Predicting Alert Source Device using Machine Learning Algorithms

Bharath M. B., D. V. Ashoka

Abstract: In a large distributed virtualized environment, predicting the alerting source from its text seems to be a daunting task. This paper explores the option of using machine learning algorithm to solve this problem. Unfortunately, our training dataset is highly imbalanced. Where 96% of alerting data is reported by 24% of alerting sources. This is the expected dataset in any live distributed virtualized environment, where new version of device will have relatively less alert compared to older devices. Any classification effort with such imbalanced dataset present different set of challenges compared to binary classification. This type of skewed data distribution makes conventional machine learning less effective, especially while predicting the minority device type alerts. Our challenge is to build a robust model which can cope with this imbalanced dataset and achieves relative high level of prediction accuracy. This research work started with traditional regression and classification algorithms using bag of words model. Then word2vec and doc2vec models are used to represent the words in vector formats, which preserve the semantic meaning of the sentence. With this alerting text with similar message will have same vector form representation. This vectorized alerting text is used with Logistic Regression for model building. This yields better accuracy, but the model is relatively complex and demand more computational resources. Finally, simple neural network is used for this multi-class text classification problem domain by using keras and tensorflow libraries. A simple two layered neural network yielded 99% accuracy, even though our training dataset was not balanced. This paper goes through the qualitative evaluation of the different machine learning algorithms and their respective result. Finally, two layered deep learning algorithms is selected as final solution, since it takes relatively less resource and time with better accuracy values.

Keywords: Fault management, unstructured data, machine learning, and event classification.

I. INTRODUCTION

Amount of unstructured data is growing rapidly, and it has outnumbered the traditional structured data format in today’s digital world. Statistical inference from this unstructured data is an important challenge in IT world. Using statistical inference is related to data-agnostic system management, which helps in automated management and monitoring in modern virtualized datacenters and their services. This work tries to address one such problem of multi class text data classification of virtualized data center alerts, based on their source device type. There are lot of application of the text classification, which typically organizes content, products, users to different class based on their stories, content, and tags it respectively. However, most of the classification example in research world and in academia are focused on binary classifications, like email spam filtering or sentimental analysis positive or negative etc. But most of the real-world problems are much more complex than binary classification. One such real world text classification example is illustrated here, which groups the datacenter alerts from distributed heterogeneous source to their originating source device. Datacenter elements (devices) are monitored using suite of applications or element managers, which track performance and fault alerts in production system. All these alerts are aggregated at one system assurance manager (SAM). These aggregated live alerts monitoring is used by Global Command Center (GCC) for 24x7 monitoring of customer datacenters. Our approach is to use the machine learning algorithm to assist the SAM with auto classification of these alerts in real time. Hence our research objective can be defined as a supervised text classification of alerts in datacenter monitoring system. Source data for this work is the last three years alerting data from SAM system. Figure-1 shows the different type of alerting sources and their aggregated alert count used in source dataset. From the figure it’s very evident that it’s unbalanced data set and there are six dominant data sources with more than 100K alerts from each of them. Figure-2 show the sample word cloud representation of this alerting text dataset.

![Figure-1. Distribution of alerts from multiple sources.](image)
This work makes two primary contributions: one is to provide an overview of the data analytics project based on the Natural Language processing techniques and second one is the qualitative evaluation of different machine learning algorithms for alert text data classification problem. We started with the traditional algorithm like Random Forest, LinearSVC, Multinomial Naïve Bayes and Logistic Regression to evaluate their relative accuracy and execution time. Then we moved to next stage of complex algorithm like word2vec and doc2vec. Which uses the vector representation of the alert text to retain the semantic meaning of the sentence. Finally, deep learning neural network algorithm is used with help of keras library with tensor-flow as backend. Which resulted highest accuracy of 99.99% with just two layers of neural networks. Following section of the paper go through the related work, basic introduction to large distributed virtual environment followed by the machine learning algorithms evaluation process and their respective results details.

