Data Analytics - Detection of difficulty areas in videos based on student viewing activity

Arinjoy Basak, Data Analytics group, Mentor: Sukla Nag,
Principal investigator: Dr. D. B. Phatak

July 3, 2015

Outline

A situation

- Students watch the videos of a course on a regular basis.
- They may have difficulty in the videos.

This may be due to (among other things)

- The course videos being difficult for the students to understand
- The videos not providing sufficient clarity in a subject

Result

- i) The students do not understand the matter clearly.
- ii) They have troubles in assimilating the matter.

A situation

- Basically, the instructor would like to know about these things,
- So that he can understand how his students are responding to his videos - enjoying them or otherwise.

Basic outline of the idea

- Analyse the student learning behaviour and activities through the events in the lecture videos.
- The locations of the pauses could be collected as events, and metrics could be developed and appropriate visualizations made to notify or warn the instructors about the difficulty faced by students in particular parts of materials.
- This could then be met by appropriate measures on the teacher's end, such as
  - Adding more explanatory material
  - Release of an additional video, or release of more lucid lecture video
  - More quizzes and practice exercises
  - .... and so on.

Steps in the project

So, given what we aim to achieve through our project, these are the steps we took, one by one:

- Extracting the data
- Extracting and Processing the data
- Creation and prototyping of the data model
- Final implementation of the data model

Technologies used

1. Prototype stage :
    - MySQL : The database management system used
    - Python : The language base used for programming in this project
2. Final implementation stage :
    - SparkSQL : The fast data processing engine, using in-memory primitives, and faster than Hadoop
    - Hive : The data warehouse infrastructure

Source of the data

- IITBombayX records the events occurring on the platform as log events in files, spanning gigabytes.
- Each of the records is a JSON object, having some keys that are common to all events, and some which are dependent on the different events themselves.

The following is an example of a log event:


```
{"username": "arinjoy15", "host": "10.105.25.74",
 "event_source": "server", "event_type": "/dashboard",
 "context": {"user_id": 11, "org_id": "", "course_id": "",
 "path": "/dashboard"}, "time": "2015-06-12T10:20:08.703318+00:00",
 "ip": "10.105.1.7", "event": "{\"POST\": {}, \"GET\": {}}",
 "agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0",
 "page": null}
```


- These logs were then flattened out, and inserted into a MySQL database. This was the source of data for my project.

Extracting and Processing the data

In order for us to determine the model on which we were to base our system, we drew our data from the IITBombayX log files, which were cleaned and made appropriately accessible through MySQL queries, to extract the features that we required. These features included the following:

For a given video:

- count of the number of times the different regions of the videos were accessed by the different students.
- total duration spent on the video by the student.

Our hypotheses: The more number of times a particular region is visited by students, and the longer time students spend there, the more difficult it is for them.

Extracting and Cleaning the required data

- A first step involved exploration of the log events and the associated attributes of the logs, to identify the features that we would require to identify each video element for each course, each video event, and to extract the data.
- At this stage, SQL queries were used to filter and select the relevant data for each video and each user watching it.

Extracting and Cleaning the required data

- To start with, we first focused our search on videos belonging only to the CS101.1x course (155 videos).
- To further restrict our search for designing the model, we selected the video with the most number of views, based on the log data, and then selected the user who has accessed that video the most number of times.
- The end result of this process was a sufficient amount of log data for a single user viewing a single video of a particular course, which would provide enough insight into the patterns of general behaviour of a student or learner.

Extracting and Cleaning the required data

- The log data also contained repetitions, redundancies and errors.
- Prior to actual processing, the data extracted from the MySQL table was cleaned further through the Python script written for the prototype.
- Certain events also had to be reordered and arranged for our purposes in order to make the resultant data more sensible.
- The development of a complex logic and error checking functions and conditions, and a thorough study of the user events and their sequencing in the logs (and even simulation by self!)
- Result: a cleaned, chronologically arranged set of events for a particular user

MySQL Queries

At this stage, primarily the data was fetched from MySQL tables for processing, through appropriate queries.
The MysQL tables were:

- UserSessionOldLog : attributes of interest were userName, moduleSysName, courseName, currVideoSpeed, oldVideoSpeed, createDateTime (nearly 4 million entries)
- CourseVideo : attributes of interest were videoSysId, videoUTubeId, videolength (a field we extracted and added later) (1600 entries)

