlate it with prior similar instances (or support tickets) and respond appropriately.
3) Investigative response - If the event is of a new type that was unobserved earlier, the abnormality requires investigation by specialist before response. Usually a new support ticket is raised.

In all the above cases, the log files containing the log events get archived continuously. I refer to this unfiltered historical log archive as the corpus. Typically the corpus runs into terabytes. Analysis performed on the log archives (if any) is done through batch process usually involving Big data tools. However the analysis and action on the real time log events has to be dynamic and immediate.

## *B. Real Time Log Categorization Problem*

Real time log categorization aims to automate the mode of response to a given real time log event. The decision whether a real time log event has similar occurrence in the past or if it is a new category has to be formed automatically. And if similar event is observed in the past, its closest category in the corpus needs to be identified, so that it can be correlated with a past response/ticket.

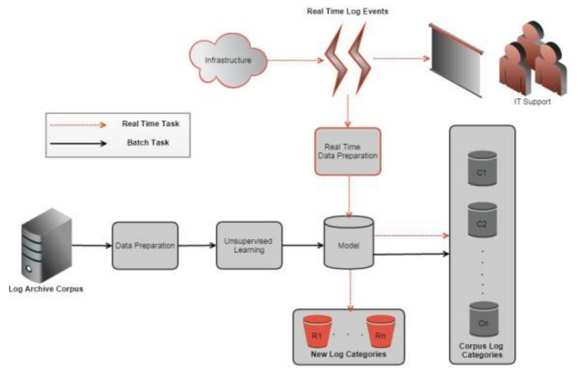


*Image 1: Industrial Framework for Real Time Log Categorization*

This problem involves 2 distinctive set of tasks.

Batch Task: This is to group the log corpus into clusters. I use unsupervised learning to split the entire log corpus into dynamic categories. Traditional document clustering techniques or its adaptations are useful here.
Real Time Task: This second task is to associate each streaming real time event to categories formed in the previous task or identify it as a new Category. This step has to be dynamic and hence should be computationally efficient with superior performance.

## *C. Challenges to Log Categorization*

### *1) Disparate formats*

The log event messages have wide variety of formats both within and outside a category. The error message from Microsoft SQL Server will obviously differ from a web-service error generated out of Oracle SOAP API libraries. However even within a specific category, the error messages differ considerably (in server names, timestamps, and unique ids e.t.c). As described in [10], consider the sample Log Events in Table 2, though there are 6 log events, there are only 2 types of events. Importantly, the events within a category do not match in syntax. Within the category 1, server ids, connection protocols, ports, class names from the libraries all differ. Within the category 2, alphanumeric user IDs, statement ids, account ids differ. However, for an experienced IT support staff, it is easy to ascertain that the 6 events belong to just 2 categories, i.e. Category 1 which deals with an Unexpected disconnection in network most likely due to server restarts and Category 2 that deals with wrong input of the mentioned organization codes and account identifiers. It is also very likely that the events within the same category require similar response. Generally logs are characterized by very few types and high volume of events within each type.

Explicit Programming is not feasible
The categories/templates of log events are rarely known in advance. Using explicit programming to identify these patterns is not feasible. A large-scale system is usually composed of various parts, and those parts may be developed with different programming languages. Despite of the availability of BSD Syslog Protocol and other standards, the contents of different components' logs can be greatly different. Within a component and category itself , explicit programming (e.g using regex pattern matching) cannot be done simply because there may be thousands of categories and different patterns of log events in the system.

Table 2: Sample Log events

| Category | Log Event |
|---|---|
| 1 | [com.tvcs.OvcsSubscriber.bcilAD] Unexpected disconnect of connection bcilConnection1[tcp://sydbcil4test1:34435]State: STREAMING -> CLOSED |
| 1 | [com. tvcs.OvcsSubscriber.options] Unexpected disconnect of connection optConnection1[tcp://sydbcil5dev1:35322] State: STREAMING -> CLOSED |
| 1 | [com.cle.CleanerEditer.cleanerEdit] Unexpected disconnect of connection clsEditer0[tcp://sydbcilclean1:15634]State: STREAMING -> CLOSED |
| 2 | [com.refdata.a3.impl.AccountCreationBuilder] Invalid organization code '9432' for account '4590XDCD6' with statement id 'US#BbCX0llQ9-E1MVPhhwA' |
| 2 | [com.refdata.a3.impl.AccountCreationBuilder] Invalid organization code '6321' for account '93DCC0485' with statement id 'US#CRXUTsRpmXq-5xW1pHA' |
| 2 | [com.refdata.a3.impl.AccountCreationBuilder] Invalid organization code '5846' for account '92BASS0028' with statement id 'US#OMXOgAVSywnVatcLLA' |

### *2) High Volume Log Corpus*

In the Big data world, systems are designed to generate massive amount of logs. The log corpus is usually a high volume and high dimensional but sparse dataset. This renders many of the traditional techniques like agglomerative clustering (E.g. Hierarchical Clustering) or even divisive clustering with an exhaustive search too slow for industrial use.

### *3) High Velocity Real Time Log Events*

High velocity is a quintessential feature of streaming real time events which makes association large computations with individual events extremely arduous. For e.g. re-clustering the real time events along with the log corpus to figure out the new categories is technically feasible but impractical

### D. *Value in Log Categorization*

The advantages of real time log categorization are

- Faster querying on organized logs
- Automatic log monitoring
- Automatic priority flagging of faults
- Ignoring insignificant alerts
- Dynamic solution recommendation
- Self healing of systems
- Reduced support staff costs

## III. VECTOR SPACE MODEL (VSM)

### E. *Log Pre-processing*

Though the format of logs is not fixed, they do have some common characteristics, for example, they may contain timestamps, and composed by upper and lower case letters, digits and special characters. As suggested by Natural Language Processing (NLP) [4], it is helpful to do some text pre-processing to the raw logs before building a TF-IDF matrix

- Remove Timestamps – The timestamp information is removed
- Remove digits – The digits in the log events are removed (or replaced with a special character)
- Lowercase conversion – All log events are converted to lowercase to provide uniformity
- Special Character removal – All special characters, including punctuations are removed and only letters are retained
- Stop Words Removal – All stop words, such as 'what', 'a', 'the', 'any', 'I', etc are removed since they are frequent and carry no information. These commonly used words may not be very useful in improving the clustering quality.

