International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 6 (2015) pp. 14319-14335
© Research India Publications
http://www.ripublication.com

Distributed Text Mining - An Approach To Sentiment Analysis

K. Shyamala
Associate Professor, Dr. Ambedkar Government College, Chennai, India
shyamalakannan_2000@yahoo.com
K. Vijay Kumar
Centre for Development of Advanced Computing, Chennai, India
vijaydharshan@gmail.com

Abstract

Data mining for Knowledge Discovery in Databases (KDD), an interdisciplinary field is required for gleaning insights out of heterogeneous datasets. This paper focuses on Text Mining, one such subfield of KDD; which has gained popularity with burgeoning usage on social media and sustenance of various formats of documentations viz., PDF, XML, HTML, DOC, etc. In this paper, we focus on the Distributed Text Mining on large corpora to showcase sentiment analysis on a movie review entity. The main target of this paper is to showcase the implementation aspects of opinion mining on distributed environment that involves choosing heuristics to discover the sentiment rating on a scale of 5. During the course of this paper, we provide cursory aspects on implementation of Sentiment Analysis to glean insights from movie review entity using Hadoop, RHadoop and its components to leverage distributed computing. The corpus dataset is stored in HDFS, processed using MapReduce and visualized using R via the tm [3, 4] package.

Key Words: Distributed Text Mining, RHadoop, Clustering, Distributed Computing

Introduction

The growing IT industry with pervasive computing in all the fields of human utility has pushed the need to contemplate on growing datasets in [9] Volume, Variety and Velocity; framed together as Big Data. The internet is Ubiquitous; the data is getting flooded from a variety of applications viz., log files, sensory, imagery, telemetry, click streams, social networking, etc. With the proliferation of electronic devices, the major chunk of textual information such as e-books, e-papers, emails, blogs, web pages etc. require the classical data mining of texts to more sophisticated formulations.

Text Mining is the process to glean insights from natural language text. Text is unstructured and amorphous and mining of this voluminous text is constrained to be processed in a single CPU processor, RAM and Storage [4]. To gain performance and reliability, the distributed processing and distributed computing in shared nothing architecture has become the norms. Text analytics term, is now used more frequently in business settings while the term text mining was used in some of the earliest application areas such as life-sciences research and government intelligence. Text mining uses statistical and machine learning methods to derive knowledge or extract patterns from large unstructured text datasets to detect lexical or linguistic usage patterns out of it [1]. Text mining basically is to convert amorphous datasets into a structured format based on term frequencies and subsequently apply heuristic techniques based on the applications.

In this paper we utilize the open source software R, an environment for statistical computing and graphics. Recently R has gained explicit text mining support via the tm [3] package. This package provides the infrastructure for sophisticated methods for document handling, filters, transformations, and data export. However, the availability of very large and always growing text corpora poses new challenges for efficient handling of these data sets mainly due to memory restrictions and architectural performance limits of single processor environments. On the other hand, we observe an increasing availability of multi-core architectures even in commodity computers and high performance computing environments, such as distributed and highly integrated computing clusters. In this context, we propose to make use of a technique called Map-Reduce [5] which is widely used in high performance computing because of its functional programming nature. With the features made available in the tm package, it is possible to add new layers to support this kind of parallelism and distributed allocation. We can tackle the compute-intensive nature of text mining by breaking up the corpora into smaller parts or entities for parallel processing made popular by recent advances in big data. A key factor in large scale text mining is the efficient management of data. Therefore, we utilize Hadoop to show how a distributed storage system such as Hadoop Distributed File System (HDFS) can be utilized to facilitate parallel processing of large and very large data sets of text data. This approach offers us a reliable, flexible, and scalable high performance computing solution for distributed text mining. This paper provides cursory aspects on implementation of sentiment analysis on the entity Movie review corpus sourced from the Rotten Tomatoes site [17] on the measurement of quality of filmed entertainment. The objective is to glean sentiment score on a scale of 5. This paper is organized into sections; Second section describes the related works in this area. Hadoop framework closely knitted with distributed file system is described in third section. A fourth section describes R integrated with Hadoop to leverage distributed computing. The fifth section elucidates and discusses our experimental work.

Related Work

There is a substantial amount of research done in the field of Text Mining. Ian H. Witten et al [4] explained text mining over data mining and how it closely embraces

Natural Language Processing. He provided a methodology to text summarization and document retrieval using language identification and authorship ascription—and a third—identifying key phrases—further discusses the extraction of structured information, both individual units or "entities" and structured relations or "templates."

Daniel Sonntag in 2004 [18] explained text mining components and text mining challenges in the distributed Data Mining Grid architecture. The complexity involved in scenarios like Distributed file access, distributed classifiers with the same classifier model and distributed learning was explained in a distributed grid architecture.

