Journal of Big Data
a SpringerOpen Journal

RESEARCH Open Access

A big data methodology for categorising technical support requests using Hadoop and Mahout

Arantxa Duque Barrachina[1] and Aisling O'Driscoll[2*]

* Correspondence:
aisling.odriscoll@cit.ie
[2]Department of Computing, Cork Institute of Technology, Cork, Ireland
Full list of author information is available at the end of the article

Abstract

Technical Support call centres frequently receive several thousand customer queries on a daily basis. Traditionally, such organisations discard data related to customer enquiries within a relatively short period of time due to limited storage capacity. However, in recent years, the value of retaining and analysing this information has become clear, enabling call centres to identify customer patterns, improve first call resolution and maximise daily closure rates. This paper proposes a Proof of Concept (PoC) end to end solution that utilises the Hadoop programming model, extended ecosystem and the Mahout Big Data Analytics library for categorising similar support calls for large technical support data sets. The proposed solution is evaluated on a VMware technical support dataset.

Keywords: Big data analytics; Distributed clustering; Parallelised programming; Hadoop; HBase; Hive; Mahout; Support call center

Introduction

In recent years, there has been an unprecedented increase in the quantity and variety of data generated worldwide. According to the IDC's Digital Universe study, the world's information is doubling every two years and is predicted to reach 40ZB by 2020 (Digital Universe Study (on behalf of EMC Corporation) [1]). This increase in data, often referred to as a "data tsunami", is driven by the proliferation of social media along with an increase in mobile and networked devices (the Internet of Things), finance and online retail as well as advances in the physical and life sciences sectors. As evidence of this, the online microblogging service Twitter, processes approximately 12 TB of data per day, while Facebook receives more than five hundred million likes per day (McKinsey Global Institute [2]). In addition, the Cisco Internet Business Solutions Group (IBSG) predicts that there will be 25 billion devices connected to the Internet by 2015 and 50 billion by 2020 (Cisco Internet Business Solutions Group (IBSG) [3]). Such vast datasets are commonly referred to as "Big Data". Big Data is characterised not only by its volume, but by a rich mix of data types and formats (variety) and it's time sensitive nature which marks a deviation from traditional batch processing (velocity) (Karmasphere [4]). These characteristics are commonly referred to as the 3 V's.

Traditional distributed systems and databases are no longer suitable to effectively capture, store, manage and analyse this data and exhibit limited scalability. Furthermore, relational databases support structured data by imposing a strict schema; however, data growth is currently driven by unstructured data by a factor of 20:1 (Karmasphere [5]). Finally, data warehouses are no longer able to process whole datasets due to their massive size; hence the information stored in these solutions is no longer statistically representative of the data from which it was extracted, making the data analytics performed on it less reliable. Big Data requires new architectures designed for scalability, resilience and efficient parallel processing.

As a result, big data processing and management (capturing and storing big data) has gained significant attention in recent years e.g. the MapReduce paradigm. However it is now recognised that it is necessary to further develop platforms that can harness these technologies in order to gain meaningful insight and to make more informed business decisions. The implementation of data analytics on such datasets is commonly referred to as *big data analytics*. Data mining and machine learning techniques are currently used across a wide range of industries to aid organisations in optimising their business, reduce risks and increase profitability. Sectors employing these techniques include retailers, banking institutions and insurance companies as well as health related fields (Huang et al. [6]; Lavrac et al. [7]). In the current marketplace big data analytics has become a business requirement for many organisations looking to gain a competitive advantage as evidenced by IBM's 2011 Global CIO study that places business intelligence and analytics as the main focus for CIOs over the next five years, on top of virtualisation and cloud computing (IBM [8]).

Importantly, this has further been recognised in the technical support space of leading technology multinationals, as call centres have begun to explore the application of data analytics as a way to streamline the business and gain insight regarding customer's expectations, a necessity in an industry challenged by economic pressures and increased competition (Aberdeen Group [9]). Thus this paper;s contributions are two-fold:

- An end to end proof of concept solution based entirely on open source components is described that can be used to process and analyse large technical support datasets to categorise similar technical support calls and identify likely resolutions. The proposed solution utilises the Hadoop distributed data processing platform, extended ecosystem and parallelised clustering techniques using the Mahout library. It is envisaged that if such a solution was deployed in commercial environments with large technical support datasets, updated on a daily basis, it would expedite case resolution and accuracy to maximise daily closure rates by providing similar case resolutions to staff when a new technical support case is received. If achieved, the reduction of resolution time would also ultimately aid technical support teams to increase customer satisfaction and prevent churn. Furthermore this solution could also be used to identify the most problematic product features and highlight staff knowledge gaps leading to more directed staff training programmes.
- Secondly an evaluation of the performance and accuracy of parallelised clustering algorithms for analysing a distributed data set is conducted using a real-world technical support dataset.