### F. *TF-IDF Representation*

A log event can be represented either in the form of binary data, when I use the presence or absence of a word (or string) in the event to create a binary vector. A more enhanced and commonly used representation would include refined weighting methods based on the frequencies of the individual words in the log event as well as frequencies of words in an entire collection of log corpus; commonly known as the TF-IDF (Term Frequency times Inverse Document Frequency) representation.[7] The TF-IDF matrix for the log events is computed as below. The term frequency of $i^{th}$ word in $j^{th}$ log event $tf_{ij}$ is defined as below

$$tf_{ij} = \begin{cases} 0 & if m_j = 0 \\ \dfrac{f_{ij}}{m_j} if m_j > 0 \end{cases}$$

Where $f_{ij}$ the frequency of $i^{th}$ word in $j^{th}$ log event is normalized by dividing it by $m_j$ , the number of words in $j^{th}$ log event.

$j = 1,2,3 \ldots .n$ where $n$ is the count of log events in the set of log events N

$i = 1,2,3 \ldots m$ where $m$ is the count of the selected vocabulary (words) set M in the collection of log events N.

i.e. $|M| = m$ and $|N| = n$

The inverse document frequency $idf_i$ of the $i^{th}$ word is defined as below

$$idf_i = \begin{cases} 0 & if n_i = 0 \\ \log \dfrac{n}{n_i} if n_i > 0 \end{cases}$$

Where $n_i$ is the number of log events where the $i^{th}$ term appears in the whole set N of log events. Now, I calculate a TF-IDF vector for every log event. The $j^{th}$ log event $l_j$ can be represented as

$$\left(tf_{1j} \times idf_1 , tf_{2j} \times idf_2 , tf_{3j} \times idf_3 , \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots tf_{Mj} \times idf_M\right)$$

So finally, the entire collection of log events can be represented as a $n \times m$ matrix

## IV. DOCUMENT CLUSTERING OVERVIEW

### G. *Similarity Measures*

Here I attempt to cluster and evaluate the log events with 2 measures of similarity; Euclidean measure and the Cosine similarity. As mentioned in [5], the choice of a similarity measure can be crucial to the performance of a clustering procedure. While Euclidean or Mahalanobis distance has its advantage in some circumstances, the Cosine similarity captures the 'directional' characteristics which is intrinsic of the log event TF-IDF vectors. The Euclidean distance between 2 log event vectors $[p_1, p_2, p_3, p_4 \ldots \ldots \ldots p_M]$ and $[q_1, q_2, q_3, q_4 \ldots \ldots \ldots q_M]$ is defined as

$$e.d(p,q) = \sqrt{\sum_{i=1}^{M} (p_i - q_i)^2}$$

The Cosine distance between 2 log event vectors $[p_1, p_2, p_3, p_4 \ldots \ldots \ldots p_M]$ and $[q_1, q_2, q_3, q_4 \ldots \ldots \ldots q_M]$ is defined as

$$c.d(p,q) = (1 - \cos\theta) = 1 - \frac{\sum_{i=1}^{M}(p_i \times q_i)}{\sqrt{\sum_{i=1}^{M}(p_i)^2} \times \sqrt{\sum_{i=1}^{M}(q_i)^2}}$$

The resulting cosine distance is a number from 0 to 1. When the result is 1, the two messages are completely different, and when the result is 0, the two messages are identical

### H. *Similarity Measures*

The clustering techniques can be loosely divided into 2 classes; Hierarchical and Partitional clustering. While hierarchical clustering algorithms repeat the cycle of either merging smaller clusters to larger ones or dividing larger clusters to smaller ones, Partitional clustering algorithms generate various partitions and then evaluate them by some distance/similarity criterion. Hierarchical clustering is traditionally slower in performance for large datasets. In addition, Partitional methods, where clusters are represented by centroids has computational advantages when predicting cluster membership for new data i.e. real time log events. Hence in this paper I consider only Partitional methods

#### 1) *Standard K-Means*

In K-Means based clustering; the centroid, a median point ascertained by some similarity measures from the set of points represents the set itself. K-means Algorithm to obtain K clusters is as follows:

- Choose random K points as the initial centroids.
- Assign every point to the closest centroid.
- Re-compute the centroid of each cluster.
- Repeat steps 2 & 3 till the centroids don't change.

The Standard K-means clustering, attempts iteratively to find clusters in a data set such that a cost function of dissimilarity (or distance) is minimized. The dissimilarity measure is chosen as the Euclidean distance. A log event set N of size n are to be partitioned into k categories, into a set of clusters $C$

$$l_j \epsilon N \;;\; j = 1,2,3 \dots n$$
$$C_i \in C \;;\; i = 1,2,3 \dots k$$

The cost function, based on the Euclidean distance can be defined as

$$R = \sum_{i=1}^{k} R_i = \sum_{i=1}^{k} \sum_{l_j \epsilon c_i} \left\| l_j - \mu_i \right\|^2$$

Where $\mu_i$ is the mean (centroid) of log events in $C_i$category

*2) Spherical K-Means*

Spherical K Means [3] is essentially K Means attempted with Cosine similarity. Just like the Standard K-means clustering, it attempts iteratively to find clusters in a data set such that a cost function of Cosine dissimilarity (or distance) measure is minimized For a log event set N of size n; $l_j \in N \;;\; j = 1,2,3 \dots n$that are to be partitioned into k categories, $C_i \in C \;; i = 1,2,3 \dots k$the cost function is defined as

$$R = \sum_{i=1}^{k} R_i = \sum_{i=1}^{k} \sum_{l_j \epsilon c_i} 1 - \cos\left(l_j , \mu_i\right)$$

Where $\mu_i$ is the mean (centroid) of log events in $C_i$cluster. For high-dimensional data such as text documents, and market baskets, Spherical K Means has been shown to be a superior measure to Standard K means.