II. RELATED WORKS

Here are few related research works in the field of text data classification. Jingnian Chen and Houkuan Huang [1] team have work on improving the feature selection logic during text classification using metrics like Multi-class Odds Ratio (MOR), and Class Discriminating Measure (CDM). Naïve Bayesian algorithm is an effective text classifier, but this algorithm is very sensitive to feature selection. Which significantly affect the final accuracy values, hence improvement of feature selection using metrics like MOR and CDM, will yield good results. Grigorios Tsoumakas and Ioannis Katakis research work Multi-Label Classification: An Overview [2] talks about the different multi label text classification algorithms and their literature elements. It organizes the related literatures into structure presentation with comparative study on them. This paper gives quick overview of text data classification in data mining. Durgesh and team [3] have used support vector machine (SVM) for classification effort. There work shows the comparative results of using different kernel function in SVM. It gives a quick introduction to method of using SVM for multi label classification use-case. SVM delivers better results as it doesn’t suffer from limitation of data dimensionality or with relatively less sample data for training. Eui-Hong, HanGeorge Karypis and Vipin Kumar [4] have proposed using Weight Adjusted k-Nearest Neighbor (WAKNN) for text classification problems. WAKNN learns feature weights from greedy hill climbing technique which doesn’t suffer from multi-modality of categories. It talks about qualitative comparison of WAKNN with traditional algorithms. Along with this it also presents the two performance optimization logics to improve the computational time of WAKNN. Ikonomakis [5] presents overview of text classification using machine learning algorithms. Bruno Trstenjak, Sasa Mikac and Dzenana Donko [6] talks about using TF-IDF (Term Frequency Inverse Document Frequency) with KNN (K Nearest Neighbors). This research work does the qualitative evaluation of KNN with TF-IDF on parameters like execution speed and quality of classification. They have presented the framework which enables using TF-IDF along with KNN for classification efforts. George Forman from HP labs, Palo Alto presented using Bi-Normal Separation (BNS) [7] in place of the IDF. This work shows the empirical evaluation of the text classification on data set using TF-BNS for Support vector machine (SVM). It gives overview of this approach with resulting dataset. Mita K. Dalal and Mukesh A. Zaveri [8] have presented the technical overview of semi-supervised machine learning classification logic, which categorize the documents to predefined set of buckets. This paper explains the generic strategy applied for these use-cases along with survey of some of the existing methodologies. In one of our previous research project tried finding the co-relation between virtualized data center performance metrics through statistical inference [9]. It uses regression algorithm to find any dependency between two performance metrics and then use this inference to build monitoring model dynamically. It builds the dependency relationship in large distributed virtual environment, helping administrator to visualize monitoring element dependencies. Dino Isa and team [10] presents a hybrid approach to text classification problem by combining naivist Bayes approach and Support Vector machine algorithm. Bayes formula is used to vectorize the document, this produce the probability distribution vector representation of the input document. Then at the second stage SVM algorithm is used for classification at multidimensional level. This method gives relatively better accuracy compared to using TF-IDF and SVM combination.

Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao used Recurrent Neural Networks (RNN) [11] for text data classification domain. This uses the deep learning algorithm technique without using any of the human designed features. It uses the recurrent structure to identify the contextual information, this reduces the noise considerably compared to traditional classification logics. Along with this it uses the max pooling layer to identify the key words in the document, which helps in classification efforts. Jinseok Nam and team have worked on the large scale multi label text classification using neural network [12]. They propose a simple neural network architecture which reduces the pairwise raking errors. Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau proposed new neural network model called C-LSTM [13], which is combination of both RNN and CNN architecture.

Figure-2. Sample word cloud representation for input alert text.
It uses the CNN to extract the higher-level phrases and feeds that to Long Short-Term Memory (LSTM) recurrent neural network to get the complete sentence representation. This approach captures both local features as well as the global semantics of that document. This hybrid approach outperforms the traditional neural network algorithms. Rie Johnson and Tong Zhang [14] has used CNN for text classification by using 1D structure of the text data and 2D structure for image data. They propose using bag of word model in convolution layer of CNN architecture.

III. MONITORING DISTRIBUTED VIRTUALIZED ENVIRONMENT

Our organization monitors around 90+ customer datacenter through centralized monitoring setup as shown in the Figure-3. Where each customer datacenters are connected to centralized system through VPN (Virtual Private Network) tunnel, which allow the flow of the alerts from source datacenter to central monitoring system. Each of these individual datacenter VPN connection lands on the isolated VM network on our side, to avoid cross talk between two different network configurations. These landing VPN tunnel entries are marked as “Isolated DMZ Hosts” as shown in the Figure-3. These DMZ hosts (Demilitarized zone) intern forwards the alerts/events to event aggregator service. Which then present alert to GCC console (Global command center console). On customer datacenter side each site has individual dedicated monitoring solution which collects alerts and faults traps from the devices and their respective element managers. Which inter forwards these live alerts through VPN tunnel to centralized monitoring system. Individual datacenter elements like Switch, Router, Storage arrays etc are connected to local monitoring system as shown in the Figure-3.

In this distributed environment identifying the alerting source device from its alert text is our problem domain. More specifically this can be classified as the multiclass text classification. Which can be defined as a classification task with more than two classes/buckets. It also makes an implicit assumption that each record belongs to only one group or label. Even though this specific research problem can be classified as NLP text classification issue, there were few specific idiosyncrasies related to our environment and input data that is used in the training phase. The alerting data generated in this distributed setup is highly unbalanced, and it changes over period of time as new device (or newer version of the same device) get added to this distributed setup. At the same time number of alerting source and their relative number keeps changes (from month to month) as new accounts are added and removed from this centralized monitoring system. Due to this source alerting dataset will always be highly unbalance. Any efforts to normalize this input data set will truncate the dataset from a newly added device, which have relatively less alerts records. Table-1 show our alert training dataset with count value for each alerting source device. From the table-1 it’s very evident that around 60% alerting source have less than 5K records (around 15 devices).

