VideoCeption: A content based Video Search

Vihari Gorripati, Sri Praneeth Iyyapu, Naresh Goud Pogakula, Mounika Yalamanchili

Abstract— The application that we proposed in this paper is a video searching web-application. Our web- application is unique because it takes the video-search to another level. This paper discusses our implementation of the application by which user can get videos based on the actual content of video instead of names or tags associated with video. It is significant because currently there is no video-search application available that performs similar operation. The implementation discusses how the big data technologies like storm, kafka and video processing techniques can be used together to get the required results. It explains how supervised and unsupervised learning algorithms can be used to build the model for sports videos, partiucularly baseball videos so that our application can match thes baseball models in the searched videos. If the model matches, then the video is classified as baseball video and displayed to the user. This way we can create a lot of models so that the video search can be spam free as the result is on completely contend based.

Index Terms—Big Data, Storm, Video Processing, Kafka, Supervised/Unsupervised Learning

1 Introduction

Several applications related to weather forecasting, medical results, security surveillance and road traffic conditions collects large volumes of video data every second where evolves the term Big data. The processing of this video data is very complex as it requires highly reliable, fault tolerant, scalable video processing techniques or frameworks. In this paper, we have discussed few frameworks and technologies for processing and analyzing the content of the video. Video processing is broadly categorized into two types: offline and online processing. Spark framework is used for offline video processing. Spark processes the image/video files by parallel computations using resilient distributed datasets (RDD). Spark is implemented using Scala, a high-level programming language.

Online data processing requires complex computations which can be achieved through Apache Storm, which is a real-time distributed data processing engine. It performs various analytical jobs on the streaming data. Kafka is used as a communication link between Spark (batch processing) and Storm (stream processing). Kafka is a high throughput distributed messaging system. The evolution of deep learning and several machine learning algorithms have taken the video processing to another level in various fields like image detection, speech recognition and activity recognition. The machine learning libraries are available in Spark, which enables us to train the video data and its evaluation/testing is performed using Storm. Using machine learning algorithms, we can even generate smart tags for the video.

The Features of our application is listed here:

1. Smart Search: This smart-search is based on the video content. The entire video is analyzed based on the visual and the audio content of the video uploaded to the application. So this is a smart way of searching the video instead of routine name based search.
2. Smart Tags: The smart-tags can be user given while uploading the video or else our video-engine analyzes the video content and gives some smart-tags that pertain to the video. The video-search is dependent on the generated smart-tags that are generated.
3. Spam Filtering: Many spam videos are uploaded every day to YouTube and other websites. Currently there is no way to possibly identify whether a video is genuine or spam. We try to analyze the content of the video and label the video as spam.
4. Video Recommendations: This feature is similar to already existing video-recommendations. But we try to give smart recommendations based on our smart-tags.
5. Parental Control: We can also categorize the videos based on content and generate a flag called "not suitable for children". Hence if the parents decide to turn on a feature called "parental control", all the inappropriate videos would be removed from the search.
6. Content based Navigation: You can navigate to the particular content/entity in the video using this feature.

2 Related work

2.1 Computation using Spark

Scheduling, load balancing and fault tolerance are the important properties of any model. Existing frameworks like Hadoop MapReduce is proved deficient for two use cases: (i) Iterative jobs: Most of the Machine learning algorithms apply same function multiple times to the similar dataset for optimizing a parameter. (ii) Interactive analytics: In Hadoop, there is a significant latency with each query, it reads the data from the disk by running a separate MapReduce job. To provide similar fault tolerance and scalability properties to MapReduce, Spark which is cluster computing framework is introduced. The key feature of Spark is resilient distributed dataset(RDD) in which resilient represents Fault tolerance, distributed represents can run on multiple machines and dataset represents collection of objects. The main use of RDD is that, we can reuse it in parallel operations by caching it in memory across multiple machines. Spark is implemented using Scala, a programming language for Java Virtual

Machine (JVM). Spark provides mainly two abstractions for RDDs and parallel operations for parallel programming. Also, Spark aids two shared variables that are implemented in functions that runs on the cluster. In Spark, Scala object represents each RDD. Creation of RDDs can be done in four ways: (i) From any shared file system like Hadoop (ii) By parallelizing Scala collection (iii) Transform existing RDD (iv) Change the persistence of an already existing RDD

Various parallel operations like reduce, collect and foreach can be implemented on RDDs. Currently grouped reduce operation is not supported by Spark. Spark helps to create shared variables like broadcast variables and accumulators.

