Near Real-Time Atrocity Event Coding

[1]Sai Vivek Kanaparthy,[2]Shiva Ganga Reddy Chennu,
[3]Satya Aravind Chowdary Obellaneni,[4]Sushma Eati,[5]Manisha Galla
Email: {[1]sxk163030, [2]sxc164430, [3]sxo160530, [4]sxe160530, [5]mxg161430}@utdallas.edu

***Abstract*—Atrocities, terrorism and political unrest proliferated human sufferings to a great extent in recent times. These activities adversely affect the life of common and innocent people leaving a negative impact on their livelihood, families etc. In this modern era, where technology is growing at a rapid speed, dreaming of a tool that predicts the occurrence of an Atrocity event would be of no surprise. This can be achieved with the help of Machine Learning Algorithms and Natural Language Processing techniques. To predict the occurrence of an Atrocity event we require structured data that contains meta data with particular fields and values. This process of prediction with utmost precision can be done with the help of common sense and encoding events from news, but this process can be slow and uncertain. So in this paper we develop a near real time machine that detects the occurrence of an Atrocity event. We design a Spark Based Distribution Framework in conjuction with web scraper to generate and gather news. Raw text processing is performed more accurately using CoreNLP and performance issues can be resolved with the help of SPARK [11].**

***Keywords*—Semapor, Spark, Kafka, TF-IDF, Logistic Regression, Naive Bayes.**

I. Introduction

In recent times, many atrocities took place. There is a need to develop a software to detect such events. This can be achieved with the use of machine learning and natural language processing techniques. In order to detect such atrocities, structured data is required with multiple fields and values like perpetrator, victim etc. In general, humans encode events based on news reports but this process is tedious and cannot be done for millions of records in near-real-time processing . Machines can be used in place of humans to increase the pace of event encoding. In this paper we implement this using Web Scraper, Apache Spark, MongoDB, SEMAFOR, Kafka, CoreNLP, MLLib, Kibana. The web scraper collects data from news reports, filters reports that are related to atrocity events. All the filtered data is sent to kafka (achieves near-real-time processing). The consumer pulls the data from kafka and sends it to Semafor. Semafor parses the data and attaches a label for every word. We extract labels for the word victim which are used to extract features in order to classify the type of victim. A classifier model is built from existing data. This classifier classifies the data that we got from the semafor.

II. Related Work

A. Scraper

The news feeds are collected from 386 websites in near real-time. we have written a python code for gathering these news feeds. Libraries like pages_scrape,pattern.web,goose are used to extract these news feeds form the listed 386 websites in near real-time. It collects the news feed periodically every 5 minutes and it ignores if there is any error in the news article. These news feeds are sent to kafka and are also stored in mongodb.

B. Semafor

Semafor is a frame semantic parser. It gets the semantic analysis of an english sentence. Semafor is used to extract semantic labels.The news feeds that are collected from the Scraper are sent to atoricity filters, then to Seamfor. CMU Semafor is used to get the labels for these news filtered articles(labels like VICTIM, WEAPONS). These labels are used to extract the features to classify.

The stories from the Parus Atrocity Coding dataset are loaded as JSON objects. These objects are further converted into sentence format which serves as the input for the semafor. SEMAFOR is a frame-semantic parser, that parses the english sentences automatically wrt to the semantic analysis in FrameNET. Frame Semantics, a linguistic theory that has been instantiated for English in the FrameNET lexicon.This tool intends to find the words in text that evoke semantic frames, and to find and label each frame's argument.

- Pre-Processing: The sentences are lemmatized, part-of-speecg tagged and syntactically parsed.
- Target identification Frame-evoking words and phrases("targets") are heuristically identified in each sentence.
- Frame Identification: A log-linear model, trained on FrameNet 1.5 data with full-text frame annotations, produces for each target a probability distribution over frames in the FrameNet lexicon.The target is then labeled with the highest-scoring frame.
- Output An XML or JSON file is produced containing the text of the input sentences, augmented with the frame-semantic information

C. Tf-Idf (Term Frequency-Inverse document frequency)

Tf-idf is one of the key feature extraction module in spark mllib which helps us to extract features from text and image datasets in a format supported by machine learning. It converts the textual information in to a vector space model. Tf-idf weight gives us the importance of particular word in a document or corpus. The Tf-idf value increases as a number of times the word show up in the document. Tf-idf can be used stated as a term-frequency times inverse document frequency. **Tf-idf(t,d) = tf(t,d) - idf(t)**
idf component can be computed as:

$$idf(t) = \log \frac{1+n_d}{1+df(d,t)}$$

In our implementation term frequency is uses the hashing trick. An index is mapped to a raw feature by applying a hash function and the term frequencies are calculated for every mapped index. In this method there may be a chance for hash collisions where the same term resulted after hashing different raw features. This problem can be overcome by increasing the bucket size of the hashtable from default value 1,048,576 to even more.