The rest of this paper is organised as follows: Section II describes the algorithms and technologies underpinning the proposed architecture along with related work in this

Section III outlines the architecture and implementation of the proposed technical support analytics platform. Section V details the performance evaluation and analysis of the proposed solution with Section VI outlining final conclusions.

Background & literature review

A brief overview of the constituent technologies is now provided. The need for efficient, scale out solutions to support partial component failures and provide data consistency motivated the development of the *Google File System (GFS)* (Ghemawat et al. [10]) and the *MapReduce* (Dean & Ghemawat [11]) paradigm in the early 2000s. The premise behind the Google File System and MapReduce is to distribute data across the commodity servers such that computation of data is performed where the data is stored. This approach eliminates the need to transfer the data over the network to be processed. Furthermore, methods for ensuring the resilience of the cluster and load balancing of processing were specified. GFS and MapReduce form the basis for the *Apache Hadoop* project, comprising two main architectural components: the *Hadoop Distributed File System (HDFS)* and *Hadoop MapReduce* (Apache Hadoop [12]). HDFS (Shvachko et al. [13]) is the distributed storage component of Hadoop with participating nodes following a master/slave architecture. All files stored in HDFS are split into blocks which are replicated and distributed across different slave nodes on the cluster known as *data nodes*, with a master node, called the *name node*, maintaining metadata e.g. blocks comprising a file, where in the cluster these blocks are located and so on. MapReduce (Bhandarkar [14]) is the distributed compute component of Hadoop. MapReduce jobs are controlled by a software daemon known as the *JobTracker*. A job is a full MapReduce program, including a complete execution of Map and Reduce tasks over a dataset. The MapReduce paradigm also relies on a master/slave architecture. The *JobTracker* runs on the master node and assigns *Map* and *Reduce* tasks to the slave nodes in the cluster. The slave nodes run another software daemon called the *TaskTracker* that is responsible for actually instantiating the Map or Reduce tasks and reporting the progress back to the JobTracker.

The extended Hadoop ecosystem includes a growing list of solutions that integrate or expand Hadoop's capabilities. *Mahout* is an open source machine learning library built on top of Hadoop to provide distributed analytics capabilities (Apache Mahout [15]). Mahout incorporates a wide range of data mining techniques including collaborative filtering, classification and clustering algorithms. Of relevance to this paper, Mahout supports a wide variety of clustering algorithms including: k-means, canopy clustering, fuzzy k-means, Dirichlet Clustering and Latent Dirichlet Allocation. *HBase* is a distributed column-oriented database that resides on top of Hadoop's Distributed File System, providing real-time read/write random-access to very large datasets (Apache Hbase [16]). Additionally, *Hive* defines a simple SQL-like query language, called HiveQL, to abstract the complexity of writing MapReduce jobs from users. Hive transforms HiveQL queries into MapReduce jobs for execution on a cluster (Apache Hive [17]).

While research associated with machine learning algorithms is well established, research on big data analytics and large scale distributed machine learning is very much in its infancy with libraries such as Mahout still undergoing considerable development. However some initial experimentation has been undertaken in this area.

Esteves et. al studied the use of Mahout k-means clustering on a 1.1GB data set of TCP dumps from an airforce LAN, while examining clustering scalability and quality (Esteves et al. [18]). The authors subsequently evaluated the applications of k-means and fuzzy c-means clustering on an 11GB Wikipedia dataset with respect to clustering algorithm and system performance (Esteves & Rong [19]). However in contrast to this work, the authors of this paper consider a 32GB dataset, the impact of five clustering algorithms and, most importantly, provide a complete end to end solution including data pre-processing, daily upload of new data, real-time access and a user interface by using the extended Hadoop ecosystem. Ericson et.al use the 1987 Reuters dataset, one of the most widely used text categorisation data sets containing approximately 21,578 documents, to evaluate clustering algorithms over Hadoop but also over their own runtime environment, Granules (Ericson & Palliekara [20]). While the performance of four clustering algorithms was evaluated, the primary evaluation concentrated on the underlying runtime environment and unlike the proposed architecture, a complete end to end solution was not provided as a clean data set was used.

In contrast, the end to end framework presented in this paper successfully integrates Hadoop and Mahout to provide a fully functional end to end solution that addresses a real world problem, facilitating the streamlining of call centre operations and evaluating multiple clustering algorithms in terms of timeliness and accuracy.

Research design and methodology

The proposed solution provides an end to end solution for conducting large scale analysis of technical support data using the open source Hadoop platform, components of the Hadoop Extended Ecosystem such as HBase and Hive and clustering algorithms from the extended Mahout library. Figure 1 illustrates the architecture of the proposed analytics solution.

