Real-time Event and Context Analytics Using Audio and Video Streaming

Ganesh Taduri, Nikhita Sharma, Aparna Pamidi, Harsha Komalla
University of Missouri - Kansas City

Abstract – *One of the major challenges today is to analyze huge amounts of real time data and get useful insights from such data instantly. One of its many applications can be in event analytics by using live audio and video data as input. We propose a system which takes live data as input and give useful analytics for business and common users.We use some of the emerging technologies in Big data processing like Storm, Spark, Kafka and other machine learning concepts and algorithms like decision tree classification model, training and testing of such models to discern classes. Two major results from the proposed application will be event context such as musical or speech oriented and number of people attending the event. Event Context is classified based in audio features from the video and count of number of attendees is found based on visual input from the video.*

1. INTRODUCTION

Videos have become a major part of the day to day functioning from providing entertainment, to real-time recording applications in the field of security, education, business and entertainment. Some of the examples are surveillance videos, teaching or informational videos, advertisements or movies, etc. Accordingly there is enormous volume of data being generated everyday and various applications developed on video processing are on huge demand in the market. But processing the huge amount of data is a very big challenge. The best solution for this challenge is to implement efficient Big Data techniques and process the data effectively with minimum amount of time. By a proper implementation of available technologies in real time video processing one can develop useful applications.

Our focus in this paper will be to analyse videos and live data of events to analyze the scenario and popularity of various events and further, use these results for comparison between events based on type of events like a musical or educational event and also conclude popularity by finding out the number of people present at the event. With the help such available data we developed and trained a model using effective machine learning techniques. We tested this model with real-time dynamic data and observed the results.

In an improving educational world it's very beneficial to keep track of analysis of an event such as how many people are present in an event at a given time, auto detection of the event context such musical event, speech event, etc. These analytics can be very helpful in developing interesting use cases such as

- popularity/success of the event by inferring number of people attending the event.
- To perform real-time live analytics on large scale events which could happen at the same time

- To assist the disabled to infer the scenario around them through textual annotations or audio results
- To track activities during an event such as number of attendees leaving or entering the event, etc.

We use machine learning concepts to train and test in order to predict the context of the event. HaarTraining algorithm for face recognition is used which is available through OpenCV library. Other useful applications of similar detection process for face or an object are in traffic surveillance, security, and other visual analytics requirements including medical research field. Such applications are widely being used and they seem to be very beneficial.

The motivation behind this system were: to build an effective system for live event analysis, to design an efficient system for post event analysis, to build an application which helps the deaf and the blind to understand the scenario around them, building applications similar to some interesting Microsoft Cognitive services and implementation of some of the emerging technologies in real-time big data analytics.

2. RELATED WORK

Extensive research is being held and published in the area of Big Data processing. The real time data processing is currently the most trending and there are thousands of research papers published in various journals to read and understand. With the increasing number of videos, their necessity to analyse and extract useful information there were many frameworks being introduced for this purpose. We studied around 13 research papers related to Big data Analytics on Streaming data and some important frameworks used to achieve big data processing. Some of them frameworks researched were: Storm, Spark, Kafka, Pipeline61 Framework, Heron, SummingBird, S4 framework, etc.

This section highlights the trending technologies for Big data processing in real-time like Apache Spark, Storm, Kafka and Cluster computing using cloud. For video processing, concepts and models like online video processing using deep intelligence framework, performance in stream of video delivery and video representation models give the required information to readers about various application scenarios.

Each technology and video processing concept discussed in this survey paper has scope of improvement and future work currently undertaken. This field of study is fast improving and better methods and tools are being designed. We look forward to better method and technological upgrades to achieve the same results with higher performance and efficiency.

With a high rise in types of big data, it is a challenge to choose the best execution framework for a specific application needs. When it comes to picking a best suited framework, pros and cons of each framework are to be discussed. Frameworks tend to be strong in terms of accuracy or performance, for only the applications they are initially designed for. Some of the factors in deciding a framework or execution environment for an application are: Ability to adapt with changing data volumes, Ability to be flexible for implementing use cases in future, Upgradeable with any other technology it depends on, etc

Some of the applications already implemented and listed in the references are Human Emergency Mobility Simulator, Traffic Flow Prediction.

Among these Storm, Spark, Heron, SummingBird are a few that support real time streaming processing and provides results in microseconds and seconds. Based on the requirements of the application the select of a frameworks depends. In our application we used the Storm, Spark and Kafka. Where the Spark is used for model prediction, Kafka as messaging system to the consumer which our case is the storm, and Storm is used for the predicting the new test features data.