D. Kafka

Apache Kafka is a distributed streaming platform. The 3 main features include

1) Kafka is similar to messaging queue, it allows us to publish and subscribe to stream of records.
2) Records are stored using fault tolerance mechanism.
3) Records are processed in the way they occur.

Kafka is used in two broader classification of applications:

- Building real time streaming applications that work on or analyze the stream of the incoming data.
- Building real time pipelines that reliably get data between systems.

The atrocity application that we developed requires live streaming of the news feeds from various website. So we have chosen Kafka to store the incoming stream of data in a fault tolerant way.

Steps Involved:

1) The scraper program crawls over all the websites and filters the news feed with words related to atrocity, kills, bombs etc.
2) Once the filtered data is obtained, the program pushes the data into Kafka with a specific topic name.
3) In the client program we obtain the list of messages under the mentioned topic name and start our analysis on the feed.

E. Spark

Technologies Considered:Map Reduce, HBase, Spark.-(Used)
We have considered spark for analyzing the data over other technologies because of the following feature set:

PERFORMANCE:

1) Spark processes data using RDD's that brisk up the processing speed by100 times when compared to Hadoop MapReduce because of in-memory processing of data.
2) Spark needs a lot of memory when the data is too large to handle, since we are working with real time news feed we used spark streaming, data is first pushed into Kafka and from there the spark code gets the live stream in batches and starts analyzing them.

EASE OF USE:

3) Built in API's are available for Spark in various programming languages like Scala, Java and Python which makes programming much easier compared to Hadoop Map Reduce.

F. MongoDB:

Since we are collecting data from various sites, the structure of the feed can vary from website to website. Since MongoDB is a NoSQL database and accepts data with various fields in a document, it suits well for the current appliacation. The use of MongoDB in this application is to store the news feeds collected from the various websites mentioned in the whitelistsurl.csv. At present, we are not using the data in the MongoDB but this data can be used as a reference to check accuracy or for any comparisons or analysis in the future.
As explained above the importance of data warehouse maintenance with application specific data and considering the complexity of accessing the folders to scrap the news feeds that help in analyzing the writing style of an author I decided to use **MongoDB**.
MongoDB (from humongous) is a cross-platform document-oriented database. This is **NoSQL** database which moves away from the traditional relational databases. This supports **JSON** type documents which is termed as **BSON** by **MONGODB**. This makes the task of combining data of different applications faster and easier. Released by the GNU Affero General Public License and the Apache License, free and it is open source.
Terminology:

- Document: single entry in JSON format
- Collection: it is the collections of Documents
- Database: collection of collections

Some important features of MongoDB are:

- Document-oriented nature of MongoDB facilitates to store whole subject oriented data in the minimal documents which is much different from the way of splitting the subject oriented data into various relational databases.
- Replication is provided by the MongoDB highly.
- Load balancing, MongoDB is called so because of its nature of scaling horizontally using sharing using the shard key which determines distribution of the data over the collections.
- MongoDB runs on multiple servers at a time, balances the load and duplication of data to keep system working during any faults.
- In a multi-machine MongoDB system, the file distribution system and duplication of files is transparent thus effectively making the systems load balanced and fault tolerant system.
- JavaScript execution on server side
- JavaScript is used in querying, aggregation methods (such as Map-Reduce), and sent to database directly for execution.

G. MLLib

Machine learning is a part of artificial intelligence that enables systems to learn based on data alone, continuously improving performance as more data is processed. Machine learning is the basis for many technologies that are part of our everyday lives. Some examples of applied machine learning algorithms include recommendation engines, natural language processing, anomaly detection etc. Until recently, data scientists had to implement and customize machine learning algorithms manually to the computing framework that they were using, resulting in a significant amount of work. Now, with Spark and MLlib, data scientists can write jobs that reference a number of predefined algorithms to build these kinds of applications. MLlib is a standard component of Spark providing machine learning primitives on top of Spark. MLlib is built on Apache Spark, which is a fast and general engine for large scale processing. Supposedly, running times or up to 100x faster than Hadoop MapReduce, or 10x faster on disk. Supports writing applications in Java, Scala, or Python.

