Scalable Human-in-the-Loop Decision Support

Ramona Georgescu, Kishore Reddy, Nikola Trčka, Mei Chen, Paul Quimby, Paul O'Neill, Taimoor Khawaja, Luca Bertuccelli, Dan Hestand, Soumik Sarkar, Ozgur Erdinc and Michael Giering
Systems Department
United Technologies Research Center
{georgera, reddykk, trckan, chenm4, quimbypw, oneillpc, khawajts, bertuclf, hestanpd, sarkars, erdinco, gierinmj}@utrc.utc.com

Abstract—**In this work, a scalable human-in-the-loop decision support system has been built around an active learning algorithm operating on time series data. Anchored in big data analytics, the system integrates an architecture component, hierarchical clustering, a random data access module, active learning, a communication and user interaction modules.**

I. Introduction

With the advent of ubiquitous sensing and advanced computation capabilities, today's industry is the generator of and exposed to high volume of data. Regular and otherwise non-intuitive actionable intelligence can be extracted from "Big Data" that can immensely help industry to optimize operations and decision-making in various domains [1].

Until now, non-engineering problems such as business analytics, social media, healthcare solutions and financial forecasting have been primary applications of the "Big Data" revolution. However, the wave has already reached the realm of core engineering applications and highly convincing use cases are emerging from various fields. For example, in the energy sector, applications such as optimizing supply-demand trade-off in smart grids, automated health monitoring and supervisory control in large-scale complex energy generation/storage systems can leverage data-driven technologies to achieve low-cost and scalable solutions. Similarly, various data-intensive applications are emerging in the manufacturing sector, e.g., discovering sources of manufacturing flaws, quality control and reducing process inefficiencies and material characteristics analysis. Other than these, general engineering service sector decision support systems (e.g., predicting failures and prescribing maintenance actions in large equipment fleets, recommendation for optimal operation step) are also beginning to leverage big data analytics techniques [2].

Along the line of such development, the issue of cyber-physical security is drawing more and more attention for safety and verifiability. Apart from the general ability to utilize the high volume of data generated by the engineering systems, it turns out that data-driven techniques are actually better suited for many of the complex cyber-physical systems mentioned above as developing reliable physics based models for analysis can be extremely difficult and expensive.

While advanced data-mining and machine learning tools (i.e., indexing/retrieval, clustering, recommending) are becoming highly sophisticated, in most of the applications, automated decision support systems using them can still be sub-optimal due to lack of efficient domain knowledge elicitation and contextual adaptation [3]. Therefore, passive decision support systems that depend solely on pre-defined strategies and models can achieve significant gain in performance with even minimal human inputs. However, such active learning schemes need to be optimal in order to keep the user engaged but not overwhelm at the same time. From a technical point of view, this task may become non-trivial as traditional Big Data ecosystems may suffer from latency issues that can defeat user interaction and visualization purposes [4].

Apart from processing and communication latency issues, an active decision support system also needs to optimize the human interaction modality. As opposed to machine data, human generated information can become rather unstructured in general. However, constraining the interaction modality and scheme can limit the possibility to extract implicit human intent that can prove to be extremely valuable for analytics and decision-making. Extracting implicit intent does not only help make better decisions, it can also help the system understand user preference so that it can further adapt to facilitate a seamless interaction.

In the following, this extended abstract discusses the individual building blocks of a practical use-case for a human-in-the loop decision support system designed for semi-automatic characterization of field service data; the focus is on the analytical tools.

II. Hierarchical Clustering

Clustering has widely been used for exploratory/statistical data analysis and machine learning. With the advent of large datasets in the order of tera bytes, performing clustering on such data became a challenge.

Hierarchical clustering uncovers a hierarchical structure in the data which is more informative than the results generated by unstructured clustering algorithms like k-means and spectral clustering which produce flat results. For this reason, hierarchical clustering is preferred for data analysis and also for performing visual analysis by drawing up the tree.