White paper on Revolution Analytics [12] elucidates deliverance of optimized statistical algorithms for the three primary data management paradigms to address scalability, heterogeneous data including file-based, MapReduce or In-Database Analytics. Open source R is memory-bound and is basically not built for Big Data Analytics. A marriage of Open Source R with Hadoop complements each other and is best fit for scalabilty, parallel processing computation and availability [11].

Stefan, Ingo Feinerer and Kurt Hornik detailed tm package in R called tm.plugin.dc [3] implementing distributed corpus class which can take advantage of Hadoop MapReduce library for large scale text mining tasks.

Hadoop Framework

Apache Hadoop, an open source software framework provides the most popular general-purpose computing platforms that can be used to solve Big Data related issues. We have already excelled in the Relational Database Technology, the solutions provided by Hadoop Framework are highly acclaimed to be reliable, scalable and optimized for distributed computing environment. We will be discussing the powerful computational platform, Hadoop, that is well suited for the creation of a new generation of powerful data science and enterprise applications, to leverage scalable distributed storage and MapReduce processing.

Hadoop stores and process large data sets by splitting the data set into equal chunks, and processes in parallel across a number of commodity hardware, thus leveraging horizontal scalability [13]. Hadoop Distributed File System (HDFS) with the default block size 128 MB manages and distributes the blocks amongst the nodes in the cluster. For processing the data, the Hadoop Map/Reduce ship code (specifically Jar files) to the nodes that have the required data, and the nodes then process the data in parallel. This approach takes advantage of data locality [2]. Though Hadoop is not a comprehensive solution to all problems, it is open source, popular, and can be deployed on a virtual machine cluster of a private cloud, thus providing flexible distributed big data processing and storage ability in our services platform. File I/O operations over it on client requests. The figure 1 shows pictorial representation of the implications involved when a client requests.

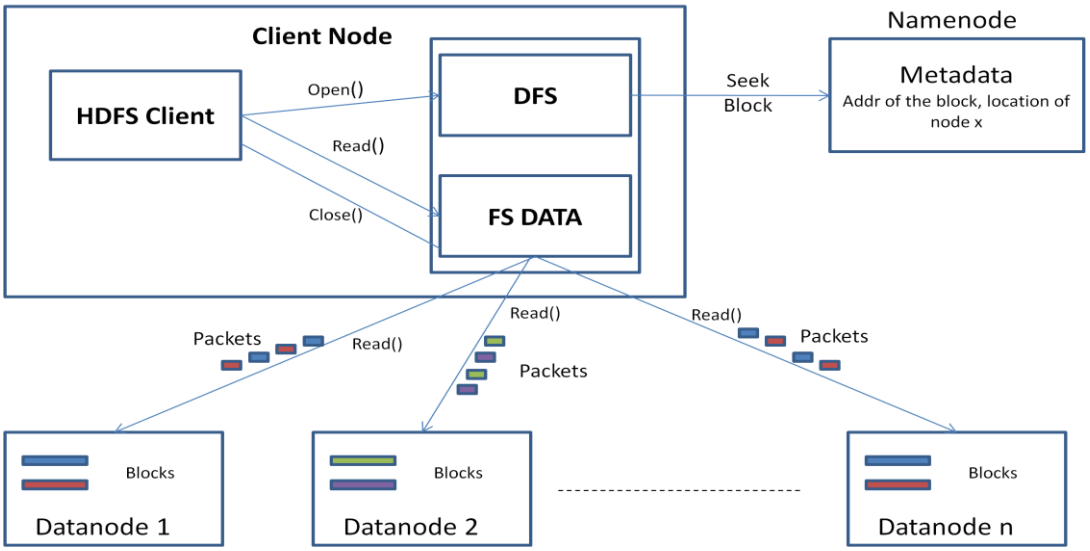

Figure 1: HDFS Architecture

To give a brief overview on the intricacies involved when a client request is made. The user request is fulfilled by HDFS client thru file system API and I/O operations. Namenode, is the masternode of HDFS cluster containing FS Image having metadata information to access blocks in the datanodes (slave machines) which stores and processes actual physical data. The status of the job process is updated to Namenode by the block reports sent by data nodes using heartbeat.

As in the above figure, HDFS client sends request for file read operation. The *Open() method* initiates Distributed File System for a read request. HDFS interacts with Namenode to get the metadata. The Namenode returns the sorted address of Datanodes based on the proximity of Datanodes thus giving weightage to data localization. Distributed File System (DFS) returns an FSData (InputStream). Client calls *Read ()* in DFSInputStream that uses a wrapper to manage I/O operations over Namenode and Datanode. Block reader contains the following information

- Block ID.
- Data start offset to read from.
- Length of data to read.
- Client name.

