applied sciences

MDPI

Article

Detection of Sensitive Data to Counter Global Terrorism

Binod Kumar Adhikari [1,2], Wanli Zuo [1,*], Ramesh Maharjan [2], Xuming Han [3] and Shining Liang [1]

1 College of Computer Science and Technology, Jilin University, Changchun 130012, China; binodkumaradhikari14@mails.jlu.edu.cn (B.K.A.); liangsn19@mails.jlu.edu.cn (S.L.)
2 Amrit Campus, Tribhuvan University, Kathmandu 44600, Nepal; ramesh.anahcolus@gmail.com
3 School of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China; hanxvming@163.com
* Correspondence: wanli@jlu.edu.cn

Received: 4 December 2019; Accepted: 20 December 2019; Published: 25 December 2019

Abstract: Global terrorism has created challenges to the criminal justice system due to its abnormal activities, which lead to financial loss, cyberwar, and cyber-crime. Therefore, it is a global challenge to monitor terrorist group activities by mining criminal information accurately from big data for the estimation of potential risk at national and international levels. Many conventional methods of computation have successfully been implemented, but there is little or no literature to be found that solves these issues through the use of big data analytical tools and techniques. To fill this literature gap, this research is aimed at the determination of accurate criminal data from the huge mass of varieties of data using Hadoop clusters to support Social Justice Organizations in combating terrorist activities on a global scale. To achieve this goal, several algorithmic approaches, including parallelization, annotators and annotations, lemmatization, stop word Remover, term frequency and inverse document frequency, and singular value decomposition, were successfully implemented. The success of this work is empirically compared using the same hardware, software, and system configuration. Moreover, the efficacy of the experiment was tested with criminal data with respect to concepts and matching scores. Eventually, the experimental results showed that the proposed approach was able to expose criminal data with 100% accuracy, while matching of multiple criminal terms with documents had 80% accuracy; the performance of this method was also proved in multiple node clusters. Finally, the reported research creates new ways of thinking for security agencies in combating terrorism at global scale.

Keywords: big data; term frequency-inverse document frequency (TF-IDF); singular value decomposition (SVD); Hadoop; lemmatization; criminal justice system

1. Introduction

Global Terrorism is the use of intentional violence against civilians and neutral militants for the purpose of political, cultural, and social benefits around the world [1]. Due to this, individuals, business owners, government, and private and public organizations are at risk [2]. The administrative expense of combating global terrorism is too high [3,4]. For example, the 9/11 terrorist attack caused an economic loss of approximately US $123 billion, while the London bombing cost UK £2 billion [5]. In the era of big data, the advent of the latest technologies, the use of social media, mobile technologies, and cloud drives has led to the generation of different types of voluminous data in the form of big data. Criminals also use all of these platforms to exploit information, harm society, and globally dispute criminal activities, thus leading to financial loss, cyber war, and cybercrime. It is a major challenge for governments, social justice organizations and security agencies to fight terrorism all around the world. This work, to the best of our knowledge, is the first attempt to retrieve sensitive data from big data through the use of Hadoop, annotations, lemmatization, TF-IDF, and SVD. As we determine criminal data from huge mass of unstructured data sets using distributed computing environments, this work constitutes a milestone for national and international agencies in combating global terrorism.

Terrorism has gained global attention due to its negative impact on the economy [6] and its effect on developing countries. It is also against international law and human rights. Therefore, it cannot be tolerated or justified. It must be prevented and fought at national and international levels [7–9]. Terrorists try to achieve political or social objectives through the use of violence by individuals or groups, while also attempting to attract the focus of a large audience compared to the immediate target. It creates unwanted pressure on governments for the purpose of political or personal benefit. In [10], sensitive data is a term used to describe the data related to murder, criminals, burglaries, explosions, domestic violence, druggists, thieves, and cybercriminals that is present in big data. Normally, terrorists use techniques like bombing, kidnapping, assassination, suicide attacks, plane hijacking, etc. [11]. Terrorist organizations make alliances with other terrorist groups, resulting in a strong and active network of terrorist organizations. This type of relationship increases a group's capacity and increases the ability to regulate deadly attacks. It is more harmful when they establish relationships with the strongest groups to whom they can gain access [12]. Although the study of terrorism and response to terrorism (START) [13] deals with a network of scholars studying the causes and consequences of terrorism, it remains a global challenge in monitoring terrorist groups to identify criminal information in big data for the estimation of the potential risk to countries, cities, and societies.

Big data is the collection of large data sets generated from many domains [14], including social media, cloud drives, sensors, geospatial locations, and mobile technologies. These data are in unstructured, semi-structured and structured formats [15]. Big data is characterized by the seven Vs: volume, velocity, variety, veracity, variability, visualization, and value [16]. Volume represents the size of the data sets, velocity represents the rate at which data is transmitted, variety represents different types of data sets, veracity refers to anomalies and uncertainties in the data, variability concerns data whose meaning is constantly changing, visualization refers to illustrating the relationship among data, and finally, value represents the context and importance of the data [17]. Among all of these characteristics, data size is the most distinctive feature of big data analytics compared to the other features [18]. The use of big data analytical tools and techniques [19] is very useful for identifying sensitive data, because they use parallel and distributed computing to process voluminous data [20]. To prevent unexpected event that may occur in future, security agencies must be one step ahead of terrorists [21]. Criminals make policies and sometimes change their moves very quickly. In these scenarios, the use of big data analytics will be helpful in identifying criminal activities even if they change their moves and policies [22]. To predict future terrorist incidents, Toure and Gangopadhyay [23] developed a risk model for computing terrorism risk at different locations, while Nie and Sun [24] carried out systematic research for counter-terrorism using quantitative analysis, demonstrating the impact of big data in fighting terrorism through data collection, processing, monitoring, and warning. However, these approaches are inefficient for processing the huge and exponentially growing amount of data, and it is also difficult for traditional statistical methods of computation to capture, store, process, and analyze such data in a given time frame [25,26].

To process massive unstructured data sets, parallel and distributed computing frameworks, along with Hadoop clusters, are widely used with Yet Another Resource Negotiator (YARN), which is employed by Apache Software Foundation in Hadoop 2.0 [27]. It uses container-based resource partitioning to assign resources to the subdivided units of computations. With the use of Hadoop, Hu et al. [28] designed and implemented a job scheduler without previous knowledge of the job size, and effectively mimicked shortest job first scheduling. This job scheduler assisted the design of multiple level priority queues, where jobs were first assigned to lower-priority queues if the amount of service reached a certain threshold. In [29], Kim et al. introduced the benefits of input split in decreasing container overhead without the modification of the existing YARN framework, presented a logical representation of HDFS blocks, and increased the input size of the container by combining the multiple HDFS blocks. Meanwhile, Zhao and Wu [30] implemented elastic resource management for YARN to increase cluster efficiency through better use of hardware resources by employing a fluctuating container size to fulfill the actual resource requirements of the tasks. Despite the fact that several studies have been completed with Hadoop and YARN, and they have demonstrated their importance in the field, scanty or no literature can be found on the retrieval of criminal data using YARN. Moreover, Hadoop and YARN alone cannot solve the problem of mining criminal data until algorithmic methods have been systematically implemented for further research using Hadoop clusters.