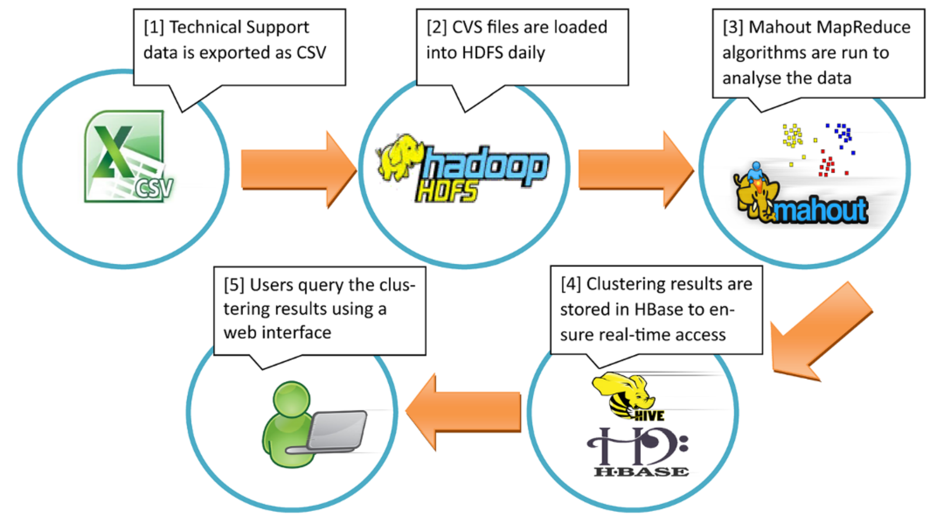

[1] Technical Support data is exported as CSV
CSV
[2] CVS files are loaded into HDFS daily
hadoop HDFS
[3] Mahout MapReduce algorithms are run to analyse the data
mahout
[4] Clustering results are stored in HBase to ensure real-time access
HIVE
H·BASE
[5] Users query the clustering results using a web interface

Figure 1 **Proposed opensource end to end solution for analysing large technical support data sets.**

Data pre processing

To allow technical support data to be processed by Mahout, it must be uploaded to HDFS and converted in text vectors. The VMware technical support data under consideration in this paper is stored in the cloud Software as a Service (SaaS) application, Salesforce, a popular Customer Relationship Management (CRM) service. Thus a Hadoop job is devised to convert the technical support data exported from Salesforce in CSV format into Hadoop *SequenceFile* format. A Hadoop Sequence File is a flat file data structure consisting of binary key/value pairs. Hadoop mappers employ an *InputReader* to parse input keys and values, which the *mapper* task subsequently processes before outputting another set of keys and values. As the default Hadoop *InputReader* is the *TextInputFormat* where every line of text represents a record, this is not applicable for CSV format as technical support calls span multiple lines. Thus a custom *input record reader* and *partitioner* were required in the proposed solution. This custom *input record reader* accumulates text from the input file until it reaches a specified end of record marker. As the *mapper* requires a key and value, the *value* is the resultant text from the *input record reader*, including the support call identifier and support call description. The *key* is set to the position in the file of that record i.e. the byte offset. The *mapper* extracts the support call identifier (passed to the *reducer* as the key) and the support call description (passed to *reducer* as the value). Finally, the *reducer* receives

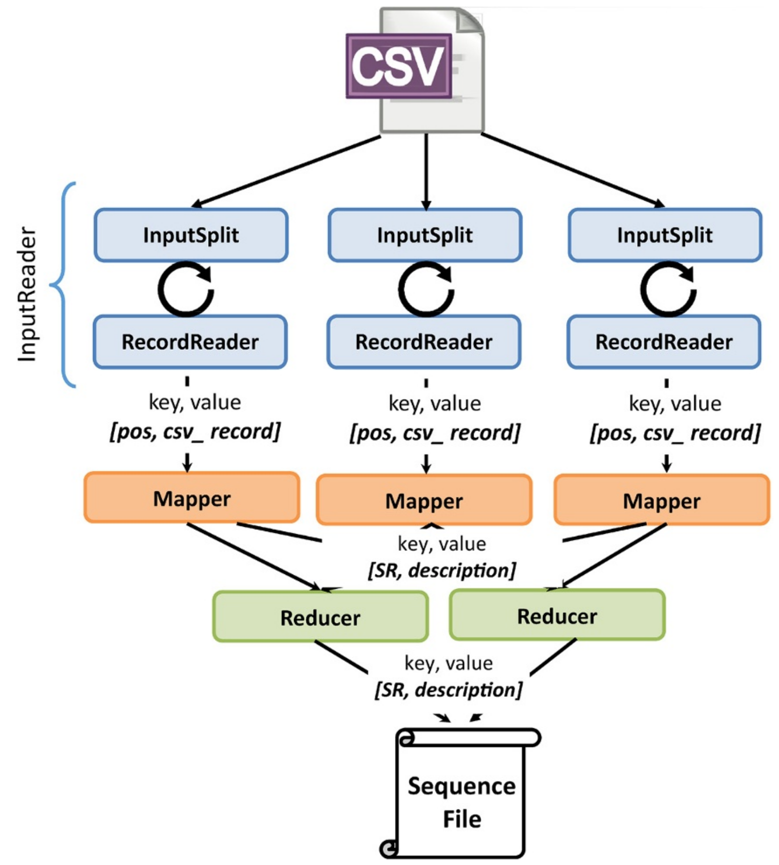


Figure 2 Design of a custom MapReduce job to import CSV salesforce technical support data into Hadoop.