Data is streamed from the Datanode back to the client in the form of packets, this data is copied directly to input buffer provided by client. The DFS client reads and performs the checksum operation and then updates the client buffer. *Read ()* is iteratively called till the end of a block is reached. When the client has finished reading, it will call *Close ()* on FSDataInputStream to close the connection. The fault tolerance is inherently managed by replicating the data in multi-system. When Datanode is down during reading or DFSInputStream encounters an error during communication, DFSInputStream will switch to next available Datanode where replica can be found. DFSInputStream remembers the Datanode which encountered an error so that it does not retry them for later blocks.

As you can see that client with the help of Namenode gets the list of best Datanode for each block and communicates directly with Datanode to retrieve the data. Here Namenode serves the address of block location on Datanode rather than serving data itself, which could become the bottleneck as the number of clients grows. This design allows HDFS to scale up to a large number of clients since the data traffic is spread across all the Data nodes of clusters.

R and Hadoop

R, an open-source software suite can well handle solving complex data analysis, statistical computation with high resolution graphic visualization. R itself does not have a big data processing capacity [10][13]. Combining the advantages of R and Hadoop, RHadoop, a software package features Big Data capabilities with strong data analytics and visualization. The software RHadoop integrates R with the key components MapReduce, HDFS and HBase. RHadoop was developed by RevolutionAnalytics [11]. With the tm package in R we can mine diverse and large text data sets [12][13]. Figure 2 depicts the communication flow between R and Hadoop [13].

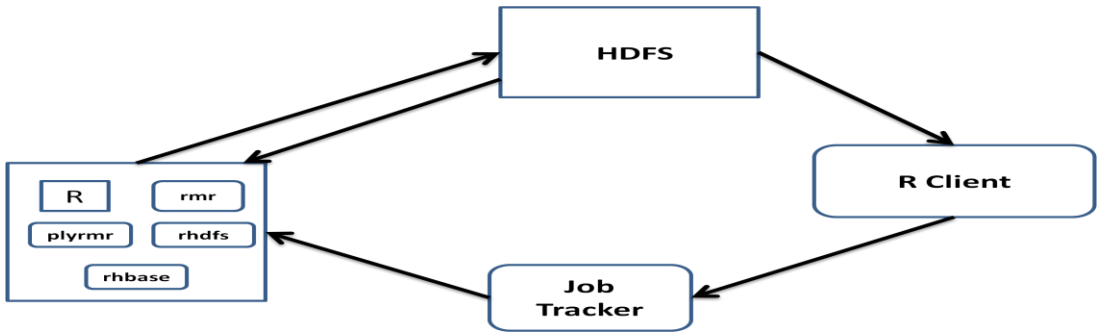

Figure 2: Communication flow between R and Hadoop

It consists of four essential packages:

- plyrmr: this R package enables the R user to perform common data manipulation operations on very large data sets stored in Hadoop.
- rmr: this R package allows an R user to perform statistical analysis via MapReduce on a Hadoop cluster.
- rhdfs: this R package provides basic connectivity to the Hadoop Distributed File System. R users can browse, read, write, and modify files stored in HDFS.
- rhbase: this R package provides basic connectivity to HBASE. R users can browse, read, write, and modify tables stored in HBASE.

In addition to these 4, we also add the tm package in R

- tm: [3] The text mining infrastructure package tm has now become the de facto standard for running text mining applications in R since it provides a

transparent way to prepare textual data for statistical analysis and offers easy extensibility via well documented interfaces.

Experimental Work

In this section, we provide cursory aspects on implementation of sentiment analysis on the entity Movie review corpus sourced from the Rotten Tomatoes site on the measurement of quality of filmed entertainment. The Objective is to glean sentiment score on a scale of 5. Heuristics underlined are on our approach from the related study [6] [7] [8] [9] [10].

- Data Pre-processing / Cleaning
- Exploratory Analysis
- Feature Identification and Weighting
- Applying Prediction Algorithms
- Evaluation of Results

Figure 3 gives clarity on Sentiment Classification Techniques [7].