*I. Evaluation Measures*

For clustering, basically two types of quality measures are used. First type allows to compare multiple cluster sets without any indicator to the actual class, hence referred to as internal quality measure. The second measure compares the categories created by the clustering techniques to original classes to evaluate the cluster hence referred to as external quality measure. I use 2 external measures Entropy, Purity and one internal measure Silhouette Index to evaluate our clustering

*1) Entropy*

Entropy[9] is an external quality measure. Initially the class distribution is calculated for every cluster, i.e., I compute $P_{ij}$, the probability that a member of cluster-j corresponds to class-i. The entropy of every cluster $E_j$ is then calculated using the formula below:

$$E_j = -\sum_{i=1}^{k} P_{ij} \log\left(P_{ij}\right)$$

Then the sum of individual entropies of clusters weighted by its size is the total entropy

$$E = \sum_{j=1}^{k} \frac{E_j \times n_j}{n}$$

Where $n_j$is the size of the cluster j, k being the number of cluster(s), and $n$ being the number of log events.

*2) Purity*

Purity[9] is another external quality measure. Purity measure is calculated by assigning each cluster to the label which is most frequent in it. Then the number of correctly assignedlog events are counted and is divided by $n$ the total number of log events.

$$P = \frac{1}{n} \sum_{i=1}^{k} max_j \left| c_k \cap z_j \right|$$

Where

$C = \{c_1, c_2, c_3, c_4 \dots\dots\dots c_k\}$is the set of generated clusters.
$Z = \{z_1, z_2, z_3, z_4 \dots\dots\dots z_k\}$is the set of original classes.

*3) Silhoutte Index*

The Silhouette Index is a combined measure of how well each data point lies within its own cluster and how dissimilar each data point is to its closest neighbouring cluster. It is computed as below [6]. A log event set N of size $n$ is partitioned into k categories, into a set of clusters $C$

Each log event $l_j$is such that

$$l_j \epsilon N \;;\; j = 1,2,3 \dots n$$
$$C_i \in C \;;\; i = 1,2,3 \dots k$$

$n_i$is the size of cluster $C_i$ i.e. $n_i = |C_i|$

The average distance $a_j^i$ between the $j^{th}$ log event$l_j$in the cluster $C_i$ and the all other log events in the same cluster is calculated as

$$a_j^i = \frac{1}{n_i - 1} \sum_{z=1, z \neq j}^{n_i} dist\left(l_j^i , l_z^i\right)$$
$$j = 1,2,3 \dots n_i \;;\; l_j^i \epsilon C_i ;\; l_z^i \epsilon C_i$$

The average distance to the neighbouring cluster $b_j^i$ is calculated next. The minimum average distance between the $j^{th}$ log event$l_j$in the cluster $C_i$ and all the log eventsclustered in the other clusters except $C_i$ is calculated as

$$b_j^i = \underset{\substack{r=1,2..k \\ r \neq i}}{min} \left\{ \frac{1}{n_r} \sum_{z=1}^{n_r} dist\left(l_j^i , l_z^r\right) \right\}$$

$$j = 1,2,3 \dots n_i \;;\; l_j^i \epsilon C_i ;\; l_z^r \epsilon C_r$$

Then as described in [10], the silhouette width of the cluster $C_i$ is defined in the following way:

$$SW_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{b_j^i - a_j^i}{max\{b_j^i, a_j^i\}}$$

And the Silhouette Index of the entire clustering is taken by average

$$S.I = \frac{1}{k} \sum_{i=1}^{k} S.W_i$$

## V. EXPERIMENTAL DATASETS

I used 3 separate log data sets pairs (for the corpus and real time data). Each of the dataset was obtained from the IT logs of a single real banking service provider. The log events typically included Overflow alerts from messaging queues, Database adapter connection failures, Application server cluster failover warnings and SOAP web-service failure messages. Each of the real time data sets contain 10 new categories and 10 old categories that exist in the Corpus. I used 2 Corpus - Real time pairs with log event counts 13200-20, 17600-20 & 18700-20. The number of words in the the pairs were 189640-327, 323950-323, 365530-370. All three corpus set had 20 categories and each real data set contained 20 old and 20 new categories. The automatic log data was fetched from the systems/servers. Being from disparate sources there was no unity in the set of original log attributes. However, in general the log data contained (not restricted to) the following original log attributes

- Time stamp of the Log Event
- Source of the log event, such as the class, function, or filename
- Log Category Information. For example, the severity - INFO, WARN, ERROR & DEBUG
- Script/Process names
- Brief Summary of the Log Event
- Detailed Description of Log Event/Error Description/Event message

Out of these log attributes only the last attribute i.e Detailed Description of Log Event/Error Description/Event message was used for this paper

## VI. LOG CORPUS CATEGORIZATION

### J. Methodology – Batch Task

The methodology to cluster log corpus log in batch mode is described below. The steps in batch process may be repeated are frequent intervals as an offline process.