these key/value pairs and writes them into a Hadoop *SequenceFile* format so they can be further processed using Mahout. Figure 2 illustrates this process by displaying the anatomy of the custom developed MapReduce job, illustrating input and output keys and values. SR represents the Service/Support Request.

Automated periodic refresh of new technical support data

As technical support data sets will be updated daily with new solutions derived, a mechanism to automatically update the technical support data from which the clustered results are derived to ensure accurate output is required. Given the nature of the data, it is sufficient to perform such a procedure on a daily basis. To avoid uploading the entire technical support data set on a daily basis, a cron job can be scheduled to automatically backup the changes that occurred over the preceding 24 hours. It is proposed that a separate Hadoop job is responsible for uploading the CSV files containing the details of the support requests received that day into HDFS.

Parallelised clustering

In order for data to be processed by the Mahout clustering algorithms, it must first be converted to vector format. Technical support data stored in HDFS is converted into vectors using Mahout's existing command line for subsequent clustering analysis. Importantly, the challenge of identifying related support calls based on their problem description is resolved by using Mahout's distributed clustering machine learning algorithms to analyse the data set, thereby identifying support calls with a similar description. Multiple clustering algorithms are evaluated in the proposed system with respect to system performance and accuracy. The specific details of the clustering evaluation and results are discussed further in Section V. It is beyond the scope of this paper to provide a detailed description of each clustering algorithm with respect to a parallelized Map Reduce job. However to as an indication, the k-means algorithm described from the perspective of a MapReduce job is outlined: Each map task receives a subset of the initial centroids and is responsible for assign each input data-point i.e. text vector to its nearest cluster i.e. centroid. For each data point, the mapper generates a key/value pair, where the key is the cluster identifier and the value corresponds to the coordinates of that point. The algorithm uses a combiner to reduce the amount of data to be transferred from the mapper to the reducer. The combiner receives all key/value pairs from the mapper and produces partial sums of the input vectors for each cluster. All values associated with the same cluster are sent to the same reducer, thus each reducer receives the partial sums of all points of one or more clusters and computes the new cluster centroids. The driver program iterates over the points and clusters until all clusters have converged or until the maximum number of iterations has been reached.

Real-time platform access

Importantly, Hadoop does not provide real-time access and is designed for batch processing. Thus, once the analysis phase is completed, a Hadoop MapReduce job stores the clustering results into a non-relational database so that technical support engineers can query the information in real-time. The information stored includes the support call number, the cluster identifier and the probability of that support call belonging to a given cluster. Given its native integration with Hadoop, HBase has been chosen for this purpose. When an engineer requests the support calls related to a particular case, the cluster to which such a case should belong is identified in HBase, with all support calls within that cluster sorted based on their cluster membership probability. An ordered list containing the support call identifier and its associated cluster membership probability is returned to the technical support engineer. The support calls displayed at the top of such list are more likely to contain similar problems to the specified unresolved technical support case and as a result are more likely to share the same resolution. Figure 3 illustrates this step in the solution by displaying the anatomy of the custom developed MapReduce job, illustrating input and output keys and values.

User query of results

Finally, it should be noted that directly querying HBase to obtain similar support calls is not user friendly and is technically complex requiring the development of a MapReduce job. To overcome this, HBase has been integrated with Hive. Such an approach allows technical support engineers to obtain similar support calls from a web-based interface that generates HiveQL to query the clustering results derived from Mahout. It can be envisaged