Term Frequency–Inverse Document Frequency (TF-IDF) is a numerical method for determining the importance of a word in a collection of documents [31]. The weight factor of TF-IDF increases with increasing number of words present in the document, but decreases with the number of documents present in the corpus. In [32], Yan and Zhou formulated a binary classification problem for modern aircraft maintenance systems in which faults occurring in past flights were identified from raw data, and Term Frequency–Inverse Document Frequency was applied for feature extraction. To classify documents for news articles in Bahasa, Indonesia, Hakim et al. [33] implemented the TF-IDF classifier along with tokenization, duplicate removal, stop word removal, and supervised word removal, and achieved terms with higher accuracy. Meanwhile, Inyaem et al. [34] employed TF-IDF to classify terrorism events through the use of similarity measures and implemented a finite-state algorithm to find feature weights and event extraction algorithms to enhance the performance. According to [35], Alghamdi and Selamat implemented TF-IDF and the Vector Space Model (VSM) to find topics in Arabic Dark websites, and the K-means clustering algorithm to group the terms, and they also implemented the concept of filtering, tokenization and stemming. In order to mine criminal information from online newspaper articles, Arulanandam et al. [36] employed Named Entity Recognition algorithms to identify thief-related information in sentences based on locations, and achieved 80–90% accuracy, but they had still employed a traditional method of computation. Most of this research adopted the numerical method of TF-IDF to determine the importance of words or to classify terrorism events, but a means for combining TF-IDF with data reduction techniques in order to reduce the size of the data and expose criminal information has yet to be reported.

Dimensionality reduction is an important data processing technique [37] when processing huge amounts of data using singular value decomposition (SVD) [38]. The complex representation of higher dimensional data makes the decision space more complex and time-consuming in big text classification [39]. Therefore, SVD is one an important tool for generating different data sets by maintaining the originality of the information. In [40], Radhoush et al. utilized SVD and the Principle Component Analysis (PCA) algorithm to feed compressed big data into the state estimation step, and showing that compressed data save time and speed up processing in distributed systems. To measure the difference between the original dataset and the distorted dataset for the degree of privacy protection, Xu et al. [38] proposed a sparsified SVD for data distortion techniques. According to [41], Feng et al. proposed an orthogonal tensor SVD method in the field of cybersecurity and cyber forensics, achieving promising results for higher-order big data reduction. In [10], Adhikari et al. implemented two different machine learning algorithms—KNN and NN—to identify sensitive data from big data with Hadoop clusters, but they only considered the performance measures for mining the criminal information. Cho and Yoon [42] presented an approach for identifying important words for sentiment classification using set operations and implementing truncated singular value decomposition and low-rank matrix decomposition techniques to retrieve documents in order to resolve issues related to noise removal and synonymy. Based on this research, we can see that SVD has been implemented for the reduction of the size of voluminous data in order to feed compressed big data into the state estimation, to measure the difference between the original dataset and the distorted dataset, and to resolve issues related to noise removal, but there exists a big literature gap with respect to identifying accurate criminal data in big data using Hadoop clusters for combating global terrorism.

According to the above studies, we argue that the existing approaches are insufficient to account for the exponential growth of data, and we also find that the traditional computational methods are ineffective in the retrieval of criminal data from voluminous data. Moreover, it is also argued that the algorithmic approach should be implemented with the Hadoop cluster to achieve higher performance for the retrieval of sensitive data. Based on these arguments, this research is aimed at the accurate identification of criminal data from the huge mass of varieties of data collected from diversified fields through the use of Hadoop clusters to support the criminal justice system and social justice organizations in fighting terrorist activities at a global scale. The achievement of this experiment is rigorously compared using the same set of hardware, software, and system configuration. Moreover, the efficacy of the experiment was tested using criminal data with respect to concepts and matching scores.

2. Research Methodology

To identify sensitive information within big data, several methods were implemented chronologically in a distributed computing environment. A systematic representation of the implementation of the methodology is depicted in Figure 1. Firstly, documents that contained criminal information were stored in the Hadoop Distributed File System. To achieve parallelism and work distribution in Spark, Resilient Distributed Datasets (RDD) were used. Moreover, Spark API was used to read the files in a folder. To provide efficacy of the work through parallelism, SparkContext Driver, Cluster Manager, Worker Nodes were communicated with each other. Moreover, to find patterns and inferences in the datasets, we applied Annotation, in which raw data were passed to the Annotation Object, and after processing with the relevant function, an annotated text was generated. In the next step, to acquire the base or dictionary form of a word from plain text, Lemmatization was implemented with the help of Standard Core NLP project. Still, there were several undesired terms that did not carry too much meaningful information for the analysis of criminal information. Therefore, to reduce the dimensional space, we implemented the StopWord Remover functions that were present in Spark API. Subsequently, we also calculated term frequency and inverse document frequency to extract features from the raw data. For this, we employed CountVector API to find term frequency and generated sparse representation of the document. For quicker analysis of the data, we instigated RowMatrix API for calculation of SVD and computeSVD function for the decomposition of the matrix into its constituent three different matrices, i.e., term matrix, diagonal matrix, and document matrix. Finally, we carried out commands to submit Spark jobs and extracted criminal information. The methodology for the mining of criminal information is systematically presented in the figure.

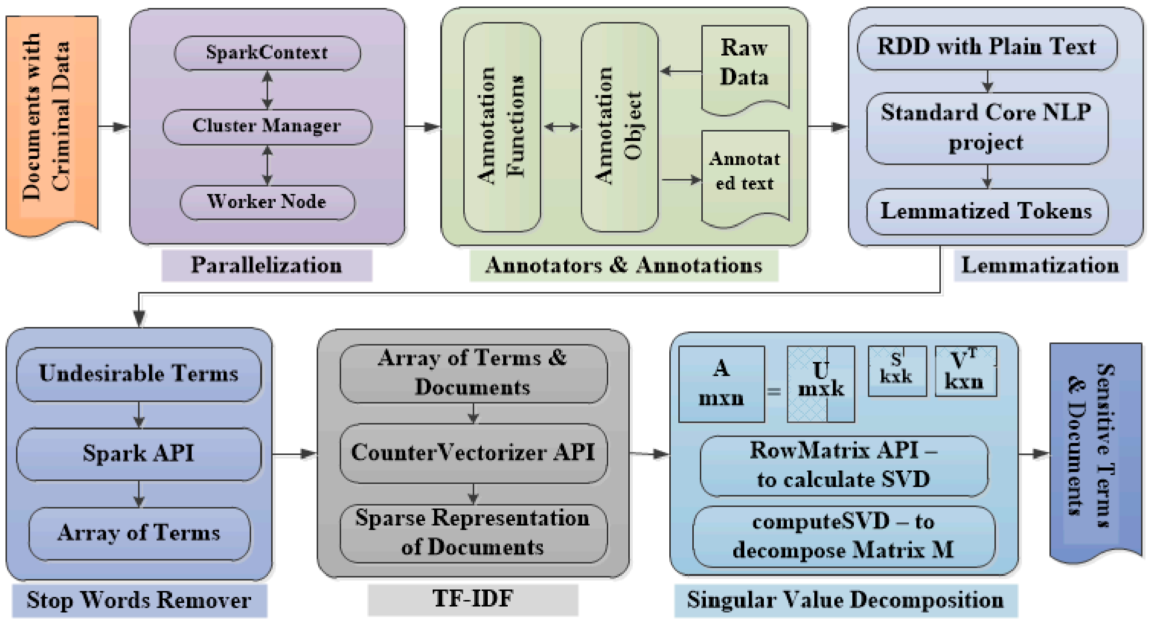

Figure 1. An overview of the proposed method.

2.1. Parallelization

A parallel processing system was devised to speed up the processing of programs by breaking single task units into multiple tasks and executing them simultaneously. In this regard, Spark is very pragmatic for scaling up data science workloads and tasks. Using Spark libraries and data frames, the massive data sets were scaled for distribution across a cluster for parallel processing. The parallelization of algorithms is most significant for resolving large scale data optimization and data analytical problems. A Spark application on a Cluster is graphically presented in Figure 2.