MySQL queries

Some of the queries are as follows:

- Fetching top 50 most watched videos in CS101.1x

```
select t.*, CV.videoUTubeId, CV.videolength from ( select moduleSysName, count(eventName) eventCount from UserSessionOldLog where courseName='CS101.1x' and moduleSysName is not NULL and eventType='video' group by moduleSysName order by eventCount desc limit 50 ) t, ( select * from CourseVideos where courseName='CS101.1x' ) CV where t.moduleSysName = CV.videoSysName order by eventCount desc;
```

- Highest watched video : videoSysId '67a8559582864d6a8148e2ef5c997e8f'; So, number of viewers in descending order of activity were found as follows:

```
select userName, count(eventName) watched from UserSessionOldLog where courseName='CS101.1x' and moduleSysName='67a8559582864d6a8148e2ef5c997e8f' group by userName order by watched desc limit 10;
```

MySQL queries

- Finally, for a given user, we found out the distinct events of interest giving us the behaviour of the user in the following manner (say, for the same video as before, but userName='ricky'):

```
select eventType, eventName, moduleSysName, eventSource, oldVideoTime, currVideoTime, createDateTime from
UserSessionOldLog where userName='ricky' and (eventType in ('video') and
moduleSysName='67a8559582864d6a8148e2ef5c997e8f' or eventType in ('video', 'navigation') and eventName in
('pageclose', 'saveuserstate') ) order by createDateTime;
```

Video length extraction using Google YouTube API

- To determine a way to measure the relative amount of time spent by the student on a particular video (such as the fraction of the video playing time)
- Additional work was done to extract the actual duration of the videos on the all courses using the YouTube id's available for the videos in the CourseVideos table.
- The Google YouTube API basically sends GET requests online to the API together with the video id and the fields of the 'video' element (a JSON object) that it has to return.
- We required only the contentDetails attribute of the video element, in which, the value corresponding to the key 'duration' gave us the duration of the video in a ISO 8601 format (PT59M59S), which was processed to make it in seconds.