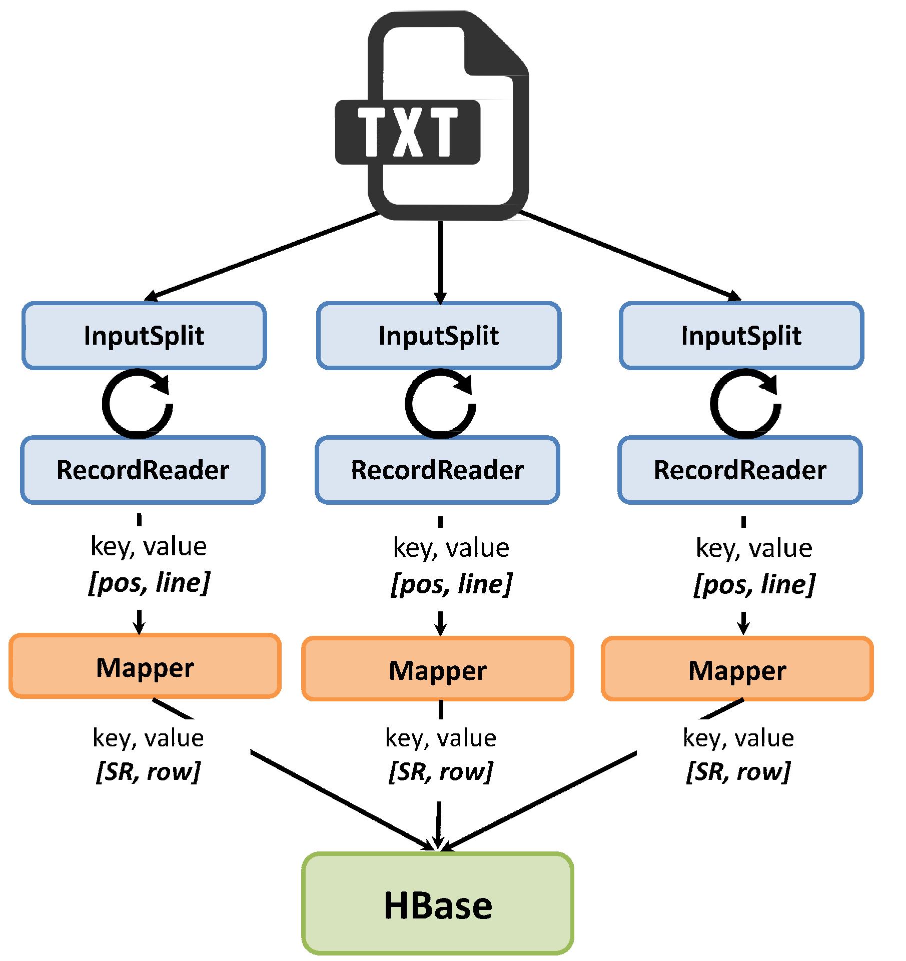


Figure 3 Design of a custom MapReduce job to transfer clustering output to HBase.

that this proof of concept web interface could be easily enhanced for integration with existing BI tools and to provide advanced technical support dashboards.

This section has outlined an end to end, open source proof of concept solution to transform, process, analyse and identify resolutions to similar technical support calls given an open case. A VMware technical support dataset is now analysed using five of Mahout's distributed clustering algorithms. The output from each of these algorithms outlining the impact of the nuances of each clustering algorithm as well as their performance is now discussed.

Results and discussion

The implemented solution outlined in the previous Section is now evaluated using a real-world VMware technical support data set of approximately 0.032 TB. This is based on an average call size of 60 KB from 4 main call centres, with an estimated 1900 calls per week. The four node Hadoop cluster is based on commodity hardware with each node following the same configuration: 8GB RAM; 2 TB hard drive; 4 MB Cache; 1 CPU Intel Core i7; 2 cores at 2.20 GHz; Onboard 100Mbps LAN; Linux CentOS with Hadoop v0.20 and Mahout v0.5.The evaluation is discussed with respect to the processing performance of the specific parallelized clustering algorithms and also with respect to the clustering accuracy.

Parallelised clustering performance

Five clustering algorithms are considered in the described evaluation: k-means, k-means with canopy clustering, fuzzy k-means, Dirichlet allocation and Latent Dirichlet Allocation (LDA). The employed clustering parameters are summarised in Table 1. The number of required clusters, *k,* was estimated by dividing the number of total support calls in the sample, approximately 10,000, by the target cluster size of 250 support calls, resulting in an estimate of 40 clusters. The well-known cosine distance is used as distance measures such as Euclidean distance are influenced by the document length rather than the content. Canopy clustering was next explored as an alternative method of creating the initial centroids and determining the number of required clusters. A well-known challenge is the derivation of meaningful values for the distance thresholds *T1* and *T2.* Multiple executions of the clustering algorithm found that values of 1.0 and 1.4 for *T1* and *T2* respectively generated a meaningful number of canopies of between 20 and 50, with resultant canopy centroids were used as an input for the k-means algorithm. The fuzzy k-means *fuzziness factor*