(i) Broadcast variables: This object enfolds the value and makes sure that it is copied to each worker only once.

(ii) Accumulators: These variables that can be only read by the driver. It provides add-only semantic that are fault-tolerant. Spark core implements RDD with the same sample interface that contains three operations: getPartitions (returns partition IDs list), getIterator (iterates the partitions) and getPreferredLocations (achieves data locality through task scheduling).

2.2 Stream processing using Storm

For processing complex computations on real time streaming data, Storm is used which is a realtime, distributed and stream processing engine. It is scalable, resilient, extensible and an efficient system. The Storm consists of three main components: spouts, bolts and topology. Streams of tuples is taken as data. Spouts are the data sources or point of entry in Storm. Bolt is a data processing unit. Topology is a network consisting of spouts and bolts.

Storm has master-slave architecture. A master node is responsible for scheduling of tasks in the cluster which is performed by Nimbus. The actual tasks are performed on worker nodes. Storm UI is used to view the clusters and the topologies. Zookeeper performs cluster management. It can have either a single or multiple Zookeeper nodes i.e. (2n+1). Supervisor supervises the process in which various worker nodes executes the code. Logviewer is used to pinpoint any issue during the execution and to review the log file on the worker nodes.

2.3 Kafka

High volumes of log data are generated at any internet based company which includes page views, clicks, user activity events associated to logins, likes, comments and search queries. Log data has always been an important metric for analytics. These analytics include recommendations, search relevance, reporting and ad targeting, newsfeed features and several security applications. The features of existing systems are low throughput, no rewindable consumption, push model and tuned for low latency. Kafka is preferred because of its high performance, scalability, low operational overhead and durability. Kafka is a publish subscribe messaging system.

The basic concepts used in Kafka are topics, producers, brokers and consumers. Topic: It is a stream of messages related to a particular type. Producer: Publishes messages to a topic. Broker: A set of servers in which the published messages are stored. Consumer: Subscribes to one or more topics from the broker.

A Kafka cluster consists of multiple brokers because of its distributed nature. A topic is further divided in to multiple partitions for load balancing. Kafka is efficient because of its very simple storage system. Unlike traditional messaging systems, an explicit message id is not associated with a message stored in Kafka. Kafka is efficient because of the careful transfer of data in and out of Kafka. Batch processing is used for sending and receiving of data. No message caching in JVM is required. It relies on file system buffering and also invokes zero copy transfer.

In distributed co-ordination, each producer publishes messages to different consumer groups that jointly consumes a bunch of subscribed topics. No co-ordination is needed across consumer groups until load rebalancing. No central master node is employed, since adding a master node further complicates the system as we need to worry about master failures.

To facilitate the co-ordination, Zookeeper is used in Kafka mainly detects removal and addition of brokers and consumers, triggering a rebalance process and maintains consumption relationship. During the startup of consumer or when consumer is informed about a broker or consumer change through the Zookeeper watcher, a rebalance process is intiated. The rebalance process algorithm is given below: At-least-once is guranteed at Kafka to keep away from lock corruption. Kafka saves a CRC for each published message which also enables us to check network erros. The main drawback of Kafka is whenever a storage system on broker is damaged, all unconsumed messages are lost forever.

From the above experimental results, we can see that on an average Kafka producer publishes messages at the rate of 50,000 messages/sec for the batch of 1 and 400,000 messages/sec for the batch of 50. When compared with ActiveMQ and RabbitMQ, Kafka producer performs better for a batch of 50. Reasons behind this are: (i) Currently Kafka producer won't wait for any acknowledgements from the Kafka brokers. (ii) It has more efficient storage layout. From the results, Kafka consumer consumed 22,000 messages/sec which is more than 4 times of what ActiveMQ and RabbitMQ have consumed. The reasons behind this are: (i) Kafka has more efficient storage layout (ii) There are no disk write activities performed on the broker.