3. PROPOSED SOLUTION:

We propose a system which take a video having both audio and visual data, as input and present results about the scenario of the event and number of attendees to the end user. Before we discuss our application in detail, let us throw some light on the technologies we used to implement our application

.

3.1 Technologies Used

Selecting an efficient framework for the system and a fast messaging system to transfer streaming data in milliseconds of time was important for achieving high performance of the application. We selected the below technologies carefully, to best optimize performance of the overall framework.

3.1.1 Spark:

Spark is a framework built based on Hadoop Mapreduce to support a class of applications where reuse of working data set across multiple parallel operations is core functionality.. It provides better scalability and fault tolerance of MapReduce. This is achieved by introducing a new abstraction called Resilient Distributed Datasets abbreviated as RDD's. It is implemented in scala, built on top of mesos supports and Map/Reduce paradigm introduced by Hadoop.

The operations on spark are 10x faster than regular Hadoop MapReduce operations because they use in memory operations to store intermediate data whereas Hadoop stores in secondary storage. Various iterative jobs such as machine learning, expectation maximization, parameter optimization and other interactive analytics can be performed effectively using Spark. Spark supports parallel programming with the help of RDD's and parallel operations. Implementation of RDD is the core part of Spark. In Spark Fault tolerance is achieved by a concept called lineage where each data object contains information about its parent object and information about how it is transformed.

3.1.2 Kafka:

Kafka is a distributed messaging system for log processing. kafka combines the benefits of traditional aggregators and messaging systems. The kafka architecture mainly deals with topic, producer, consumer, broker and clusters. Topic: stream of messages of same type, Producer: publishes messages to a topic, brokers: the servers that store published message, Consumer: subscribes and consumes the data related to a topic by pulling data from broker. A consumer can subscribe to more than one topic by creating a message streams for the topic and Cluster: a group of brokers working together.

The kafka architecture ensures at least once and exactly once data delivery. Kafka system has a simple storage structure where each partition of a topic represents a logical log. Each time producer publish a message to topic they are appended to the segment files. After a significant number of

messages are stored in the segment file it is flushed to the disk for consumption by the consumer. Consumer continuously consumes the messages based on the offset value of message at which it last consumed.The use-cases that kafka supports vary from batch processing to real-time processing.

3.1.3 Storm:

Storm is a real time distributed processing system. The architecture of storm makes itself fault tolerant and scalable. Storm works using topologies, which can be visualized as a DAG, showing the flow of data. Typically a topology consumes the streaming data, performs partitioning among the data streams as required by the computation. A topology consists of spouts: which is a data source and bolts: which performs the actual processing on the data. Storm has the capability to integrate with any of the the queueing and database system. Spouts are responsible for managing the integration with different queueing systems like kafka. Unlike MapReduce DAG the storm topologies run until explicitly killed by a user.

In Storm, the interaction mechanism between clusters is done with the help of components like Nimbus: the main interaction server between client and the Storm architecture; Worker: where the actual data processing takes place; Supervisor: supervises the activities of workers monitoring its failure and reporting to Zookeeper and Nimbus; Zookeeper: to help coordinate between Nimbus and supervisors. Concepts like heartbeat protocol, various types of threads running in a Supervisor and a worker, are explained in detail. The authors shows great attention to detail by bringing forward the per tuple processing topology and working of “at least once” and “at most once” processing of each tuple. This is achieved by acker bolt implementation in the topology.

3.1.4 MongoDB:

MongoDB is a database which provides easy scalability, high availability and performance. Database is a group of collections and each collection is a group of MongoDB documents. Each document has key-value pairs like JSON documents. MongoDB can be used as Data Hub to store large amounts of data. The stored data can be accessed through API calls. MongoDB has official drivers to support wide range of programming environments. MongoDB can act as a file system that supports data replication among multiple machines and also load balancing.

3.2 Architecture:

The system architecture includes Client application, server and MongoLabs Database. Fig 1 shows a simple representation of the architecture of our system. The client application has a UI interface for the user to select input type which is either static video files stored locally or live video capture tool. The server has Spark, Storm and Kafka environments installed. For storing our feature vectors, model and result, we use MongoDB which provides a RESTful way of pushing and fetching data from the database.

During the testing and training phases, we performed audio feature extraction on input video. These features were passed to Kafka through Rest API deployed on Tomcat Server. Kafka messaging system was used to send these features to Spark and Storm environment for model building and recognition, respectively.