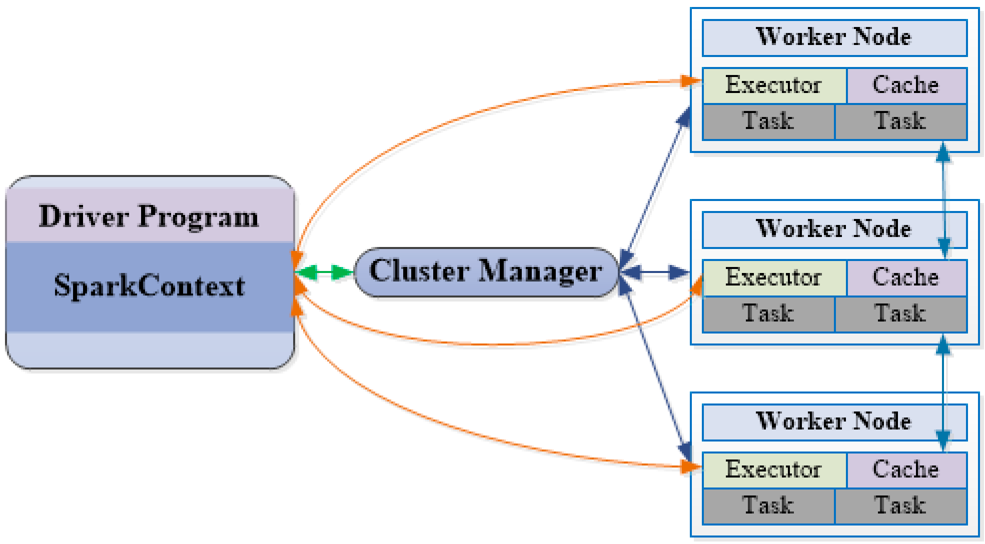

Figure 2. Working mechanism of a Spark Application on a Cluster.

For processing, the job is submitted to the SparkContext driver program, which transfers applications to the cluster manager. The cluster manager communicates with the worker node to create containers that open Executors, and the Executor again interacts with the SparkContext Driver program to carry out parallel processing. The work distribution in Spark is accomplished by means of

Resilient Distributed Datasets (RDDs) that can run a wide range of parallel applications, and many programming models that have been projected for interactive computation.

To speed up the work, we stored our data in the Hadoop Distributed File System and implemented Spark API to read files and folders using sc.wholeTextFiles(). The DataFrame with filename was adopted as the label column, while all texts were accepted as sentence columns. As DataFrame is an optimized version of RDD, it has been selected as a suitable API for various machine learning algorithms.

Dataframes were chosen as suitable APIs developed for different machine learning algorithms. The pseudocodes for implementing parallelization are presented in Appendix A to read data and then convert it to the DataFrame. Then the DataFrame is distributed to executors using mapPartitions API, which distributes the partitions equally to Executors for processing. Spark provides parallelism by using all of its components, but to achieve our goal, we implemented annotators to find patterns and inferences among the data.

2.2. Annotators and Annotations

Annotation is the process of preparing data for finding patterns and inferences by adding metadata tags to web documents, social media, image data, video data, and text documents. The research conducted by Yang and Lee [43] proposed an effective method for the recommendation of tags to annotate a web page and developed the relationships between web pages and tags based on clustering results. In other research, Liu et al. [44] also worked for semantical enhancement of web documents for recommendation and information retrieval systems, but they used a semi-automatic annotation system to annotate metaphors through the use of high abstraction terms. In the research, Ballan et al. [45] reviewed the state-of-the-art approaches to tag refinement for social images and also presented localization of web video sequences and tag suggestions by the use of automatic annotations. Beltagy et al. [46] presented a well-organized annotation system by breaking down the documents into segments, mapping the segment headings to concepts to extend the ontology. Motivated by all of these works, we implemented annotation to extract concepts from the huge mass of text documents. The working mechanism of the annotators is illustrated in Figure 3.

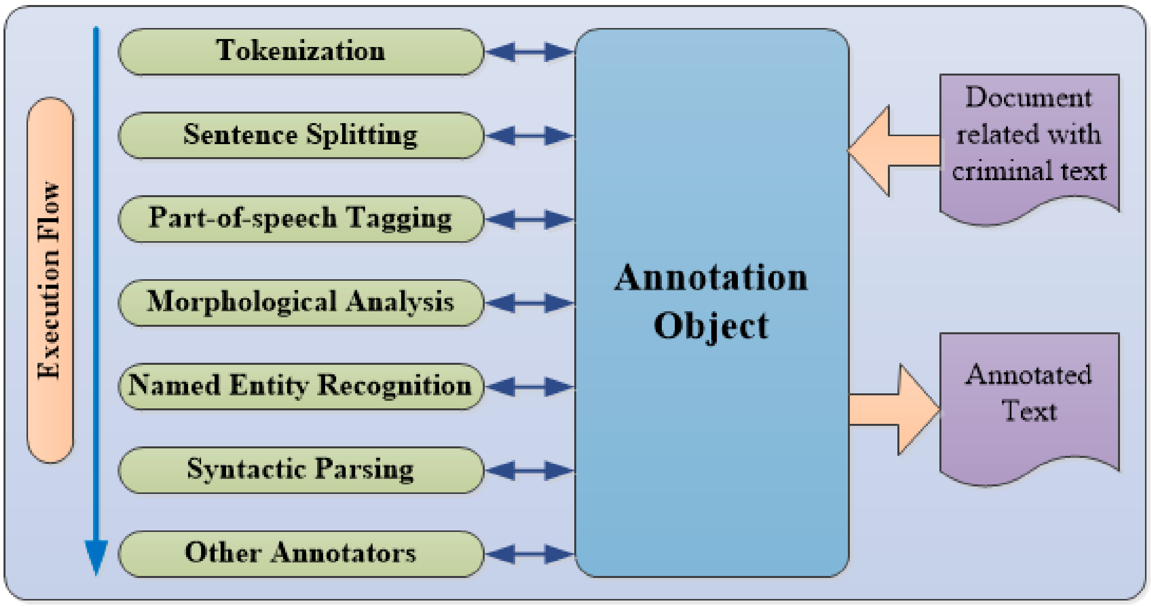

Figure 3. Working mechanism of annotators.

In our work, functions like tokenize, parse, NER tags, ssplit, and pos were employed for tokenization, sentence splitting, part-of-speech tagging, morphological analysis, named entity recognition, and syntactic parsing of the text data. The annotated texts were generated by passing raw text to the Annotation Object and by the implementation of all the defined functions. The pseudocodes for tokenizing the words for lemmatization and breaking the annotated text into sentences are presented in Appendix B.

2.3. Lemmatization

With the annotated text in hand, next, we turned it into a bag of terms in their original forms through morphological analysis and the elimination of inflectional endings. Words with the same meaning can take different forms. For example, "criminal" and "criminals" do not justify separate terms. Merging these different inflectional terms into a single term is called stemming or lemmatization [47]. Stemming implements a heuristic-based technique [48] to remove characters at the end, while lemmatization practices employ principled approaches to reduce inflectional forms to a common base form [49] by recursive processing in different layers. Miller [50] extracted words from the WordNet dictionary, but this was limited to the convenience of human readers and a combination of traditional computing with lexicographic information. As our work is related to extracting the base terms from annotated text, the Stanford Core NLP project [51] provided the best solution. It has an excellent lemmatizer, which uses Java API with the Scala. The pseudocodes presented in Appendix C employed the RDD with plain text documents for lemmatization. Then, the lemmatized tokens are converted to DataFrames for further processing.

