Scalable Sentiment Classification for Big Data Analysis Using Naïve Bayes Classifier

Bingwei Liu*, Erik Blasch†, Yu Chen‡, Dan Shen* and Genshe Chen*
* Intelligent Fusion Technology, Inc. Germantown, Maryland, USA
Email: {bingwei.liu, dshen, gchen}@intfusiontech.com
† Air Force Research Laboratory, Rome, New York, USA
Email: erik.blasch.1@us.af.mil
‡ Binghamton University, Binghamton, New York,USA
Email: ychen@binghamton.edu

***Abstract*—A typical method to obtain valuable information is to extract the sentiment or opinion from a message. Machine learning technologies are widely used in sentiment classification because of their ability to “learn” from the training dataset to predict or support decision making with relatively high accuracy. However, when the dataset is large, some algorithms might not scale up well. In this paper, we aim to evaluate the scalability of Naïve Bayes classifier (NBC) in large datasets. Instead of using a standard library (e.g., Mahout), we implemented NBC to achieve fine-grain control of the analysis procedure. A Big Data analyzing system is also design for this study. The result is encouraging in that the accuracy of NBC is improved and approaches 82% when the dataset size increases. We have demonstrated that NBC is able to scale up to analyze the sentiment of millions movie reviews with increasing throughput.**

Keywords*—*Cloud computing, Big data, Polarity mining, sentiment classification

I. Introduction

Data on the web has been explosively increasing in the past few decades. The ability to automatically mine useful information from massive data has been a common concern for organizations who own large datasets. The MapReduce framework [1] is commonly used to analyze extremely large datasets such as tweets collections, online documents or large-scale graphs [2] [3] [4]. The framework provides a simple and powerful interface for programmers to solve large-scale problems using a cluster of commodity computers.

A typical method to obtain valuable information is to extract the sentiment or opinion from a message. Sentiment classification is useful for the business consumer industry or online recommendation systems. For example, online product reviews are usually analyzed by manufacturers to decide what products they will produce in the future to reduce risk. Machine learning technologies, such as Naïve Bayes and support vector machine, are widely used in sentiment classification [5] [6] [7] [8] [9] [10] because of their ability to “learn” from the training dataset to make decisions with on-line data, to predict salient features, and to provide real-time analysis with relatively high accuracy.

Naïve Bayes (NB) classifiers are widely used in information fusion. Current trends in data fusion include machine analytics for big data [11], use of NB for cloud computing applications of simultaneous target tracking and classification [12], and robotics control [13]. Techniques for big data analysis are needed imaging, text, and cyber analysis which includes scalable and elastic learning methods [14].

When the datasets are large, some information fusion algorithms might not scale up well. For example, if an algorithm needs to load data into memory constantly, the program may run out of memory for large datasets. One promising approach is to utilize and adapt MapReduce for some machine learning technologies to resolve these large-scale problems. Apache Mahout[1] is a machine learning library for clustering, classification and filtering, implemented on top of Hadoop[2], the open source version of MapReduce. Although there are some machine learning algorithms implemented in Mahout, it is still helpful to study how to convert a machine learning algorithm to a Hadoop program and to optimize the algorithm scalability in large datasets.

In this paper, we aim to evaluate the scalability of Naïve Bayes classifier (NBC) in large-scale datasets. Instead of using Mahout library, we implemented NBC to achieve fine-grain control of the analysis procedure for a Hadoop implementation. The result is encouraging in that the accuracy of NBC is improved and approaches 82% when the dataset size increases. We have demonstrated that NBC is able to scale up to analyze the sentiment of millions movie reviews with increasing throughput.

The rest of this paper is organized as follows. Section II introduces the background of this study. Section III illustrates the system we design to analyze big datasets. The system is built on top of Hadoop basic components. The details of implementing Naïve Bayes classifier are addressed in section IV. Section V shows the experiment setup and results. Finally, we conclude in Section VI.