2.4 Integration of online processing, offline processing and deep learning

Some applications like security surveillance and smart transportation handles huge volumes of video data. It requires reliable, quick and fault tolerant video processing. The latency of processing such videos should be in the order of seconds. The advancements in the field of deep learning has greatly impacted image and video processing research. Using deep learning, researchers used a deep convolutional neural network (DCNN) for problems

like activity recognition and face detection. By considering complexity, velocity and high volumes of video data, cloud computing technologies has been introduced. To engage cloud computing technology with deep learning, a deep intelligence framework has been developed which integrates batch processing, stream processing and runtime deep learning. The effectiveness of this framework can be evaluated using some key attributes like performance, availability, scalability, modifiability, portability and usability. Some software architecture styles have been mapped to these quality attributes like: (i) Service oriented architecture (SOA): This architecture empowers functionality upgrades, portability and modifiability. (ii) Publish-subscribe: This mechanism handles decoupling of producers and consumers, by monitoring the running status of the framework which enables availability. (iii) MapReduce: This paradigm is used to analyze large amounts of video data through parallel processing which helps to achieve performance availability and scalability. (iv) Shared data: Video data is fed to different processing components like video summary and background subtraction. Shared video pattern enhances the performance. (v) Layered architecture: This framework segregates different concerns during video processing, which in turn helps to achieve usability, modifiability and portability.

The architecture consists of four layers: Data retrieval layer, Data processing layer, Data service layer and Domain service layer. Data retrieval layer: It collects video data from different cameras through webcam API and media server. The output of this layer is fed as input to data processing layer in the form of video stream or images. Data processing layer: It consists of two packages like online processing and offline processing. The offline processing depends on Apache Hadoop which performs some processing tasks like encoding and decoding, video summary extraction. Similarly, online processing depends on Apache storm for performing background subtraction. Data service layer: Intermediate results of video processing from data processing layer is fed as input to data service layer. This layer performs tasks like Decision making (classification, clustering, statistics) and Resource scheduling (association analysis, regression analysis for offline data mining and metrics sensing, bottleneck detection for online data mining) Domain service layer: This layer is used by developers for creating various domain applications like smart campuses and smart transportation. The main task of this layer is to aggregate the results of online and offline processing.

To evaluate the effectiveness of the framework, traffic statistics application based on DCNN is considered as a use case that counts number of motorcycles, cars and pedestrians detected by the cameras. The dataset contains 46,906 images as training set and 8,274 images as testing set. Based on the results, we can say that as the image size increases, the number of layers and accuracy also increases. Also, the framework achieves the real-time performance in the order of seconds.

the scalability of the framework is tested and we've concluded that as the number of cameras are increased over 100, the corresponding processing time increases on a large scale. Fault tolerance of the framework is evaluated by killing some nodes in Storm. This doesn't impact video processing as Storm restarts the workers of the failed node on another node and the processing time is also stable (fig. b) which clearly indicates that the framework is fault tolerant.

2.5 Approach for boosting streaming video

In paper titled "Boosting streaming video delivery with WiseReplica", authors proposed a new framework to stream videos online very fast without compromising on the quality of the video. As the number of users who are interested in viewing online videos are exponentially growing day by day, the traditional content delivery networks lacks in sending quality videos in time. As Internet is accessible to almost every person on the globe and the cost of internet data rates are becoming cheap, services such as online streaming, live-video streaming and video on demand services are becoming more popular among users. Users wants to watch any video such as movies, games, tv series etch on demand but they expect the less re buffering time and more quality of the video.