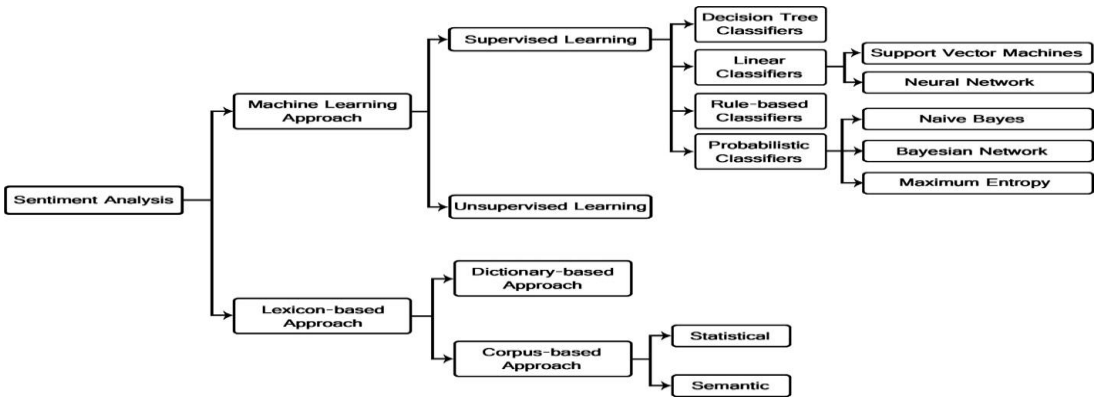

Figure 3: Classification of Sentiment Analysis Techniques

Data Pre-processing /Cleaning

This process involves multiple stages, of which the first step is preprocessing the data is to tune the data to our requirement. This involves changing all letters to lower case, removing punctuation marks, removing numbers, outliers and errors from the text. Filling in the missing values or words with approximates. Then the input, text data are selected and loaded into the mechanism. The first stage involves the removal of stop words from the text file, Stop-words are commonly used words in English grammar which do not have much overall relevance to the text mining process. Words such as prepositions such as 'for', 'on', 'in', 'at', 'is', 'which', etc. The list of stop words can be defined by the user to include words such as 'the', 'their', 'then' etc. After the removal of stop words from the input text, extra white space between the words is removed. The next step on the text is stemming, which is a process that uses a stemming algorithm for linguistic normalization, in which the various forms of a word are reduced to a root word or commonly known form, For example, 'connection',

'connections', 'connective', 'connected' and 'connecting' can all be normalized to the word 'connect', in order to perform stemming the text mining system must have a dictionary of indexed derived and root words already available to it. Next step is to take only words that have a minimum word length of 3.

Exploratory Analysis

Traditional data mining assumes that the information being "mined" is already in the structured format. Unfortunately, for many applications, electronic information is only available in the form of free natural-language documents rather than structured database [9]. These documents have to be extracted from their source documents using extraction algorithms then finally the data is summarized, clustered and used to yield knowledge for decision makers who can use the knowledge to make better decision for some kind of benefit. For the purposes of this paper, we have used several parsers and Natural Language Processing [7] to analyze a document to perform sentiment analysis, which can be used to gauge opinions in a recommendation system.

Exploratory Analysis phase involves various stages such as information extraction, topic tracking summarization, categorization, clustering, Concept Linkage, Information Visualization and finally Question Answering [6]. A starting point for computers to analyze unstructured text is to use information extraction. The Information extraction software identifies key phrases and relationships within the text. It does this by looking for predefined sequences in the text, a process called pattern matching. The software infers the relationships between all the identified people, places, and time to provide the user with meaningful information. This technology can be very useful when dealing with large volumes of text. Below are the sample test data.

Phrase Id	Sentence Id	Phrase
156076	8546	Kidman is really the only thing that 's worth watching in Birthday Girl , a film by the stage-trained Jez Butterworth -LRB- Mojo -RRB- that serves as yet another example of the sad decline of British comedies in the post-Full Monty world .
156077	8546	Kidman
156078	8546	is really the only thing that 's worth watching in Birthday Girl , a film by the stage-trained Jez Butterworth -LRB- Mojo -RRB- that serves as yet another example of the sad decline of British comedies in the post-Full Monty world .
156079	8546	is really the only thing that 's worth watching in Birthday Girl , a film by the stage-trained Jez Butterworth -LRB- Mojo -RRB- that serves as yet another example of the sad decline of British comedies in the post-Full Monty world
156080	8546	is really
156081	8546	is
156082	8546	really
156083	8546	the only thing that 's worth watching in Birthday Girl , a film by the stage-trained Jez Butterworth -LRB- Mojo -RRB- that serves as yet another example of the sad decline of British comedies in the post-Full Monty world
156084	8546	the only thing
156085	8546	the
156086	8546	only thing
156087	8546	only
156088	8546	thing
156089	8546	that 's worth watching in Birthday Girl , a film by the stage-trained Jez Butterworth -LRB- Mojo -RRB- that serves as yet another example of the sad decline of British comedies in the post-Full Monty world

As we are working on movie review corpora, we describe the frequency of terms that occur in a collection of documents. The below matrix shows the sparsity of word term frequency. Further word cloud is constructed from DTM such that words are scaled based on their frequency of occurrences. At the sentence level, 'film' (1295) and 'movie' (1173) is found to be the most frequent words.

```
<<DocumentTermMatrix (documents: 4, terms: 10)>>
Non-/sparse entries: 4/36
Sparsity           : 90%
Maximal term length: 10
Weighting          : term frequency (tf)

    Terms
Docs america amiable amid amounts amuses analyze antics anything apart appreciate
  27       0       0    0       0      0       0      0        0     0           0
  28       0       0    0       1      1       0      0        0     0           0
  29       0       0    0       1      1       0      0        0     0           0
  30       0       0    0       0      0       0      0        0     0           0
```

We took N-Gram-Based approach to text categorization that is tolerant of textual errors by calculating and comparing profiles of N-Gram frequency [15].