2.4. StopWords Remover

StopWords do not carry much meaning in the text and can be removed from the text analysis process. For example, words like "the", "is", "a" take up space but do not carry valuable information for the model. The spatial dimensions can be reduced by removing all the StopWords from the text by using algorithms or functions. Magdy and Jones [52] applied the StopWords removal technique in information retrieval text pre-processing, but this was only limited to the Machine Translation corpus. As our work is related to the RDD environment, we found an efficient method to remove StopWords from the text using Spark API. The snippet used for the removal of StopWords is presented in Appendix D.

2.5. Term Frequency–Inverse Document Frequency (TFIDF)

After the successful implementation of Lemmatization and StopWord Remover, we had an array of terms corresponding to the document. The next challenge was to compute the frequencies of the terms within the document, as well as in the entire corpus so that the importance of the words can be found. For this, we implemented Term Frequency–Inverse Document Frequency. The term frequency counts the number of terms and defines the weight of the terms in the document. The mathematical derivation for the calculation of the term frequency is as follows:

Term frequency is used to extract features from raw data. It is as follows:

$$tf(t,d) = \sum_{i \in d} f(i,t) \tag{1}$$

where $f\ (i,\ t)$ is a function which is defined as

$$f(i,t) = \begin{cases} 1 & if\ i = t \\ 0, & Otherwise \end{cases} \tag{2}$$

The mathematical representation of inverse document frequency is as follows:

$$idf(t) = \log \frac{|D|}{1 + |\{d : t \in d\}|} \tag{3}$$

where $|\{d : t \in d\}|$ represents the term t which is present in document d when $tf(t, d) \neq 0$, and 1 is added to the formula to avoid zero division.

The *tf-idf* combines term frequency and inverse document frequency and its mathematical derivation is as follows:

$$tf - idf\ (t) = tf(t,d)\ x\ idf(t) \tag{4}$$

The CountVectorizer algorithm selects top-frequency words to form vocabulary and generates a sparse representation for the documents over the vocabulary [53]. CountVectorizer is chosen against HashingTF, as HashingTF is dependent on hashing of terms, and collision may occur between terms while hashing. Vocabulary size is limited, with numTerms being variable. In our work, we implemented CountVectorizer API to find the term frequencies and pseudocodes, which are presented in Appendix E, to describe the steps to find the term frequencies. For the generation of a sparse representation of the document and to calculate TF-IDF, the snippets of codes are presented in Appendix E. IDF is an estimator which takes the vectorized counts produced by the CountVectorizer (TF algorithm used) and gives a weighting to the terms which are frequently used in the document.

2.6. Singular Value Decomposition (SVD)

At this point, we have constructed the term-document matrix M, which can be processed further for factorization and dimensionality reduction. The SVD takes the $m \times n$ matrix as the input and returns three different matrices, i.e., term matrix, diagonal matrix, and document matrix.

$$M = U\ S\ V^T \tag{5}$$

- U is a $m \times k$ matrix where the terms are represented by rows, while columns correspond to concepts.
- S is a $k \times k$ diagonal matrix whose entries represent the strength of the individual concepts. The magnitude of the individual values is signified by the importance of the concepts, and this importance can be shown by the variance in the data. A key perception of the LSI can be explained by the small number of concepts that are used to describe the data. The entries in the diagonal matrix S are directly related to the importance of each concept.
- V is a $k \times n$ matrix where documents are represented by documents, while the concepts are characterized by columns.

In Latent Semantic Analysis (LSA), m represents the number of terms, while n represents several documents. The decomposition of the matrix is parameterized with a number k, which is less than or equal to the original number n, which indicates how many concepts should be considered during analysis. If $k = n$, the product of the matrices constitutes the original matrix. However, if $k < n$, the product of the matrices generates a low-rank approximation of the original matrix. Normally, k is selected to be much lower than n in order to explore the concepts signified in the documents.

As we had RDD data to analyze, we implemented RowMatrix API to calculate Singular Value Decomposition. The snippet of the codes is presented in Appendix F. A RowMatrix is backed up by rows of an RDD, where each row represents a local vector. This is a row-oriented distributed matrix that has no meaningful row indices, and it was used to calculate the column summary statistics and decomposition. The computeSVD is a dimension reduction function that is used to decompose a matrix into its three different constituent matrices.

2.7. Command to Execute Spark Job

To prove the efficacy of our work, we extensively evaluated the experiment against criminal data by executing the Spark job by using commands. The commands for running the code are presented in Appendix G.

2.8. Running Environment

Experiments were conducted on a five-node cluster, with the configuration of each node being an Intel®® quad-core CPU @2.70 GHz, 8 GB RAM and 500 GB storage with Ubuntu 16.04LTS operating system. The master and the slave environments were also set up for the experiment. For the experiment, four text files were collected from website: http://www.gutenberg.org/ and employed for the processing. These files contained varieties of sensitive text data in an unstructured way. The files were "mob_rule_in_new_orleans.txt", "murder_in_the_gunroom.txt", "the_Leavenworth_ case.txt", and "the_way_we_live.txt" and their data sizes were 1.41 MB, 4.02 MB, 6.435 MB, and 3.88 MB, respectively. SVD is an algorithm that transforms a corpus/documents into matrix format so that the searching of texts is easy and fast. Therefore, the training and test data depend on the number of terms used and the decomposition factor (k) used. the higher the values, the higher the volume of training and test data, and vice-versa. In this experiment, validation data is manual, which involves looking at the matching score when certain words were searched.

3. Results and Discussion

To prove the efficacy of our methodology, we conducted experiments on five-node clusters as well as on a one-node cluster by maintaining the proper empirical environment, and calculated the average execution time in seconds against number of concepts (the value of k) and the number of terms (the value of numTerms). Moreover, we calculated the matching scores with respect to terms and documents. The matching score of the terms is presented in the following format.

$$\text{(Terms, Matching Score)} \tag{6}$$

where "Term" represents the text used for queries, and "Matching Score" represents the degree to which the terms matched with the queries. If the matching score is "1", this represents complete matching, but if it is "0", this represents no matching at all.

For the validation of the result, other queries were performed to find the top documents that contained more criminal information. The matching score of the top documents is presented in the following format.

$$\text{(Documents, Matching Score)} \tag{7}$$

If the matching score is "1", this indicates that the document only contains criminal information, while if it is "0", this indicates that the document contains no sensitive information.

3.1. Five-Node Cluster

The average time taken for processing was calculated for varying numbers of concepts and numbers of terms in five-node clusters. It was found that if k = 1210 and numTerms = 11,500, then execution time was 927 s; and if k = 20 and numTerms = 13,000, then the execution time was 152 s. This indicates that as the number of concepts and the number of terms increases for execution, the time taken for processing also increases for identifying criminal activities. The graphical representation of the result is illustrated in Figure 4.

If we are able to identify criminal activities, as described in [10,11], it will help in controlling the negative impact on the economy, and in fighting terrorist activity at national and international levels. This will also restrict the establishment of relationships among terrorist groups. Moreover, it provides solutions for monitoring terrorist groups and minimizes the potential risk to countries, cities, and societies.