Another benefit of employing hierarchical clustering lies in this being an unsupervised technique only requiring a designated similarity measure; the algorithm does not need initialization parameters to be set up or knowledge of number of clusters. The work relies on PARABLE, a parallel implementation of hierarchical clustering within a MapReduce framework that successfully addresses large datasets and was proposed by Wang and Dutta [5]. Here, PARABLE was implemented using Apache Spark [6] resilient distributed datasets (RDDs) as shown in Figure 1.

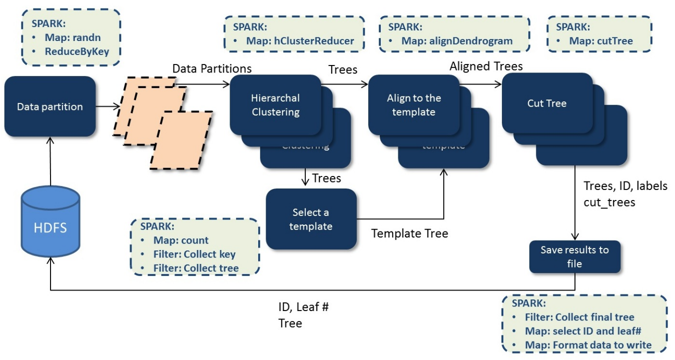


Fig. 1. PARABLE Hierarchical Clustering Implemented with Apache Spark RDDs.

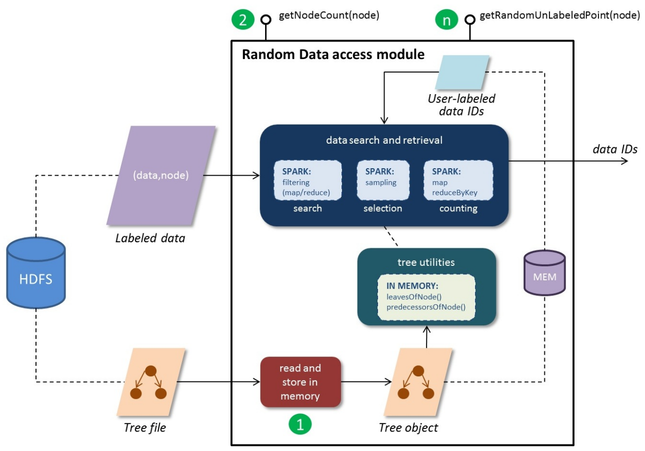


Fig. 2. Random Data Access with Apache Spark MapReduce Engine.

III. Random Data Access

The data access module is organized as in Figure 2. Satisfying the requirements of the active learning component, the user of the data access module can request a random (not already labeled) point from a node in the clustering tree (getRandomUnLabeledPoint(node)), or to get the total number of data points in a node (function getNodeCount()).

All operations are performed as Apache Spark MapReduce jobs, using the Python API. The data counting process is a simple application of a map to a single key followed by the reduceByKey operation. The data retrieval process is a combination of a MapReduce-based search and the random sampling operation provided by Spark.

All points labeled in a single session are stored in local memory, which is justified by the fact that the size of this data must be bounded by the total number of iterations between the system and the human user. While the (labeled) data points reside in HDFS, the clustering tree is assumed to be of manageable size and is thus stored in memory, for fast access and small query times. The "tree utilities" component handles various operations on the tree that are called by the main modules.

IV. Active Learning

Active learning has been studied and applied in general learning schema for a long time with significantly lower label complexity [7], [8]. In this work, we followed the idea of querying the user for labels – especially the ones lying on boundaries– with the goal of high confidence data distribution learning.

Active learning was conducted on a hierarchical clustering tree which is constructed so that a pruning of it is weakly informative of the class labels. The algorithm identifies pure nodes with proper labels and selects best combinations of leaves/nodes at different tree levels so that a pruning which covers all data points is generated. This pruning contains nodes and labels indicative of the true data distribution which can be used for regular supervised learning.