Table 1: Table showing frequency of words

Word	Frequency
film	1295
movie	1173
one	577
like	558
make	437

Figure 4 shows frequently occurring unigrams, some actor names identified as bi-grams and some movies identified as tri-grams.

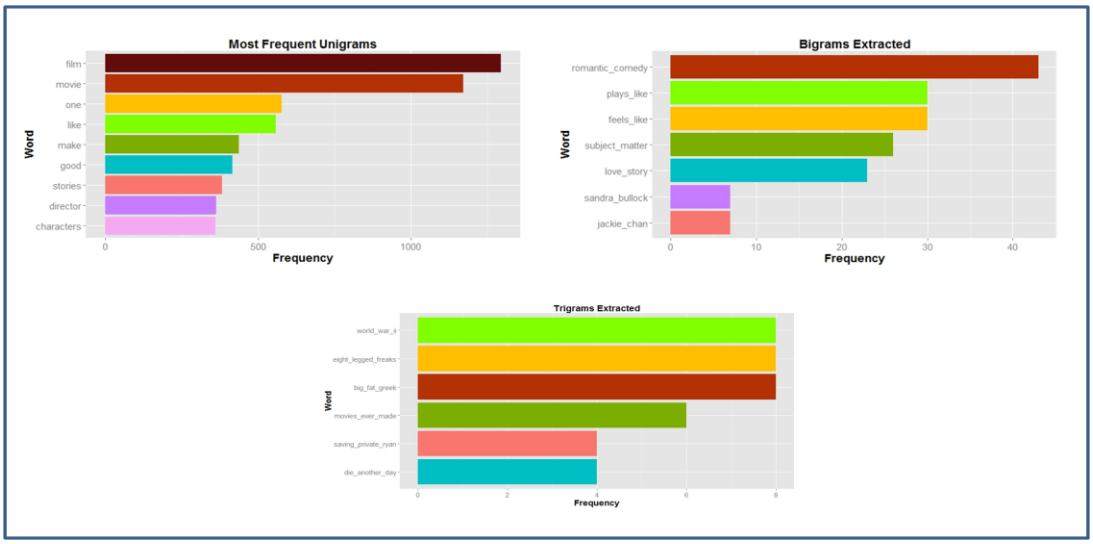

Figure 4: Bar chart showing frequently occurring unigrams, some actor names identified as bi-grams and some movies identified as tri-grams

Sentence level has Less of neutral reviews, while Phrase level has more of neutral reviews. Aspect Identification: Tag annotator is used to identify various parts of speech from the text. Subsequently probable candidates for aspects of movies ie,

Nouns are separated out and tabulated based on frequency. Relevant aspects like genre are extracted to understand the genre-wise number of reviews and their rating distribution.

V1	tags
A	DT
comedy-drama	NN
of	IN
nearly	RB
is	VBZ
romantic	JJ
comedy	NN
run-of-the-mill	NN
action	NN
flick	NN

Figure 5: Snapshot of Tags annotated

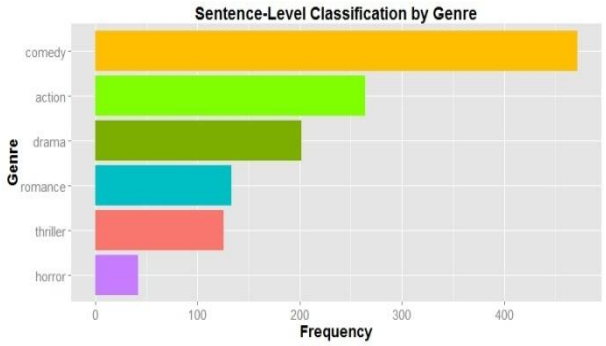

Figure 6: Movie genre were identified as aspects


Genre: Comedy
Frequency
60
40
20
0
0
1
2
3
4
Rating


Genre: Drama
Frequency
80
60
40
20
0
0
1
2
3
4
Rating


Genre: Thriller
Frequency
40
30
20
10
0
0
1
2
3
4
Rating

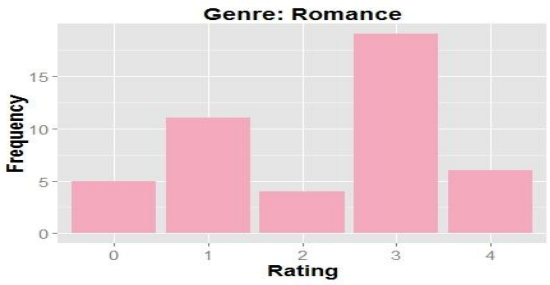
Genre: Romance
Frequency
15
10
5
0
0
1
2
3
4
Rating