Table-1. Training dataset with alert count for each device type

| Sl.No. | Alert Source Device | Number of Alerts |
|-------|---------------------|-----------------|
| 1     | EMC VMAX            | 430075          |
| 2     | Cisco Switch        | 383466          |
| 3     | EMC VPLEX           | 322929          |
| 4     | EMC VNX             | 210385          |
| 5     | EMC Isilon          | 128131          |
| 6     | Brocade Switch      | 120197          |
| 7     | EMC DataDomain      | 15382           |
| 8     | EMC RecoverPoint    | 13501           |
| 9     | Array               | 6906            |
| 10    | EMC VNXFile         | 4928            |
| 11    | vCenter             | 4693            |
| 12    | EMC XtremIO         | 2971            |
| 13    | SRM Health          | 2780            |
| 14    | Host                | 2287            |
| 15    | Chassis             | 1623            |
| 16    | EMC ECS             | 1015            |
| 17    | EMC ATMOS           | 883             |
| 18    | SAN Switch          | 728             |
| 19    | Port                | 584             |
| 20    | NetApp              | 568             |
| 21    | EMC Centera         | 400             |
| 22    | EMC Unity           | 297             |
| 23    | IBM                 | 30              |
| 24    | ViPR                | 4               |
| 25    | Unknown             | 2               |

Hence our challenge is to build a machine learning algorithm which can predict alerting source with relatively high accuracy in spite of this imbalanced dataset. Second challenge is that, model need to be retrained on regular basis like on monthly basis.
Predicting Alert Source Device using Machine Learning Algorithms

So that any new device or alert type added to this system can be handled by this algorithm. Hence its desirable to have algorithm which can be easily retrained (take less time during training phase). These are the two primary constraints that need to be taken care by our proposed solution.

IV. PROPOSED SYSTEM

Our organization has a centralized monitoring system, which collects events/alerts/faults from multiple customer datacenter. These alerts are manually grouped (or classified) by the GCC team. Since this is a manual task it is prone to errors. Adding to this problem is that, most of the team members in GCC are relatively less experienced (freshers or new college hires). Hence, we regularly see the issue of wrong classification – which adds to the delay in processing that issue as it gets routed to wrong group. In order to address this issue, we tried automating through rule-based solution. But the problem with that approach is that- they get outdated very quickly (once in every six months) as you add new device type or upgrade existing device firmware’s in customer datacenters. As new alerts or new version of monitoring solution are used at datacenter our rule becomes redundant. Finally, our focus turned towards the machine learning model to address this problem. Our approach is to use last three years alert data to train the model and used it to help the GCC to auto classify the alert by its source. Figure-4 shows the abstract dataflow architecture of this proposed system. Whereas Figure-5 represent the production deployment architecture.

Figure-4. Proposed phase-I architecture for Alert source Classification.

This new proposed solution uses the machine learning algorithm to predict the alert source. Where model is trained based on the previous alerting data. In order to include any newly added device alerts to this model. We have automated retraining of our model once in a month. This will make sure that our model should be able take care of any new alert(s) or device type(s) addition at the custom datacenter. Hence this model becomes self-trained solution which can sustain in long run. Along with this additional check have been put in place to make sure that model prediction rate doesn’t deteriorate over period of time. If so, it will be retrained again with new data set to update the model with incoming new alerts in our setup.

Figure-5. Proposed phase-II architecture for Alert source prediction.

A. Proposed Algorithm for Alert Source predication

The proposed algorithm is briefly described here. This algorithm takes last three years alert data in archive logs as input and output will be machine learning model to predict the alert source in large distributed virtualized environment. Here we briefly explain the algorithm steps followed by sudo code. Table-2 list all the symbols used in the algorithm sudo code. At the end of this section briefly describes the different machine learning techniques used in multi class text data classification. There are two phases in this proposed solution first we train the model using historical data and then second phase uses the model built in first phase for predicting the alert source in production (live environment). Let’s start with training phase first.

1. Read the archive logs files for training phase
2. Extract alert text, source details and ticketing information from alert data. Each alert will be tagged with source details by using ticketing and other additional supporting details from the archive log.
3. Normalize the data by removing noise content in alert text. (purges all the IP addresses, Hex decimal values, multiple white space, special characters etc)
4. Python pandas library data-frame are used for in-memory data source and cleanup activity.
5. From data frame extract the features (bag of words) using term’s frequency (TF) and its inverse document frequency (IDF)
6. Split the data into training and test data set
7. Create a machine learning models to predict the source by event text.
8. Test the model with LinearSVC, LogisticRegression, MultinomialNB, RandomForestClassifier, Word2Vec and Doc2Vec using LogisticRegression. Finally, with two-layer neural network architecture using Keras libraries. Each of these algorithms are evaluate for their relative performance and time complexity values.
9. Finally select the model with best results.
10. Use this model to build the predicting phase of our solution.