The cluster-adaptive active learning implemented here iterates through the following 6 steps. Figure 3 shows the associated pseudocode. The details of the algorithm are left to the original paper [8].

1) Pick node v in the hierarchical clustering-generated tree for querying. Node v is selected by an active learning rule which discourages sampling nodes that are currently fairly pure in their labels. The backup sampling option is random sampling.

   Select an unlabeled data point z that belongs to node v and query user for its label.

   The query results will contain the label and confidence level of an imperfect expert (will not be covered in this paper). In future work, dynamically updating the labels will be considered.

2) Update n_u, $p_{u,l}$, i.e. the number of points sampled from node u and the fraction of label l in node u, for all nodes u on path from z to the root of the subtree T_v rooted at v.

3) Calculate $p_{v,l}^{LB}$ and $p_{v,l}^{UB}$, where these are the lower and upper bounds of the confidence in node v having label l.

   Update the admissibility set A consisting of (node, label) pairs based on the majority label criterion.

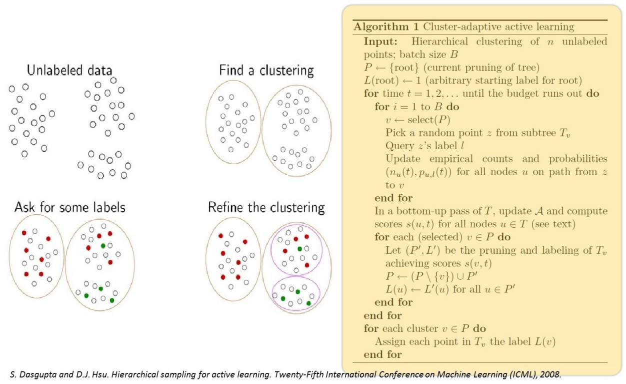
   Compute $\tilde{\epsilon}_{v,l}$, i.e. the error induced by a proposed labeling and the score $s(v)$, i.e. the error of the best admissable pruning anf labeling of T_v.

4) In a bottom up pass, calculate all possible prunings for the entire tree.

5) Update admissible prunings of the entire tree from the calculated all possible prunings.

6) Select the best pruning and label of entire tree, i.e. the pair achieving the best score.

   Finish by calculating the confidence matrix CM describing the worst case error in labeling.



Unlabeled data
Find a clustering
Ask for some labels
Refine the clustering
S. Dasgupta and D.J. Hsu. Hierarchical sampling for active learning. Twenty-Fifth International Conference on Machine Learning (ICML), 2008.

Algorithm 1 Cluster-adaptive active learning

```
Input: Hierarchical clustering of $n$ unlabeled points; batch size $B$
$P \leftarrow \{\text{root}\}$ (current pruning of tree)
$L(\text{root}) \leftarrow 1$ (arbitrary starting label for root)
for time $t = 1, 2, \dots$ until the budget runs out do
  for $i = 1$ to $B$ do
    $v \leftarrow \text{select}(P)$
    Pick a random point $z$ from subtree $T_v$
    Query $z$'s label $l$
    Update empirical counts and probabilities $(n_u(t), p_{u,l}(t))$ for all nodes $u$ on path from $z$ to $v$
  end for
  In a bottom-up pass of $T$, update $\mathcal{A}$ and compute scores $s(u,t)$ for all nodes $u \in T$ (see text)
  for each (selected) $v \in P$ do
    Let $(P', L')$ be the pruning and labeling of $T_v$ achieving scores $s(v,t)$
    $P \leftarrow (P \setminus \{v\}) \cup P'$
    $L(u) \leftarrow L'(u)$ for all $u \in P'$
  end for
end for
for each cluster $v \in P$ do
  Assign each point in $T_v$ the label $L(v)$
end for
```

Fig. 3. Active Learning Pseudocode.

