Brief Article

The Author

April 27, 2015

Abstract

This document is a report on the study of the relationships between feedback and student performance in the context of Educational Data Mining. Data from an online educational social network was analysed in order to try to find correlations between students' behaviour and final grades. The students' behaviour was characterised with feature vectors to make differentiation and comparison of students possible. The main aim was to find a method that would allow the classification of students from the analysis of log data containing interactions with an educational social network. The feature vectors considered did not allow for the differentiation of users thus making impossible their classification. The methods used to analyse the data belong to the Data Mining field and are based in classification by k-means and dimensional reduction by PCA.

The introduction section discusses pedagogical terms such as assessment and feedback as well as a personal experience in two higher education institutions that had different approaches to assessment. Additionally, it explains how the birth of educational social networks sparked a whole new field called Educational Data Mining which aims to apply Data Mining algorithms to answer educational research questions.
The background section contains an overview of the methods and papers published in the field of Educational Data Mining as well as a description of some of the technologies used to analyse the data. An overview of Data Mining and Educational Data Mining is provided followed by examples of applications of these fields to the analysis of social networks and educational data.
The Data section contains a description of the data analysed in the project as well as the technologies used to perform the analysis. The different databases are described as well as the process to build them.

Two studies in the data are presented: On one hand, the Initial Study contains simple statistics about the data. The aim of this study was to explore the data and present simple statistics that would offer clues about how to analyse it. Statistics about social network usage are presented as well as final grades and activity after feedback. On the other hand, the Feedback Study aims to study the effects that feedback in the context of online education and weather there was a correlation between level of feedback given or received and final grade.

The last section contains a reflection on the project as a whole. It compares the project to the initial proposal and tries to explain the causes of divergence. Additionally, it describes the difficulties experienced during the project and it proposes future work in other to continue the research further.

1 Introduction

In the context of education, assessment is a fundamental aspect of any learning process. An assessment is defined as a judgement which can be justified according to a weighted set of goals the result of which is usually a comparative or numerical rating [18]. The process of assessment consists in the methodology used to produce the judgement. One can not produce a judgement without having a metric to which compare. In education, the metric used is usually a summary of the most relevant aspects about the curricula being assessed. This is usually translated as learning goals and in order to assess, the student knowledge is tested against a set of weighted learning goals. Assessment in itself is present in most aspects of daily life: A solicitor might assess the chances of wining a case, a doctor might assess the evolution of a patient, etc... However, its effects on education are dramatic: Assessment is not only responsible for rating students' knowledge but it can also play a fundamental role in aiding the learning process. There are two types of assessment in an educational context: Summative assessment is responsible for producing a grade. It is usually carried out at the end of a course and its aim is to encapsulate students' knowledge in order to measure it (often by examination). Formative assessment is used to aid learning and prepare for a summative assessment. It is generally carried out through a course and in order to aid learning it requires feedback. Note that according to these definitions the summative and formative assessment are the same process, the only difference being that formative assessment requires feedback.

Formative assessment is fundamental to education nowadays and till some point it is a measure of the quality of education. It has been adopted by education curricula in most of countries, it is preferred by education professionals and students alike. An example of this is the National Student Survey. It is an annual survey taken by higher education students at their final year, the questions relate to formative assessment which is just another prove that shows how feedback is considered fundamental by educational institutions and students alike.
The word feedback is present in many different contexts and therefore there are numerous definitions in the literature. Some authors consider feedback to be information about the difference between the actual level of a piece of work and a reference level which is used to alter this gap in some way [Ramaprasad]. Other authors argue that feedback requires knowledge of the goals and skills in making comparisons and suggestions to reduce the gap between what is produced and the goal [Sadler]. A definition of feedback from an AI perspective is provided by [13]: 'Feedback is conceptualised as information provided by an agent (e.g., teacher, peer, book, parent, self, experience) regarding aspects of one's performance or understanding. [...] Feedback thus is a consequence of performance'.

On this project feedback is considered to be the information about the gap between the work produced and the gold standard, it requires skills to make the comparisons and experience in transmitting suggestions effectively that will modify the quality of the work. Additionally the agents [13] talk are users of a social network assessing each other using summative and formative assessment.

My experience in higher education comprises two degrees in different countries with completely different approaches to education. I studied physics in a Spanish university where assessment was completely summative. The subjects where very similar in content to those of any other university and the material was delivered by a lecturer in weekly sessions. These courses were very structured, at the start of the course the topic to be studied was exposed and in the following sessions the lecturer proceeded with mathematical demonstrations. Each topic was rarely covered in one session, some of them needed months or even a full academic year to cover. The lecturers frequently followed famous text books that contained the different demonstrations. There was no interaction between teacher and students during the lectures, the students were free to ask questions but the atmosphere in the lecture was usually not encouraging. Additionally, the gap between the course content and the knowledge of students was enormous making it very difficult to even be able to ask for a question.
Apart from theory lectures there were practical sessions that consisted only in the resolution of exercises. The function of these sessions was to provide formative assessment (i.e feedback to students' about their knowledge in the topic) as well as a preparation for the final exam. Unfortunately, the exercises were too complicated to even ask questions and the lectures consisted simply in the lecturer resolving problems himself or herself with no interaction with the students who limited to copy form the whiteboard. When I started my physics degree I was fascinated by the subject but over time my interest on it diminished dramatically. Part of that was the total lack of formative assessment. Assessment was seen as a punishment in spite of just another tool for learning. The summative assessment created the impression that grades were more important than actual learning. There was a point when I realised that I was passing exams with very little knowledge about the subject and I decided to stop my studies.

Most courses I studied at Goldsmiths were assessed with formative assessment. However, some of them had a summative assessment at the end of the course (an exam) that produced the final grade. As in my experience in my previous degree, the type of assessment ha a dramatic impact in the engagement I felt towards a topic or a subject. Through formative assessment I felt passionate about learning and more importantly I learned. With summative assessment I felt the need to memorise (which rarely involves any learning) information in order to write it down efficiently on a paper. My personal experience demonstrates how critical feedback is. When feedback is present I learn, when feedback is not present I disengage and memorise. I argue that the type of learning experienced following summative assessment is poor as opposed to the rich and fulfilling process of learning through formative assessment.

With the popularisation of social networks, the internet has become a repository of information where data is intensively shared, reviewed and discussed by communities

of users. One might argue that the internet is commonly used for learning not only by students but also by professionals looking for hints or even complete solutions to problems. If one accepts that life is a constant learning process, then this learning has been accelerated and optimised by the Internet whose role is to facilitate the access to information and allowing for a constant interchange of feedback between people. This type of interaction between humans and machines was described by Tim Barners-Lee in [10] as a system where humans do the creative work (formative assessment) and machines do the administrative part. Such a system is called a *social machine* and for it to exist it need from humans as well as technology.