One of the critical steps in the training phase is the text preprocessing (step-3 in proposed algorithm). Once data set is imported, input text data need to be preprocessed. Contrary to traditional belief in any real-world project most of time and effort is spent on this preprocessing phase. Actual model building phase looks trivial compared to this data preparation phase.
In put alert text need to be normalized by removing all special character, IP address, hexa decimal numbers and unwanted spaces etc. Then we remove all single characters in words like Switch’s, first punctuation character is replaced with single space then input word “Switch’s” becomes “Switch s”. This single character doesn’t help in our model building process rather it added unwanted noise. So we purge them using python regular expression “^[a-zA-Z]+$”. resulting multiple space will be removed in later stages. During this stage text data lemmatization is done. This avoid creating feature data which are semantically similar (like devices or device, routers or router etc) but syntactically different (device and devices are considered as two different features). Feature data which are semantically similar (like devices or device, routers or router etc) but syntactically different (device and devices are considered as two different features). This will improve the accuracy and substantially reduces the noise. In Step-5 bag for world are created using TfidfVectorizer in python text library. In our conversion min_df is set as 25 mean minimum 25 different alert records.

Table-2. List of Symbols used in the proposed algorithm with details

| Sl. No | Symbol(s)         | Description |
|--------|-------------------|-------------|
| 1.     | $\varepsilon_{\text{Record}}$ | Event Records |
| 2.     | $L_i(\varepsilon)$ | List of Event Records |
| 3.     | $\beta_{\text{w}}$ | Bag of word dictionary |
| 4.     | $\rho_{\text{model}}$ | Machine learning model |
| 5.     | $a_{\text{txt,are}}$ | Alert/Event Text |
| 6.     | $\Omega^i_1$ | Event Text normalization function |
| 7.     | $\beta_{\text{av}}$ | TFIDF vector |
| 8.     | $D_{\text{Train}}$ | Training Data Set |
| 9.     | $D_{\text{Test}}$ | Testing Data Set |
| 10.    | $\Psi(\text{Selected model})$ | Fitting machine learning model with our 8wi and 8wi vectors |
| 11.    | $\rho_{\text{Model}}$ | Model accuracy evaluation |
| 12.    | $a_{\text{src}}$ | Alert source predicated using model |

Algorithm-I: Training Phase

Input: $L_i(\varepsilon)$ List of alerts records

Output: $\beta_{\text{w}}$, Bag of words and $\rho_{\text{Model}}$

Method Phase-I: Training the Algorithm

for each $L_i(\varepsilon)$ where $1 \leq i \leq n$ where $n$ is total number of event records

$a_{\text{txt,are}} \leftarrow L_i(\varepsilon)_{\text{Record}}$ for each record extract the event text and source details

for each $a_{\text{txt,are}}$ in $L_i(\varepsilon)$

$a_{\text{stop}} \leftarrow \Omega^i_1$

if ngram=(2,3) \&\& min_diff $\geq 25$

if $a_{\text{w}}$ != ‘English’

$\beta_{\text{av}} \leftarrow \beta_{\text{av}}(L_i(\varepsilon))$

end

end

end

$D_{\text{Train, Test}} \leftarrow \text{Spilt} (L_{i}(\varepsilon),(0.7,0.3))$

$\rho_{\text{Model}} \leftarrow \Psi_{\text{LSTM}}$ FIT ($D_{\text{Train}}$)

$\rho_{\text{Model}} \leftarrow$ Evaluate($\rho_{\text{Model}}$

Algorithm: Training phase ends

This second phase of the algorithm explain the source predication logic when it deployed in production. It directly gets the events text data from the smarts console which is passed as an input to the model and expected output is the predicted source domain name. This part is shown in the data flow architecture diagram Figure-5.

Algorithm-II: Predicting Phase

Input: $\varepsilon_{\text{Record}}$ alerts records with event text details (cleaned input data)

Output: $a_{\text{txt,src}}$ Predicted source value from $\Psi(\text{Selected Model})$

Method Phase-2: Predicting Alert Source

for each $L_i(\varepsilon)_{\text{Record}}$

$a_{\text{txt,src}} \leftarrow L_i(\varepsilon)_{\text{Record}}$

$a_{\text{txt,src}} \leftarrow \Omega(\text{txt,src})$

$a_{\text{txt,src}} \leftarrow \Psi(\text{Selected Model}, \text{PREDICT}(a_{\text{txt,src}}))$

Algorithm: predicting phase ends

B. NLP Vectorization techniques

NLP-Natural Language processing is stream of machine learning techniques, which helps in using human language text in statistical model. Hence it acts as bridge between human text and computers. Most of the statistical algorithm cannot handle the text data, hence they are transferred into numerical vectors. First step in any NLP project is this conversion of text data to vectorized feature set. There are several different methods for this conversion. Some of the commonly used techniques are briefly explained in this section.