Step 1: Pre-process the corpus log data as described in Section 3.1 to clean and filter the data

Step 2: Build TF-IDF matrix from the corpus dataset i.e. a $T_{n \times m}$ matrix where $m$ is the total count of words in the collection excluding the stop words and $n$ is the number log events in the log corpus

Step 3: Apply the document clustering algorithm (Standard K Means / Spherical K Means ) to the generated TF-IDF matrix. Generate a centroid set $C_{k \times m}$ matrix of dimension $k \times m$ where $k$ is the number of clusters and $m$ is the total count of words in the collection excluding the stop words

For the purpose of the experiment, each of the two algorithms Standard K Means & Spherical K Means are run on each of the 3 corpus datasets (P1-Corpus, P2-Corpus, P3-Corpus) for 10 iterations each. The clustered data was evaluated for Purity, Entropy and Silhouette Index scores.

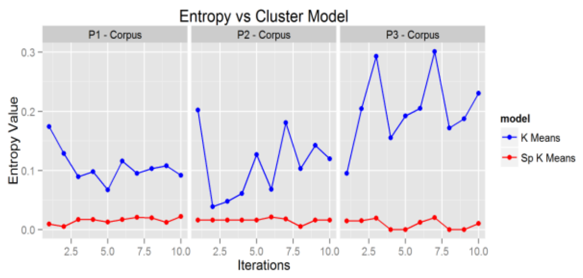


*Figure 1: Entropy Comparison*

### K. Results and Evaluation

Because of the high number of dimensions in the dataset, it is difficult to present visual representation of the clusters. The performance measures are shown in Figure 1, 2 & 3. The Entropy results for each of the 3 datasets (P1- Corpus, P2-Corpus, and P3-Corpus) are shown in figures 1 - smaller is better. The Purity comparison is shown in figures 2 - higher is better. The Silhouette Index comparison is shown in figures 3 - higher is better. Notice that Spherical K Means outperforms Standard K Means in all the three measures for each of the 3 data sets. Hence I can reasonably conclude that Spherical K Means is best suited for corpus log categorization.

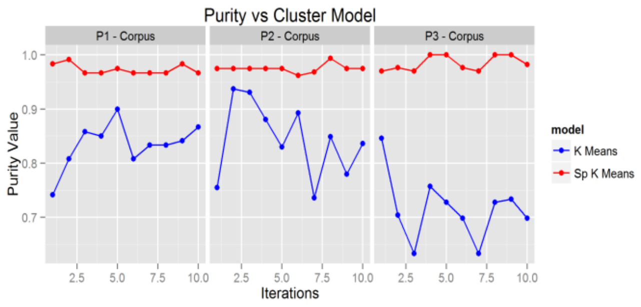


*Figure 2: Purity Comparison*

Another interesting aspect that can be observed from the plots is that the performance of Standard K Means degrades when the number of dimensions in the log event data set increases. Note that the difference in the performance between Standard K Means and Spherical K Means is the highest in P3 Corpus. All other factors (iterations, number of clusters) remaining the same, it is safe to assume that this difference is due to the dimensionality. The performance of Spherical K Means remains fairly consistent

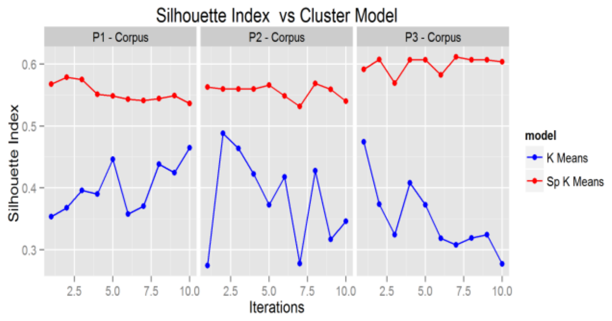


*Figure 3: Silhouette Index Comparison*

## VII. REAL TIME LOG CATEGORIZATION

Here the real rime log event has to be associated to one distinct category from the clustered log corpus or identify the event as a new category unobserved in the log corpus. This is an online process

### L. Methodology – Real TimeTask

Similar to description in [10], the methodology to dynamically categorize real time log events is described below

Step 1: For each incoming real time log event apply the pre-process step as described in Section 3.1 to clean and filter the data

Step 2: Generate a TF-IDF matrix for the real time log event based on the pre-built TF-IDF matrix of the log corpus. If the corpus TF-IDF is a $n \times m$matrix, then the real time log event TF-IDF would be a $1 \times (m + \Delta m)$ matrix where $m$ is the count of words in corpus and $\Delta m$ is the count of unseen words disregarding the oft repeated stop-words in the streaming log event. Explanation provided in Sect:7.2

Step 3: Adapt the centroid generated while clustering the log corpus to suit the TF-IDF vector of the real time log event. If the log corpus centroid set is a $k \times m$ matrix, then the adapted centroid set would be a $k \times (m + \Delta m)$matrix where $k$ is the number of clusters from the corpus, $m$ is the word count in the corpus and $\Delta m$ is the unseen word count disregarding the oft repeated stop-words in the streaming log event. Explanation provided in Sect:7.3

Step 4: The closest median point is then selected (based on the above described distance measures) from the adapted K-centroid collection. However, If distance from the nearest centroid to the real time event is larger than the appropriate Critical Distance Measure (RMWSS, Cluster Radius, Silhouette Threshold), then classify the real time event with an unseen label.

Step 5: If not, the cluster with the nearest centroid is assigned as the label.

For the purpose of the experiment, the above steps are run on each of the 3 real time datasets (P1-Real Time, P2-Real Time, P3-Real Time) against the clusters built from their corresponding corpus datasets (P1-Corpus, P2-Corpus, P3-Corpus) for 10 iterations each. The steps are repeated with Euclidean distance measure on the Standard K Means generated centroids and with Cosine distance on the Spherical K Means generated centroids. To evaluate the different critical distance measures, the accuracy for real time events classified as old or new (similar events observed or unobserved in the original corpus) are calculated. Similarly the accuracy scores are calculated for real time events classified correctly into its true class from original corpus.