II. Background

A. MapReduce

The MapReduce framework has been used to process large datasets since the original paper [1] was published. Google’s clusters process more than 20 Petabytes of data every day by running one hundred thousand MapReduce jobs on average [15]. Using this framework, programmers only need to focus

[1]http://mahout.apache.org
[2]http://hadoop.apache.org/

on problem solving versus implementation. The MapReduce runtime system will take care of the underlining parallelization, fault tolerance, data distribution and load balance. Google file system (GFS) is a distributed file system that MapReduce uses for the storage of large amount of data across inexpensive hard drives. The availability and reliability of underlining unreliable hardware are provided by replicating file blocks and distributing them across different nodes.

A MapReduce job consists of at least a map function and a reduce function, called mapper and reducer respectively. The mapper takes as input a pair of key/value and produces a set of key/value pairs. All key/value pairs are sorted by their keys and sent to different reducers according to the key. Each reducer receives a key and a set of values that has the **same** key. This makes MapReduce an excellent tool for computations that need sorting or counting. The map and reduce functions are left to the user to implement their desired functionalities to process each key/value pair.

Hadoop is an open source implementation of the MapReduce framework that is commonly used by academic and industry for Big Data analysis. In the core of Hadoop are Hadoop MapReduce and Hadoop Distributed File System (HDFS), the open source counterpart of GFS. There are also a bundle of Hadoop-related projects supported by Apache Foundation, such as HBase (database), Hive (data warehouse), Pig (high-level data-flow), Zookeeper (high-performance coordination) and Mahout (scalable machine learning and data mining). Therefore, we choose Hadoop as the develop platform to study the scalability of Naïve Bayes classifier.

B. *Naïve Bayes Classification*

Naïve Bayes has proven to be a simple and effective machine learning method in previous text classification studies. It is even optimal in some cases [16] [17].

Suppose there are m possible classes $\mathbb{C} = \{c_1, c_2, \cdots, c_m\}$ for a domain of documents $\mathbb{D} = \{d_1, d_2, \cdots, d_n\}$. Let $\mathbb{W} = \{w_1, w_2, \cdots, w_s\}$ be the set of unique words, each of which appears at least once in one of the documents in $\mathbb{D}$. The probability of a document d being in class c can be computed using Bayes' rule:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}. \tag{1}$$

Since $P(d)$ is a constant for the known data set size, the denominator of (1) is typically not calculated for maximum a posteriori (MAP), common for parametric statistical problems. A Naïve Bayes model assumes that each term or word, w_k, in a document occurs independently in the document given the class c. Therefore, equation (1) becomes:

$$P(c|d) \propto P(c) \prod_{k=1}^{n_d} [P(w_k|c)]^{t_k},$$

where n_d is the number of unique words in document d and t_k is the frequency of each word w_k. To avoid floating point underflow, we use the equivalent equation:

$$\log P(c|d) \propto \log P(c) + \sum_{k=1}^{n_d} [t_k \log P(w_k|c)]. \tag{2}$$

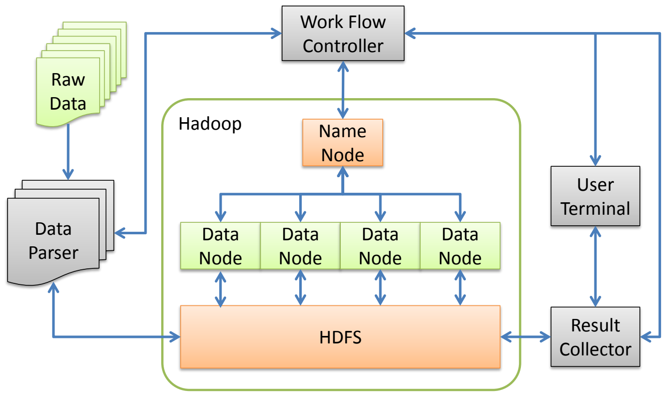

Fig. 1. A simple system to process data using Naïve Bayes classifier on Hadoop.