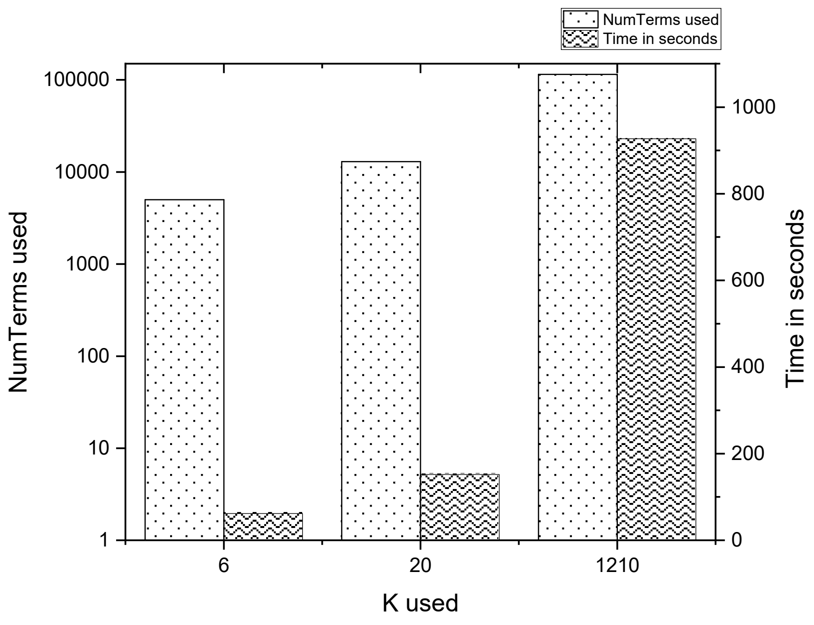

Figure 4. Average execution time based on number of concepts and terms for 5-node clusters.

3.1.1. For k = 1210 and numTerms = 115,000

The top terms for the crime with their matching score are presented in Table 1.

Table 1. Top Terms for the Crime.

Terms	Matching Score	Terms	Matching Score
crime	0.9999999999999999	party	0.9999982845529726
hypertext	0.9999982258389528	great	0.9999979315862131
hand	0.9999978642889132	resist	0.9999977194098939
impose	0.9999974439918138	believe	0.9999961301511628
hart	0.9999948639086803	narrow	0.9999948603199577

For the experimental results of $k = 1210$ and numTerms = 115,000, the matching score of the criminal terms within the documents are presented in Table 2. It was found that the term "crime" within the document had matching score 0.9999999999999999. This indicates that criminal terms had a maximal matching with the text present in the file, while the term "narrow" had a matching score of 0.9999948603199577, indicating that the term "narrow" had less matching compared to the term "criminal".

Table 2. Top Docs for TermQuery: Prison, Kill, Destroy.

Documents	Matching Score
the_way_we_live.txt	0.796688711889935
mob_rule_in_new_orleans.txt	0.19917217797246592
the_leavenworth_case.txt	0.09958608898622971
murder_in_the_gunroom.txt	-2.6454533008646308E-17

In these experimental results, terms like a prison, kill, and destroy were queried against the files containing criminal data, and it was found that the file "the_way_we_live.txt" had more matching criminal terms compared to "murder_in_the_gunroom.txt".

3.1.2. For *k* = 20 and numTerms = 13,000

The top terms for the term "crime" are presented in Appendix H. For the experimental reuslts of *k* = 20 and numTerms = 13,000, most of the terms had a score of 1.0, showing that the matching term frequencies were approximate one.

The top docs for TermQuery: prison, kill, and destroy are presented in Appendix H. In these experimental results, it was found that the file "the_way_we_live.txt" had more criminal information compared to the file "murder_in_the_gunroom.txt", because the file "the_way_we_live.txt" had a higher matching score compared with the others.

3.1.3. For *k* = 6 and numTerms = 5000

The top terms for the term "crime" are presented in Appendix I. For the experimental results of *k* = 6 and numTerms = 5000, the matching score for the term "crime" was 1.0, while the term "building" was 0.99998513214733l8. This shows that the term "crime" has more similar terms than the term "building".

The top docs for TermQuery: prison, kill, and destroy are depicted in Appendix I. In these experimental results, it was found that the file "the_way_we_live.txt" contained more criminal words than the file "murder_in_the_gunroom.txt".

From all the above results, we can say that although there was a huge number of different types of unstructured data [14,16–18] collected from a diversity of fields, our studies helped to identify criminal data in several documents, as well as those documents containing the maximum numbers of criminal terms. In this regard, this research could help to notify security agencies in advance [21], even if criminals change their moves [22].

3.2. *One-Node Cluster*

Experiments were completed for the one-node cluster by changing the values of *k* and numTerms in accordance with three different scenarios, and the average time taken for processing was recorded. The obtained results are presented in Figure 5, which depicts the average execution time for determining whether the criminal data is proportional to the increase in the number of concepts and the number of terms used.

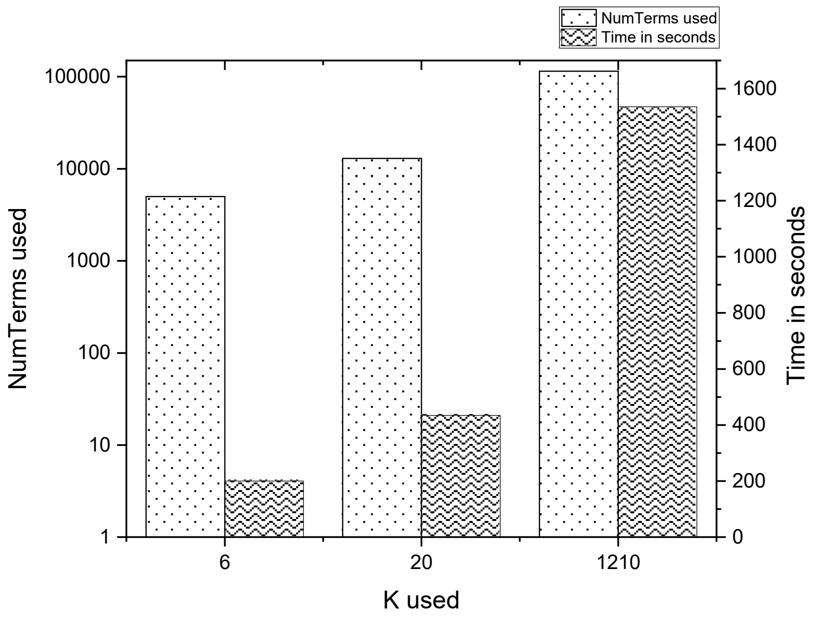

Figure 5. Average execution time based on number of terms and concepts for 1-node cluster.

Despite the fact that several studies [28–30] have employed Hadoop in different fields in order to distribute the work among several computers in a cluster, this work also adds one milestone, using YARN for the detection of sensitive data and helping the criminal justice system to control criminal activities. Moreover, most of the research combined the numerical methods of TF-IDF [32–35] with different approaches to present its importance. In this regard, we instigated TF-IDF with a data reduction technique in order to identify criminal information and demonstrate its importance. Furthermore, in [36], Arulanandam et al. executed Named Entity Recognition algorithms to identify thief-related information, achieving 80–90% accuracy; however, with respect to the retrieval of criminal terms, we achieved 100% accuracy, and in matching multiple terms with the document, we obtained an accuracy of up to 80% in the distributed computing environment using Hadoop Clusters. As we implemented our work using several computers in a cluster in order to mine sensitive information from big data, this exceeds the ability of traditional statistical methods of computation to store, capture, process and analyze the data within a fixed time frame [25,26]. We also conducted experiments on five-node clusters and one-node clusters in order to compare single-node and multiple-node processing, and we found that the performance for criminal data retrieval was more effective when using the five-node clusters than with the one-node clusters.

For further analysis, the terms were queried against the listed files, and their matching scores were as follows:

3.2.1. For $k = 1210$ and numTerms = 115,000

Top terms for the term "crime" are tabulated in Appendix J. In these experimental results, it was found that the term "crime" had a matching score of 1.000000000000000011, while the term "anybody" had matching score of 0.9964465159802695. This demonstrates that terms related to "crime" can be obtained based on the maximum response in comparison with other terms.