The machine learning algorithms supported by MLlib are: Classification:

- Support Vector Machine
- Logistic Regression

Logistic Regression:

- Logistic Regression
- Lasso
- Ridge Regression

Clustering:

- KMeans

Collaborative Filtering:

- Alternating Least Squares

Gradient Descent Primitive:

- Gradient Descent

H. CoreNLP

Core NLP provides a list of natural language processing analysis tools. It gives the basic forms of words their parts of speech.

1) It is reliable and fast with arbitary texts.
2) Interfaces are available for all high level programming languages.
3) Highest quality for text analytics.

III. Implementation

To classify the real time news feed we need labeled training data. We got this data(human encoded training data) form the internet. So, Input data is the human encoded news feeds.

DataSet Description: This data set [12] contains the whole data in structured format each news article is in its corresponding date folder and the date folder has three subfolders which are codingfile, downloads and json. codingfiles has the stories which contains news articles, downloads folder has the headlines and the json has the structured file which contains the class Fields and labels in json format

A. Semafor

Each news feed is sent into semafor to get the semantic analysis for each line and label them as Victim, Weapons, etc. Semafor that we have used is CMU Semafor. we have written a python code to use this semafor. Input to the semafor is a single line from each article and get labels for the article and store these labels in a document. we now use the required labels like Victim. there are different classes for these labels, these labels are given in the input data. From the input data we found that, there are six different labels(which are classified based on the event that has occurred) labels for these victim are,

- victimcomb
- victimethn
- victimpoli
- victimrand
- victimreli
- victimsoci

We have used this information as classes to the input news feeds and used labelled them. this labelled data is used to train the classifier

B. TF-IDF(Term Frequency- Inverse Document Frequency)

TF-IDF is implemented using the MLLib library in Spark. HashingTF is used for getting the term-frequency and IDF is used to get the Inverse Document Frequency. This is used as a feature for each document. tf-idf vector that is generated using MLLib can be directly used as an input to any of the classifiers.

C. Model Building

As the data that we are using is multi-class data we have to implement linear classifiers using one vs rest technique, in this technique we use one of the class as the first class, rest of the classes as second class and repeat this for all the classes. we have one vs rest technique is used with the logistic regression and Multinomial naive bayes classifier.

Logistic Regression: MLLib has an inbuilt library that can be used to train the logistic Regression classifier. we have implemented this in the spark to make it much faster. input data is split into train and validation. we have used randomSplit to split it into 80%(train) and 20%(test).

Multinomial Naive Bayes: MLLib has an inbuilt library that can be used to train Multinomial naive bayes classifier. input data is split into train and validation. train data is used to train the classifier and validation data is used to validate the model and to check the goodness of the model.

SVM: SVM is also inbuilt in MLLib. we use this built-in library directly to built the model, as per the performance the logistic regression has the highest accuracy. Even in SVM we split the input data into train and valid data, train is used to built the model and valid is used to validate the model

The evaluation metric that we have used is accuracy from the MulticlassClassificationEvaluator(Pyspark). the accuracy for the naive bayes model is more than the logistic regression model

D. Testing Real Time News Feeds:

Real time news feeds from approximately 400 websites are taken(if there are any errors in the news feeds they are ignored) and these news feeds are sent to Kafka steaming. Now these feeds are processed through semafor where we get the labels(semantic) for the news feeds now these feeds are predicted using the model built from the above classifier. these predicted class is again sent into model to get the class label.

IV. Feature Extraction

Pre-Processing is a crucial step in extracting knowledge from data available because the data contains information irrelevant to the context. It ignores any errors present in the data. An initial set of measured data is used to build derived values (features) intended to be informative and non-redundant. It is the first phase of feature extraction. It facilitates the subsequent learning and generalization steps. It leads to better human interpretations.
Feature extraction involves reducing the amount of parameters required to represent a large set of data. When analysis of complex data is performed, one of the significant problems that occur is the number of variables involved. A significantly large amount of memory and high computational power are the problems which arise when analyzing a large number of variables. Similarly, a classification algorithm which over-fits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

A. Traditional Features

Some of the important semantic labels are listed below:

- Victim
- Weapon
- Actor
- Event

B. Construction of bi-grams

Step 1: In order to evaluate the bi-grams, we need the parsed tree of a sentence. Therefore, the first step for evaluating the bi-grams is the generation of a parse tree. To generate the parsed tree of a sentence, Stanford Parser is being used as mentioned in [9]. A NLP parser is a module to branch out the sentences grammatical structure, e.g. whichever word groups go along (as phrases) and which are the object or subject of a verb. Probabilistic parsers use hand-parsed sentences knowledge to make the most suitable analysis of sentences that are new. Though they still make mistakes, it works satisfactorily. Fig. 1 represents the parsed sentence structure of the sample sentence.