## *M. Real Time TF- IDF generation*

For each real time log event, the TF-IDF vector has to be generated dynamically. The following factors should be considered while determining the real time log event TF-IDF matrix. To measure the distance to the centroids, the TF-IDF vector of the real time log event should be comparable to the corresponding matrix built from the corpus. The real time event might contain a subset of the current vocabulary from the corpus or the unseen word collection w.r.t the corpus (in any log events) or a combination of both (most likely). The process to generate the TF-IDF vector of the real time log event should be computationally inexpensive. Although adding the new the real time event to the whole corpus and regenerating the TF-IDF matrix will theoretically yield the same result, this is not computationally realistic for a dynamic real time process.

$N$is the set of all events from corpus.
$N = \{l_1, l_2, l_3, l_4 \dots\dots\dots l_n\}$ ; $l$represents the individual log events
$M$is the set of the selected vocabulary (words) in the set of log events $N$ in the Log Corpus.
$M = \{w_1, w_2, w_3, w_4 \dots\dots\dots w_m\}$ ; $w$represents the individual words$m$ and $n$ are the number of vocabulary and number of log events from corpus respectively.

$$|M| = m and |N| = n$$

$T_{n \times m}$is the TF-IDF matrix built from the corpus dataset and $T_{pq}$ is matrix entry for the $p^{th}$ row and the $q^{th}$ column i.e. the $q^{th}$ word in $p^{th}$ log event
$M^R$is the set of the selected vocabulary from the real time event.

$$M^R = \{w_1^R, w_2^R, w_3^R, w_4^R, \dots\dots\dots w_r^R\}$$

$w_i^R$represents the $i^{th}$ terms from the real time event and $r$ is word count from the real time event

If $T^R_{1 \times (m + \Delta m)}$ is the TF-IDF vector of the real time log event that need to be generated and $\Delta m$ is the count of unseen words disregarding the oft repeated stop-words in the streaming log event. The algorithm to generate TF-IDF vector of the streaming event is described in Algorithm 1.

-------------------------------------------------------------------------
Algorithm 1: TF-IDF for Real Time Events
-------------------------------------------------------------------------

```
Initialize $T^R_{1 \times m}$matrix ; {With same columns as $T_{n \times m}$ }
for each word $w_i^R \in M^R$ ; $i = 1,2,3 \dots r$
    if$w_i^R \notin M$ then
        $M \leftarrow \{M, w_i^R\}$ ; {Add the unseen term to collection}
        $m \leftarrow m + 1$ {up collection size by 1}
        $T^R_{1m} \leftarrow \frac{count(w_i^R)}{r} \times \frac{\log(n+1)}{1}$
    else
        Find $j$ where $w_j = w_i^R$; $w_i^R \in M^R, w_j \in M$
        $c(w_i^R) = rowcount(T_{n \times m}) \mid T_{pj} \neq 0 ; p = 1,2,3 \dots n$
        $T^R_{1j} \leftarrow \frac{count(w_i^R)}{r} \times \frac{\log(n+1)}{c(w_i^R)+1}$
end if
end for
```

-------------------------------------------------------------------------

First the new TF-IDF vector; $T^R_{1 \times m}$ of the real time log event that need to be generated is initialized. Both this vector and the original matrix has the exact same number of fileds, with the vector containing a single record. I loop over every term in the real time event in next step. If the word is not present in the log corpus Vocabulary, you need to add it to $T^R_{1 \times m}$. This is accomplished in step 4 & 5. Once the TF IDF is computed in the next step using the document size to ascertain the IDF now is $(n + 1)$ i.e. the corpus size + one for the real time event. The denominator is 1 because this word is not found in any other log event. I take the valid row count for the word in the original TF-IDF matrix from corpus in step 9. This count is used to calculate the IDF in step 10.

**Computational Advantage**

In this algorithm I iterate only over the $r$ dimensions of the real time log event. Since $r \ll m$ this algorithm is better in terms of computational efficiency when compared to regenerating the whole TF-IDF matrix

## *N. K-Centroid dimension adaptation*

While trying to compare and find the distance between the real time event to each centroid from the set of corpus log clusters, it is essential that the column types of the matrices match. Hence there is a need to adapt the centroid matrix generated while clustering the log corpus to suit the TF-IDF vector of the real time log event in the run time.
Original dimension of Centroid Matrix $= k \times m$
Dimension of TF-IDF of real time event

$$T^R_{1 \times (m + \Delta m)} = 1 \times (m + \Delta m)$$

where$\Delta m$ is the unseen word count in the streaming event that was absent in the log corpus. Dimension of adapted Centroid

International Journal of Data Mining Techniques and Applications
Volume: 06 Issue: 01 June 2017 Page No.17-25
ISSN: 2278-2419

Matrix $= k \times (m + \Delta m)$

The adaptation is done by simply adding $\Delta m$ new columns representing the new words to the original centroid matrix. Each entry for the $k$ rows in the $\Delta m$ columns is initialized to zero.

## *O. Critical Distance Measure*

Once the TF-IDF of the streaming event is generated and the k centroids are adapted to suit them, I took the least dissimilarity approach to find the closest centroid. However, along with finding the nearest category in the log corpus for the streaming event, it has to be identified if the streaming event is a new category, unobserved in the log corpus.