The top docs for TermQuery: prison, kill, and destroy are tabulated in Appendix J. In these experimental results, it was found that the file "the_way_we_live.txt" contained more sensitive data in comparison to the file "murder_in_the_gunroom.txt", because the file "the_way_we_live.txt" had a higher matching score.

3.2.2. For $k = 20$ and numTerms = 13,000

The top terms for the term "crime" are tabulated in Appendix K. In these experimental results, it was found that all the terms related to "crime" could be identified, while the term "president" was less related to criminal data.

The top docs for TermQuery: prison, kill, and destroy are tabulated in Appendix K. For the document analysis, it was found that "the_way_we_live.txt" contained more criminal-related data, while the file "murder_in_the_gunroom.txt" contained less sensitive data, because the file "the_way_we_live.txt" had a greater matching score compared to the other files.

3.2.3. For $k = 6$ and numTerms = 5000

The top terms for the term "crime" are presented in Appendix L. In the case of the experimental results of k = 6 and numTerms = 5000, the information retrieval for sensitive data was more than for the other terms, because the matching score for the term "crime" had a higher matching score.

The top docs for TermQuery: prison, kill, and destroy are tabulated in Appendix L. In this experiment, it was found that there were more terms related to criminal information in the file "the_way_we_live.txt" than there were in the remaining files, because that file had a higher matching score than the other files.

In the above results, we achieved mining of criminal terms with 100% accuracy, and documents containing maximum criminal terms with 80% accuracy. Although there are several studies [38,40–42] that implemented Singular Value Decomposition to prove its importance in related fields, we executed SVD with annotation and annotators, Lemmatization, StopWord Remover, and TF-IDF to retrieve criminal information in a distributed computing environment and proved its importance. Furthermore, in [10], the researchers only considered evaluation as a performance measure, while we also considered the efficiency and accuracy of the work in demonstrating its significance.

Overall, from all of the above results and discussion, we can say that our work is unique and demonstrates its importance in the field and could help the criminal justice system and social justice organizations to fight criminals at national and international levels [7–9], while assisting administrative authorities in minimizing expense [3,4] and controlling negative impacts on the economy. Moreover, our work could help to antagonize other terrorist groups [12]. Furthermore, our research is a milestone in combating global terrorism.

4. Conclusions

Due to the fact that terrorism creates a negative impact on the economy, establishes relationships with other terrorist groups, and creates potential risks for societies at national and international levels, this paper chose to study methods of identifying sensitive data from big data using a distributed computing environment, annotation, the StopWords Removal technique, Lemmatization, Term Frequency and Inverse Document Frequency, and Singular Value Decomposition. From the experimental results, it was found that the identification of sensitive terms was carried out with 100% accuracy, while matching of different crime-related terms with documents was performed with 80% accuracy. Furthermore, this study was carried out using a huge number of different types of unstructured data collected from diversified fields through the successful implementation of the Hadoop cluster. Therefore, it exceeds the ability of traditional statistical methods of computation to store, capture, process and analyzes data within a fixed time frame in order to support the criminal justice system in supervising criminal activities. Moreover, to identify criminal data within the huge mass of data, several different algorithms, including parallelization, annotator and annotation, Stop Word Remover, Lemmatization, TF-IDF, and SVD, were implemented in combination to achieve higher accuracy and performance, so that this research can help social justice organizations and the criminal justice system to minimize administrative expenses and combat terrorism at a global scale.

In order to identify criminal data from within the huge mass of different types of data, this research could be further extended through the implementation of classification algorithms like support vector machine, K-nearest neighbor algorithm, etc., in distributed computing environments in order to improve efficacy. Furthermore, the existing approach could be applied to cloud computing platforms so that the performance and accuracy could also be evaluated online to detect criminal data to fight global terrorism.

Author Contributions: Conceptualization, B.K.A.; Data curation, R.M. and S.L.; Formal analysis, R.M.; Funding acquisition, W.Z. and X.H.; Methodology, B.K.A.; Project administration, X.H.; Resources, S.L.; Supervision, W.Z.; Writing—original draft, B.K.A.; Writing—review & editing, W.Z. and R.M. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: The research was supported by the National Natural Science Foundation of China under grant No. 61976103, 61472049, 61402193 and 61572225; the National Society Science Foundation of China under grant No.15BGL090; the Jilin Development and Reform Commission Project; the Education Department Science and Technology Project No. JJKH20181048KJ; the Jilin Department of Science and Technology Research Project No. 20180101330JC, 20190302029GX, 20190302071GX.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Parallelization

Pseudocode 1: Reading data and converting to dataframe.

```
val sentenceData = sqlContext.createDataFrame(sc.wholeTextFiles(input)      .map(x =>
(x._1.substring(x._1.lastIndexOf("/") + 1), x._2.replace("\n", " ")))).toDF("label", "sentence").as[table]
```

Appendix B. Annotators and Annotations

Pseudocode 2: Annotator to tokenize for lemmatizing the words

```
val props = new Properties()
props.put("annotators", "tokenize, ssplit, pos, lemma")
val pipeline = new StanfordCoreNLP(props)
val doc = new Annotation(text)
pipeline.annotate(doc)
```

Then the annotated texts are broken down to sentences using the following codes:

Pseudocode 3: Annotated text broken down to sentences

```
val sentences: java.util.List[CoreMap] = doc.get(classOf[SentencesAnnotation])
```

The sentences are tokenized and finally tokenized words are used for lemmatization.

Appendix C. Lemmatization

Pseudocode 4: Lemmatization of tokenized word

```
for (sentence <- sentences; token <- sentence.get(classOf[TokensAnnotation]))
{
val lemma = token.get(classOf[LemmaAnnotation])
}
```

Appendix D. Stop Words Remover

Pseudocode 5: The snippet used for removal of StopWords are as follows:

```
import org.apache.spark.ml.feature.StopWordsRemover
val remover = new StopWordsRemover().setInputCol("terms") .setOutputCol("filtered")
val filtered = remover.transform(lemmatized).select(col("title"), col("filtered").as("terms"))
```

Appendix E. Term Frequency—Inverse Document Frequency

In our work, we implemented CountVectorizer api to find the term frequencies and its pseudocodes are presented below:

Pseudocode 6: CountVector api to find the term frequncies

```
val countVectorizer = new CountVectorizer().setInputCol("terms").setOutputCol("termFreqs"). setVocabSize
(numTerms)
val vocabModel = countVectorizer.fit(filtered)
val docTermFreqs = vocabModel.transform(filtered)
val termIds = vocabModel.vocabulary
```

Pseudocode 7: Generation of sparse representation for the document

```
val idf = new IDF().setInputCol("termFreqs").setOutputCol("tfidfVec")
val idfModel = idf.fit(docTermFreqs)
val docTermMatrix = idfModel.transform(docTermFreqs).select("title", "tfidfVec")
val termIdfs = idfModel.idf.toArray
```

Appendix F. Singular Value Decomposition

Pseudocode 8: RowMatrix api for calculation of SVD

```
val vecRdd = docTermMatrix.select("tfidfVec").rdd.map
{
row =>Vectors.fromML(row.getAs[MLVector]("tfidfVec"))
}
val mat = new RowMatrix(vecRdd)
val svd = mat.computeSVD(k, computeU=true)
```

Appendix G. Command to Execute the Spark Job

After setting up all the environments for the work, we had submitted following command to execute the Spark job.
spark-submit –class LSI –master yarn –deploy-mode client machinelearning.jar <input> K numTerms
Where k is a constant which defines the number of concepts to be formed and numTerms is used for limiting the number of Terms used by CountVectorizer for selecting the top frequency words for the formation of vocabulary.