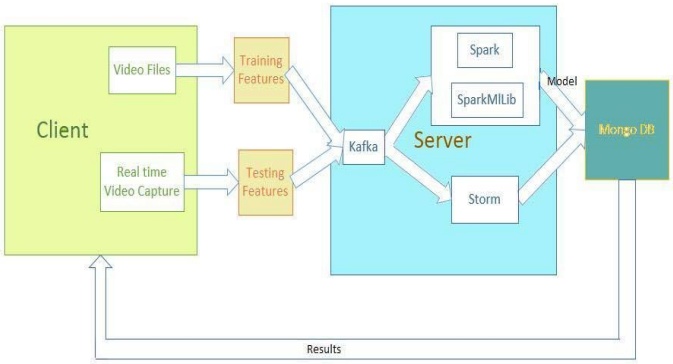

Fig 1: Architecture Diagram

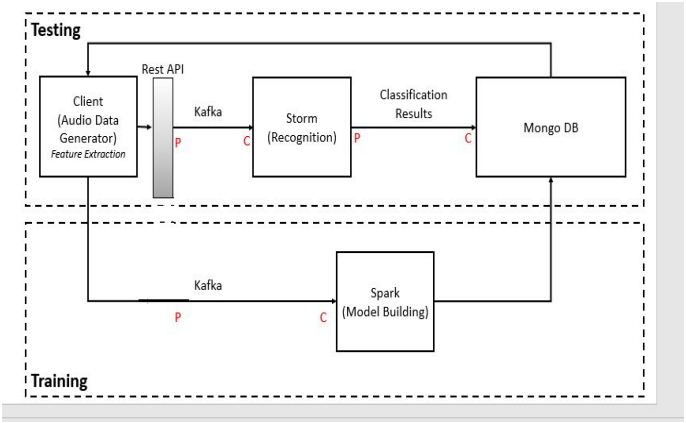

Fig 2: Workflow Diagram

The final decision tree model and recognition results were stored in MongoDB so that they can be updated and retrieved through API calls at any point of time, by the client application

4. IMPLEMENTATION

In this section, we will discuss the implementation of our application. This section includes overall workflow, model generation, training and testing of the model, techniques used for face recognition, performance comparison of Spark and Storm and challenges faced during the implementation process.

4.1 WorkFlow:

Fig 2 shows the workflow of the system. We divided the workflow into two parts. One for training and one for testing. For training the model we collected videos files, extracted audio from them and then extracted features from audio files, sent these features from client to kafka producer and saved the features into a vector file, generated the model, and pushed the model into mongoDB.

For testing the model we sent audio features from client to Kafka spout and classified the data using three bolts in the storm topology and saved the results into Mongodb.

4.1.1 Feature Extraction:

First, Collected around 25 videos from various sources like Youtube, Ted.com, and some other random videos then we extracted audio from the video files and then we used Jaudio for the feature extraction. we classified the features into music features, speech features and other features. We selected a set of six features which suited best for classifying our data. The features we selected are meanZCR, meanMFC, meanSpectralRollOff, meanPeakValue, meanRMS, meanCompactness. The features were then passed using Kafka Producer-Consumer Pub-Sub messaging System to the server and stored them in as a feature vector file that can be used in generating the model.

4.1.2 Model Building and Training:

We generated the model using the features vector file saved at server which was received from kafka consumer. We implemented decision tree algorithm in Spark MLlib to train the model the model. As our model have only three classes for the decision tree, we observed that the model generated was not so complex and easily it's interpretable. Moreover we observed that Mean MFC is a major decision making feature for music recognition and mean RMS for speech recognition. This model is then pushed to mongoDB which will be accessed later during the testing phase.

As an alternative we used random forest algorithm to generate our model with an intention to improve the accuracy. On contrary we observed that random forest does not work better for audio and image features.Accuracy of the model generated by random forest is less compared to the model generated by decision tree.

4.1.3 Recognition and Testing:

For testing our model at real time, we created a Storm topology with one spout and three bolts, the spout acts as a data source i.e it receives the new audio features to be tested from client through kafka. The bolts perform the actual task in our topology, all the logic related to parsing the decision tree , classify the feature into a particular class is written in the bolt, we created three bolts each bolts the depicts one class. The bolts receives the input features data from the spout parse them them through the decision tree, once the processing is done respective bolt outputs a class label to which the given input data belongs to, in our scenario the class labels are either Music or Speech or Other. The topology we generated is shown in the Fig 3.

There results were then extracted at the client end and annotations were displayed on the video with respect to the results along with the count of number of people in it.