Social networks have provided a new paradigm on education where learning and especially feedback is delivered on the web. This is manifested in how users use popular social network commenting tools (forums and discussion threads) to learn about a particular topic. In some cases users look for straight answers to specific problems, however, on other cases users seem to be 'interested' in the topic. This later situation is very interesting and I argue that it has been maximised by the extensive use and adoption of social networks. It is consistent with the learning achieved through formative assessment where students are engaged with the subject and discussions about topics are spontaneous, it creates communities of learners where ideas are exchanged and reviewed in a process that can be described as giving and receiving feedback.
From the very popular social networks used for learning Youtube is probably the most popular. A good example of how this social network i used in an informal educational context is [5]. It is a Youtube Channel owned by the University of Nottingham Department of Chemistry, it has 2298 videos about chemistry with a total of 291658881 views, each video has hundreds of comments where a substantial number of them initiate discussions about chemistry. The list of Youtube Channels dedicated to education is immense (there's no way to know how much there are....?). As pointed out, most of these channels are targeted to non formal learning (it's the same as watching a tv show) and most of their popularity and success is probably due to this reason.

Over the last 10 years (sure?) higher education institutions have been transitioning to virtual learning environments (VLE) to support their education services. Universities use systems like [4] that provide a virtual environment where students and teachers can interact with each other. VLEs tipically contain course materials, assignment submission tools and forums where students can engage in discussions among other tools. However, frequently the social potential of the VLEs are not exploited or used very scarcely. An important detail is that VLEs are closed to the public meaning that only students of a particular course (or institution) have access to them.

The idea of an online course open to anyone on the Internet crystallised in MOOCs (Massive Open Online Courses) [7]. MOOCs are courses usually produced by major educational institutions that have a high number of students and are free of charge. The content of MOOCs usually consists in video lectures, reading lists and exercises but also in-lecture quizzes and interactive forums to support community of learners. These types of courses are usually distributed in online platforms offering a wide range of courses from different subjects. MOOC platforms are monopolised by three compa-

nies: Coursera, edX and Udacity. All three appeared in 2012, a fact that brought the New York Times to coin 2012 as 'The Year of the MOOC' [7]. One can think of these platforms as a new type of 'university', they offer different courses on a varied range of subjects from some of the most prestigious educational instittutions.
One of the issues MOOCs need to address is that the title achieved after completing successfully one usually lacks of recognition. On that sense there is still a long way to go to be able to compare MOOCs with higher education courses. *should I talk a bit more about this?*
MOOCs provide a privileged platform to encourage a more social and interactive education. As exposed above, online forums have been used for a while as contexts for communities of learners. However, there has recently been a shift towards a type of formative assessment called *peer feedback*. Peer feedback consists is an interchange of feedback between two students. By using an online peer feedback system students can support, express concerns or find solutions to problems. More importantly they can teach and learn together thus creating communities of learners through feedback and social interaction. Online peer feedback systems allow student to formative assess each other anytime anywhere.

We have seen how important is the role of feedback on learning and how formative assessment can be optimised through peer feedback. Additionally, I presented my personal experience on how summative and formative assessment are needed in order to allow learning. An interesting question that arises is weather one can establish the relation between these two types of assessment, in other words, weather or not it is possible to quantify the effects of feedback on a learning process. In order to measure that magnitude, one needs access to big amounts of data relating student behaviour, feedback and performance. The idea is to combine data mining techniques with pedagogical theory and empirical evidence in order to add some light into this issue. The surge in student behaviour data provided by the online systems described above allowed researchers to develop methods to study educational questions. This new emerging field encapsulate multidisciplinary techniques under the name of Educational Data Mining, in other words, the application of Data Mining methods to educational data with the objective of resolving educational research questions.
On this project I try to use data mining techniques in order to classify student performance from their behavioural patterns after receiving feedback. The data used was generated in a case study involving Music Circle, a social network for learning developed within the PRAISE research project at the computing department at Goldsmiths. The students undertook a creative programming MOOC that run on Coursera [1]. It lasted for six weeks and it was completed by 3715 users. Students had access to video lectures where theory and exercises were presented. The exercises were audio-visual programming exercises and once completed they had to be screen recorded, uploaded to Vimeo and then submitted to Music Circle for discussion with peer students. For each of the users we know their final grade and the grade they got in the quizzes during the lectures. Music Circle logs all the mouse events of the users that use the system, from this data one can infer when feedback takes place and observe the user response.

1.1 Report structure

This document is a report on my final major project in Creative Computing. The *Introduction* section sets the context of the project, it explains what it consists in and its interest from the research perspective. It is followed by the *Data Preparation* section which describes the data pipeline: From the data preprocessing to the database technology used. The next two sections are different analysis on the data. The *Initial Study* measures gross statistics about the data and tries to obtain simple correlations. *Feedback Study* is a more in depth analysis of the data which tries to find more intricate meaning in the data. Finally there is a conclusion section that summarises the project and exposes the results followed by a reflection section that contains a self-assessment.

2 Related work

This section contains an overview of the research about the intersections between data mining and education. It contains a general description about data mining and educational data mining followed by concrete examples on data mining applied to education.

2.1 Data Mining

Data mining (also known as Knowledge Discovery for Data) is a multidisciplinary subject which aims to discover patterns in very large datasets, it uses algorithms from computer science, artificial intelligence, machine learning and database theory.

From a historical point of view Data Mining can be viewed as an evolution of information technology. As with any other technological field, database technologies have been following a rapid expansion over the last fifty years: The limitations of early database systems that allowed primitive file processing prompted the development of database management systems in the early 70s. These systems introduced new paradigms such as hierarchical, network and relational databases, indexing and accessing methods, query languages such as SQL, user interface and finally online transaction processing. In the mid-80s advanced database systems were developed, this systems included advanced data models and they were applied to a wide range of data from engineering to multimedia.
It is in the late 80s where the design of databases considered the analysis of the data they contained. Data warehouse technology was developed as well as a new field which aimed to extract knowledge from the databases, this new subject was Data Mining. Data Mining continued its growth with the invention of the Internet and it surged with the increase of data driven technologies such as mobile devices and social networks.

Nowadays, data is generated at increasing rates aided by the popularisation of mobile devices (and sensors) as well as social networks. This situation allows systems to build very large datasets in a fast and inexpensive way. These large datasets are usually referred as 'Big Data' and they are the target of Data Mining. Depending on the nature of the data, the knowledge extracted can be used in a wide range of applications that range from business analysis to medical applications or scientific research.

With this knowledge one can build models of the data and then use those models to classify data instances or make predictions. Two of the main problems in Data Mining are classification and regression. A regression problem is focused in producing a value whereas a classification problem is focused on assigning classes to data instances.

In a nutshell the Data Mining process can be summarised as follows [12]:

- Data cleaning: Removal of any noise or inconsistent data
- Data integration: Combination of multiple data sourcest
- Data selection: Retrieval of relevant data from the database
- Data transformation: Consolidation of data into appropriate forms for mining
- Data mining: Application of algorithms to extract knowledge
- Pattern evaluation: Identification of interesting patters using metrics
- Knowledge presentation: Presentation of results

2.2 Eduational Data Mining

Educational Data Mining (EDM) is an emerging discipline that aims to apply data mining techniques to educational data in order to answer questions about education research [2]. EDM is concerned in the development of methods for making discoveries within the unique kinds of data that come from educational settings, and using those methods to better understand students and the settings which they learn in [9]. The increasing use of the Internet for education and the development of online education systems generates large datasets that contain information about teacher-student and student-student interaction. The aim of EDM is to use these datasets to better understand the learning process and to develop systems that optimise the learning experience.
There are three main research areas in EDM [17]:

- Offline education that transmits knowledge through in-person interaction and tries to understand how humans learn. It borrows from psychometrics and it applies statistical techniques to data gathered in classroom environments.
- E-learning and Learning Management Systems are new educational platforms that deliver their content through the Internet. They intend to be a virtual environment that emulates the traditional teaching setting.
- Intelligent Tutoring and Adaptative Educational Hypermedia System are systems that focus on the optimisation of the learning experience on the web therefore providing a system that adapts to every student needs. This systems tend to generate vast amounts of data such as log files or user models that can be analysed using DM algorithms.