Table 1 Clustering parameters

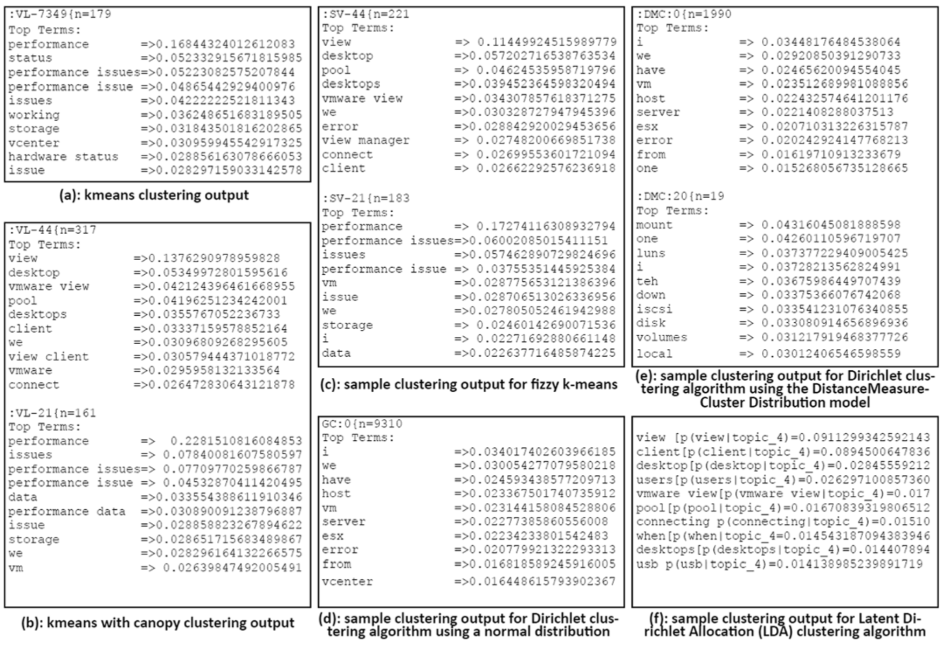
Parameter	*Value*
k (k-means)	40
T1 (Canopy clustering)	1.0
T2 (Canopy clustering)	1.4
Fuzziness Factor (Fuzzy-kmeans)	1.05
Dirichlet Clusters	50
Max Dirichlet Iterations	10
LDA Clusters	50
Words in Corpus	12886
Max LDA Iterations	30

Table 2 Mahout algorithms and their execution times

Algorithm	*Mahout command*	*Fixed clusters*	*Partial membership*	*Execution Time (ms)*
K-means	kmeans	Y	N	61096
Canopy	canopy	Y	N	121178
K-means	kmeans			
Fuzzy k-means	fkmeans	Y	Y	644471
Dirichlet	dirichlet	N	Y	15011080
LDA	lda	Y	Y	3373535

determines the degree of overlap between the generated clusters and analysis of the evaluated data set found that a value of 1.05 generated well-defined and differentiated clusters. To provide consistency with canopy clustering, Dirichlet clustering also specified 50 clusters with 50 topics also specified for LDA.

Table 2 summarises the characteristics of the clustering algorithms available in Mahout and their incurred execution times on the 4 node cluster. The simplicity and scalability of the k-means and canopy clustering algorithm result in very low execution times of approximately 1 minute and 2 minutes respectively. More complex clustering techniques such as fuzzy k-means and in particular Dirichlet clustering and Latent Dirichlet Allocation incur significantly higher execution times of over 10 minutes, approximately 4 hours and 56 minutes respectively. The execution delays experienced by the LDA clustering algorithm are dependent on the number of words in the corpus. LDA calculations are CPU intensive when performed on large datasets such as the support calls under study. The CPU utilisation of the slave nodes of the Hadoop cluster was analysed using the 'top' command (measures CPU utilisation, process statistics and memory utilisation) while running the LDA algorithm. Hadoop slave nodes showed an average CPU utilisation of



```
:VL-7349{n=179
Top Terms:
performance        =>0.16844324012612083
status             =>0.05233291567181598 5
performance issues=>0.0522308257520784 4
performance issue =>0.04865442929400976
issues             =>0.0422222252181134 3
working            =>0.03624865168318950 5
storage            =>0.03184350181620286 5
vcenter            =>0.03095994554291732 5
hardware status    =>0.02885616307866605 3
issue              =>0.02829715903314257 8
```

(a): kmeans clustering output

```
:VL-44{n=317
Top Terms:
view               =>0.1376290978959828
desktop            =>0.05349972801595616
vmware view        =>0.04212439646166895 5
pool               =>0.0419625123424200 1
desktops           =>0.0355767052236733
client             =>0.0333715957885216 4
we                 =>0.0309680926829560 5
view client        =>0.03057944437101877 2
vmware             =>0.0295958132133564
connect            =>0.02647283064312187 8

:VL-21{n=161
Top Terms:
performance        =>  0.2281510816084853
issues             => 0.07840081607580597
performance issues=> 0.07709770259866787
performance issue => 0.04532870411420495
data               =>0.03355438861191034 6
performance data   =>0.03089009123879688 7
issue              =>0.02885882326789462 2
storage            =>0.02865171568348986 7
we                 =>0.02829616413226657 5
vm                 => 0.02639847492005491
```