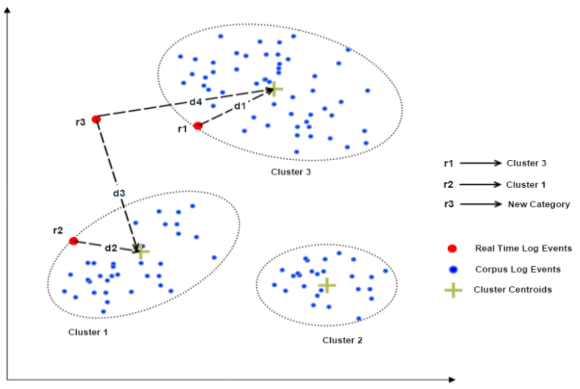


*Figure 1: Real Time Log Event Categorization*

To do this, I propose using a Critical Distance measure. If the distance of real time log event to the closest centroid is greater than the Critical Distance, then it is considered as a new category of log event. As shown in Figure1, real time log events r1 and r2 are assigned to Cluster 3 and Cluster 1 respectively. However the distance of r3 to its nearest cluster, Cluster 1 is greater than the Critical Distance and hence I associate it with a new category.

Table 4 : Clustered Corpus Log Events

| Cluster | Corpus Log Event |
|---|---|
| 1 | [RequestHandler.D.42] Book Request is rejected because of unexpected exception: Invalid customer account: LBM2XT12. |
| 1 | [RequestHandler.D.42] Book Request is rejected because of unexpected exception: Invalid customer account: AVBGV711. |
| 1 | [RequestHandler.D.42] Book Request is rejected because of unexpected exception: Invalid customer account: ATUIF169. |
| 1 | [RequestHandler.D.42] Book Request is rejected because of unexpected exception: Invalid customer account: CRJAY123. |
| 2 | [bcil.TradeClient.java] ProtocolException Unexpected format Soap@17fcvve9 <?xml msg = Ordinal Request for Price Lead Cancel Stream bound/> |
| 2 | [bcil.TradeClient.java] ProtocolException Unexpected format Soap@25515779 <?xml msg = Trade and balance loading work flow run in progress/> |
| 2 | [bcil.TradeClient.java] ProtocolException Unexpected format Soap@gcfd88a1 <?xml msg =T he view selected contain risk from the Fraud System/> |

**Critical Distance chosen should be dynamic**

For Higher accuracy categorization, it is essential that the Critical Distance chosen is dynamic i.e. it should vary according to the distribution of log events in the cluster. This reason is explained in the example below

Table 5: Real time Log Events

| Cluster | Real Time Log Event |
|---|---|
| NEW | [RequestHandler.D.42] Cancel Request is rejected because of unexpected exception: Invalid organization account: CRTPBLA. |
| 2 | [bcil.TradeClient.java] Protocol Exception Unexpected format Soap@gcfd88a1 <?xml msg = Micro trades sent to target successfully by overflow/> |

Consider the real examples in Table 4 which contains the clustered log events from the corpus and the examples in Table 5 that has the incoming real time events. The first real time event in Table 5 looks surprisingly similar to the category of log events in cluster1. On word to word comparison with Category 1, only 3 words differ, out of an average length of the log event 13; i.e. an approximate similarity of 10/13 = 76%. However on closer observation, it is obvious that this real time event is a new category. Note that in this example the variation within the cluster1 is very less ; only 1 word typically varies in the events i.e. an approximate similarity of 12/13 = 92%. This is a very compact cluster. The second real time event in Table 5 varies in approximately 9 words out of an average length of the log event 16 i.e. an approximate similarity of 7/14 = 50%. Even with this poor similarity, this event most likely belongs to category 2. Note that the Cluster 2 is has a high radius , due to the higher variation within its members. I propose three different Critical Distance Measures beyond which the event belongs to a new category and evaluate each of them against the test datasets in Table 3.

## *P. RMWSS*

The Root Mean Within Sum of Squares (RMWSS) is a conservative measure of Critical Distance for each cluster. For a log event set N of size n that are clustered into k categories, into a set of clusters $C$ ,

$$l_j \epsilon N \,;\, j = 1,2,3 \ldots n$$
$$C_i \in C \,;\, i = 1,2,3 \ldots k$$

The $RMWSS$ of Cluster $C_i$ , $RMWSS_i$ can be defined as

$$RMWSS_i = \sqrt{\frac{1}{n_i} \sum_{l_j \epsilon C_i} \left[dist\left(l_j, \mu_i\right)\right]^2}$$

Where $\mu_i$ is the mean (centroid) of log events in $C_i$ category and $n_i$is the number of log events in cluster $C_i$. The $dist\left(l_j, \mu_i\right)$is the Euclidean or Cosine distance between $l_j \& \mu_i$

## *Q. Cluster Radius*

The cluster radius is defined as the distance from the cluster centroid to the farthest point within the cluster. It is the maximum distance among distance of each entry from cluster and its centroid. For a log event set N of size n that are clustered into k categories, into a set of clusters $C$

$$l_j \epsilon N \,;\, j = 1,2,3 \ldots n$$
$$C_i \in C \,;\, i = 1,2,3 \ldots k$$

The Cluster Radius of Cluster $C_i$, $CR_i$ can be defined as

$$CR_i = \max\{dist(l_j, \mu_i)\} \quad ; l_j \epsilon C_i$$

Where $\mu_i$ is the mean (centroid) of log events in $C_i$ category and $n_i$is the number of log events in cluster$C_i$. The $dist(l_j, \mu_i)$is the Euclidean or Cosine distance between $l_j \& \mu_i$

## R. *Silhouette Threshold*

The Silhouette Threshold of a cluster is a combined measure of the silhouette width of the cluster and the Radius of that cluster. As mentioned in [10], the Silhouette Width of an individual cluster is an indication of how perfectly each entity resides within a cluster and how distant is each entity from its nearest neighbouring cluster.