Authors suggest many areas in the application for EDM [11],[9],[17]. These applications use traditional DM techniques such as classification, clustering and association rules but also other methods that are not exclusively considered DM such as visualisation and regression.

The analysis and visualisation of data highlights useful information and supports decision making. It can be used to analyse students activities to provide a general view of a course. The techniques used in this task are statistics and data visualisation. Statistics provide the analysis of the data whereas the visualisation aids the interpretation of it. Feedback for supporting instructors suggest actions for teachers or course administrators that would optimise the learning experience. Association rules are the most common DM technique used for this task. Association rule algorithms find relationships between data in large datasets presenting them as strong rules according to metrics measuring how interesting they are [15].
Another interesting EDM task is to build a systems that makes direct recommendations to students thus offering personalised learning. The most commonly used DM techniques in this case are association rules, clustering and sequential pattern mining. Predicting student performance consists in the estimation of an unknown variable that describes the student (knowledge, score, grade, etc...). This value can be continuous (regression task) or a categorical attribute (classification task). Regression analysis finds the relationship between a dependant set of variables whereas classification assigns classes to individual items based on quantitative information regarding items' features. Some of the DM techniques used for this task are: Neural networks, Bayesian networks, rule-based systems, regression and correlation analysis.

2.3 Analysis of user activity in social networks

The ubiquitous use of social networks in recent years has sparked research about how to analyse their users behaviour. Traditional methods in the characterisation of user behaviour are not suitable in the online context where users interact with a website, an application or other users through interfaces that implement the social network functionality.
As seen above social networks generate large amounts of data regarding user behaviour. However, usually this data is in its brute form and it does not by itself show features useful for an analysis process. Generally the data comes from log files that capture user events or explicit activity that happens on the social network meaning that in order to extract useful knowledge from the data one needs to use DM methods.
Unfortunately, DM does not perform well when applied to raw data. The data needs to be reduced in a way that expresses desired features, the features relating to the same item are put together in a vector which is fed to DM algorithms. The process of obtaining features from the raw data is not trivial and it makes up for most of the pre-processing stage in a DM analysis.

The authors in [14] propose a methodology for characterising and identifying user behaviours in online social networks. First, they crawled data from YouTube and used a clustering algorithm to group users that share similar behavioural pattern. Next, they showed that attributes that stem from the user social interactions, in contrast to attributes relative to each individual user, are good discriminators and allow the identification of relevant user behaviours. Finally, they present and discuss experimental results of the use of proposed methodology.
Of particular interest is a section describing their methods to build feature vectors that

allow to discriminate between users. Each user was assigned a nine dimensional feature vector containing features unique to the user but also features about the interaction with other users. The features uniquely relating to a user are: Number of uploads, number of views, number of channel views, system join date and age. The features relating to a community of users are: Clustering coefficient, reciprocity, out-degree and in-degree. The clustering coefficient measure the interconnection between users and their neighbours. The reciprocity measures the probability of mutual subscriptions. The out-degree is the number of subscriptions made by the user and finally the in-degree is the number of subscriptions received by the user.
Once the feature vector is constructed it is normalised to allow for the features to have comparable weights. The authors then apply a k-means algorithm to the set of feature vectors, similar users cluster together forming groups thus manifesting different types of users.

2.4 Analysis of student behaviour in educational systems

We have seen an example on how feature vectors can be used to characterise user behaviour in social networks. One of the most interesting aspects of EDM is that there is not yet formal methods to analyse student behaviour, this meaning that the methods to obtain feature vectors out of educational data are not well defined.
The authors in [8] presented models which can detect student engaged concentration, confusion, frustration, and boredom solely from students' interactions within a Cognitive Tutor for Algebra. These detectors were designed to operate solely on the information available through students' semantic actions within the interface, making these detectors applicable both for driving interventions and for labelling existing log files in the PSLC DataShop, facilitating future discovery with models analyses at scale.

The data collected consisted in log files from the tutoring system and field observation of students behaviour. The log files contained the interaction of students with the interface whereas the filed observation consisted in judgement of the students' state or behaviour on the work context, actions, utterances, facial expressions, body language, and interactions with teachers or fellow students. At the time of analysis the log file data and the filed observations were synchronised and the features were extracted. A total of 232 features were extracted from the data for each student, these features fell into four categories showed in Table 1. Each vector contained 232 features. In order

Category	**Feature**
Engaged concentration	Minimum number of previous incorrect actions
Boredom	Average time the student took to respond
Confusion	Percentage of actions taking longer than 5 seconds after incorrect answers
Frustration	Percentage of past actions on the skills involved that were incorrect

Table 1: Categories with examples of features

to optimise the results, these features were selected (also known as feature reduction) using forward selection, where the feature that most improves model goodness is added

repeatedly until adding additional features no longer improves model goodness. Once the dimensional reduction was applied to the feature vectors, these were fed into DM algorithms which included J48 decision trees, step regression, JRip, Nave Bayes, and REP-Trees.
The results obtained were better than chance but left room for improvement. Different algorithms performed better depending on the category. For engaged concentration, the best algorithm was K*. For confusion, the best algorithm was JRip. For frustration, the best algorithm was REPTree. Finally, for boredom, the best algorithm was Nave Bayes.

2.5 Prediction of student performance in online discussion systems

One of the oldest applications of EDM is to predict students performance based on other information that relates to the learning experience of the student. This is a difficult problem to solve since the performance of a student is influenced by a large number of variables that are difficult to control such as socio-economical background or psychological profile. Authors in [16] proposed the use of different data mining approaches for improving prediction of students' final performance from quantitative and qualitative participation indicators in social network forums. Their objective was to determine how the selection of instances and attributes, the use of different classification algorithms and the date when data was gathered affected the accuracy and comprehensibility of the prediction. Depending on the variable to be predicted one must use one technique or another: On one hand, classification algorithms are useful when the students performance can be described with a categorical value. On the other hand, regression provides a method to assign a numerical value to the predicted performance. Additionally, one can use a density estimator in order to obtain a probability function of the output variable. An important detail might be to notice that most of the research in EDM focus on higher education or university students and more specifically to e-learning. It is this later online systems that provide with huge amounts of data regarding student behaviour which is valuable to solve this type of problem. Various publications investigate the link between the use of online discussion systems and student performance finding that it is a strong indicator of student performance (citations here?). In general, these publications analyse both quantitative or qualitative features independently but not in conjunction.
The paper describes the prediction of students' performance based on quantitative, qualitative and social network information about forum usage by using classification algorithms and clustering.