Figure 7: Lens view of aspects

Feature Identification and Weighting

The movie review corpora after above steps is required for training the Machine Learning Algorithm. NLTK library (Python) is employed to identify the most informative, positive and negative features. During this course, most frequent bi-gram, tri-gram and most informative uni-grams were placed as features in predicting. Total around 2700 features were chosen.

Figure 8: Features with weightage

Weighting of Terms:

The cell W_{ij} indicates the "weight" of a term in the phrase (or sentence); zero means the term has no significance in the phrase or it simply doesn't exist in the review. Most frequent bi-gram, tri-gram and most informative unigrams represent the "terms".

Table 2: Matrix format for recording weight of terms

	Term 1	**Term 2**	**….**	**Term t**
Phrase 1	W_{11}	W_{21}	….	W_{t1}
Phrase 2	W_{12}	W_{22}	….	W_{t2}
…	….	….	….	….
Phrase n	W_{1n}	W_{2n}	….	W_{tn}

Table 3: Matrix format for recording term frequency

	Term 1	**Term 2**	**….**	**Term t**
Phrase 1	f_{11}	f_{21}	….	f_{t1}
Phrase 2	f_{12}	f_{22}	….	f_{t2}
…	….	….	….	….
Phrase n	f_{1n}	f_{2n}	….	f_{tn}

Term Frequency - Inverse Document Frequency (TF-IDF) weighting was used in place of frequency as weights.

Sentiment score (y) depends on TF-IDF of 2700 terms (x_1,x_2,....x_{2700})

TF-IDF weighting = W_{ij} = TF * IDF

Term Frequency (TF)

More frequent terms within a document are more important, ie more indicative of the topic f_{ij} - frequency of term i in review j

Normalize *term frequency (tf)* by dividing the frequency of the most common term in that review: $tf_{ij} = f_{ij} / max_i\{f_{ij}\}$

Inverse Document Frequency (IDF):

Terms that appear in many reviews are less indicative of the overall topic.

df_i - document frequency of term i (ie., number of documents containing term i)

idf_i - Inverse document frequency of term i = $\log_2(N / df_i)$ where N is the total number of documents.

TF-IDF Weighting = $W_{ij} = tf_{ij} * idf_i = tf_{ij} * \log_2(N / df_i)$

A term occurring in particular review, but rarely in the rest of the collections gives higher weight.

Prediction Algorithms

Prediction means same as finding the most likely, as inferred from the past behavior. In this experiment we have evaluated machine learning approach in supervised learning heuristics [7].

- Support Vector Machine
- Neural Network
- Random Forest

The approach called "kernel trick" [16] is followed for the classification model for a non-linear decision boundary obtained by enlarging feature space using Radial (Gaussian) Kernel. Selection of one class over other classes is used for applying SVMs. Each class is meant to be an SVM of best fit. 5 classes here mean that 5 SVMs need to be fit, iteratively comparing one of the 5 classes with the remaining 4 classes. Sentiment assigned to the class with maximum distance-value (indicates the confidence level) among the 5 SVMs.

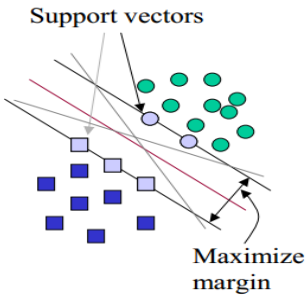

Figure 9: Using Support Vector Machine to classify Sentiments

Neural Networks are used to capture and depict complex relationships for input and output [17]. Using least squares, each logistic unit in the hidden and output layers is trained after initialization with random weights. Criteria enforced, Number of units used in the hidden layer is 2 and maximum iterations are 1000 with total number of input units after removing sparse terms is taken to be 2700. Snippet of R code is given below.

```
for test
container <- create_container(doc_matrix, RT_sentences$Sentiment[1:1000], trainSize=1:900,testSize=901:1000, virgin=FALSE)
#Neural Networks
NNET <- train_model(container,"NNET",size = 2)
NNET_CLASSIFY <- classify_model(container, NNET)
#SVM Classifier
SVM <- train_model(container,"SVM")
SVM_CLASSIFY <- classify_model(container, SVM)
#Random Forest Classifier
RF <- train_model(container,"RF")
RF_CLASSIFY <- classify_model(container, RF)
#To Find precision recall and f-score
analytics <- create_analytics(container, cbind(SVM_CLASSIFY,RF_CLASSIFY,NNET_CLASSIFY))
summary(analytics)
```