The class of d is decided as the class, c^*, which maximizes $\log P(c|d)$ in equation (2).

$$c^* = argmax_{c \in \mathbb{C}} \left\{ \log P(c) + \sum_{k=1}^{n_d} [t_k \log P(w_k|c)] \right\} \tag{3}$$

When applying Naïve Bayes classifier (NBC), we can estimate $P(c)$ and $P(w_k|c)$ as:

$$\widehat{P}(c) = \frac{N_c}{N} \text{ and } \widehat{P}(w_k|c) = \frac{N_{w_k}}{\sum_{w_i \in \mathbb{W}} N_{w_i}}$$

where N is the total number of documents, N_c is the number of documents in class c and N_{w_i} is the frequency of a word w_i in class c. With these estimations, the calculation of the right hand side of equation (3) is essentially a counting problem. This makes MapReduce a suitable framework for the implementation of NBC in large-scale datasets.

III. System Description

On top of Hadoop MapReduce and HDFS, we designed a Big Data analyzing system to evaluate whether Naïve Bayes classifier can scale up to classify millions of movie reviews. This section explains the four modules of the system and important steps of the work flow.

A. *System Components*

As shown in Fig. 1, the system adds four modules on top of Hadoop: the work flow controller (WFC), the data parser, the user terminal and the result collector. We design this system based on our need to generate different sizes of datasets and test the Hadoop program on them respectively. We also need to perform ten-fold cross validation for accuracy computation that requires calling the same program multiple times.

Our raw data comes from large sets of movie reviews collected by research communities. The data parser is responsible to produce the desired data format to assist the program to efficiently process each review. The user submits jobs through the user terminal. Experiment results are also accessible through the user terminal after the result collector finishes collection. The work flow controller manages the work flow of the whole system, which includes:

1) Instruct data parser of the format of input data and the desired output;
2) Transmit source code to the name node and execute Hadoop jobs; and
3) Trigger the result collector to collect computing results once they are available on Hadoop Distributed File System (HDFS).

B. Dataset

In our experiments, we use two datasets: the Cornell University movie review dataset[3] and Stanford SNAP Amazon movie review dataset[4]. The Cornell dataset has 1000 positive and 1000 negative reviews, which makes $P(c)$ in equation (3) 0.5 for both class. A review's class is then decided by the frequency of each word that appears in the model obtained from training dataset.

The Amazon movie review dataset is organized into eight lines for each review, with additional information such product identification (ID), user ID, profileName, score, summary etc. Since it is a 5 points rating system, we simply divide all reviews into positive and negative subset using 3.0 as a threshold. We consider unigrams only for the Naïve Bayes model.

C. Overall Work Flow

1) Pre-processing Raw Dataset: The data parser first pre-processes all reviews into a common format. After the processing, each review is one line in the dataset, with document ID and sentiment (positive or negative) prefixed. This is useful because by default MapReduce splits the input files by line and passes each line to a mapper. To pre-process a raw review from the dataset, we simply delete unwanted context such as punctuation, special symbols and numbers. We did not introduce a lexicon or vocabulary to filter out meaningless words.

Table I lists four sample reviews after pre-processing. Each review is one line in the dataset file. We use two prefixed tags: sentiment of either positive or negative and document ID, which is a number. The two tags are both surrounded by colons to help the program differentiate them from review text. Note that there are some isolated single letters resulting from the remove of punctuation. For example, the "p.s." in review ID 41 becomes "p s " after removing the two dots. The pre-processing procedure didn't remove these meaningless letters. All pre-processed reviews are stored in the name node as a repository, waiting for further sampling.

2) Preparing Input Datasets: The WFC and the data parser work together to prepare input datasets for all test trials. When the WFC requests a dataset with certain size, the data parser extracts from the repository the desired number of positive reviews, as well as the same amount of negative reviews. The result is an input dataset of two equal-size classes of movie reviews. After a dataset is generated, the data parser divides it into 10 subsets for the convenience of ten-fold cross validation. The WFC then moves them to the right locations in HDFS for every trial and calls the Hadoop NBC program.