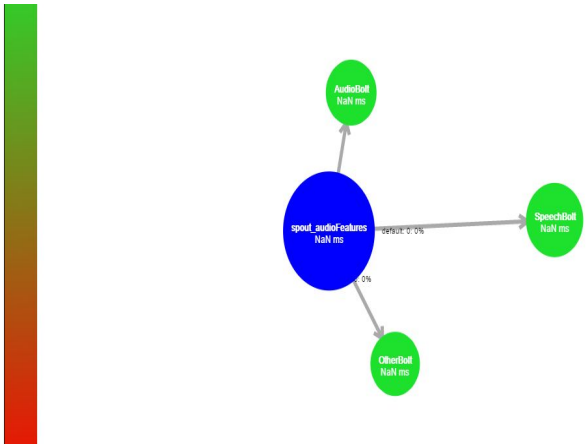

Fig 3: Storm Topology

4.2 Face detection using OpenCV and Haar Algorithm

We used audio features for classification of the video context such as music , speech or other classes and Video frames for face detection to count the number of faces. We implemented the OpenCv library and used Haar detector methods to identify a human face in a video frame. Once the face got detected a squared box boundary over the face is drawn in the video. Then we counted number of number of boxes generated in a video which reflects the number of faces ultimately counting number of people in the video. We annotated the video with the population dynamically.

GitHub link:

https://github.com/nikhitasharma/RTBigDataAnalytics_Project

Project Video:

https://www.youtube.com/watch?v=1JMl7ekg2UY

5. RESULTS AND EVALUATION

The system is trained on 25 videos under 3 different categories Music Speech and other. We tested our model on Local Spark as well on server using storm topology for real time data. The model on local machine on spark has better accuracy rate compared to model on the server on topology. We observed that model on local spark has about 90% accuracy. The reason for this high accuracy is that the deciding factor between speech audio and music audio is quite distinguishable with the features set we choose for training.

Then, we tested our model that has been deployed on Storm topology and we observed that accuracy is a little less in this case. We tested for around 8 videos - 4 music videos, 2 speech videos without music and 2 other random videos with speech and music. We observed our model correctly identified only 2 out of 4 music files as music and rest 2 it identified as other videos. It correctly identified the 2 speech videos as speech and 2 other category videos as other. Over for this short testing our model displayed 75% accuracy. We plotted a short confusion matrix depending on the result and calculated the other factors like precision, recall and f-measurement. All the calculations are shown below.

The final result of application at the client end are:

- Annotating the video with number of people in it.
- Extracting the results from mongoDB to identify the type of the audio.
- Annotating the video with results.

5.1 Observed Metrics:

Below are the model evaluation results for the decision tree classification model generated. We use the conventional evaluation methods for a model such as representing a confusion matrix which shows the actual and predicted results from the model. Other metrics like accuracy of the model, precision, recall and F-measure for each classification class and overall error rate of the model is also calculated based on the conventional formulae involving the True Positive (tp) , True Negative (tn) , False Positive (fp) and False Negative (fn) measures.

Confusion Matrix:

Predicted / Actual	Music	Speech	Other	
Music	2	0	2	4
Speech	0	2	0	2
Other	0	0	2	2
	2	2	4	8

Accuracy:

tp + tn / total

Observed Accuracy = 6/8 = 0.75 (75%)

Precision:

tp / (tp + fp)

Music class = 2/2 = 1

Speech class = 2/2 = 1

Other class = 2/4 = 0.5

Recall :

tp / (tp + fn)

Music class= 2/4 = 0.5

Speech class =2/2 = 1

Other class = 2/2 = 1

F-measure:

2*(Precision*Recall)/(Precision+ Recall)

Music class = 0.66

Speech class = 1

Other class = 0.66

Error Rate:

(fp + fn) / total cases

Observed Error Rate= 2/8 = 0.25 (25%)

5.2 Storm Vs Spark

The above discussed application workflow was implemented in Spark for testing the static data in addition to Storm which test the real time data and compared the performances of both the frameworks. Accuracy observed in Spark was above 90 percent based on the classification and recognition performed on Spark. Whereas the accuracy in Strom was about 70 percent.

6. CHALLENGES

One of the major challenge we faced was to integrate all the systems Spark, Kafka and Storm together. Kafka and storm both support java while Strom supports Java, Python and Scala. First, we developed our Spark code in Python as we faced complex issues while integrating it with other technologies we switched to Scala. We observed that kafka system is stable we did not face many issues in implementing it. On the other hand strom is not totally a stable one so debugging issues was a difficult task. Therefore in our future work we would implement heron instead of Storm which is stable than prior one.

The minimum RAM size required for all the systems is 8GB. Though our machine has minimum requirement we faced a lot of memory issues , therefore it's better to have a minimum of 16 GB RAM on the machine that implements these technologies.