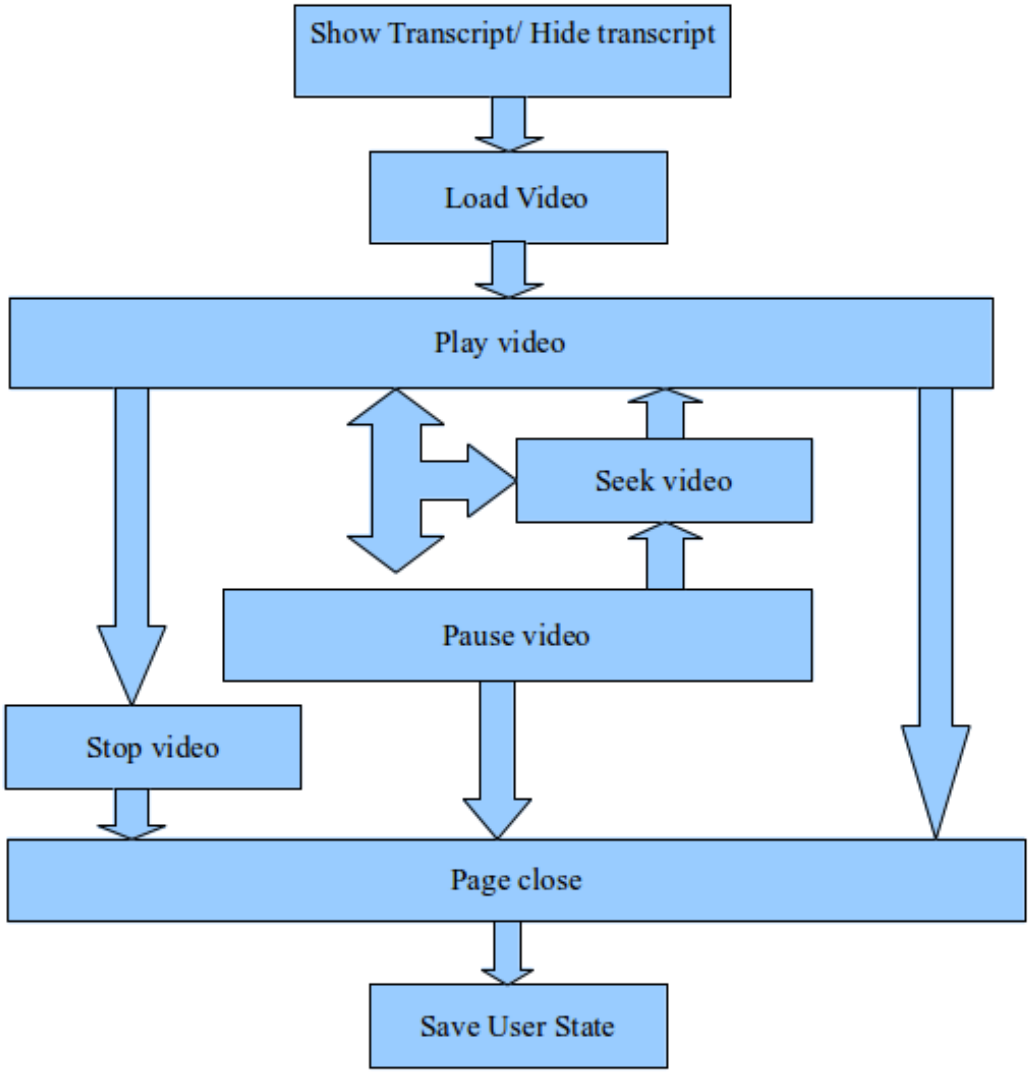
Redundancy removal and Error correction

- Removing the redundant data that could be represented by one instance, in order to shorten processing time.
- This was done programmatically, using complex logic.
- The following is a simple description of the criteria used to filter out unnecessary data from the results of the query by comparing the rows pairwise, in order of the createDateTime field:
  if not ((activity1.eventName == activity2.eventName) and (activity1.oldVideoTime == activity2.oldVideoTime) and (activity1.currVideoTime == activity2.currVideoTime))
  Then keep the 'activity' in the final list of rows.
  else Discard.

Creation and prototyping of the data module

- Following the extraction of the data and its cleaning, we had to process the data in order to extract the two features we had proposed earlier.
- timeFrame : a segment of the video of a certain length (here, we considered 4 seconds)
- To find the number of accesses to a timeFrame and time duration spent in a timeFrame by a user, given his activities.
- For this purpose, we considered the type of events we were dealing with: loadvideo, playvideo, pausevideo, seekvideo, stopvideo, saveuserstate, pageclose - and the relation between these events and their sequence of ordering.
- Inferring actual time spent by the student from the sequence of events, involving tracking whether the video was being played or paused, and when the page was opened or closed.

A simplified and ideal sequence of video watching



Creation and prototyping of the data model

- The interval between a play and a pause event was considered as the time spent by a student.
- The time spent upto a seek was considered the time spent by a student, and the seek marking a region that is probably being visited a second time.
- This stage also involved checking for errors and discrepancies in the data, which was dealt with appropriately to find the proper amount of time spent by a student watching the video.
- Most of these errors in the timing were probably due to disturbances in the networks, or interleaving of events that is beyond our control at this stage (for example, saveuserstate event always follows pageclose event, but sometimes, it appeared in logs before it - probably due to different ordering in arrival)