The silhouette width of the cluster $C_i$is defined as below:

$$SW_i = \frac{1}{n_i}\sum_{j=1}^{n_i} \frac{b_j^i - a_j^i}{max\{b_j^i, a_j^i\}}$$

The silhouette width varies from -1 to 1.
The cluster radius of the cluster $C_i$is defined as below:

$$CR_i = \max\{dist(l_j, \mu_i)\} \quad ; l_j \epsilon C_i$$

Then I define Silhouette Threshold as:

$$ST_i = CR_i[1 + SW_i]$$

## S. *Results & Evaluation*

First I analyze the accuracy of categorizing the real time events as old or new category (observed or unobserved category in the original corpus) and compare them for each of the Critical Distance Measures (RMWSS, Cluster Radius and Silhouette Threshold). The experiments are carried out for each of the 3 datasets (P1- Real Time, P2-Real Time, and P3-Real Time) against both Euclidean and Cosine dissimilarity measures. The Accuracy is calculated as

$$Accuracy = \frac{NoofCorrectlyCategorized\ RealTimeEventsasOldorNew}{NoofTotalRealTimeEvents}$$

The Accuracy measures are shown in Figure 4 & 5.

## T. *Critical Threshold Measures Differentiation*

The Cluster radius and RMWSS does not perform well as Critical distance measures. It is mainly because there is no positive/negative limiting factors to these measures. For example, for a cluster with high variation within its members, the probability of wrongly categorizing a streaming event as belonging to that cluster is high. If one considers Radius/RMWSS as the threshold measures, there are no limiting factors in these measures to contain the wrong categorization because of poor clustering performance.

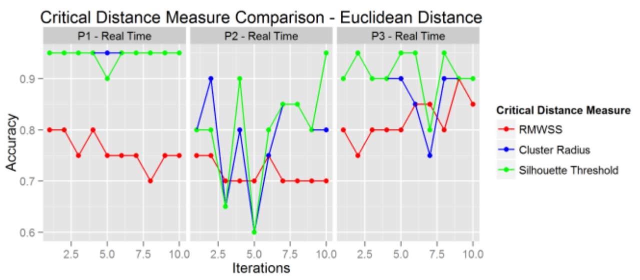


*Figure 4: Accuracy vs. Iterations – Euclidean Distance*

It is evident that the Silhouette Threshold as a Critical Distance Measure outperforms RMWSS and Cluster Radius for each of the datasets for both Cosine and Euclidean dissimilarity. The accuracy levels of Cluster Radius though relatively better than RMWSS, it is not as good as Silhouette Threshold for most iterations.

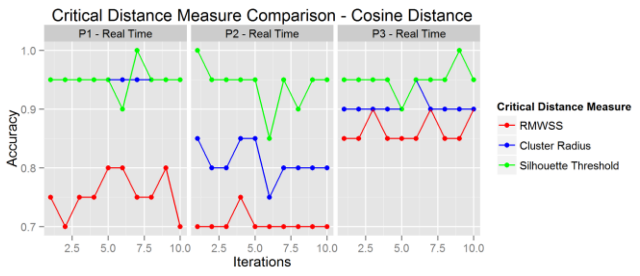


*Figure 5: Accuracy vs. Iterations – Cosine Distance*

Next I also analyze the accuracy of correctly categorizing a real time event to its true category given that similar events are previously observed in the original corpus. The experiments are carried out for each of the 3 datasets (P1- Real Time, P2-Real Time, and P3-Real Time) against both Euclidean and Cosine dissimilarity measures. This comparison is presented in Figure 6. Among the real time events that was present in the original corpus, I find the accuracy as below.

$$Accuracy = \frac{NoofCorrectlyCategorizedRealTime\ Eventstoitsorginalclass}{NoofTotalRealTimeEvents}$$

It can be seen that in case of prediction of the class, there is no significant distinction between Euclidean and Cosine distance measure. Hence once the original corpus is clustered with high accuracy, the distance measure chosen to find the closest centroid/category in the corpus is relatively insignificant.

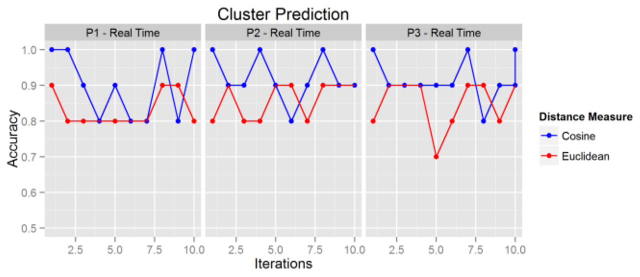


*Figure 6: Real Time Events – Closest Category Prediction*

## U. *Why Silhouette Threshold Performs Better*

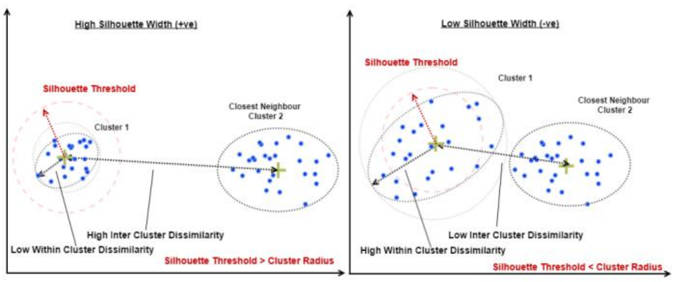

High Silhouette Width (+ve)
Silhouette Threshold
Cluster 1
Closest Neighbour Cluster 2
High Inter Cluster Dissimilarity
Low Within Cluster Dissimilarity
Silhouette Threshold > Cluster Radius
Low Silhouette Width (-ve)
Silhouette Threshold
Cluster 1
Closest Neighbour Cluster 2
Low Inter Cluster Dissimilarity
High Within Cluster Dissimilarity
Silhouette Threshold < Cluster Radius

*Figure 7: Sample Clustering with High vs. Low Silhouette Width*

As mentioned before the Silhouette Width is defined as,.