Bag of Words: Is one of the most popular models for text feature representation. It represents the work in the form of a binary vector. Which is a representation of occurrence count of any word in input vocabulary list. It’s easy to use, but it doesn’t preserve the semantic meaning of related words in binary vector.
Predicting Alert Source Device using Machine Learning Algorithms

**Harsh:** a generic hashing function convert the arbitrary input text data to fixed size vector set. Same sequence of text string result in same hash function test result. Hash clash are common problem and it’s not easy to scale as the input text size increase. But relatively easy to implement for small input dataset.

**IT-IDF:** Term Frequency-Inverse Document Frequency rescales the frequency count of any word with respect to how often it appears in all documents. Term Frequency assign score of frequency count to each word in current document corpus. Whereas Invers Document Frequency finds out how rare this word across the corpus. With this approach it selects words, which are relatively rare, but they can act as selected feature set, which easily differentiate each documents or group of text.

**Word2vec:** Converts the sequence of words into vector formation, thus preserving semantic meaning and relationship of words. Same sequence of words results in same vector, thus maximizing the probability of observing a word in a context. Both continuous bag of words (CBOW) and continuous skip gram methods try to predict the context form presence of a word. Word2vec is relatively complex and computation take more time, but relatively accurate.

**Doc2vec:** It is similar to Word2vec and uses the network of paragraphs and words. It uses the Distributed Bag of words (DBOW) like skip-grams in word2vec. A sentence can be thought as sequence of words in a document.

**CNNs:** Is several sequential layers of convolutions with nonlinear activation (trained by input data) and applied to produce to final results. Convolutions are result of local neuron connection, where each region of the input is connecting to resulting output. Each successive layer applies different set of filters and combination of their result. Originally neural network was built for computer vision and with enhancement of easily library wrapper for neural network enable their use for other fields like text categorization etc.

V. EXPERIMENTAL SETUP AND RESULTS

The project started with extracting last three years alerts data from the smarts global console archive logs. Initial dataset size was around 1.6 million records (exactly 16,54,765 alerts). Normally any classifiers algorithm (or any machine learning algorithms) cannot process the RAW text records in their native form. Most of these algorithms expect the fixed size numerical feature vector, compared to regular text data which is of variable length. Hence these RAW event records are converted to more manageable dataset. Out of 26 attributes of the alerts data only 4 columns (or attributes) are used for this research. They are event class name, instance name, event type and event text. Mainly event text data is our main source of text input to the model and expectation is to predict its source.

There were three difference phases with seven different models in this research project. We started with traditional machine learning classifier algorithms then moved on to word2vec and doc2vec feature representation and finally neural network is used for text classification use-case. Simple two layered neural network is built using keras libraries. In this section will explain each of these three phases along with their respective results values. Figure-6 shows the histogram of alert text length data, from the figure it’s very evident that most of the alert text had 150 to 300 words, with every few alerts having up to 350 words. This gives a quick overview of the input dataset used in our experiment.

The first phase of the project started with bag of words model (along with traditional machine learning classifier algorithms) to extract the useful features set from the text input data. Presence of a group of words (frequency of occurrence) is consider without any order dimensionality considerations. Occurrence of order of these group of words are ignored while calculating the frequency count. We used ngram of 2 and 3 (bigram and trigram), that mean group of two- or three-words occurrence in an alert text. For each term in our dataset, algorithm calculates the term frequency and inverse document frequency normally referred as TI-IDF. We used sklean python library TfidfVectorizer class function for these calculations. Random Forest, Multinomial Naive Bayes, Logistic Regression and LinearSVC algorithms are evaluated with TI-IDF vectors. It’s important to note that these algorithm accuracies drastically improve as the initial text cleanup procedure matured over the course of this project. Among the four LearnSVC emerged as clear winner with 99% accuracy. Table-3 and Figure-7 shows these values.