The data was gathered from Moodle [4] forums by developing a special module that allowed an instructor to obtain a summary of the activity. The instructor had to grade each of the forum messages according to metrics that measured the degree of online participation. The quantitative indicators were: Number of messages, threads, words, sentences, reads and time. Qualitative indicators consisted in an average score of all the messages sent by each student, the degree of centrality, the degree of prestige and the final mark. The feature vectors contained both the quantitative and qualitative

indicators.

The next step was to pre-process the data by performing instance selection where only a subset of instances was selected, attribute selection (feature reduction) where the dimensionality of the problem was reduced in order to ease the calculations and finally perform data transformation that converted data to an appropriate input format for the DM algorithms.
In this study the authors were interested in categorising the students into two classes: Fail and pass. Therefore, this was a classification problem as opposed to a regression problem. Two data mining methods were proposed: On one hand, they proposed to use a traditional classification method since it is the standard DM approach to classification problems. On the other hand, they propose clustering as an alternative method. The authors then compare all the different algorithms against performance metrics and conclude that EM (Expectation Maximization) clustering algorithm provides the best result.

3 Data preparation

A crucial task on Data Mining is the preparation of the data for analysis, a process also known as data pre-processing. The databases used for DM are large, often in the Gigabytes order of magnitude. Such systems are complex and the data sources are often heterogeneous which results in data being noisey, missing or inconsistent. Data with those features has low quality, meaning that DM algorithms will produce poor results when using it. Therefore, the aim of data pre-processing is to improve the quality of the data so that DM algorithms produce useful results.
Different techniques have been proposed to improve the quality of the data: Data cleaning minimises noise and corrects inconsistencies. Data integration combines data from different sources creating a coherent database. Data transformation manipulates the data by finding representations that maximise its potential. Finally, data reduction selects the most important features of the data thus reducing its size, increasing processing speed and improving the results.

Data cleaning techniques focus on filling gaps in the data, smoothing values and correcting inconsistencies. Several options are available when dealing with missing values:

- Discart the tuple: If one component of the tuple is missing, then discard the sample
- Filling the missing value manually: This approach is time consuming and not feasible for large datasets
- Use a constant: When a missing value is encountered, then substitute it for a constant value
- Use the attribute mean: Calculate the mean of the attribute considering all samples and substitute by that value
- Use the most probable value: Use machine learning algorithms to predict the value

Noise is a random error or variance in the data. These are several techniques that can help to minimise noise:

- Binning: These methods smooth the data by comparing it with similar samples which are grouped into bins. The samples' values are substituted by the bin mean, bin median or bin boundaries
- Regression: The data is fitted to a function and sample values are substituted by the output of the fitted function
- Clustering: Clustering is an effective way of detecting outliers. Any sample that does not fall into a cluster is eliminated.

Data integration consists in merging data from various sources in order to produce a coherent database that can be used by DM algorithms. As an example consider a system that uses a database to perform its operations. The database of such a system will be optimised for the retrieval and modification of data according to that particular system. That database might not be ideal to perform data mining operations, therefore, a new database can be created which will contain the system's data stored in a way that makes it efficient for DM algorithms.

Data transformation aims to transform and consolidate the data into appropriate representations for mining. Some of the common methods for data transformation are:

- Smoothing: Remove noise from data with the techniques described above
- Aggregation: Similar to data integration
- Generalisation: Low-level values are replaced by high-level attributes
- Normalisation: Scale the data between [-1,1] or [0,1]
- Attribute construction: Generation of new features from the given set of attributes

Data reduction aims to find a representation of the data that is smaller in size but retains the basic properties of the original data set. That means that DM on the reduced data set produces similar results yet increasing the processing speed.

One of the most common data reduction techniques is Principal Components Analysis (PCA). Given a set of k-dimensional vectors, the aim of PCA is to find n orthonormal vectors where $n < k$, the dimension of the vector space is reduced thus allowing to represent the data set into a dimensional smaller space. PCA is an orthogonal transformation that transforms a possibly correlated set of vectors into a linearly uncorrelated vectors called principle components. These principal components are sorted in a decreasing variance value. The principal components then represent a new set of axes that contain information about the variance of the data. PCA combines attributes from the different vector to create a smaller representation of the data that usually reveal dependancies that were not trivial.

4 Data and databases

4.1 Music Circle

Music Circle is an educational social network developed within the PRAISE research project at the Goldsmiths University Computing Department. The aim of Music Circle is to allow communities of students to learn online together by providing an online environment with specific tools to give and receive feedback. Music Circle was designed considering the fundamental role of feedback in learning. In the introduction and background sections we say the dependancies between formative assessment, feedback, student performance and learning. Additionally, we saw how valuable is the data generated by students using online systems. Music Circle was designed thinking in the collection of student data for posterior EDM analysis. The content of Music Circle is time based media that users can upload. This media can be content generated by the user such as an audio or video recording, or content shared from other social networks such as Youtube or Vimeo. Once the content is uploaded, users can share it with different communities. Once a media item is shared in a community, it becomes visible to all the members of the community and they can start discussions about it.

Music Circle is a web service, meaning that it is formed by two systems: The server-side software interfaces with the database in order to insert, update or retrieve items. The client side is a Javascript web application that implements all the functionality of Music Circle. The server-side and the client-side are connected with a RESTful API. Figure 1 shows the Music Circle web application.

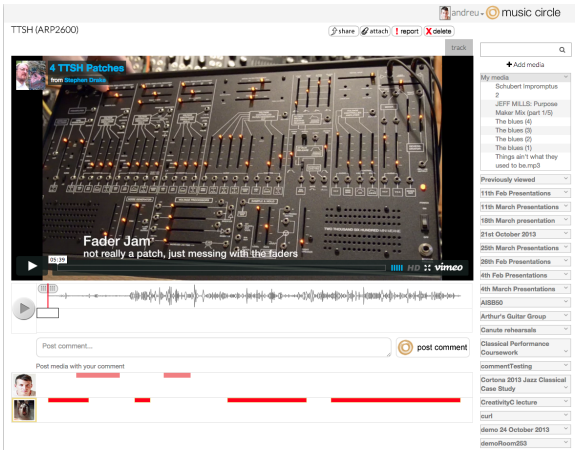

Figure 1: Music Circle web application

The tool that allows for the discussion of media is the social timeline and it sits at the core of the social network functionality. Using the social timeline, users can select a time region of the media and attach text comments or other media comments to

that particular region. Commented regions become can become threaded discussions by adding replies to comments. The selected regions appear as coloured blocks next to the avatar of the user, figure x shows Music Circle's main screen. The coloured blocks represent the time regions selected that have been annotated. Comments that relate to the same region stack on the top of each other, additionally, comment from different users are stacked in layers, thus differentiated by vertical position and colour.