$$SW_i = \frac{1}{n_i}\sum_{j=1}^{n_i} \frac{b_j^i - a_j^i}{max\{b_j^i, a_j^i\}}$$

$$SW_i = \begin{cases} 1 - \frac{a^i}{b^i}\, if a^i < b^i \\ 0 \qquad if a^i = \ b^i \text{Hence } -1 \le SW_i \le 1 \\ \frac{b^i}{a^i} - 1\, if a^i > b^i \end{cases}$$

For a cluster with high avg inner cluster distance and low within cluster distance will have a silhouette width greater than 0. Conversely, for a cluster with low avg inner cluster distance and high within cluster distance will have silhouette width less than 0. This is shown in Figure 7. Silhouette Threshold is chosen as:

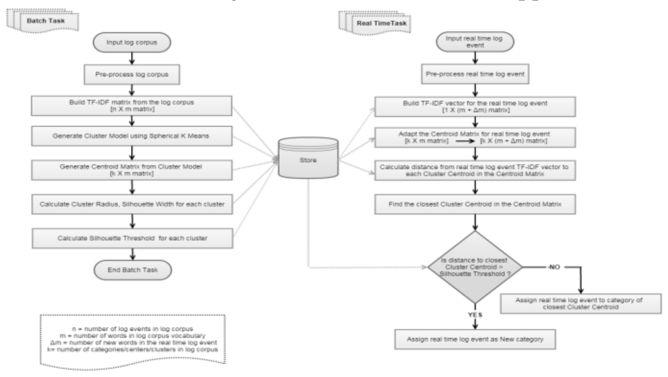
$$ST_i \ = \ CR_i[1 + SW_i]$$

For a cluster with high variation within its members, the probability of wrongly categorizing a streaming event as belonging to that cluster is high. Hence the Critical Distance should be less than the Cluster Radius. So hypothetically, for a really bad cluster, $(a^i \gg b^i)$ Silhouette Index would be -1 and the Silhouette Threshold would be zero indicating that it is better not to associate any Real Time Event with it. Conversely, for a cluster with high inter-cluster dissimilarity, there is more leeway to assign a new real time log event as belonging to that cluster. Hence the Critical Distance may be more than the Cluster Radius. Silhouette Width acts a balance measure in choosing the Silhouette Threshold as the Critical Distance Measure

## VIII. CONCLUSION

In this paper, I have introduced various techniques forreal time log categorization. First I have experimentally evaluated traditional document clustering techniques for categorization of the corpus. Then I have proposed computationally efficient strategies for categorizing real time events. However, the major contribution of this work lies in the introduction of Critical Distance Measures to identify if similar real time events are unobserved in the corpus. I have found that among the centroid methods that I have evaluated, Spherical K Means performs well with log events. I have also found that the new measure Silhouette Threshold that I have introduced performs well as a Critical Distance Measure to identify new log event categories.Finally, there is lot of scope for future work. It would be interesting if the mechanism proposed is combined with the ordering of terms in the log events. An ordering weighted spherical k-means is likely to produce much better results in log categorization. Similarly, a framework that uses the proposed mechanism along with the trouble tickets would be of considerable industrial interest.

**Appendix**
Attached is a summary workflow for industrial applications


Batch Task
Input log corpus
Pre-process log corpus
Build TF-IDF matrix from the log corpus [n X m matrix]
Generate Cluster Model using Spherical K Means
Generate Centroid Matrix from Cluster Model [k X m matrix]
Calculate Cluster Radius, Silhouette Width for each cluster
Calculate Silhouette Threshold for each cluster
End Batch Task
n = number of log events in log corpus
m = number of words in log corpus vocabulary
Δm = number of new words in the real time log event
k= number of categories/centers/clusters in log corpus
Store
Real TimeTask
Input real time log event
Pre-process real time log event
Build TF-IDF vector for the real time log event [1 X (m + Δm) matrix]
Adapt the Centroid Matrix for real time log event [k X m matrix] → [k X (m + Δm) matrix]
Calculate distance from real time log event TF-IDF vector to each Cluster Centroid in the Centroid Matrix
Find the closest Cluster Centroid in the Centroid Matrix
Is distance to closest Cluster Centroid > Silhouette Threshold ?
NO
Assign real time log event to category of closest Cluster Centroid
YES
Assign real time log event as New category

**Acknowledgment**
I sincerely thank Mr. Sudheer Koppineni, Digital Enterprise, TCS and Danteswara Rao Pilla, TCS for their guidance and contributions

**References**
[1] K. Hammouda and M. Kamel, "Collaborative Document Clustering," Proc. Sixth SIAM Int'l Conf. Data Mining (SDM'06), pp. 453-463, Apr. 2006.
[2] Michael Steinbach, G. Karypis and Vipin Kumar," A comparison of document clustering techniques." KDD Workshop on Text Mining, 2000
[3] Kurt Hornik, Ingo Feinerer, Martin Kober, Christian Buchta "Spherical k-Means Clustering" JSS Vol. 50, Issue 10, Sep 2012
[4] P. Spyns "Natural language processing," Methods of information in medicine, vol. 35, no. 4, pp.285–301, 1996.
[5] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. "Clustering on the unit hyperspherenn using von mises-sher distributions." Journal of Machine Learning Research, 6:1345-1382,2005.
[6] Bolshakova N. and Azuaje F., "Cluster Validation Techniques for Genome Expression Data", Signal Processing, 83, 2003, pp. 825-833.
[7] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries", (2012) September 19, http://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf.
[8] Li Weixi, "Automatic Log Analysis using Machine Learning : Awesome Automatic Log Analysis version 2.0", (2013) November 19, http://uu.diva-portal.org/smash/get/diva2:667650/FULLTEXT01.pdf
[9] Stanford University - Evluation of Clusteringhttp://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html
[10] Jayadeep Jacob, "Real-time categorization of log events", (2015) March 17, https://www.google.co.uk/patents/US20160196174