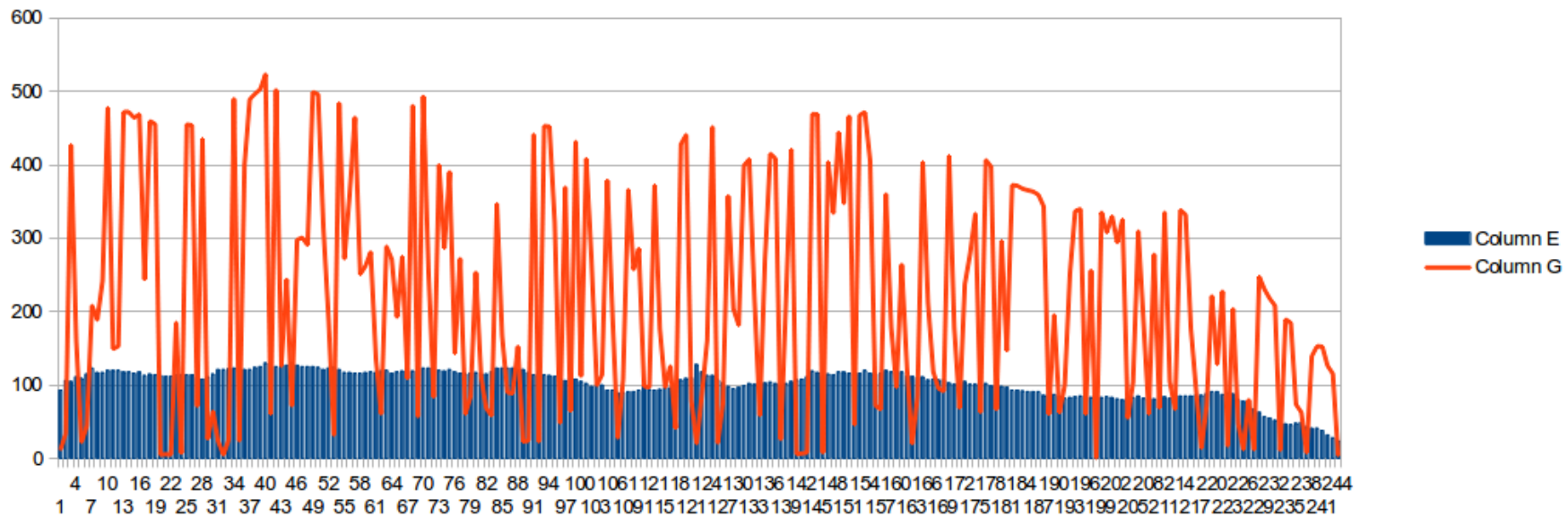
Creation and prototyping of the data model

- The output of this process would be entries of a MySQL table, having a primary key consisting of:
  videoSysId, userName, timeFrameId
  and the attributes frameAccessCount and frameDuration.
- Finally, we can perform our visualizations, inferences, etc. based on this data.
- For example, :
  - Total time spent by a user on a particular video
  - Total time spent by all users in a particular timeFrame
  - Total number of accesses to a particular timeFrame

  which could be made available in the corresponding summary tables as well.

An example graph

The following graph was plotted on the basis of the data obtained by running the module on a single video, and its top 50 users.



Legend:X-axis - Time frame number. (dimensionless)
Red curve - The total time spent by all the users in a particular timeFrame. (seconds)
Blue Columns - The number of accesses to those timeFrames. (dimensionless)

Final implementation of the data model

- After the data model was finalized, we took it the project to the final step.
- Implement the data model in Spark
  1. Draw the data from populated Hive tables containing the cleaned Users log data.
  2. Put the processed data back into the Hive tables.
  3. Create summary data that would be populated into the MySQL Analytics database tables, used for visualization

Why Hadoop, Hive and Spark?

- The data module would be working as a live application in a Big Data environment
  - Deal with tens of gigabytes of log data generated by the IITBombayX platform.
- Hadoop (for distributed filesystems)
- Hive (for creating data warehouses)
- Spark (for running SparkSQL queries and executing operations on the data)
  - Spark API and SparkSQL supports MapReduce workflow, which equals to = Parallelization
  - Which equals to = Faster processing!

Extracting the data from Hive tables

- To transfer the User Log data from the MySQL database to the Hive tables, we used the Sqoop utility.
- While Spark does run very fast, it is required that the data must be in the main memory in order for the processing to be very efficient.
- So, specified a single query to draw the required user data, using the concept of RDDs and collect()

Query used:

SELECT DISTINCT orgname, coursename, username, modulesysname, eventtype, eventname, oldvideotime, currvideotime, createdatetime FROM userlogsmall WHERE eventtype=video OR (eventtype=navigation AND eventname=page_close) ORDER BY orgname, coursename, username, createdatetime

Extracting the data from Hive tables

- To begin with, in order to test the final implementation of the data model, we selected a subset of the original records to contain only the records of the top 30 most active users on the platform, using the following MySQL/Hive query

```
create table userlogsmall ( select USOL.* from UserSessionOldLog USOL, (select userName, count(eventType) cet from UserSessionOldLog where userName is not NULL and userName not in () and courseName=CS101.1x group by userName order by cet desc limit 30) t1 where USOL.userName = t1.userName and courseName=CS101.1x)
```

Filtering of events per user and per video

- Events for each user and each video were separated out from the event logs.
- The result of this process was
  - A single key for each orgname, coursename, videoname and username
  - Its corresponding value being a list of events for that user and that video in that course
- Conditions for filtering (examples)
  - For each log event of the type video, append to the list of events for the corresponding key.
  - If a log event is the very first log event in the sequence of events for a particular user, if not a save user state or a page close event, then append to new list of corresponding key.