The WiseReplica Approach discussed in the paper overcome the challenges in the delivering videos. They have integrated a machine learning model in their framework so that it can detect the more popular videos dynamically and so provide more storage and bandwidth for that demanded videos. In the learning phase, the framework takes the training data set and draw some conclusions on the type of video popularity and quality. Then based on the conclusions it will categorize the streaming videos so that more resources can be provided to stream videos that are really being watched by more number of users. This approach ensures that the average bit rate for video data is maintained so that end users can watch videos without buffering and also the minimize the resource allocation for the videos which are not very popular or having less probability to watch by more users. The simulations performed on the YouTube channel indicates that the proposed framework satisfies end users in watching online videos.

2.6 Learning representations of Video data

In the quest of searching videos by their content, there are lot more research is going on to detect the underlying patterns in the video. In order to find the patterns, first the system need to be learned from the training data sets. But here there two kind of environment i.e. supervised learning and unsupervised learning. In supervised learning the training data is labelled so that it can learn and draw logical conclusions from each labelled dataset so that it can perform really well in test data or real data. This is good in various kind of pattern recognition requirements like labelling images or speech recognition etc. But the videos normally consist of very large frames and there are more number of videos available on the internet. So in order to find representation in the video, there needs to provide lot of labelled trained data to the system. But this kind of supervised learning is not feasible in this situation. So the research is focusing on the

unsupervised learning. It will enable to find the video representations more effectively and It can predict the representations of the real time videos.

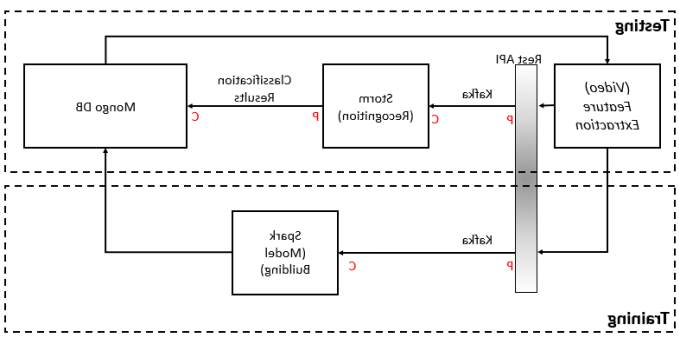
The paper on "Unsupervised learning of video representations using LSTMS' focused on considering the unsupervised environment. The authors proposed a new model in which they have used long short term memory networks in order to find the representations from the video. These networks take the input sequence of the video frames and do encoding using LSTM to produces some useful representation. This representation can be decoded with other LSTM to see if it matches with the input sequence or It can use to detect the future representations of the particular video. The authors done simulations using proposed model and they got very good results in finding good representations in the video with the LSTM model.

2.7. Real Time Video Processing Application:

There are a lot of other real time applications associated with video processing. The surveillance video processing is very useful in detecting the abnormal patters and alerts the authorities about the detection. As the video is processed without human interaction, it will have a lot of real time applications associated with it. Another such example is discussed in the paper "Big data analytics on CCTV images for collecting traffic information". In this paper authors discussed the processing of real time CCTV images using big data analytics and notifies real time traffic information to the public.

3 Proposed Solution

The proposed solution is to develop a front end web application from which user enters search keywords like baseball etc. Then our application redirects that search results to youtube and gets the videos from youtube in real time. Here the the videos which are returned by youtube is based on the name and tags of the videos given by uploader. So now our system will take these videos and send it to storm in real time. Before this happens, we have builded few models which can recognize certain patterns in videos for example ball and bat models can be used to detect the baseball videos. We have developed these models in spark using video processing techniques and build the models using decision tree alogrithms. We have uploaded these decision models in online file storage like mongodb.



So now when the videos returned from youtube is fed into storm in real time. Based on the search keyword, strom can bring the corresponding models from mongodb and compares the videos with the fetched models. If the content really contains the features of models, then that video is returned to our website so that we can display the videos to the user so that user can see spam free videos.

4 Implementation

The idea is to train the system to find out the videos related to sports field (especially baseball). To achieve this, we have created objects like ball, bat etc. and stored these models in mongo DB and these models are used to match with the objects found in other videos. If the same type of object present in the videos, we classify them as baseball videos. So these videos can be sent to user interface so that user can see these videos only. As this search is not based on the name or tags of the video and only based on the content of the video, user can't see spam videos.