![Figure 6. Alerts text length histogram plat.](image)

After carefully analyzing our results in Table-3 it’s very evident that our bigram and trigram uniquely identify their source. We transformed our RAW alert text into vector of numbers using TF-IDF weight vectors. After having this vector representation, one can easily train any supervised classifier. We train the model to predict the source of any alert text. Out of four classifier algorithm LinearSVC emerged as clear winner with 99% accuracy (as highlighted in the Table-3). Here is the accuracy rate for our alert source prediction dataset for large virtualized environment. Table-3 show the model name and their respective accuracy rate as per the f1-score values. Confusion metric for some of these models are presented in the following section.
VI. RESULT AND DISCUSSION

This section briefly discussed the model evaluation number by looking at confusion matrix and f-score test results. Due to space constrain only four out of seven algorithm testing results are shown in this section. Starting with phase one classifier algorithm, Linear SVC yielded highest accuracy of 0.99995. Figure 8 shows the classification test results for the LinearSVC. Except the two unknow alert record LinearSVC had almost 100% hit rate for most of the other sources. Whereas the Figure 9 shows the confusion matrix for test dataset. It is interesting to observe that few alerting sources have minor false positive result for devices like Array, Cisco Switch and EMC XtremIO. On a similar note Figure 10 shows the f1-test scores for the Logistic Regression algorithm. This achieves the prediction accuracy rate of 0.999987. From the test score one can notice that IBM device have a lowest score of 0.91 and most of the other device type reaches near to 1.0. Figure 11 shows the confusion matrix for the Logistic Regression algorithm.

Table-3 Modeling accuracy details

| Sl. No. | Model Name               | Accuracy | Avg. CPU | Avg. Memory | Execution Time (minutes) |
|--------|--------------------------|----------|----------|-------------|--------------------------|
| 1      | RandomForestClassifier   | 0.912390 | 18 %     | 16 GB       | 320 minutes              |
| 2      | MultinomialNB            | 0.999659 | 22 %     | 17.5 GB     | 380 minutes              |
| 3      | LogisticRegression       | 0.999987 | 18 %     | 16 GB       | 355 minutes              |
| 4      | LinearSVC                | 0.999995 | 20 %     | 15 GB       | 290 minutes              |
| 5      | Word2Vec                 | 0.999981 | 24 %     | 28.5 GB     | 430 minutes              |
| 6      | Doc2Vec                  | 0.999959 | 25 %     | 32 GB       | 460 minutes              |
| 7      | Keras Sequential         | 0.999999 | 78 %     | 16 GB       | 38 minutes               |

Table-4. Sample alert data with its predicted values from ML model.

| Sl. No. | Alert Test from different source | Expected results | Prediction Values |
|---------|----------------------------------|------------------|-------------------|
| 1       | watch net agg event              | EMC VNX          | EMC VNX           |
| 2       | srm event clm emc vplex virtualstore clm vplex clm wrtlattency delta flat latem vplex cluster wretlattency delta hortlandia ibm tag tag sm vplex perf alert vplex cluster wretlattency delta | EMC VPLEX | EMC VPLEX |
| 3       | watch net agg event file switch port fail port linkfailures mhe srm fileswitch port linkfailures east windsor brocade switch threshold port link failures | Brocade Switch | Brocade Switch |
| 4       | watch net agg event symmetry controller synchrony financial srm vnxma director currentutilization dallas vnxma director currentutilization | EMC VMAX | EMC VMAX |
| 5       | watch net agg event c cisco switch port c d e synchrony financial srm fileswitch port linkfailures dallas cisco switch threshold port link failures | Cisco Switch | Cisco Switch |
| 6       | watch net agg event caa event srm data domain disk caax disk diskstatus flat emea datadomain disk status milan datadomain disk status | EMC DataDomain | EMC DataDomain |
| 7       | watch net agg event phsua emc isilon node cpu phsua node node novartis srm node cpu util quot boston phsua isilon data cpu current util quot | EMC Isilon | EMC Isilon |
| 8       | watch net agg event wtgbtemcavg emc vnxfile vnxfile replication wtgbtemcavg vnxfile replication files _dell_rep availability dell amex srm vnx file replication availability ctc vnx file replication availability | EMC VNX File | EMC VNX File |
| 9       | watch net agg event apg ip pgpm inter intern smrhealth mysql ap pgpm inter intern mysql ap qquerycount pgpm srm mysql rarewiralarms in alert database srm queries | SRM Health | SRM Health |
| 10      | watch net agg event recovpoint rpa rpa status vnxstream srm rpa status vienna prop mithirdparty rpa status | EMC RecoverPoint | EMC RecoverPoint |
| 11      | watch net agg event mdc pod saas chassis server host interface mdc pod saas chassis server host interface dce interface operationalstatus csm od operators interface dce interface operationalstate csm od operators interface dce interface operationalstate pd csm od operators | Host | Host |
| 12      | watch net agg event undefinability mapp volume quotafiltered quota event status | NetApp | NetApp |
| 13      | watch net agg event minerror mapp volume quotafiltered quota event status | EMC Unity | EMC Unity |
| 14      | watch net agg event denu par array controller denu par controller write percent nor par par controllers high write percent | Array | Array |
| 15      | watch net agg event rdpupcra fave ea ventter processor rdpupcra fave ea processor currentutilization csm hig cpu utilization rdc vcenter high cpu utilization | vCenter | vCenter |
| 16      | srm event vnx device device symmetric device device state has changed online degraded chrysler device_status detroit vmax traps devices device status changed | EMC VMAX | EMC VMAX |
| 17      | watch net agg event clm bov brocade switch port chs wot dce port ologgess novartis srm fileswitch port losossignal boston brocade switch threshold port losossignal | Brocade Switch | Brocade Switch |
| 18      | watch net agg event vplex frm emc vplex port vplex frm chlcovplex director port eth availability csmis srm vplex port availability acdc vplex port availability | EMC VPLEX | EMC VPLEX |
| 19      | watch net agg event fileoan cisco switch port fileosan port availability csmis srm cisco port statuschange acdc cisco port statuschange | Cisco Switch | Cisco Switch |
| 20      | watch net agg event flxinem emc vnx block port flxinem port availability csmis srm vnx block port statechange acdc vnx block port statechange | EMC VNX | EMC VNX |