(b): kmeans with canopy clustering output

```
:SV-44{n=221
Top Terms:
view               => 0.11449924515989779
desktop            =>0.05720271653876353 4
pool               => 0.04624535958719796
desktops           =>0.03945236459832049 4
vmware view        =>0.03430785761837127 5
we                 =>0.03032872794794539 6
error              =>0.02884292002945365 6
view manager       => 0.02748200669851738
connect            => 0.02699553601721094
client             => 0.02662292576236918

:SV-21{n=183
Top Terms:
performance        => 0.17274116308932794
performance issues=>0.0600208501541115 1
issues             =>0.05746289072982469 6
performance issue => 0.03755351445925384
vm                 =>0.02877565312138639 6
issue              =>0.02870651302633695 6
we                 =>0.02780505246194298 8
storage            => 0.02460142690071536
i                  => 0.02271692880661148
data               =>0.02263771648587422 5
```

(c): sample clustering output for fizzy k-means

```
GC:0{n=9310
Top Terms:
i                  =>0.03401740260396618 5
we                 =>0.03005427707958021 8
have               =>0.02459343857720971 3
host               =>0.02336750174073591 2
vm                 =>0.02314415808452880 6
server             =>0.0227738586055600 8
esx                =>0.0223423380154248 3
error              =>0.02077992132229331 3
from               =>0.01681858924591600 5
vcenter            =>0.01644861579390236 7
```

(d): sample clustering output for Dirichlet clustering algorithm using a normal distribution

```
:DMC:0{n=1990
Top Terms:
i             => 0.03448176484538064
we            => 0.02920850391290733
have          => 0.02465620094554045
vm            => 0.023512689981088856
host          => 0.022432574641201176
server        => 0.0221408288037513
esx           => 0.020710313226315787
error         => 0.020242924147768213
from          => 0.01619710913233679
one           => 0.015268056735128665

:DMC:20{n=19
Top Terms:
mount         => 0.04316045081888598
one           => 0.04260110596719707
luns          => 0.037377229409005425
i             => 0.03728213562824991
teh           => 0.03675986449707439
down          => 0.03375366076742068
iscsi         => 0.033541231076340855
disk          => 0.033080914656896936
volumes       => 0.031217919468377726
local         => 0.0301240654659855 9
```

(e): sample clustering output for Dirichlet clustering algorithm using the DistanceMeasure-Cluster Distribution model

```
view [p(view|topic_4)=0.0911299342592143
client[p(client|topic_4)=0.0894500647836
desktop[p(desktop|topic_4)=0.02845559212
users[p(users|topic_4)=0.026297100857360
vmware view[p(vmware view|topic_4)=0.017
pool[p(pool|topic_4)=0.016708393198065 12
connecting p(connecting|topic_4)=0.01510
when[p(when|topic_4=0.01454318709438394 6
desktops[p(desktops|topic_4)=0.014407894
usb p(usb|topic_4)=0.0141389852398917 19
```

(f): sample clustering output for Latent Dirichlet Allocation (LDA) clustering algorithm

Figure 4 Clustering Algorithm output comparison.

99.2% during this period. Thus it can be determined that k-means or fuzzy k-means offers better performance and scalability with comparable clustering results as discussed in the next section.

The output of the clustering algorithms is now examined as well as the accuracy of the clusters.. Figure 4 shows the output of all considered clustering algorithms for clusters including support calls related to "performance" issues. It can be noted that the results are quite similar for k-means, k-means with a pre-processing step using canopy clustering and fuzzy k-means. The number of support calls included in the clusters are comparable using all techniques. The output of the probabilistic clustering algorithms is the most applicable but given the prohibitive delays incurred with Dirichlet, the output of the LDA clustering algorithm is deemed the most appropriate with the relevant generated topics problems matched with those frequently encountered by VMware customers. However, it can be noted that current clustering output has limitations. The relevance of the output is limited. This problem is not related to the proposed system so much as the method used to generate the text vectors. At present, Mahout only supports TF or TF-IDF weighting (Owen et al. [21]). Although TD-IDF weighting was used to convert the support calls descriptions into vectors, stop words still appeared in the clustering results. While not currently supported in Mahout, more advanced techniques exist to transform text documents into vectors is required. As described in (Soucy & Mineau [22]), supervised weighting methods such as *Gain Ratio* based on statistical confidence intervals could be used to improve the quality of the text vectors and, as a consequence, the clustering results. Furthermore it is necessary to consider collocation.