A. *Number of labels*

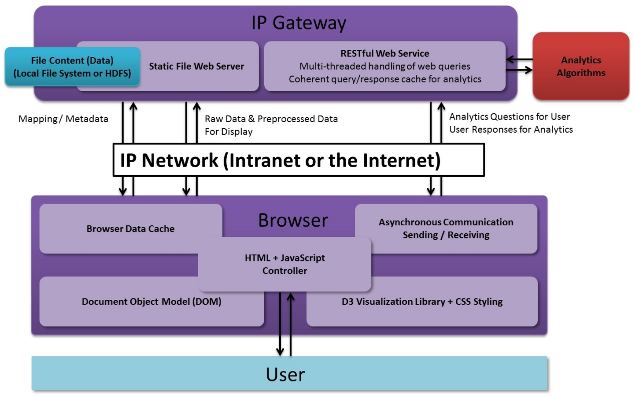
This work extended the original algorithm to the case of three label types $\beta = 3$. In this case, the criterion for determining the majority of $A_{v,l}$ needed an update (see step 3 above). Originally, $p_{v,l}^{LB} > 2p_{v,l'}^{UB} - 1, \forall l' \neq l$, for $l = 1, \dots, K$ with $\beta = 2$, where the algorithm is designed to incur at most β times as much error with the labeling it recommends than with any other label. For any given v, t, several different labels l might satisfy this criterion, for instance if $p_{v,l}^{LB}(t) = p_{v,l}^{UB}(t) = \frac{1}{K}$ for all labels l [8]. Then, $p_{v,l}^{UB} = 1 - \sum_{k \neq l} p_{v,k}^{LB}$. The detailed calculation is given by:

$$
\begin{aligned}
p_{v,l}^{LB} &> 2(1 - \sum_{k \neq l'} p_{v,k}^{LB}) - 1; \\
p_{v,l}^{LB} &> 2 - 2p_{v,l}^{LB} - 2 \sum_{k \neq l, l'} p_{v,k}^{LB} - 1; \\
3p_{v,l}^{LB} &> 1 - 2\frac{K-2}{K}. \qquad (1)
\end{aligned}
$$

In the case of two labels $K = 2$, $p_{v,l}^{LB} > \frac{1}{3}$. When $K = 3$, then $p_{v,l}^{LB} > \frac{1}{9}$. Now, the calculation of $p_{v,l}^{LB}$ will be:

$$
\begin{aligned}
p_{v,l}^{LB} &> 1 - \beta(1 - \sum_{k \neq l'} p_{v,k}^{UB}); \\
p_{v,l}^{LB} &> 1 - \beta + \beta \sum_{k \neq l'} p_{v,k}^{UB}; \\
p_{v,l}^{LB} &> 1 - \beta + \beta(1 - \sum_{k \neq l'} p_{v,k}^{LB}); \\
p_{v,l}^{LB} &> 1 - \beta + \beta - \beta p_{v,l}^{LB} - \beta \sum_{k \neq l, l'} p_{v,k}^{LB}; \\
(1 + \beta) p_{v,l}^{LB} &> 1 - \beta \frac{K-2}{K}. \qquad (2)
\end{aligned}
$$

When the number of labels K gets large, the r.h.s of the above equation suggests that if we still allow the majority label to incur at most twice as much error as the other labels, then any label would fit this criterion. Therefore, we need to use a smaller β in equation (2) to determine the node purity. We choose $\beta = 1 + \frac{1}{K}$ so that β being slightly larger than 1 is still a strong enough assumption to allow the majority label to have a little more error than any other labels.



IP Gateway
File Content (Data) (Local File System or HDFS)
Static File Web Server
RESTful Web Service
Multi-threaded handling of web queries
Coherent query/response cache for analytics
Analytics Algorithms
Mapping / Metadata
Raw Data & Preprocessed Data For Display
Analytics Questions for User
User Responses for Analytics
IP Network (Intranet or the Internet)
Browser
Browser Data Cache
Asynchronous Communication Sending / Receiving
HTML + JavaScript Controller
Document Object Model (DOM)
D3 Visualization Library + CSS Styling
User

Fig. 4. User Interaction is Achieved Through the Communication and Visualization Modules.