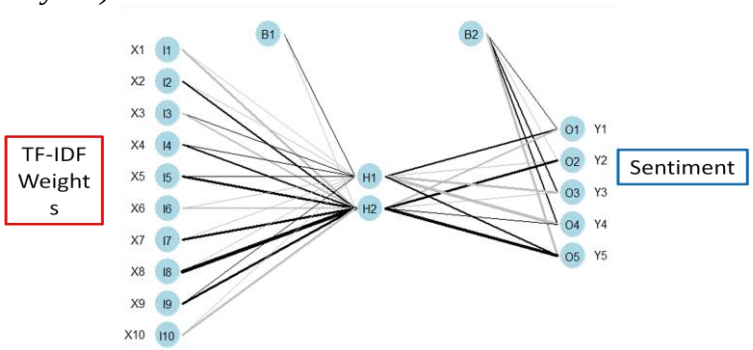

Figure 10: Neural Network representation

- Bagging used along with a random subset of features is used to train the model.
- Collection of decision trees trained with 200 trees and through random selection of ~52 from ~2700 features passed.
- Prediction is based on majority votes from decision trees

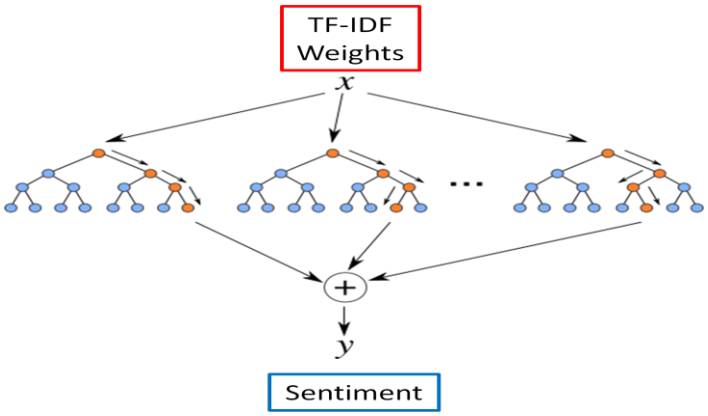

Figure 11: Random Forest representation

Measure Performance

For each class (sentiment rating), a confusion matrix is built.

Table 4: Table showing Confusion Matrix

	Predicted as Class 1	**Predicted as Other Class**
Actually Class 1	tp (True Positive)	fn (False Negative)
Actually other Class	fp (False Positive)	tn (True Negative)

Table 5: Table showing formulae]

$$\text{Precision} = \frac{tp}{tp+fp}$$

$$\text{Recall} = \frac{tp}{tp+fn}$$

$$\text{F-Score} = \frac{2*\text{Precison}*\text{Recall}}{(\text{Precision}+\text{Recall})}$$

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn}$$

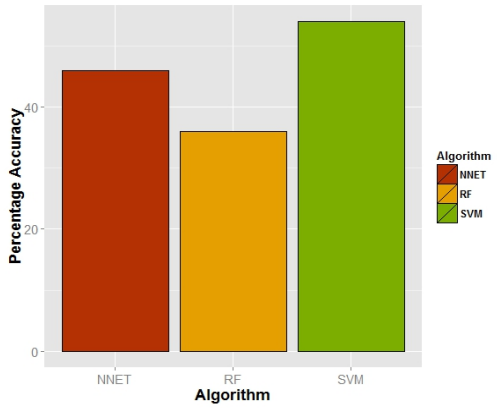

Figure 12: At Sentence Level: Prediction Accuracy in %

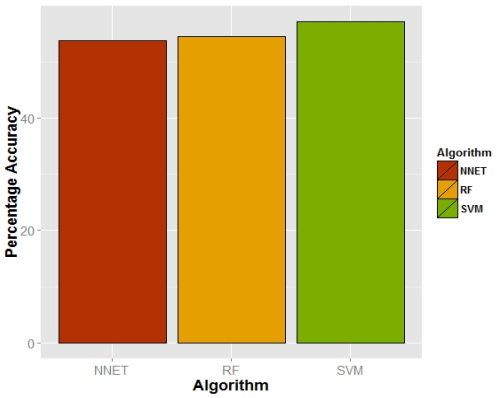

Figure 13: At Phrase Level: Better than at sentence level, because here most reviews fall into the 'neural rating'.