Conclusions

The research presented in this paper presents a complete open source solution for processing and categorisation of similar service calls within large technical supports data sets to allow for identification of similar calls with potential for faster resolution. The solution was evaluated using a subset of VMware technical support data with the output and accuracy of the five commonly employed clustering algorithms examined. Although, this paper discusses the analysis of VMware support data in particular, the described techniques and procedures are generally applicable to other organisations providing similar services, thereby providing a proof of concept Industry framework. Future work will examine alternative text vectorisation methods to TD and TD-IDF to further improve the quality of the clustering results and to consider word collocation. Additionally, orchestration tools such as Oozie could be considered to automate the steps required to identify related support calls.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
ADB designed the study, developed the methodology, implemented the system, performed the analysis and evaluation and wrote the manuscript. AOD supervised the project. Both authors read and approved the final manuscript.

Author details
[1]VMware, Inc, Cork, Ireland. [2]Department of Computing, Cork Institute of Technology, Cork, Ireland.

Received: 14 June 2013 Accepted: 20 February 2014
Published: 24 June 2014

References

1. Digital Universe Study (on behalf of EMC Corporation) (2012) Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. In: http://idcdocserv.com/1414
2. McKinsey Global Institute (2011) Big data: The next frontier for innovation, competition, and productivity. In: http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
3. Cisco Internet Business Solutions Group (IBSG) (2011) The Internet of Things: How the Next Evolution of the Internet is Changing Everything. In: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
4. Karmasphere (2011) Deriving Intelligence from Big Data in Hadoop: A Big Data Analytics Primer.
5. Karmasphere (2011) Understanding the Elements of Big Data: More than a Hadoop Distribution.
6. Huang DW, Sherman BT, Lempicki RA (2009) Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources. Nat Protoc 4(1):44–57
7. Lavrac N, Keravnou E, Zupan B (2000) Intelligent data analysis in medicine. In: Encyclopaedia of Computer Science and Technology, vol 42. pp 113–157
8. IBM (2011) The Essential CIO. In: http://www-935.ibm.com/services/uk/cio/pdf/CIE03073-GBEN-01.pdf
9. Aberdeen Group (2010) Unlocking Business Intelligence in the Contract Center.
10. Ghemawat S, Gobioff H, Leung S (2003) The Google File System. In: SOSP '03 Proceedings of the 19th ACM symposium on Operating Systems Principles., vol 6. ACM, pp 10-10. http://dl.acm.org/citation.cfm?id=945450.
11. Dean J, Ghemawat S (2004) MapReduce: Simplified Data Processing on Large Clusters. OSDI' 04 Proceedings of the 6th symposium on Operating System Design and Implementation, San Francisco, CA Vol 6. pp 10-10. http://dl.acm.org/citation.cfm?id=1251264.
12. Apache Hadoop (2012), http://hadoop.apache.org/
13. Shvachko K, Kuang H, Radia S, Chansler R (2010) The Hadoop Distributed File System. In: IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST' 10). ACM pp 1–10. http://dl.acm.org/citation.cfm?id=1914427.
14. Bhandarkar M (2010) MapReduce programming with apache Hadoop. In: IEEE 24th International Symposium on Parallel & Distributed Processing (IPDPS' 10)
15. Apache Mahout (2012), https://mahout.apache.org/
16. Apache Hbase (2012), http://hbase.apache.org/
17. Apache Hive (2012), http://hive.apache.org/
18. Esteves RM, Pais R, Rong C (2011) K-means Clustering in the Cloud – A Mahout Test. In: IEEE International Conference on Advanced Information Networking and Applications (WAINA '11). IEEE pp 514–519. http://www.ieeeexplore.us/xpl/articleDetails.jsp?tp=&arnumber=6133195&queryText%3Desteves+clustering+wikipedia.
19. Esteves RM, Rong C (2011) Using Mahout for Clustering Wikipedia's Latest Articles: A Comparison between K-means and Fuzzy C-means in the Cloud. In: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom '11). IEEE pp 565–569. http://www.ieeeexplore.us/xpl/articleDetails.jsp?tp=&arnumber=5763553&queryText%3Desteves+K-means+Clustering+in+the+Cloud+%E2%80%93+A+Mahout+Test.
20. Ericson K, Pallickara S (2012) On the Performance of High Dimensional Data Clustering and Classification Algorithms. Elsevier Future Generation Computer Systems, Available from: http://dl.acm.org/citation.cfm?id=2435540
21. Owen S, Anil R, Dunning T, Friedman E, Manning Publications (2012) Mahout in action. Chapter 8. Shelter Island, N.Y, pp 130–144
22. Soucy P, Mineau GW (2005) Beyond TFIIDF Weighting for Text Categorization in the Vector Space Model. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI '05). ACM pp 1130–1135. http://dl.acm.org/citation.cfm?id=1642474.

doi:10.1186/2196-1115-1-1
Cite this article as: Duque Barrachina and O'Driscoll: **A big data methodology for categorising technical support requests using Hadoop and Mahout.** *Journal of Big Data* 2014 **1**:1.