The data generated by Music Circle consists in the logs generated by the user events when uploading, viewing, commenting and replying. The data has quantitative and qualitative features. On this project only quantitative measures are used, however, it would be interesting to use a qualitative study on the data on future studies. The log data consists in the events generated by the students using the interface and it is described in detail below. (here I need to add a pic of Music Circle)

4.2 MongoDB

Traditional databases store the data according to a relational model where the data is separated into tables that represent the different attributes. Data instances are stored in rows that have a unique key and relations between tables are possible by storing the unique key of the row that relates to the row it should be linked. The relational model of data is very popular and many database technologies make use of it. Moreover, these databases use SQL (Structured Query Language) to interact with the database. The structure of the database is defined when creating the database meaning making them not suitable when data changes its structure frequently. This feature makes them more suitable for an environment where the data changes often such as in a web context.

MusicCircle uses a MongoDB database. MongoDB is a document oriented database that provides high performance, high availability, and easy scalability. [3]. Document oriented databases rely on the internal structure of the data to infer type information and store all the related information together allowing instances to be different from each other.
A MongoDB system can contain several databases. Each of this databases contains a set of collections. The collections hold a set of documents. Documents are at the core of the database and they are the object that stores the data. They are set of key-value pairs and have a dynamic schema that allow document in the same collection to have different structure or different types of data. In a document data is stored in BSON (Binary JSON) serialisation format which consists in a binary representation of JSON objects. Therefore, the language used to define documents is JSON and MongoDB language translates it to BSON. The maximum size of a document is 16 megabytes.
The queries in MongoDB imply the use of the find() method with a number of query operators in order to select documents from the collections. The query operators consist in exact equality matches and conditionals.
MongoDB true power shows in cluster servers where high performance is needed. Replica sets provide high performance replication with automated failover, while sharded

clusters make it possible to partition large data sets over many machines transparently to the users. MongoDB users combine replica sets and sharded clusters to provide high levels redundancy for large data sets transparently for applications.

4.3 Databases

Two different databases where used for this project. The Music Circle database is the one that sits on the server and relates to the web application. It is responsible for storing and serving the data in order to allow the Music Circle clients to work. It is 19.24 Gb in size and contains the following collections:

- Activity: Replies
- ActivityDefinition: Comments
- ActivityDefinitionTagLinker: Comments tags
- ActivityDefinitionTagLinker: Replies tags
- aggLeastCommened: List of least commented items
- aggMostCommented: List of mosts commented items
- AudioContent: Media Items
- AudioContentTrackSetLinker: Track sets
- DeletedComments: Deleted comments
- DeletedReplies: Deleted replies
- GenericContent: Files attached to media items
- log: Log data generated by users events
- MediaViewLog: Log data generated by users events when watching media
- Opinion: Rating assigned to a media item, a comment or a reply
- PraiseUser: User
- PraiseUserActivityDefinitionStatus: Indicates if a user has not read all the comments
- PraiseUserAudioContentStatus: Indicates if a user has not played all the media items
- PropertySet:
- Session: User session data
- Tag: A tag
- TrackSet: Media items group
- UserGroup: Group of users

4.4 Research database

The Music Circle database was built in order to serve data to the different clients. However, the schema of the database is not optimal for performing Data Mining on its data, therefore, a research database was built considering the analysis of the data. The research database contains documents that relate users with their actions and their items, on this sense, the user field allows every element in the database to be unique. The user field contains an id in order to make it possible to link with the Music Circle database. The activities field is equivalent to the log data in the Music Circle database, it contains the type of activity, the id the element that the activity took place on and a timestamp when it happened. Additionally, the media field contains all the media items for all the users. The id filed contains the media item id. The fmt field is the media item format. The owner filed is the user id. The title is the title of the media item. The ts filed is the timestamp when the file was uploaded. The schema of the research database is presented below, it has two collections: The Actors collection contains users and their actions whereas the Media collection contains all the media items on the system.

```
Actors: [
    {
        _id: element id,
        activities: [
            {
                at: activity type
                id: id of the element the action took place on,
                ts: timestamp
            }
        ],
        idx: user id on the Music Circle database,
        name: name oft heuser,

    }
],
```

Where the possible values for 'at' are: 0=upload, 1=view, 2=comment, 3=reply, 4=login, 5=play, 6=log data (mouse events).

```
        Media: [
            {
                _id: element id,
                fmt: media format,
                owner: userid,
                title: title of the media item,
                ts: timestamp,
            }
        ]
```

Where the possible values for 'fmt' are: 0=video, 1=audio.

This schema was initially proposed by Dr. Chris Kiefer who previously did research on the same dataset and it was adapted on this project to accommodate all the data needed for the particular studies.

To build the research database was difficult. The process of importing the data from the Music Circle database into the Research database consisted in iterating through various of the collections in the Music Circle database, putting the data in the right format and inserting it into the right element in the research database.
The PraiseUsers collection is iterated and for every user a new document is inserted into the Actions collection which contain the user id, the username and an empty activities array.
The AudioContent collection is iterated and for every media item a new document is inserted into the Media collection which contain the media id, the media format, the user id that owns the media item, the title of the media item and a timestamp.
The ActivityDefinition collection is iterated and a new object is built which contains the type of activity (comment activity in this case), the id of the comment, the timestamp when it happened, the media time id it belongs to and the text of the comment. The user id of the owner of the comment is known and used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the *activities* array on the correct document on the Actors collection.
The Activities collection is iterated and for each reply a new object is built which contains the type of activity (reply in this case), the id of the activity, the timestamp when it happened and the comment id it belongs to. The user id of the owner of the reply is known and used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the *activities* array on the correct document on the Actors collection.
The MediaViewLog collection is iterated and for every log element a new object is built which contains the type of activity (media view in this case), the timestamp when it happened and the id of the media item that the user viewed. The user id of the owner of the reply is known and used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the *activities* array on the correct document on the Actors collection.
The Sessions collection is iterated and for each session a new object is built which contains the type of activity (logging in in this case) and the timestamp when it happened. The user id of the owner of the reply is known and used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the *activities* array on the correct document on the Actors collection.
Finally, the log collection is iterated and its data is imported. The log collection is the largest one in the Music Circle database and contains mouse user events from the usage of the Music Circle web client. The size of this collection means that one might not be able to use the approach to import the data described above since it takes a lot of time. When the simple method (the one from above) was tried, the script

run on the machine for 2 days having completed only a quarter of the collection size. The nature of the log data also meant that it was difficult to predict how much time it would take for the import to complete. The size of individual elements in the log collection varies greatly, meaning that the speed at which the data was imported into the research database also varied.
One of the approaches to try to lower the importing time was to process the data in batches. This way, the amount of data to be processed was low and in theory the computational speed faster. However, the process that slowed down the importing was to update the *activities* array and specially pushing elements at the end of it. Unfortunately, batch processing did not increase the computation speed enough. After this, bulk operations were tested. Bulk operations are standard in Mongo and allow to do a large number of operation in an optimised way, unfortunately it did not work either. The detail that provided a hint to solve this issue was that when inspecting the Mongo process where the Mongo client connects to. When the Mongo process was inspected it showed a CPU usage close to 100%. Considering that the machine used contained 4 CPUs, it was clear that by using parallel computing the importing speed could be increased dramatically (at least by a factor of 4). In this sense the research database had to become a distributed system that allowed multiple Mongo instances running in subsets of the data, this approach would allow the maximisation of CPU usage.