7. CONCLUSION

The system is a complex one as it involved various system integrations and deals with real time data. But the solution we provided can increase the efficiency and accuracy of the model. We successfully implemented our model to the below application use cases:

- Automating analysis on number of people attending different types of events.
- The context of the event such as a Musical event or a speech oriented event such as TedTalk or Conference proceedings.
- Presenting results to the Client in real-time

8. FUTURE WORK

We would like to extend this project by adding a few more sub classes to music class such that the system can also detect the type of music playing on in the room. This can be achieved by collecting a lot more training data and by building a betted decision tree with maximum depth possible. to achieve this in real time we need to add a few more bolts to topology according the number classes need to be extended.

This model can further be extended to Face recognition system in a classroom scenario to automate attendance.We can achieve this by selecting a good SIFT features and also we can add more analytics like gender and age of attendees for event analysis. We can extend this project in many different ways a few of the interesting use cases are listed below.

- Adding Audio annotations to help the blind.
- Combining audio and video analytics for more results like group discussions, live-theatre plays, surveillance implementations, etc for the user.
- Addition of additional features such as surveillance system for exit and entry of students in the class at various points of time during the class
- This project can also be extended to surveillance systems for public meetings or gatherings.

ACKNOWLEDGEMENT:

We'd like to thank the ideas, guidelines and suggestions given by Dr. Yugyung Lee, Mayanka Chandrashekar, Ravikiran Yadavalli and Naga Krishna Vadlamudi. We'd also like to thank the authors of OpenCV and Storm research papers, upon which our application is developed.

REFERENCES

[1]. Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.

[2]. Zaharia, Matei, et al. "Spark: cluster computing with working sets." HotCloud 10 (2010): 10-10.

[3]. R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, H. Jiang, T. Qiu, A. Reznichenko, D. Ryabkov, M. Singh, and S. Venkataraman. Photon: Fault-tolerant and scalable joining of continuous data streams. SIGMOD, 2013

[4].B. D. Ziebart, A. Maas, J. B., and Dey, A. K. 2008b. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. Proc. of Ubicomp. 322–331.

[5].C. L. Philip Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," Inf. Sci., vol. 275, pp. 314–347, Aug. 2014.

[6].N. Zhang, F.-Y. Wang, F. Zhu, D. Zhao, and S. Tang, "DynaCAS: Computational experiments and decision support for ITS," IEEE Intell. Syst., vol. 23, no. 6, pp. 19–23, Nov./Dec. 2008.

[7].Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. 2011.

[8].http://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_1441:Video Representations

[9].http://spark.incubator.apache.org/docs/latest/streamingprogramming-guide.html: Spark Streaming.

[10].http://openimaj.org/openimaj-image/faces/dependencies.html: Open IMAJ documentation for face recognition

[11].http://openimaj.org/tutorial/sift-and-feature-matching.html: Open IMAJ documentation for SIFT feature matching

[12]. http://openimaj.org/tutorial/eigenfaces.html: Eigen faces for face recognition using OpenIMAJ

[13]. http://klresearch.org/IJMSTM/papers/v2i3_2.pdf: Class Room Attendance System Using Facial Recognition System

[14].http://blog.mashape.com/list-of-10-face-detection-recognition-apis/ : Face Detection / Recognition APIs, libraries, and software

[15].Kyuwoong Hwang and Soo-Young Lee, Member, IEEE. Environmental Audio Scene and Activity Recognition through Mobile-based Crowdsourcing. http://ieeexplore.ieee.org.proxy.library.umkc.edu/stamp/stamp.jsp?tp=&arnumber=6227479:

[16].http://storm.apache.org/releases/1.0.2/Concepts.html: Storm Concepts and Components

[17]. https://kafka.apache.org/documentation: Kafka messaging system concepts, components and implementation Documentation

[18].http://spark.apache.org/docs/latest/mllib-decision-tree.html : decision tree documentation using Spark

[19].https://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html : Help on Apache Tomcat application deployment.

[20]. https://en.wikipedia.org/wiki/Confusion_matrix: Model Evaluation Metrics based on Confusion Matrix.

[21].http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html: Documentation on Face Recognition using OpenCV

[22]. Cascade Classifier Training using HaarTraining Algorithm: http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html

[23]. https://en.wikipedia.org/wiki/Haar-like_features: Haar-like features for object recognition

[24]. Cascading Ttrainign classifiers for machine learning: https://en.wikipedia.org/wiki/Cascading_classifiers

[25] Apache Aurora. http://aurora.incubator.apache.org

[26] Apache Samza. http://samza.incubator.apache.org