Figure-7. Accuracy rates of the classifier algorithms.
Figure-8. Classification report for difference source by Linear SVC

Figure-9. Confusion matrix for Linear SVC

Figure-12. Shows the classification test results for the Doc2Vec mapping with Logistic Regression algorithm. Most of the alerting source have f1-scare as 1.00 or 0.99. This approach gives relative high accuracy results, but its computational complexity is very high. On 8 processor with 32 GB RAM machine it took nearly 5 hours to complete the initial training and testing phase. It also demands very high resource utilization in terms of memory, we observed that during model fitting stage it utilized almost 28 GB of memory on our test setup. On the similar note Figure-13 shows the confusion matrix for Doc2Vec logic. From the figure it’s evident that few array types did not had 100% match like Isilon, VMAX, Array, SRM Health, Chassis etc had few false positive results.

Figure-10. Classification report for different alerting source using Logistic Regression

Figure-11. Confusion metric for Logistic regression algorithm

Figure-14 shows the confusion matrix for the two-layer neural network. It’s important to note that our simple neural network had almost 100% prediction for each class which is highlighted in the diagram axis in the confusion matrix. Here is the sample list of alerts that were predicted by our neural network model. Due to space limitation only small subset of resulting prediction are shown in Table-4 to demonstrate the heterogenous nature alert text in our sample data. Table-4 lists alert text data (cleaned alert text data) and model predicted values as source, along with its expected results which is the right classification for that input alert text.
Figure-12. Classification report for difference source by Doc2Vec

| Source          | Precision | Recall | f1-score | Support |
|-----------------|-----------|--------|----------|---------|
| Cisco Switch    | 1.00      | 1.00   | 1.00     | 3057    |
| Brocade Switch  | 1.00      | 1.00   | 1.00     | 7516    |
| Brocade S4      | 1.00      | 0.99   | 0.99     | 534     |
| EMC VMAX        | 1.00      | 1.00   | 1.00     | 137552  |
| EMC Unity       | 2.00      | 1.00   | 1.00     | 466     |
| EMC VPLEX       | 1.00      | 1.00   | 1.00     | 206     |
| EMC VNX         | 1.00      | 1.00   | 1.00     | 491     |
| EMC RecoverPoint| 1.00      | 1.00   | 1.00     | 5688    |
| EMC VERSIPLEX   | 1.00      | 1.00   | 1.00     | 11254   |
| EMC DataDomain  | 1.00      | 1.00   | 1.00     | 4456    |
| SAS Switch      | 1.00      | 1.00   | 1.00     | 37496   |
| IBM Health      | 1.00      | 1.00   | 1.00     | 85404   |
| IBM BCS         | 1.00      | 1.00   | 1.00     | 1979    |
| IBM ADVICE      | 2.00      | 1.00   | 1.00     | 215978  |
| IBM XIVendio     | 1.00      | 1.00   | 1.00     | 936     |
| VCServer        | 1.00      | 1.00   | 1.00     | 995     |
| NetApp          | 1.00      | 1.00   | 1.00     | 11      |
| Classis         | 1.00      | 1.00   | 1.00     | 257     |
| Port            | 1.00      | 1.00   | 1.00     | 109     |
| Net            | 1.00      | 1.00   | 1.00     | 257     |
| IBM Centers     | 1.00      | 0.99   | 1.00     | 966     |
| VMware          | 0.99      | 0.99   | 0.99     | 806600  |
| IBM            | 1.00      | 1.00   | 1.00     | 1       |
| Unknown         | 1.00      | 1.00   | 1.00     | 1369    |
| micro avg       | 1.00      | 1.00   | 1.00     | 806600  |
| macro avg       | 0.96      | 0.96   | 0.96     | 806600  |
| weighted avg    | 2.00      | 1.00   | 1.00     | 806600  |

VII. CONCLUSION

Our research work demonstrates that one can use machine learning algorithm to predict the alerting source in large virtualized environment. This research work experimented with all popular natural language processing technique for text data classification. From the results its very evident that machine learning algorithm can be easily used in an environment with heterogeneous alerting source which has imbalanced data set. One of the important realization observed during this research work was that, ML model accuracy is directly related to quality and sophistication of our preprocessing and cleaning stage. As the cleaning logic mature our prediction accuracy also increased. Even simple regression algorithm resulted better prediction accuracy.