B. *User Interaction*

The user interaction was set up in two stages: a communication module and a visualization one as shown in Figure 4. The communication consists of a data web server Python implementation that takes in as input the request from the active learning algorithm for a data point z to be labeled by the user. The RESTful interface protocol development provides multi-threaded handling od queries with coherent query and response cache as it serves the file (which resides on Hadoop HDFS) corresponding to the data point over the IP network to the visualization module.

The browser client takes the file and relies on the D3 JavaScript library to display the necessary information to the user. The user is then able to label the data point to be sent back to the active learning algorithm. Note that the back and forth information exchange is done asynchronously both on the sending and the receiving ends.

V. Results

An overview of the integrated system is displayed in Figure 5. On the left, data storage in Hadoop's HDFS file system supports the raw csv files. The interactions with the algorithmic side on the right are given in terms of inputs and outputs. Note that both the hierarchical clustering and active learning algorithms run in distributed fashion by calling Apache Spark within Python code while the user interface is designed to be currently generic yet customizable for multi-user capability.

Figure 6 shows results on a real fleet monitoring dataset where hierarchical clustering created a tree with 9 nodes and the active learning algorithm works on 2 labels (here, red and black). The label probabilities for the 5 leaf nodes are given after 99 queries to the user[1]. As expected, the confidence bounds become tight in the case of the nodes that have many data points labeled, e.g. after 41 queries, node 2 has 22 data points with label 1 and 2 data points with label 2 while after 99 queries, node 2 has 37 data points with label 1 and 3 data points with label 2 resulting in probability (node 2 majority label = label 1)=0.925 and probability(node 2 majority label =label 2) = 0.075 with confidence interval $[0.85, 0.99]$. Additionally, the

[1]The algorithm was initialized with a batch set of labels from the user for speedup purposes only.

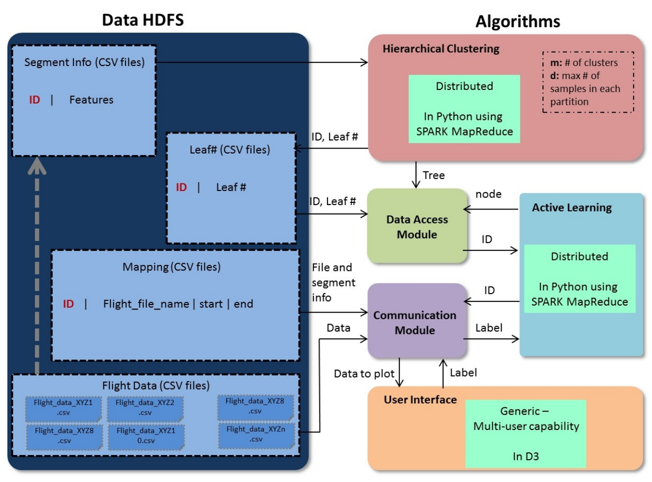
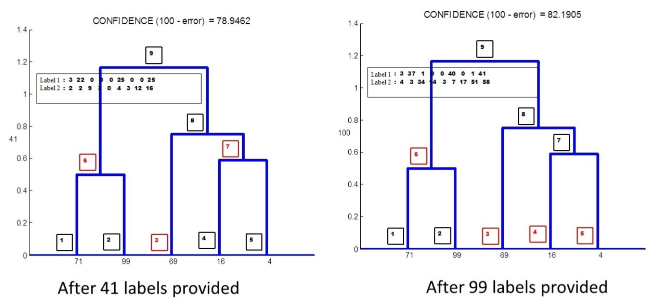


Fig. 5. Integration.