Sample Sentence: A large company needs a substantial business model.

```
(ROOT
  (S
    (NP (DT A) (JJ large) (NN company))
    (VP (VBZ needs)
      (NP (DT a) (JJ substantial) (NN business) (NN model)))
    (. .)))
```

Fig. 1. A Parsed Sentence

Step 2: The bi-grams of the parts of speech of the words are considered as features. So the parts of speech of each word are to be known as per the syntactic structure. So the parts of speech of each sentence are evaluated. Stanford Log-linear Part-Of-Speech Tagger [7] is used to evaluate the parts of speech of a sentence such as a noun, verb, adjective, etc., although computational applications use more fine-grained POS tags like 'noun-plural'.

Tagged Sentence: The tagged sentence of the above stated smaple sentence is as follows:

A/DT large/JJ company/NN needs/VBZ a/DT substantial/JJ business/NN model/NN

V. Classification

Classification is the processing of categorizing given documents into predefined classes.
In a supervised learning method,
Given:
V a description of an instance, x $\in$ X, where X is the instance space.A fixed and predefined set of classes are:

$$C = c_1, c_2, c_3 .... c_n$$

To determine the category of x : c(x) belongs C, where c(x) in classification function whose domain is X and range is C

A. One vs Rest:

One vs Rest is a machine learning technique present in ML-Lib that performs multi-class classification when a classifier which performs binary classification is given. Given n classes, it considers class i as one class label and all the remaining classes as another class label. This process is repeated for n classes. It is also known as One vs All. As we have six classes of victims, we used One vs Rest with Logistic Regression and Naive Bayes Classiers.

B. Logistic Regression

Logistic Regression is a predictive analysis. It is used when the dependent (response)variable is binary. It explains the relation between dependent (binary) variable and independent (explanatory) variables. Here, the values of the dependent variables are obtained from the combination of values chosen by the independent variables.
Logistic regression can be used in the following cases:
when the probability of the dependent variable needs to be predicted from a function of set of independent variables.
when the response variable needs to be classified into two classes for example, predicting a person as literate or illiterate when we need to summarize the variations between two people in distinct categories as a function of independent variables
It is important to consider model fit while using logistic regression. Statistical variability increases as the independent variables gets added to the model and over fitting occurs by adding many variables to the model.

C. Naive Bayes

Nave Bayes classification technique falls under supervised learning algorithms. It is based on Bayes theorem. It can perform better than other classifiers. It considers its features to be independent of each other but the features may be dependent in some cases. This is reason for it being called nave. It classifies given a set of features using posterior probability (combining prior probability and likelihood).

$$p(\frac{c}{x}) = \frac{p(\frac{x}{c})p(c)}{p(x)}$$

D. Support Vector Machines

Vapnik [2] developed the fundamental concept of Support Vector Machines. The SVMs concept is based on the idea of structural risk minimization which minimizes the generalization error (i.e., true error on unknown examples) which is limited to the sum of the training set error and a term which relys on the Vapnik-Chervonenkis (VC) examples. The use of structure risk minimization performance measure is in contrast with the empirical risk minimization approach used by multiple classifiers. Conventional classifiers attempt to minimize generalization error. Therefore, SVMs have theoretically a greater ability to generalize.

Ideally, each SVM is provided with all of the emails, but the large size of the dataset makes this infeasible. To improve the ability of an SVM to distinguish between its associated class and all others, we ensure that this set of emails includes the input closest classes as determined by the Euclidean distance between the class centroids. A linear kernel is used for classification. A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Though the training time of SVMs can be slow, they are highly accurate, owing to their ability to model complex nonlinear decision boundaries. C is a regularization parameter in SVM, which controls the trade-off between achieving a little error on the training data and minimizing the norm of the weights. It is the penalty imposed on training data that fall on the wrong side of the hyperplane (class decision boundary). C is a parameter which can be viewed as a way to control overfitting: it trades off the relative importance of maximizing the margin and fitting the training data.

VI. Proposed Algorithm

Algorithm 1 Atrocity Event Coding

1: **procedure** Real-Time news feeds classification
2: **input:**human labelled data set
3: **begin:**
4: *data* $\leftarrow semafor($*input data*$)$
5: *trainingData* $\leftarrow add.victimlabels($*data*$)$
6: *model* $\leftarrow LogisticRegression($*trainingData*$)$
7: *realtimeNewsFeed* $\leftarrow Scrapper()$
8: *testData* $\leftarrow Atrocityfilter(realtimeNewsFeed)$
9: *test* $\leftarrow semafor($*testData*$)$
10: *prediction* $\leftarrow Model.predict($*test*$)$
11: *label* $\leftarrow AddLabel($*test*$)$
12: **end**
13: **end procedure**

VII. Experimental Results and Discussion

A. Victim Classification $word_cloud$

WordCloud is a python library used to display the most frequent words that are present in the given document. Victim is classified depending upon the following words in the news feed.