Two common techniques to scale distributed systems are vertical scaling and sharding: Vertical scaling consists in adding more CPU and storage resources to increase capacity. Scaling by adding capacity has limitations: high performance systems with large numbers of CPUs and large amount of RAM are disproportionately more expensive than smaller systems. Additionally, cloud-based providers may only allow users to provision smaller instances. As a result there is a practical maximum capability for vertical scaling.
Mongo response to distributed systems is sharding also known as horizontal scaling. It consists in dividing the data and distribute the data over multiple servers or **shards**. Each shard is an independent database, and collectively, the shards make up a single logical database. Sharding addresses the challenge of scaling to support high throughput and large data sets: Sharding reduces the number of operations each shard handles. Each shard processes fewer operations as the cluster grows. As a result, a cluster can increase capacity and throughput horizontally. For example, to insert data, the application only needs to access the shard responsible for that record. Sharding reduces the amount of data that each server needs to store. Each shard stores less data as the cluster grows. For example, if a database has a 1 terabyte data set, and there are 4 shards, then each shard might hold only 256GB of data. If there are 40 shards, then each shard might hold only 25GB of data.
Mongo supports the sharding of databases through the configuration of a sharded clusters. A sharded cluster contain these elements: Query routers, config servers and shards.
In a sharded cluster, Mongo clients connect to the query routers and these redirect the queries to the appropriate shards. The query router targets operations to the shards and then returns the results to the clients. In real world deployments of sharded clusters there are more than on query router in order to balance the request load even

though a particular client sends queries to only one query router at a time.
Config servers contain the cluster metadata. They contain a mapping between the clusters data and the shards and the query routers use this metadata to target the shards. A Mongo sharded cluster contains exactly three config servers.
Shards are he structures that store the data. In a Mongo sharding the data gets distributed at a collection level, that is to say that the shards will contain data from a particular collection. The distribution of the data using a shard key. A shard key is either an indexed field or an indexed compound field that exists in every document in the collection. MongoDB divides the shard key values into chunks and distributes the chunks evenly across the shards. To divide the shard key values into chunks, MongoDB uses either range based partitioning or hash based partitioning. Mongo manages the distribution of chunks in the different shards. Chunks are splitted when they grow beyond a set chunk size and migrates them when a shard contains too many chunks compared with the other shards.

The sharded cluster used had 4 shards so that potentially the four CPUs of the machine could be used simultaneously. Importing the log data using a sharded cluster took 12 hours. On the client side the log collection was processed in batches containing log data between two particular dates. There were four clients connected to the query router each one importing data from a quarter of the time. For each fraction of time, the log collection was iterated. Each element in the log collection is an object containing a user id and a set of logs corresponding to that user. The set of logs is iterated and for each log a new object is built that contains the activity type (log type in this case), the timestamp when the event happened, the type of event and the type of element where the event took place. The user id of the owner of the log event is known and used to find the element for that particular user in the Actors collection. The object previously built is then pushed into the *activities* array on the correct document on the Actors collection.
With this last step the research database is built and ready to use. Scripts A.1, A.2, A.3, A.4 implement the importing methods that build the research database.

Figure 2 shows the data pipeline.

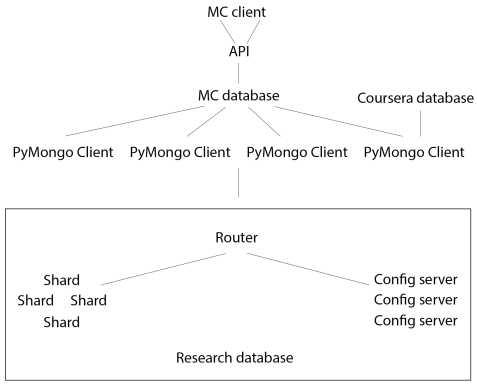

Figure 2: Data pipeline

5 Initial study

5.1 Aim

The aim of this study was to explore the data and generate simple statistics linking student activities after feedback with student performance. It was an opportunity to test the research database and make the necessary adjustments in order to make it robust. Additionally, it was an opportunity to familiarise myself with Python and its scientific tools.

5.2 Methods

The first step was to explore the Music Circle database by showing the activities per day. Script [A.5] iterates through various collections in the Music Circle database and counts how many activities happened per day. Figures 3, 4, 5, 6, 7 show the plots generated. The spikes in the plots correspond to a surge in activity due to the approach of assessment dates.

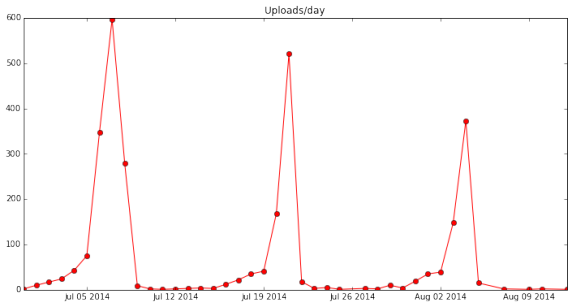

Figure 3: Uploads per day

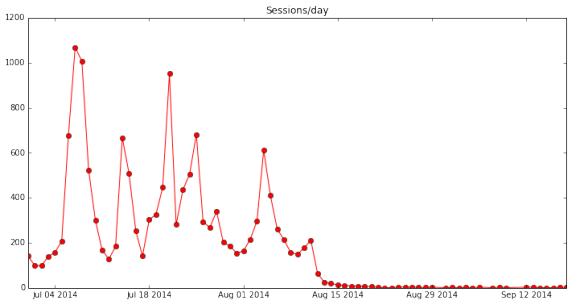

Figure 4: Sessions per day

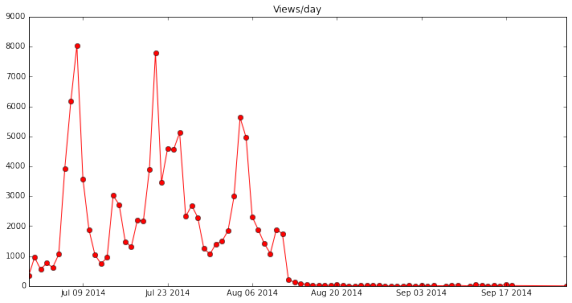

Figure 5: Views per day

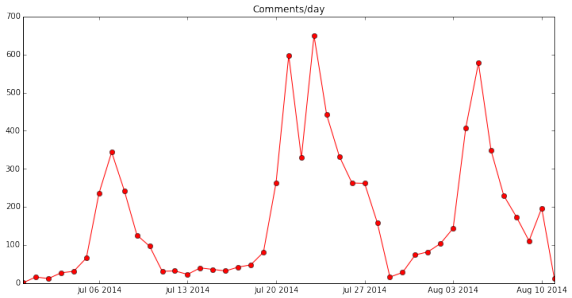

Figure 6: Comments per day

After this initial statistics, the data in the Research database was explored. A necessary step was to clean the data as much as possible. After trying to implement initial statistics I realised that some user events were logged more than once. Those duplicates had to be removed since they would interfere with the results. Script [A.6] removes the duplicates in the activities array. For each user it iterates through the activities array, it finds the duplicates and it removes them from the database. As said

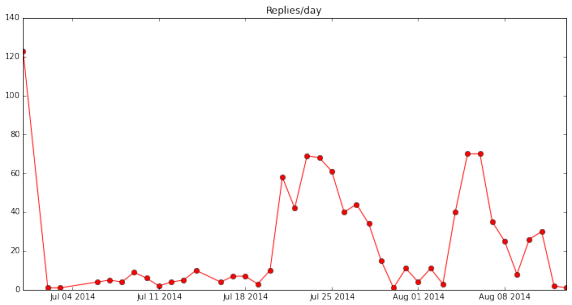

Figure 7: Replies per day

before, this study was an opportunity to explore the data. Particularly interesting were the extreme cases of very active or very inactive users. Therefore, a script [A.7] that added some information into the activities array about actors' level of activity was implemented, specially since Mongo query modifiers can not operate array length queries.