[3]http://www.cs.cornell.edu/people/pabo/movie-review-data/
[4]http://snap.stanford.edu/data/web-Amazon.html

TABLE I. SAMPLE REVIEWS AFTER PRE-PROCESSING.

:POS: :41: i disagree with the reviewers who said the movie was predictable and drawn out it was a movie with heart and you could feel the main characters plight when he lost his companion being an animal lover i was pulling for the happy ending of course i am disney s biggest fan and i love this movie right along with the others p s i am a grandmother to eleven thank heavens for disney movies
:POS: :85: sit back and enjoy the interesting and exciting story of the count of monte cristo great rainy day movie
:POS: :95: a very well done film and an excellent cast i d put it right up with the three and four musketeers movies york reed chamberlain heston etc
:POS: :96: this is an excellent movie and i never read the book the acting and the plot was very nice done it is one of my favorite movies

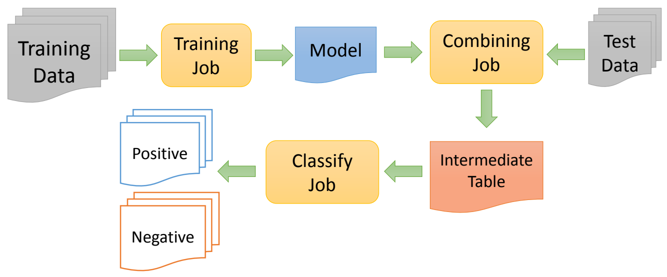

Fig. 2. Job sequence of Naïve Bayes Classifier on Hadoop.

3) Sentiment Classification Using Hadoop: The sentiment classification is the key step in the work flow. Fig. 2 shows the job sequence of this step. Once the training data and test data are ready in HDFS, the WFC starts the training job to build a model. The combining job then combines test data with the model, resulting an intermediate table. Finally, the classify job simultaneously computes the probabilities of each review in the two classes respectively and makes a decision about the sentiment of this review. The statistics of true positive, true negative, false positive and false negative are also recorded.

4) Result Collection: After the classify job finished, the result collector retrieves the model, intermediate table, classification results and statistics of the test from HDFS.

D. Automatic Scheduling

The WFC coordinates the automation of the whole system. For the Amazon dataset, we conducted 120 test trials in total on twelve sizes of datasets. All test trials are automatically scheduled by the WFC without supervision. The only parameters that need to be decided by human is the sizes of experiment datasets. This automatic scheduling method can be easily applied to other programs with minor change of the parameters.

IV. IMPLEMENTATION OF NAÏVE BAYES IN HADOOP

A. Defining the Problem

We first define our task as to classify the sentiment of a movie review, "positive" or "negative". Hence, we only have two classes of documents. To simplify the problem, we choose the same number of positive and negative reviews, which makes the $P(c)$ in equation (3) a constant. The estimation of $P(w_k|c)$ is computed by the relative frequency of w_k in all documents in c. The classification problem is then converted to a counting problem on the training and test datasets.

Algorithm 1 Training Job

```
Input: The training reviews.
Output: Model.
function MAPPER($key$, $value$)
    $s$ = parseSentiment($value$)
    for all $w \in d$ do
        emit($w, 1 : s$)
    end for
end function
function REDUCER($key$, $values$)
    $posSum = 0; negSum = 0;$
    for all $value \in values$ do
        [$s, count$] = parse($value$)
        if $s == positive$ then
            $posSum+ = count$;
        else
            $negSum+ = count$;
        end if
    end for
    emit($key, posSum, negSum$)
end function
```

Algorithm 2 Combining Job

```
Input: The test reviews and the model.
Output: Intermediate results for classify job input.
function MAPPER($key$, $value$)
    if $value \in model$ then
        [$word, pos_sum, neg_sum$] = parse($value$)
        emit($word, pos_sum, neg_sum$)
    else
        [$docid$, $sentiment$] = parse($value$)
        for all $word \in value$ do
            emit($word$,1,$docid$,$sentiment$)
        end for
    end if
end function
function REDUCER($key$, $values$)
    for all $value \in values$ do
        if $value \in model$ then
            outstr.append($value$)
        else
            [$count$, $docid$] = parse($value$)
            docs[$docid$].add($count$)
        end if
    end for
    for all $docid \in docs$ do
        outstr.append($count$,$docid$)
    end for
    emit($word$, $outstr$)
end function
```

B. Algorithms

After the simplification of the problem, the task can be divided into three sequential jobs as follows.