Fig. 2. Words which classified the victim as combat

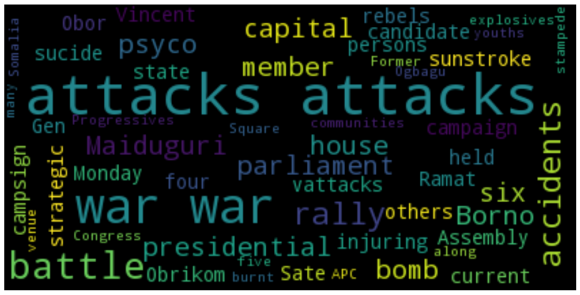

Fig. 3. Words which classified the victim as random

B. Relevant Discussion

The data set used for training data is Data set[14]. This data set contains the whole data in structured format each news article is in its corresponding date folder and the date folder has three subfolders which are codingfile, downloads and json. codingfiles has the stories which contains news articles, downloads folder has the headlines and the json has the structured file which contains the class Fields and labels in json format.

The dataset contains 6 victim classes as mentioned above the distribution of the classes over the dataset is as follows

TABLE I. VICTIM CLASS DISTRIBUTIN OVER THE INPUT DATASET

victim class	victim type	number of articles
victimcomb	Combat	56
victimethn	Ethnic	334
victimpoli	Political	490
victimrand	Random	1604
victimreli	Religious	502
victimsoci	Social	332

From the above table we can observe that there are more random events where the atrocity events takes palce.
From this table we an say that our training biased not equally distributed.

TABLE II. NUMBER OF CLASSES PER DOCUMENT

victim class per document	number of articles
0	262
1	5748
2	495
3	30
4	1

From the above table we can conclude that there are 90% of the classes where there are only one class labels.

TABLE III. METRICS EVALUATION METRICS

Classifier	recall	precision	F1 Score	Accuracy
Multinomial Naive Bayes	0.527	0.517	0.511	0.528
Logistic Regression	0.51	0.512	0.512	0.518

Using Algorithm 1, we can classify the real time news feed and predict the atrocity events that are occurring in near real-time.

It is found that, Multinomial Naive Bayes classifier used with one vs rest classifier techniques outperforms the SVM classifier even Logistic Regression has the similar performance with the Naive Bayes. SVM performs well when the number of classes is few. As the number of classes increase, the performance of the algorithm descents. The performance of SVM drastically falls when the number of classes is increased. So, Support Vector Machines Classification is preferred when the number of classes is very low.

VIII. CONCLUSION

In the Project we have successfully implemented the paper [1]**Near Real-Time Atrocity Event Coding**. We have introduced a spark-based model and a machine learning classifier to predict the atrocity events in near real time and also predicted the label of the predicted class of the victim.

REFERENCES

[1] Solaimani, Mohiuddin, et al. "Near real-time atrocity event coding." Intelligence and Security Informatics (ISI), 2016 IEEE Conference on. IEEE, 2016.

[2] LIBSVM. "Library for Support Vector Machines, Chih-Chung Chang and Chih-Jen Lin." Computer Science Department, National Taiwan University. Available: http://www. csie. ntu. edu. tw/cjlin/libsvm (2004).

[3] Geurts, et al. "Extremely randomized trees." Machine learning 63.1 (2006): 3-42.

[4] Hastie, et al. Unsupervised learning. Springer New York, 2009.

[5] Louppe,et al. "Ensembles on random patches." Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2012. 346-361.

[6] Bauer,et al. "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants." Machine learning 36.1 (1998): 2.

[7] Toutanova,et al. "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger." Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13. Association for Computational Linguistics, 2000.

[8] Toutanova,et al. "Feature-rich part-of-speech tagging with a cyclic dependency network." Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 2003.

[9] De Marneffe,et al. "Generating typed dependency parses from phrase structure parses." Proceedings of LREC. Vol. 6. No. 2006. 2006.

[10] www.cs.cmu.edu/ ark/SEMAFOR/

[11] https://spark.apache.org/docs/2.1.0/api/python/pyspark.mllib.html

[12] http://eventdata.parusanalytics.com/data.dir/atrocities.htm

[13] https://pypi.python.org/pypi