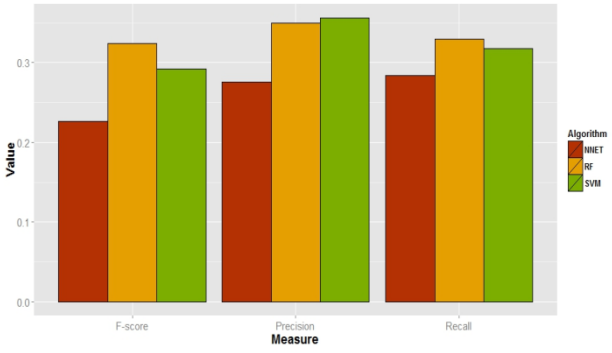

Figure 14: At Sentence Level: Precision, Recall and F-Score

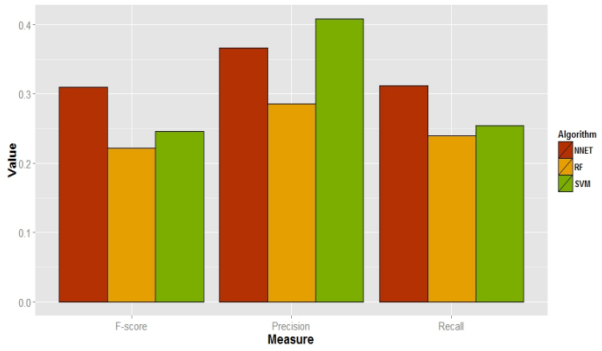

Figure 15: At Phrase Level: Precision, Recall and F-Score

Conclusion

Text mining, an interdisciplinary field, requires number of techniques to seek information from unstructured textual data like machine learning, data mining techniques, natural language processing, statistics, etc. Our goal in this paper was to showcase the approach on gleaning high quality insights from implementation aspects of opinion mining on distributed environment that involves choosing heuristics to discover the sentiment rating on a scale of 5. In this paper, readers are provided with the richness of text mining tasks for opinion data and review data using RHadoop and its components. A comparison was made on the performance to get sentiment analysis on the movie review entity. Overall, we conclude among the three algorithms selected, Random Forest had least performance at phrase level. At sentence level, Neural networks had a comparatively lesser measure of performance, while SVM had overall good performance and also computationally efficient over others.

References

[1]. Defining Text Mining and Analytics http://en.wikipedia.org/wiki/Text_mining#cite_note-1

[2]. What is Hadoop Distributed File System HDFS? http://www-01.ibm.com/software/data/infosphere/hadoop/hdfs/

[3]. Ingo Fingerer, Kurt Horne, and Stefan. “A tm Plug-In for Distributed Text mining in R. Journal of Statistical Software”, Volume 51, Issue 5, November 2012,.

[4]. Ian H. Witten. Text Mining, Computer Science, University of Waikato, Hamilton, New Zealand email ihw@cs.waikato.ac.nz.

[5]. Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified data processing on large clusters”. In OSDI’04, 6th Symposium on Operating Systems Design and Implementation, p:137–150, 2004.

[6]. Vishal Gupta, Gurpreet S. Lehal. "A Survey of Text Mining Techniques and Applications". Journal of emerging technologies in web intelligence, vol 1 (1), August 2009.
[7]. Walaa Medhat, Ahmed Hasan, Hoda Korashy. "Sentiment Analysis algorithms and applications: A survey". Ain Shams Engineering Journal, 5:1093-1113, 2014.
[8]. Bo Pang and Lillian Lee. "Opinion Mining and Sentiment Analysis; Foundations and Trends in Information Retrieval". Vol.2, No 1-2, p:1-135, 2008.
[9]. ChandhanaSurabhi.M, "Natural Language Processing Future". Proceedings of International Conference on Optical Imaging Sensor and Security, Coimbatore, Tamil Nadu, India, July 2013.
[10]. Pattern-Based Strategy: Getting Value from Big Data. Gartner Group press release. July 2011. Available at http://www.gartner.com/it/page. jsp?id=1731916
[11]. S. Das, Y. Sismanis, K.S. Beyer, et al. "Ricardo: Integrating R and Hadoop". New York, Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, p:987-998, 2010.
[12]. White paper on Advanced 'Big Data' Analytics with R and Hadoop, info@revolutionanalytics.com.
[13]. Tom White, Hadoop, The Definitive Guide; Published by O'Reilly Media Inc, May 2012.
[14]. Hearst, M.A. (1999) "Untangling text mining." Proc Annual Meeting of the Association for Computational Linguistics ACL99, University of Maryland, June.
[15]. William B. Cavnar, John M. Trenkle. "N-Gram-Based Text Categorization". Environmental Research Institute of Michigan, Ann Arbor, MI 48113-4001.
[16]. http://en.wikipedia.org/wiki/Kernel_method.
[17]. www.rottentomatoes.com.
[18]. Daniel Sonntag, "Distributed Text Mining and Natural Language Processing for DataMining Grid," DaimlerChrysler Research and Technology, RIC/AM, 89013 Ulm Germany, March 9, 2004.