1) Training job (Algorithm 1). All training reviews are fed into this job to produce a model for all unique words with their their frequency in positive and negative review documents respectively.
2) Combining job (Algorithm 2). In this job, the model and the test reviews are combined to a intermediate table with all necessary information for the final classification.
3) Classify job (Algorithm 3). This job classifies all reviews simultaneously and writes the classification results to HDFS.

Algorithm 1-3 are the pseudo-codes for the three jobs. These jobs are executed in sequence because the dependencies between them, as shown in Fig. 2. The training job produces a model that is used to compute the probability of each word in the two classes. The combining job then associates the test data with the model, excluding the words that appear in test data but not in the model. The intermediate table produced by the combining job is then fed to the classify job. By the end of classify job, all reviews are classified into positive or negative classes.

The actual Hadoop code we develop is less than five hundred programming lines, including comments and package import statements. With such a simple program, the results are surprisingly good, considering the difficulties of mining sentiment using computer programs.

Algorithm 3 Classify Job

```
Input: The intermediate results.
Output: Classification results and accuracy.
function MAPPER($key$, $value$)
    for all $item \in value$ do
        [$docid, count, pos_sum, neg_sum$] = parse($value$)
        emit($docid, count, pos_sum, neg_sum$)
    end for
end function
function REDUCER($key$, $values$)
    [$docid, trueSentiment$]=parse($key$)
    for all $value \in values$ do
        calculate $pos\_prob$
        calculate $neg\_prob$
    end for
    $predict$ = ($pos\_prob > neg\_prob$) ? $pos$ : $neg$
    if $predict = trueSentiment$ then
        $correct$ = true
        $correct\_count + +$
    else
        $correct$ = false
    end if
    emit($docid, predict, correct$)
end function
```

V. Experimental Study

A. Cloud Infrastructure

Virtual Hadoop cluster is a fast and easy way to test a Hadoop program in the Cloud, although the performance might be weaker compared to a physical Hadoop cluster. Our cloud infrastructure is built on a Dell server with 12 Intel Xeon E5-2630 2.3GHz cores and 32G memory. We use Xen Cloud Platform (XCP) 1.6 as the hypervisor. On top of XCP 1.6, we built a virtual Hadoop cluster of seven nodes. There is one special node, the manager, that we use to manage the Hadoop

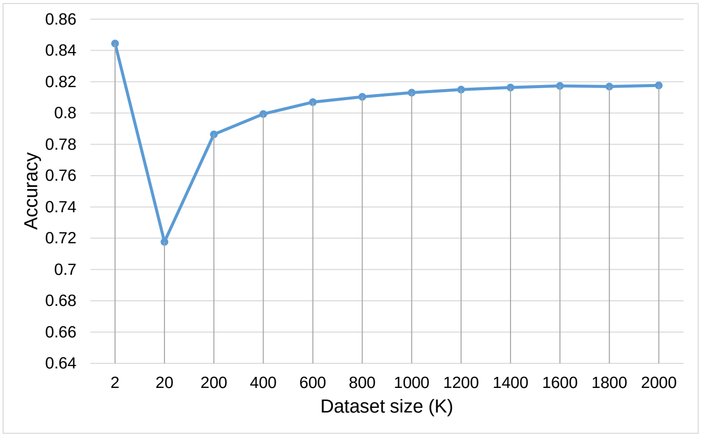

Fig. 3. The accuracy of Naïve Bayes classifier when the dataset size increases.