Training**:** First we have extracted the features and send those model to spark machine learning library. After that we have stored that model in mongo DB. Testing**:** Then other videos are searched using the model. These tasks are executed in storm. Kafka is used to send and receive the video in between them

The first part fo the implementation: In order to build the models, we have given the sample baseball videos to the spark where features of the video are generated using video processing techniques. This features are taken as input and developed model using decision tree algorithm and the model is stored in mongodb. The second part of the implementation is to create storm topology and storm bolts and Kafka. The Kafka framework is used to send and recive from storm. We have created two bolts in which 1 bolt is realted ball bolt and other blot is for base ball bat. These bolts take the input video and compares the features of that particular video to model stored in mogodb. The final results whether that video is realted baseball or not is stored in mogodb. The third part of implemention is to design web application from which user enters search keywords and the final results are displayed to user. The complete implementation of the project can be found in the following link:
Github URL:
https://github.com/niCEnANi/VideoCeption
Youtube URL:
https://www.youtube.com/watch?v=enNJF0dKHGI&feature=youtu.be

5 Results and Evaluation

The proposed application is tested with the different categories of videos. Our system detects the videos with 55-65% accuracy. As we have developed few models for the baseball video, sometimes our application doesn't able to recognize baseball videos. The training data in our use case is video so it takes lot of time to generate features of that video. So we have trained our system using 20 to

35 distinct baseball palying videos. But the models are need to be increased paritcualr to the type of videos and so that the propsosed sytem can be worked more efficiently.

6 Conclusion

The paper discusses the need to have a content based video search and thoroughly discusses our implementation of such system using big data technologies, video processing techinques and machine learning algoirthms. This paper also expalined the implementation of the proposed sytem.

7 Future work

We are planning to train more data in order to build more efficient models so that the accuracy of the final results can be increased. At the same time, we have focused on the one single type of videos like baseball but we would like to build models on different domains of videos and extends this application more user friendly.

References

[1] Bekris, Kostas, et al. "Cloud Automation: Precomputing Roadmaps for Flexible Manipulation." Robotics & Automation Magazine, IEEE 22.2 (2015): 41-50.

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

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

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

[5] Zhang, Weishan, et al. "A deep-intelligence framework for online video processing." IEEE Software 33.2 (2016): 44-51.

[6] Silvestre, Guthemberg, et al. "Boosting streaming video delivery with wisereplica." Transactions on Large-Scale Data-and Knowledge-Centered Systems XX. Springer Berlin Heidelberg, 2015. 34-58.

[7] Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhutdinov. "Unsupervised learning of video representations using lstms." CoRR, abs/1502.04681 2 (2015).

[8] Twitter Heron - Stream Processing@Scale by Karthik Ramasamy, Twitter

[9] Wu, Dongyao, et al. "Building Pipelines for Heterogeneous Execution Environments for Big Data Processing." IEEE Software 33.2 (2016): 60-67.

[10] Boykin, Oscar, et al. "Summingbird: A framework for integrating batch and online mapreduce computations." Proceedings of the VLDB Endowment 7.13 (2014): 1441-1451.

[11] Lv, Yisheng, et al. "Traffic flow prediction with big data: a deep learning approach." Intelligent Transportation Systems, IEEE Transactions on 16.2 (2015): 865-873.

[12] Kulkarni, Sanjeev, et al. "Twitter heron: Stream processing at scale." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

[13] Neumeyer, Leonardo, et al. "S4: Distributed stream computing platform."2010 IEEE International Conference on Data Mining Workshops. IEEE, 2010.

[14] Song, Xuan, et al. "A Simulator of Human Emergency Mobility following Disasters: Knowledge Transfer from Big Disaster Data." Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

[15] Wu, Dongyao, et al. "Building Pipelines for Heterogeneous Execution Environments for Big Data Processing." IEEE Software 33.2 (2016): 60-67.

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