Node #	1	2	3	4	5
P(Label 1)	0.43	0.925	0.02	0	0
P(Label 2)	0.57	0.075	0.97	1	1
Bounds (majority)	[0.24,0.90]	[0.85,0.99]	[0.91,1.00]	[0.92.1.00]	[0.66,1.00]

Fig. 6. Active Learning Results.

total confidence for the tree labeling has improved between 41 user queries (78.94) and 99 user queries (82.19). This metric rapidly grows with the first queries and then tends to level off.

A. Scalability

We define scalability as the ability of the system to achieve equivalent computational efficiency when the data quantity and the distribution of the system increases. The results of this simulation study are subject to the limits of the use of the data, e.g. visualization or caching versus disk storage and are bounded from below by the algorithmic computational complexity. By "equivalent computational efficiency" we mean that the latency, throughput and accuracy are affected only by an additive constant characteristic of the system architecture.

The primary architectural driver is to maintain scalability as the system architectural complexity grows. As Figure 7 shows, the goal has been met as both the random data access and hierarchical clustering respond linearly in latency to the linear increase in data points.

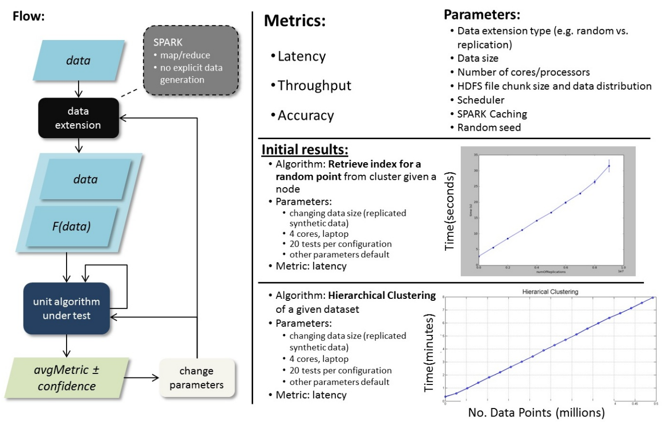


Fig. 7. Scalability Study Results.

VI. Conclusion

Active learning is an attractive algorithm due to its flexibility. It can incorporate any clustering algorithm, runs in a completely unsupervised setting, does not require balanced datasets (of high importance when working on field collected data) and pleasantly, its query strategy can be modified to accomodate mode realistic scenarios.

In this work, a scalable human-in-the loop decision support system was built to support fleet monitoring in fielded systems. At its core, experts supplement a dataset with domain knowledge / semantic knowledge via labeling. The active learning algorithm chooses as few samples as possible to ask the expert to label. At any given time, algorithm can "label" all remaining instances, and able to provide "confidence" metrics for each automatically assigned label.

Future extensions (both short and long term) include: streaming data clustering, incorporating feature-salience queries and/or feature weighing and multiple experts / imperfect experts.

References

[1] http://www.kdnuggets.com/

[2] Z. Zheng, J. Zhu and M.R. Lyu. *Service-Generated Big Data and Big Data-as-a-Service: An Overview*. IEEE Intl. Congress on Big Data, 2013.

[3] D. R.Holmes. *Keynote address: Clinical Decision Support: The challenge of big data and big computation*. IEEE Intl. Symposium on Workload Characterization (IISWC), 2012.

[4] C. Hansen. *Keynote speaker: Big data: A scientific visualization perspective*. IEEE Pacific Visualization Symposium, 2013.

[5] S. Wang and H. Dutta. *PARABLE: A PArallel RAndom-partition Based HierarchicaL ClustEring Algorithm for the MapReduce Framework*. Technical Report CCLS-11-04, 2011.

[6] http://spark.apache.org/documentation.html

[7] D. Cohn, L. Atlas and R. Ladner, *Improving generalization with active learning*. Machine learning, Vol. 15, No. 2, pp.201-221, 1994.

[8] S. Dasgupta and D. Hsu. *Hierarchical sampling for active learning*. Intl. Conf. on Machine learning (ICML), 2008.