Processing and generation of results for final summary table : MapReduce

- At this stage and in the summary stage, the concept of MapReduce workflow was used to speed up the process of analysing the data generation and processing tasks.
- The concept of MapReduce revolves around the fact that the data can be converted into a set of key-value pairs by passing it to a mapper function, and these key value pairs are then merged together using a reducer function to create the final result.

Processing and generation of results for final summary table

- RDD (Resilient Distributed Database) was created for each of the (key,value) pairs
  ((uIITBombayX, uCS101.1x, ufa1f6040f46a43298cc25fc33db89a83, uShrikrishna), [(uvideo, uload_video, 0.0, 0.0, u2015-02-17 08:22:03.0), (uvideo, uplay_video, 0.0, 0.0, u2015-02-17 08:29:32.0), (uvideo, upause_video, 0.0, 11.525, u2015-02-17 08:29:32.0), ... and so on....])

- Sent to a mapper function called processing(), which essentially performs the analysis required on the key-value pairs, as described in the section on the development of the data model prototype.
  eventMap=sc.parallelize(eventDict.items()).map(lambda p: (p[0], p[1]))

  videoDetails = eventMap.map(processing)

Processing and generation of results for final summary table

- The output produced key-value pairs having the same keys, but having the timeFrameBuckets list as the value.
  - timeFrameBuckets list is a list of bi-tuples (x, y), where x denotes the number of accesses to a particular timeFrame, and y denotes the time spent by the user in that timeFrame of the video.
- The result would look like this:

  ((uIITBombayX, uCS101.1x, ufa1f6040f46a43298cc25fc33db89a83, uShrikrishna), [[4, 16.0], [4, 15.05], [2, 8.0], ...and so on ....])

Processing and generation of results for final summary table

- The processed data generated would be stored in the corresponding Hive table
  - Primary key : ⟨ an incrementing unique numeric id ⟩, orgName, courseName, videoSysId, userName, timeFrameId
  - Non-key attributes : frameAccessCount and frameDuration

Processing and generation of results for final summary table

- One of the possible visualizations that can be generated from this data is the total number of times accessed and the total time spent by users in a particular timeFrame of a particular video.
- What this would give is an overall analysis of the number of views and the time of the views on the video in ques- tion.
- An instructor would be able to use this analytical data to determine what are the regions of the videos that are being watched by users (who have watched this video), most frequently.

Processing and generation of results for final summary table

- The summary data is generated as follows.
  - First, the results of the previous analysis stage would be converted into an RDD through the parallelise function, and then converted into appropriate key-value pairs.
  - The result again parallelized to get an RDD, on which, we ran a final map-reduce job; reducer function we used added the tuples together to sum up the total number of accesses and the total time spent in a timeFrame.

Processing and generation of results for final summary table

- The summary data generated would be stored in the corresponding MySQL table (video_difficulty_details) :
  - Primary key : id ⟨ an incrementing unique numeric id ⟩, orgName, courseName, videoSysId, videoFrame
  - Non-key attributes : count and timeSpent
- The summary data was then drawn from these two tables using an R data visualization script using the googleVis library to plot a graph on the analytics dashboard.
- The graph had two y-axes: one for the timeSpent on the video timeFrames (indicated on the X-axis), and one for the count field, giving the number of accesses to the timeFrame in the video by all the users.

Future Work

- The clustered regions for different videos in a course could be compared with each other in order to find the regions of difficulty, and report them to the instructor.

Conclusion

We wish to extend the edX Insight module to be able to aid the students and instructors in ways both direct and indirect, so that their learning experience is enriched, and the students are able to spend a worthwhile time on the MOOCs they participate in. This project takes a small step in that direction by providing a method for the teachers to find out directly from the activities of the students whether they are facing any difficulty in the videos, without having to communicate with them directly - and ultimately, going towards improvement, and better courses.

THANK YOU