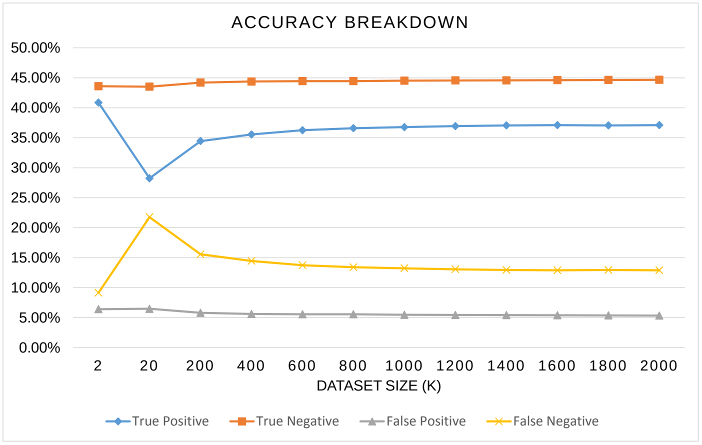

Fig. 4. The accuracy breakdown of Naïve Bayes classifier when the dataset size increases.

cluster. The rest of six nodes form the actual Hadoop cluster, with one name node and six data nodes. We allocate each VM two virtual CPU and 4GB of memory to provide sufficient computing resources for each Hadoop node.

B. Experiment Setup

First we tested our code on Cornell dataset and resulted in a 80.85% average accuracy. Without changing the Hadoop code, our program was able to classify different subsets of Amazon movie review dataset with comparable accuracy. To test the scalability of Naïve Bayes classifier, the size of dataset in our experiment varies from one thousand to one million reviews in each class.

C. Results

The result statistics include the classification accuracy, the computation time and the throughput of the system.

Fig. 3 shows the average accuracy of our NBC program on various sizes of datasets. Each accuracy number is the average of ten trials. When the dataset is relatively small, the accuracy is unstable because the training data are not big enough for the model to learn enough knowledge about each class. As the dataset size increases above 400K, the

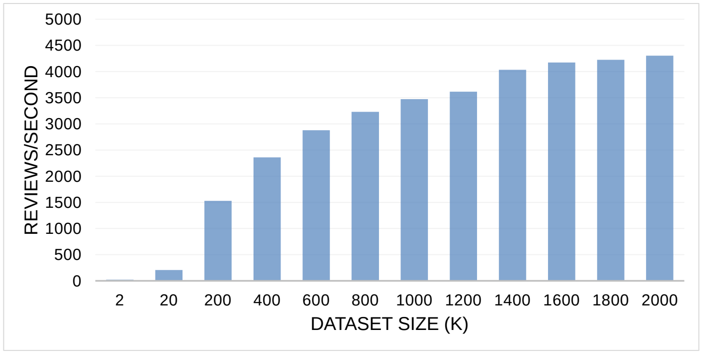

Fig. 5. Throughput of the System with respect to dataset size.

TABLE II. PROCESSING TIME (SECOND) EVERY 1000 REVIEWS.

Dataset size (K)	2	20	200	400	600	800
Second/10K reviews	455.1	48.15	6.57	4.24	3.47	3.11
Dataset size (K)	1000	1200	1400	1600	1800	2000
Second/10K reviews	2.88	2.77	2.47	2.4	2.37	2.33

accuracy gradually climbs above 80% and approaching 82%. This demonstrates that the accuracy of NBC is stable when the dataset increases. To further examine the classification results, we plot in Fig. 4 the breakdown of accuracy into true positive and true negative, accompanied by false positive and false negative. As we expected, the true positive and true negative increase with respect to the dataset size, while the false positive and false negative decrease.

Table II shows that the processing time for every ten thousand reviews in our Hadoop NBC program decreases when the dataset size increases. A dataset of 2K reviews did not benefit from the parallelization of Hadoop because the input data is smaller than the size of one block in HDFS. Although three replicas are distributed in different nodes, there are at most three nodes in the Hadoop cluster that can access the input data locally. After the input data increasing to a certain size, the advantage of Hadoop starts to appear in that the processing time for the same amount of reviews is drastically reduced compare to the 2K case.

