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

June 23, 2015

What is edX Insight?

- InSight is a BI (Business Intelligence) platform that was built by edX for the purpose of communicating the details of the courses, and analytical data concerning the student activity on the various courses to the administrators and course instructors.
- The primary aim of this platform is not only to monitor the student performances and validation of the choices made in designing the course, but also allows the designers to re-think the choices, and improve the course in its different aspects for creating a better experience for the students.

Why do we need InSight?

edX Insight in its current state provides access to various graphs, metrics and reports for analysing the student behaviour and activity in different aspects, such as :

- Course Enrollment data : daily student enrollment chart, enrollment metric, and enrollment over time metrics.
- Engagement Data : weekly engagement charts, content engagement breakdown report.
- Demographic Data : analytics on age bands and educational backgrounds.
- Location/Geographic Data : enrollment geography.
- Data on graded and ungraded contents : Find points of difficulty, and attempts to solve the problems.

Why do we need InSight?

"Necessity is the mother of invention"

So, we wish to extend the edX InSight platform by adding our own data modules, to get a visual feel of the data we want to see, and make our own decisions (aided or unaided).

Let's talk about one situation that edX does not address in its current state.

A situation

- Students watch the videos of a course on a regular basis.
- However - doubts may (and do) creep up regarding the lectures!

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.

So.... a "NECESSITY"

So, What do we do?

- 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.
- It wouldn't be of much credit if the instructor started giving lessons that were too difficult for the students to understand.

That's the problem we want to solve in this project.

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 expansion 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
- Processing the data
- Creation and prototyping of the data module
- Final implementation of the data module

Extracting 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.

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.