Appendix H. For k = 20 and numTerms = 13,000

Table A1. Top terms for term crime.

Terms	Matching Score	Terms	Matching Score
crime	1.0	tell	1.0
would	1.0	look	1.0
think	1.0	lady	1.0
one	1.0	time	1.0
come	1.0	find	1.0

Table A2. Top Docs for TermQuery: prison, kill, and destroy.

Documents.	Matching Score
the_way_we_live.txt	0.7966887118899392
mob_rule_in_new_orleans.txt	0.19917217797246578
the_leavenworth_case.txt	0.09958608898622988
murder_in_the_gunroom.txt	-2.2985086056692694E-17

Appendix I. For k = 6 and numTerms = 5000

Table A3. Top terms for term crime.

Terms	Matching Score	Terms	Matching Score
crime	1.0	hurry	0.9999986123095814
car	0.9999985895469953	kind	0.999997758568858
upper	0.9999949902400325	belief	0.9999935062735485
future	0.9999905181559015	opinion	0.9999872392433281
rather	0.9999868020185473	building	0.9999851321473318

Table A4. Top Docs for TermQuery: prison, kill, and destroy.

Documents	Matching Score
the_way_we_live.txt	0.796688711889875
mob_rule_in_new_orleans.txt	0.19917217797247003
the_leavenworth_case.txt	0.09958608898623503
murder_in_the_gunroom.txt	2.951198313505543E-16

Appendix J. For k = 1210 and numTerms = 115,000

Table A5. Top terms for term crime.

Terms	Matching Score	Terms	Matching Score
crime	1.0000000000000000	past	0.999283970251538
chief	0.9989530950843942	store	0.9989184455648151
immediately	0.9982570426229813	native	0.9976558984517548
across	0.9974812337215039	necessary	0.9973599839279012
active	0.9970177153885116	anybody	0.9964465159802695

Table A6. Top Docs for TermQuery: prison, kill, and destroy.

Documents	Matching Score
the_way_we_live.txt	0.7966887118899464
mob_rule_in_new_orleans.txt	0.19917217797246276
the_leavenworth_case.txt	0.09958608898623116
murder_in_the_gunroom.txt	-2.3099360965672666E-14

Appendix K. For k = 20 and numTerms = 13,000

Table A7. Top terms for term crime.

Terms.	Matching Score	Terms	Matching Score
crime	1.0	justify	0.9999870694427458
source	0.9999883358181888	live	0.9999849222467917
stop	0.9999865140050637	sight	0.9999781311786513
height	0.9999801708501026	box	0.999973434101214
dark	0.9999669274497553	president	0.9999610395060395

Table A8. Top Docs for TermQuery: prison, kill, and destroy.

Documents	Matching Score
the_way_we_live.txt	0.7966887118899374
mob_rule_in_new_orleans.txt	0.19917217797246542
the_leavenworth_case.txt	0.09958608898623014
murder_in_the_gunroom.txt	−1.0018028073766061E-16

Appendix L. For k = 6 and numTerms = 5000

Table A9. Top terms for term crime.

Terms	Matching Score	Terms	Matching Score
crime	1.0000000000000000	certain	0.9999940855675256
set	0.9999865953543935	member	0.9999657992901405
much	0.9999570363529842	satisfied	0.9999361561508657
whistle	0.9999361561508657	direction	0.9999252086255683
red	0.9999252086255683	mississippi	0.9999252086255683

Table A10. Top Docs for TermQuery: prison, kill, and destroy.

Documents	Matching Score
the_way_we_live.txt	0.796688711889875
mob_rule_in_new_orleans.txt	0.19917217797246992
the_leavenworth_case.txt	0.09958608898623521
murder_in_the_gunroom.txt	4.679416576447437E-16

References

1. Dartnell, M. A Legal Inter-Network For Terrorism: Issues of Globalization, Fragmentation and Legitimacy. *Terror. Polit. Violence* **1999**, *11*. [CrossRef]
2. Weinberg, L.; Pedahzur, A.; Hirsch-Hoefler, S. The challenges of conceptualizing terrorism. *Terror. Policical Violence* **2004**, *16*, 777–794. [CrossRef]
3. Koc-Menard, S. Trends in terrorist detection systems. *J. Homel. Secur. Emerg. Manag.* **2009**, *6*. [CrossRef]
4. Blalock, G.; Kadiyali, V.; Simon, D.H. Driving fatalities after 9/11: A hidden cost of terrorism. *Appl. Econ.* **2009**, *41*, 1717–1729. [CrossRef]
5. Strang, K.D.; Alamieyeseigha, S. What and where are the risks of international terrorist attacks: A descriptive study of the evidence. *Int. J. Risk Conting. Manag. (IJRCM)* **2015**, *4*, 1–20. [CrossRef]
6. Arfsten, K.-S. Before, Now, and After the Event of Terror: Situational Terror Awareness for Civilians in US Homeland Security. *Eur. J. Secur. Res.* **2019**. [CrossRef]
7. Bowman Grieve, L.; Palasinski, M.; Shortland, N. Psychology perspectives on community vengeance as a terrorist motivator: A review. *Safer Communities* **2019**, *18*, 81–93. [CrossRef]
8. Matusitz, J. Symbolism in Female Terrorism: Five Overarching Themes. *Sex. Cult.* **2019**, *23*, 1332–1344. [CrossRef]
9. Bordo, J. Walter Benjamin at the City Library of Berlin: The New Library as Incident Room. *Images* **2019**, *12*, 97–121. [CrossRef]
10. Adhikari, B.K.; Zuo, W.L.; Maharjan, R.; Guo, L. Sensitive Data Detection Using NN and KNN from Big Data. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, Guangzhou, China, 15–17 November 2018; pp. 628–642.
11. Sandler, T.; Enders, W. Economic consequences of terrorism in developed and developing countries. *Terror. Econ. Dev. Polit. Openness* **2008**. [CrossRef]
12. Horowitz, M.C.; Potter, P.B.K. Allying to kill: Terrorist intergroup cooperation and the consequences for lethality. *J. Confl. Resolut.* **2014**, *58*, 199–225. [CrossRef]
13. Santos, C.; El Zahran, T.; Weiland, J.; Anwar, M.; Schier, J. Characterizing Chemical Terrorism Incidents Collected by the Global Terrorism Database, 1970–2015. *Prehospital Disaster Med.* **2019**, *34*, 385–392. [CrossRef] [PubMed]
14. Chen, C.L.P.; Zhang, C.-Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Inf. Sci.* **2014**, *275*, 314–347. [CrossRef]
15. Adhikari, B.K.; Zuo, W.; Maharjan, R. A Performance Analysis of OpenStack Cloud vs Real System on Hadoop Clusters. In Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 194–201.
16. Uddin, M.F.; Gupta, N. Seven V's of Big Data understanding Big Data to extract value. In Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education, Bridgeport, CT, USA, 3–5 April 2014; pp. 1–5.
17. Gobble, M.M. Big data: The next big thing in innovation. *Res. Technol. Manag.* **2013**, *56*, 64–67. [CrossRef]
18. Strang, K.D. Selecting research techniques for a method and strategy. In *The Palgrave Handbook of Research Design in Business and Management*; Springer: New York, NY, USA, 2015; pp. 63–79.
19. Adhikari, B.K.; Zuo, W.L.; Maharjan, R.; Yadav, R.K. Use of Big Data Analytics in WASH Sector. In Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 14–15 June 2018; pp. 1185–1190.
20. How to Detect Criminal Gangs Using Mobile Phone Data. Available online: https://www.technologyreview.com/s/526471/how-to-detect-criminal-gangs-using-mobile-phone-data/ (accessed on 1 October 2019).