The aim of this study was to provide with an overview of the data and simple statistics of activity after feedback. An interesting question was how to detect that feedback had happened. In this sense one way to detect that feedback has happened is when a user clicks on an element in the Music Circle web app. User clicks always trigger information to be displayed on the screen: An assumption is made that feedback is produced after a mouse click. A more difficult situation is to determine for how long a particular feedback action (a click) has effect. On this study it is assumed that the effects of feedback last through the whole session period. A session is the period of time between the login of the user and the logout. A script [A.8] imports the sessions for each user. It finds all the actors in the research database that have activities or uploads, it then iterates through the found actors and finds all the sessions in the Music Circle database for each user. Once the sessions are retrieved, the right actor in the research database is found and a new field called *sessions* is inserted.

Part of this study aim was to make the research database robust, that is to say to explore the data and find missing features in the research database. The data preprocessing described above was essential and this is why it was decided to alter the database. In other mild cases or cases where the data was inconsistent the approach was to reformat the data in the scrip without changing the database. As seen before, the research database took time and effort to build, therefore, the less operations were done to it, the better. In three occasions the research database was broken, which meant to re-build it again by setting the sharded cluster environment and importing the log data, a process that is difficult and consumes time.

With script [A.8] the research database was ready for the next stage, to build feature vectors. Finding an appropriate set of feature vectors is the most important task on this project as a whole. The feature vectors build on for this study consisted in cumulative measures of five types of click events: Clicking play on a media item, clicking

on an owned media item in order to display it, clicking on a media item from a community in order to display it, click on a comment to display it or click to upload a new media item. Script [A.9] builds the feature vectors. The first step to build the feature vectors is to make a list of objects that contain the starting an ending date consecutive sessions for each user. Then the list of activities is iterated making chunks of activities that happened in a particular session. These activities are stored in a matrix, each row contains a list of activities that happened during a particular session. The feature vectors are built by iterating the matrix and counting how many uploads, plays, clicks on owned media items, clicks on community media items and click on comments are per session (per row). These values are put in a 5 dimensional vector. Therefore, for each session there is one feature vector. This process is repeated for all users and all the feature vectors are written in a Json file. The data from the activities have some inconsistencies with missing fields therefore before doing any operations checks on the consistency of the data were performed. The reason to store this data in a file and not into the database was to avoid doing operations on the database that could damage it. While cleaning the data the database was broken several times and as expressed above, to rebuild the database was a time expensive process.

Once the feature vectors were built it was time to link it with the data ground truth. In the PraiseUser table in the Music Circle database there is a field called *courseraId* which in theory allows to link with the Coursera database that contains the final grades student achieved. Unfortunately, this is not the case and none of the *courseraIds* were present in the Coursera database. This was a major issue since without the being able to link both databases this project could not be completed. The script that solves this issue is [A.10] and this is the approach considered: Each submission on Coursera contains a session id and a link to a particular item in Music Circle. The session id matches a row in the Coursera database. The link to Music Circle contains a media id at the end of it. Media items in the Music Circle contain the user id that owns them. This way there is a link between the Coursera database and the Music Circle database. The data was inserted in a new collection in the research database called ActorsGrade where each element contains the Music Circle user id, the grade from achieved from the Coursera database and the total number of activities performed by the user.

The final step was to generate some plots using the data calculated. Of particular interest were plots that compared general activity to the grade achieved. Below there are the plots generated by [A.11] script:

5.3 Conclusion

These study had different aims: To complete the research database, to explore the data and to present some simple statistics that would hopefully offer clues on how to analyse the data. From the plots it is obvious that there is not a predominant correlation between activities and grades for any of the measures. The analytical results when trying to fit a linear function support this by showing a coefficient of correlation close to 0 r^2=0.50665379655859477 for the uploads, r^2=0.15757119550297927 for comments, r^2=0.12108309234311032 for owned tracks, r^2=0.10335127031735547 for community

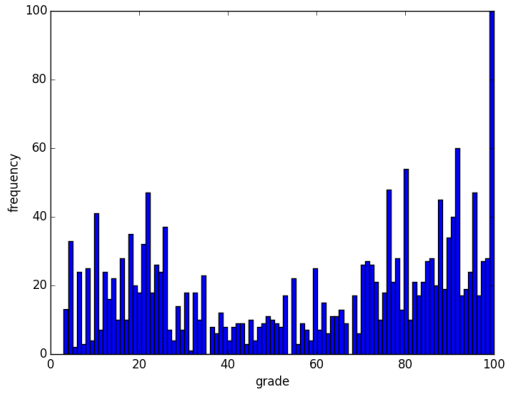

Figure 8: Distribution of grades

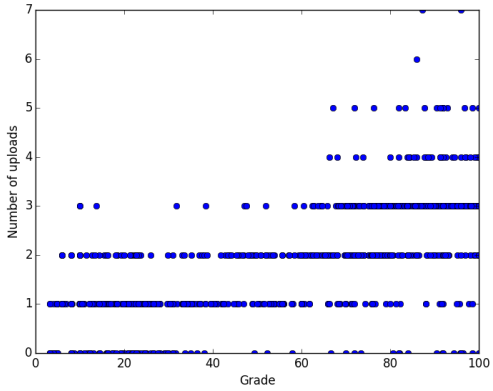

Figure 9: Uploads vs grades

and r^2=0.11542833494381988 for plays. Trying to fit non linear functions might improve the results. The data does not show any signs of a nice probability distribution, making it difficult to decide what the next steps are.