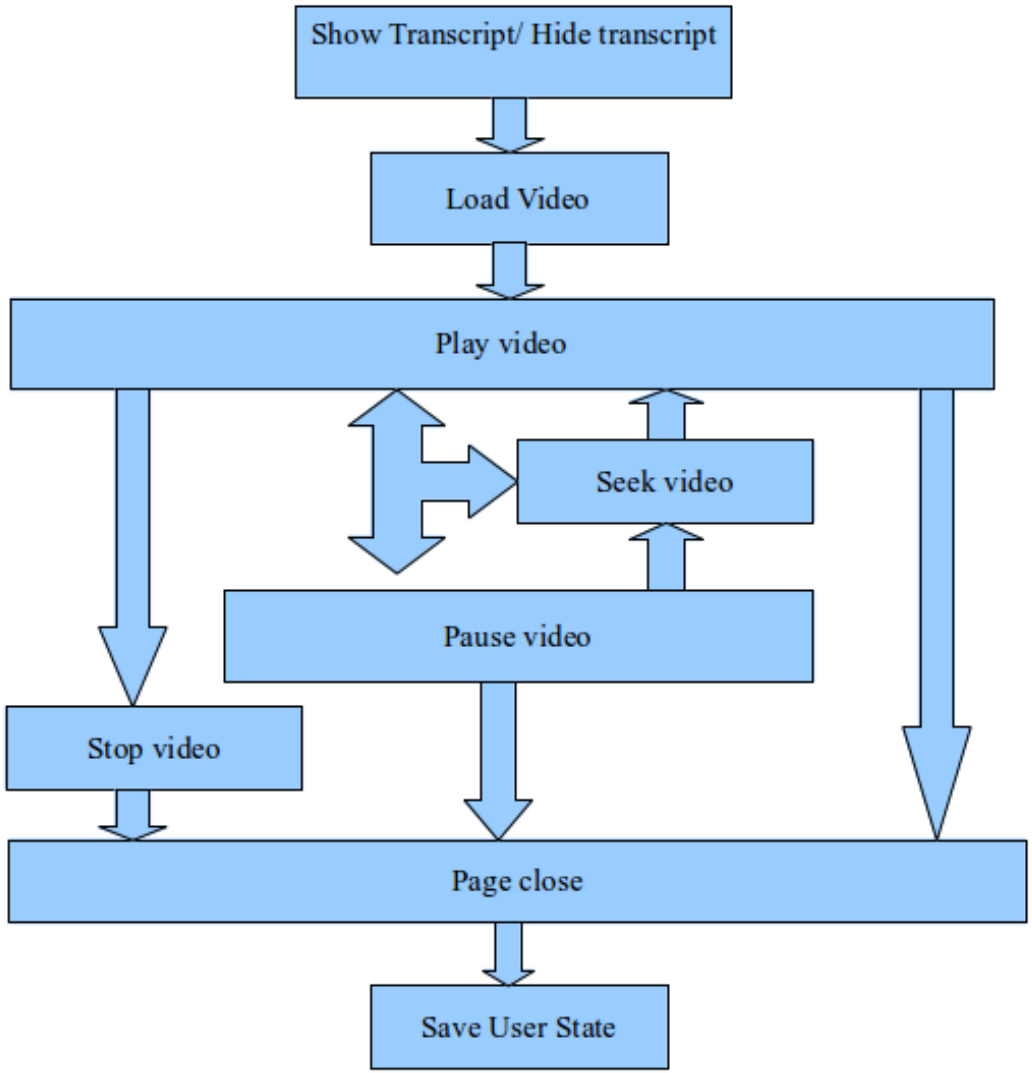
Redundancy removal and Error correction

- 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 video 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 module

- 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)

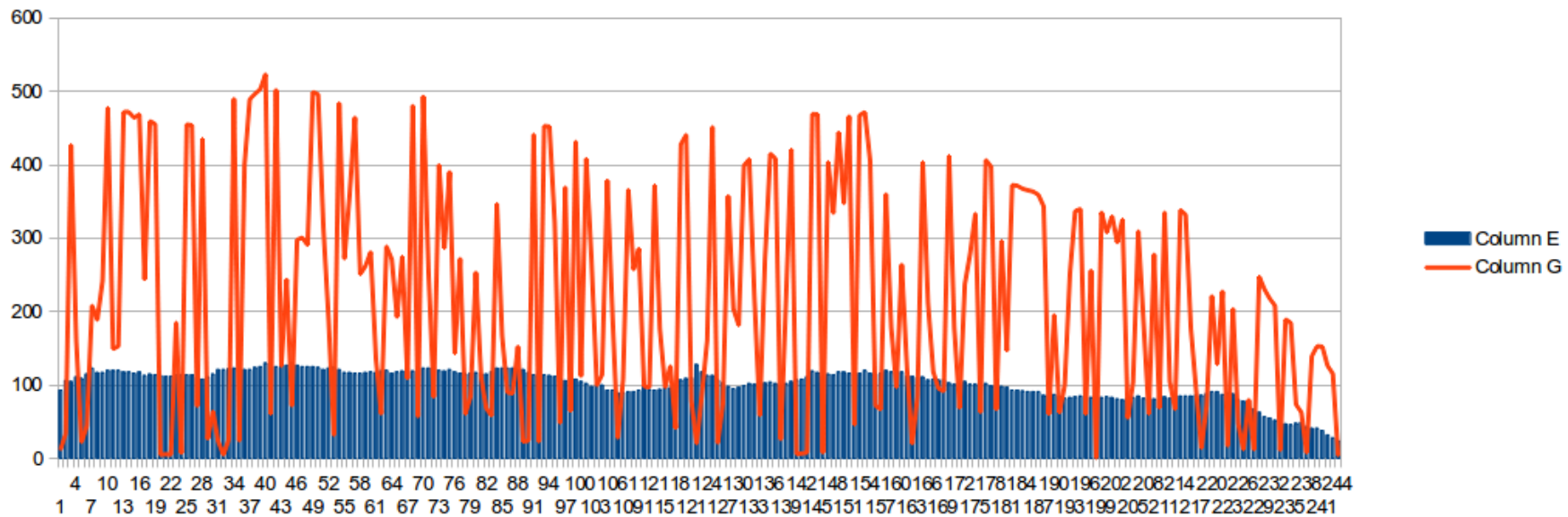
Creation and prototyping of the data module

- The output of this process would be entries of a MySQL table, having a primary key consisting of:
  videoSysId, userName, timeFrameId
- 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)

Further work in progress

The done so far has been outlined and described so far. The following are the tasks that are in progress:

- Spark implementation: The final module would be working in a big data environment - therefore, it needs to be efficient and fast. For this purpose, the final module would be implemented in Spark, and would draw input data from Hive tables through queries and write the result data to hive tables.
- Further, this data can then be summarized and stored in the corresponding summary tables, which could be accessed during the visualizations provided by the edX Insight module interface provided to the Course Instructor.

Technologies used

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

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 spent 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