21. Sanderson, T.M. Transnational terror and organized crime: Blurring the lines. *SAIS Rev. Int. Aff.* **2004**, *24*, 49–61. [CrossRef]
22. Data Protection Act. Available online: https://www.huntonprivacyblog.com/wp-content/uploads/sites/28/2016/11/big-data-and-data-protection.pdf (accessed on 1 October 2019).
23. Toure, I.; Gangopadhyay, A. Real time big data analytics for predicting terrorist incidents. In Proceedings of the 2016 IEEE Symposium on Technologies for Homeland Security (HST), Waltham, MA, USA, 10–11 May 2016; pp. 1–6.
24. Nie, S.; Sun, D. Research on counter-terrorism based on big data. In Proceedings of the 2016 IEEE International Conference on Big Data Analysis (ICBDA), Hangzhou, China, 12–14 March 2016; pp. 1–5.
25. Adhikari, B.K.; Zuo, W.; Maharjan, R.; Han, X.; Ali, W. Statistical Analysis for Detection of Sensitive Data Using Hadoop Clusters. In Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 10–12 August 2019; pp. 2373–2378.
26. Snijders, C.; Matzat, U.; Reips, U.-D. "Big Data": Big gaps of knowledge in the field of internet science. *Int. J. Internet Sci.* **2012**, *7*, 1–5.
27. Polo, J.; Carrera, D.; Becerra, Y.; Torres, J.; Ayguadé, E.; Steinder, M.; Whalley, I. Performance-driven task co-scheduling for mapreduce environments. In Proceedings of the 2010 IEEE Network Operations and Management Symposium—NOMS 2010, Osaka, Japan, 19–23 April 2010; pp. 373–380.
28. Hu, Z.; Li, B.; Qin, Z.; Goh, R.S.M. Job scheduling without prior information in big data processing systems. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 572–582.
29. Kim, W.; Choi, Y.-R.; Nam, B. Mitigating YARN container overhead with input splits. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 204–207.
30. Zhao, Y.; Wu, G. Yadoop: An Elastic Resource Management Solution of YARN. In Proceedings of the 2015 IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco Bay, CA, USA, 30 March–3 April 2015; pp. 276–283.
31. Leskovec, J.; Rajaraman, A.; Ullman, J.D. *Mining of Massive Datasets*; Cambridge University Press: New York, NY, USA, 2014.
32. Yan, W.; Zhou, J.-H. Predictive modeling of aircraft systems failure using term frequency-inverse document frequency and random forest. In Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 10–13 December 2017; pp. 828–831.
33. Hakim, A.A.; Erwin, A.; Eng, K.I.; Galinium, M.; Muliady, W. Automated document classification for news article in Bahasa Indonesia based on term frequency inverse document frequency (TF-IDF) approach. In Proceedings of the 2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, 7–8 October 2014; pp. 1–4.
34. Inyaem, U.; Meesad, P.; Haruechaiyasak, C.; Tran, D. Ontology-based terrorism event extraction. In Proceedings of the 2009 1st International Conference on Information Science and Engineering (ICISE), Nanjing, China, 26–28 December 2009; pp. 912–915.
35. Alghamdi, H.M.; Selamat, A. Topic detections in Arabic dark websites using improved vector space model. In Proceedings of the 2012 4th Conference onData Mining and Optimization (DMO), Langkawi, Malaysia, 2–4 September 2012; pp. 6–12.
36. Arulanandam, R.; Savarimuthu, B.T.R.; Purvis, M.A. Extracting crime information from online newspaper articles. In Proceedings of the Second Australasian Web Conference, Auckland, New Zealand, 20–23 January 2004; Volume 155, pp. 31–38.
37. Ali, M.; Jones, M.W.; Xie, X.; Williams, M.J.T.V.C. TimeCluster: Dimension reduction applied to temporal data for visual analytics. *Vis. Comput.* **2019**, *35*, 1013–1026. [CrossRef]
38. Xu, S.; Zhang, J.; Han, D.; Wang, J. Singular value decomposition based data distortion strategy for privacy protection. *Knowl. Inf. Syst.* **2006**, *10*, 383–397. [CrossRef]
39. Salem, R. A manifold learning framework for reducing high-dimensional big text data. In Proceedings of the 2017 12th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 19–20 December 2017; pp. 347–352.

40. Radhoush, S.; Shabaninia, F.; Lin, J. Distribution system state estimation with measurement data using different compression methods. In Proceedings of the 2018 IEEE Texas Power and Energy Conference (TPEC), College Station, TX, USA, 8–9 February 2018; pp. 1–6.
41. Feng, J.; Yang, L.T.; Dai, G.; Wang, W.; Zou, D. A Secure Higher-Order Lanczos-Based Orthogonal Tensor SVD for Big Data Reduction. *IEEE Trans. Big Data* **2018**, *5*, 355–367. [CrossRef]
42. Cho, H.; Yoon, S.M. Improving sentiment classification through distinct word selection. In Proceedings of the 2017 10th International Conference on Human System Interactions (HSI), Ulsan, Korea, 17–19 July 2017; pp. 202–205.
43. Yang, H.-C.; Lee, C.-H. Mining tag semantics for social tag recommendation. In Proceedings of the 2011 IEEE International Conference on Granular Computing (GrC), Kaohsiung, Taiwan, 8–10 November 2011; pp. 760–766.
44. Liu, C.-H.; Chen, H.-C.; Jain, J.-L.; Chen, J.-Y. Semi-automatic Annotation System for OWL-based Semantic Search. In Proceedings of the 2009 International Conference on Complex, Intelligent and Software Intensive Systems, Fukuoka, Japan, 16–19 March 2009; pp. 475–480.
45. Ballan, L.; Bertini, M.; Uricchio, T.; Del Bimbo, A. Social media annotation. In Proceedings of the 2013 11th International Workshop on Content-Based Multimedia Indexing (CBMI), Veszprem, Hungary, 17–19 June 2013; pp. 229–235.
46. El-Beltagy, S.R.; Hazman, M.; Rafea, A. Ontology based annotation of text segments. In Proceedings of the 2007 ACM Symposium on Applied Computing, Seoul, Korea, 11–15 March 2007; pp. 1362–1367.
47. Ghanbari-Adivi, F.; Mosleh, M. Text emotion detection in social networks using a novel ensemble classifier based on Parzen Tree Estimator (TPE). *Neural Comput. Appl.* **2019**, *31*, 8971–8983. [CrossRef]
48. Xiang, L.; Bao, Y.; Tang, L.; Ortiz, D.; Salas-Fernandez, M.G. Automated morphological traits extraction for sorghum plants via 3D point cloud data analysis. *Comput. Electron. Agric.* **2019**, *162*, 951–961. [CrossRef]
49. Keith, J.; Westbury, C.; Goldman, J.J.B.R.M. Performance impact of stop lists and morphological decomposition on word–word corpus-based semantic space models. *Behavior. Res. Methods* **2015**, *47*, 666–684. [CrossRef] [PubMed]
50. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]
51. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP natural language processing toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, Maryland, 23–24 June 2014; pp. 55–60.
52. Magdy, W.; Jones, G.J. An efficient method for using machine translation technologies in cross-language patent search. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, Glasgow, Scotland, UK, 24–28 October 2011; pp. 1925–1928.
53. Sajjad, R.; Bajwa, I.S.; Kazmi, R. Handling Semantic Complexity of Big Data using Machine Learning and RDF Ontology Model. *Symmetry* **2019**, *11*, 309. [CrossRef]

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).