Eventually simple neural network built using keras is selected as final model as it demands less computation resources and completes training phase much faster than the other traditional models. This source predication automation has replaced the current manual mapping method in production GCC process. Currently system is under performance monitoring and we have observed around 98% accuracy rate as system matures over period of time with more training data set added to model. This automation has reduced the downstream side effects of wrong classification in production system substantially. As future enhancement more, advanced algorithms like BERT (Bidirectional Encoder Representations from Transformers) or FastText algorithms can be explored for this problem domain.

REFERENCES

1. Jingnian Chen, Houkuan Huanga, Sheng feng Tiana and Youli Qua "Feature selection for text classification with Naïve Bayes" Expert Systems with Applications Volume 36, Issue 3, Part 1, April 2009, Pages 5432-5435
2. Grigorios Tsoumakas and Ioannis Katakis "Multi-Label Classification: An Overview" International Journal of Data Warehousing and Mining (IJDWM) Volume 3, 2007 Page 13
3. Durgesh K. Srivastava, Lekha Bhambhu "DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE" Journal of Theoretical and Applied Information Technology Volume 12, 2010, Page 1
4. Han EH, Karypis G, Kumar V. (2001) Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. In: Cheung D., Williams G.J., Li Q. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2001. Lecture Notes in Computer Science, vol 2035. Springer, Berlin, Heidelberg
5. M. IKONOMAKIS, S. KOTSIANTIS and V. TAMPAKAS "Text Classification Using Machine Learning Techniques" WSEAS TRANSACTIONS on COMPUTERS, Issue 8, Volume 4, August 2005, pp. 966-974
6. Bruno Trstenjak, Sasa Mikac and Dzenana Donko, "KNN with TF-IDF based Framework for Text Categorization" Procedia Engineering Volume 69, 2014, Pages 1356-1364
7. George Forman "BNS feature scaling: an improved representation over tf-idf for svm text classification" Proceeding CIKM ’08 Proceedings of the 17th ACM conference on Information and knowledge management, California, USA — October 26 - 30, 2008 Pages 263-270
Predicting Alert Source Device using Machine Learning Algorithms

8. Mita K. Dalal and Mukesh A. Zaveri "Automatic Text Classification: A Technical Review" International Journal of Computer Applications Volume 28, Page 37-40, August 2011

9. Bharath M.B., Ashoka D.V. (2017) Algorithm to Find the Dependent Variable in Large Virtualized Environment. In: Satapathy S., Prasad V., Rani B., Udgata S., Raju K. (eds) Proceedings of the First International Conference on Computational Intelligence and Informatics. Advances in Intelligent Systems and Computing, vol 507. Springer, Singapore

10. Dino Isa, Lam H. Lee, V.P. Kallimani, R. RajKumar “Text Document Preprocessing with the Bayes Formula for Classification Using the Support Vector Machine” Published in: IEEE Transactions on Knowledge and Data Engineering (Volume: 20, Issue: 9, Sept. 2008) Page(s): 1264 – 1272

11. Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao "Recurrent Convolutional Neural Networks for Text Classification" Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Feb 2015, Page 2267 – 2273

12. Nam J., Kim J., Loza Mencía E., Gurevych I., Fürrnkranz J. (2014) Large-Scale Multi-label Text Classification — Revisiting Neural Networks. In: Calders T., Esposito F., Hüllermeier E., Meo R. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2014. Lecture Notes in Computer Science, vol 8725. Springer, Berlin, Heidelberg

13. Chunting Zhou, Chonglin Sun, Zhiquan Liu, Francis C.M. Lau "A C-LSTM Neural Network for Text Classification" Proceedings of International Joint Conferences on Artificial Intelligence, Nov 2015.

14. Rie Johnson, Tong Zhang "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks" Proceedings of International Joint Conferences on Artificial Intelligence, March 2015.

AUTHORS PROFILE

Bharath M. B. is working as Senior Solution Architect in Dell EMC Bangalore, He received his M.Tech in computer science and engineering from VTU Belgaum. His fields of interest are Operating System, Resource Virtualization and Knowledge management, Machine Learning, NLP etc.

Dr D. V. Ashoka is working as Professor in the Department of Information Science and Engineering at JSS Academy of Technical Education, Bangalore. He received his M.Tech in computer science & Engineering from VTU and Ph.D degree in Computer Science and Engineering from Dr. MGR, University, Chennai. He has 25 years of academic and research experience. His fields of interest are Requirement Engineering, Operating System, Computer Organization, Software Architecture, data mining in image processing and Cloud Computing.