To observe the resutls in another dimension, Fig. 5 shows the throughput of the system with respect to the size of dataset. The number of reviews that the system can processes in one second increases from 22 (2K case) to 4304 (2000K case).

Overall, our implementation of NBC is able to scale up to two million reviews sampled from the Amazon dataset. The accuracy tends to stable when the dataset size increases. These results are based on the simple processing of review texts. Further filtering of the input data might be able to increase the accuracy.

VI. CONCLUSION

In this paper, we presented a simple and complete system for sentiment mining on large datasets using a Naïve Bayes classifier with the Hadoop framework. We implemented the NBC on top of Hadoop framework, with additional modules to automate the experiment. We also provide the implementation details for converting a classifier to Hadoop program, of which any machine learning algorithm could be replaced for the NBC.

Our results show that NB classifier can scale up easily, even without a database. Because of our simplified setup, the average accuracy stays below 82% in all cases. An intelligent filter might be helpful to increase the accuracy.

We believe that our work is just a beginning of employing machine learning technologies in large-scale datasets. Future work will include using our framework for information fusion over imagery and text, distributed robotics applications, and cyber analysis using cloud computing.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI'04. Berkeley, CA, USA: USENIX Association, 2004.

[2] U. Kang, D. H. Chau, and C. Faloutsos, "Pegasus: Mining billion-scale graphs in the cloud," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012, pp. 5341–5344.

[3] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 607–614.

[4] U. Kang, D. H. Chau, and C. Faloutsos, "Mining large graphs: Algorithms, inference, and discoveries," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, 2011, pp. 243–254.

[5] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 2002, pp. 79–86.

[6] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 2004, p. 271.

[7] P. Chesley, B. Vincent, L. Xu, and R. K. Srihari, "Using verbs and adjectives to automatically classify blog sentiment," *Training*, vol. 580, no. 263, p. 233, 2006.

[8] M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger, "Pulse: Mining customer opinions from free text," in *Advances in Intelligent Data Analysis VI*. Springer, 2005, pp. 121–132.

[9] A. Kennedy and D. Inkpen, "Sentiment classification of movie reviews using contextual valence shifters," *Computational Intelligence*, vol. 22, no. 2, pp. 110–125, 2006.

[10] M. Thomas, B. Pang, and L. Lee, "Get out the vote: Determining support or opposition from congressional floor-debate transcripts," in *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006, pp. 327–335.

[11] E. Blasch, A. Steinberg, S. Das, J. Llinas, C. Chong, O. Kessler, E. Waltz, and F. White, "Revisiting the JDL model for information exploitation," in *Int'l Conf. on Info Fusion*, 2013.

[12] E. Blasch, Y. Chen, G. Chen, D. Shen, and R. Kohler, "Information fusion in a cloud-enabled environment," *High Performance Semantic Cloud Auditing, Springer Publishing*, 2013.

[13] B. Liu, Y. Chen, E. Blasch, K. Pham, D. Shen, and G. Chen, "A holistic cloud-enabled robotics system for real-time video tracking application," in *Intl Workshop on Enhanced Cloud Fusion, in conjunction with Future Tech, Korea, Sept. 2013*, Sept.

[14] B. Liu, E. Blasch, Y. Chen, A. J. Aved, A. Hadiks, D. Shen, and G. Chen, "Information fusion in a cloud computing era: A systems-level perspective," *Submitted to IEEE AES Magazine*, 2013.

[15] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[16] D. Lewis, "Naïve (bayes) at forty: The independence assumption in information retrieval," *Machine Learning: ECML-98*, pp. 4–15, 1998.

[17] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[18] H. Karloff, S. Suri, and S. Vassilvitskii, "A model of computation for mapreduce," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2010, pp. 938–948.