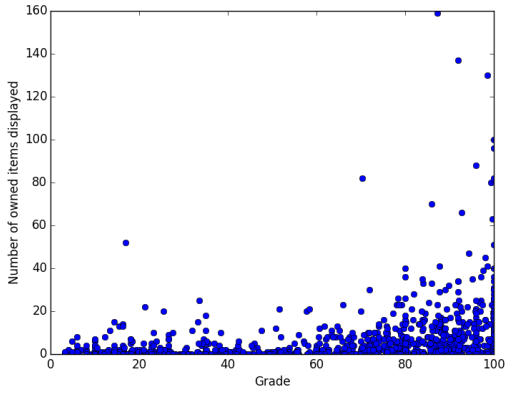

Figure 10: Owned track views vs grades

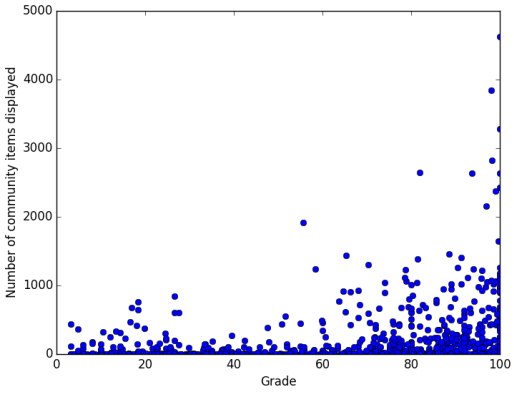

Figure 11: Community track views vs grades

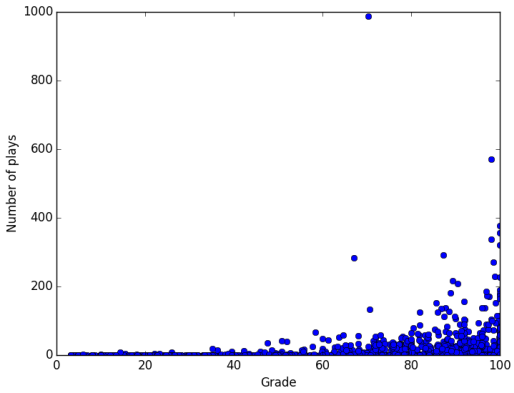

Figure 12: Number of plays vs grades

6 Feedback study

6.1 Aim

The aim of this study was to analyse the activity of students after feedback and to classify students academic performance using Data Mining algorithms.

6.2 Methods

The Initial Study provided with a set of feature vectors that described the activity after feedback of students using Music Circle. The Feedback Study tries to analyse those feature vectors and apply Data Mining algorithms in order to classify students' performance into two binary classes, 'good performance' and 'bad performance'. A crucial step for this study is the validation of the feature vectors calculated in the Initial Study. This meant to demonstrate that the feature vectors represented the students' behaviour and could be used to differentiate between types of student. There is a set of feature vectors for each user and the hypothesis is that each of those sets can characterise a particular user. One can think of each of those sets as different clusters. If the feature vectors described users appropriately then by taking all the feature vectors corresponding to all users and applying a k-clustering algorithm where k is equal to the number of users should produce a similar clustering as the calculated in the Initial Study. The cluster generated in the Initial Study was considered as the ground truth clustering and was used to measure the success of the clustering.

The clustering algorithm used on this study is k-means, it forms part of a wider group of methods called partition methods which are defined as follows: Given D, a data set of n objects, and k, the number of clusters to form, a partitioning algorithm organises the objects into k partitions ($k < n$), where each partition represents a cluster. The clusters are formed to optimise an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are similar, whereas the objects of different clusters are dissimilar in terms of the data set attributes [12].

The k-means algorithm takes as the k parameter and partitions the data into k clusters that have high intracluster similarity and low intercluster similarity. The cluster similarity is measured considering the mean of the elements in a cluster. Below there is a description on how the algorithm works:

- Select k random elements in the data that will be the initial cluster centres
- Each of the elements in the data is assigned to the centre cluster it is most similar according to a distance metrics (usually euclidian distance)
- The previous two steps iterate until a certain stoping criteria is fulfilled

The stopping criteria is usually determined by the square-error defined as follows

$$E = \sum_{i=1}^{k} \sum_{p \in C} |p - m|^2$$

k-means was chosen because of its simplicity and intuitive nature.

Script A.12 is responsible for the clustering of the feature vectors. The clustering algorithm is provided by *scikit-learn* which is a machine learning library for Python that provides simple and efficient tools for data mining and data analysis. It is built on Numpy, SciPy and matplotlib, making it fully compatible with Python scientific programming tools. A k-means clustering with k equal to the number of users is performed on all the feature vectors. Figure 13 shows the result of the clustering. From a simple observation it is obvious that the two clustering are different. However, in order to prove how different they are similarity metrics are needed. Figure 14 shows

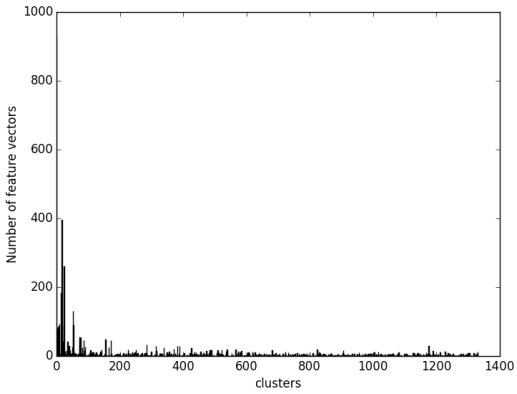

Figure 13: Clustering generated

the ground truth clustering. The next step is to compare the ground truth clustering

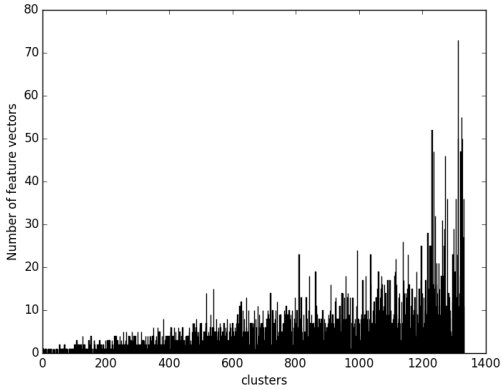

Figure 14: Clustering ground truth

with the generated cluster. Script A.13 performs the clustering comparison by using different metrics. The adjusted random index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings [6]. An adjusted random index with value 0 indicates that there is no difference between the calculated clustering and a random assignment whereas a value of 1 indicates perfect clustering. The Mutual Information is a function that measures the agreement of the two assignments, ignoring permutations. Two different normalised versions of this measure are available, Normalised Mutual Information(NMI) and Adjusted Mutual Information(AMI). NMI is often used in the literature while AMI was proposed more recently and is normalised against chance. Values close to 0 indicate a random assignment whereas values close to 1 indicate equal clusters.
Homogeneity measures that each cluster contains members of a single class, completeness measures that all members of a single class are assigned to the same cluster. Additionally, the v-measure is the harmonic mean between homogeneity and completeness. Applying this measures to the ground truth and calculate clustering produces these results:

- Adjusted random index = 0.0004
- Adjusted mutual Information based scores = 0.0045
- Homogenity = 0.5380
- Completeness = 0.6433
- V-measure = 0.5860

The next step was to consider a dimensional reduction technique in order to try to improve the results. Script [A.14] performs a Principal Component Analysis on the data. It first normalises the feature vectors, it performs a PCA on the normalised vectors and finally tries to cluster them. These results were obtained using the same measures as before:

- Adjusted random index = 0.0012
- Adjusted mutual Information based scores = 0.0110
- Homogenity = 0.4799
- Completeness = 0.6299
- V-measure = 0.5436

Figure 15 shows the clustering after performing PCA on the data.

6.3 Conclusion

The results for this study show unambiguously that the feature vectors do not represent users accurately enough to differentiate them. The metrics used to compare the clusters indicate that the clustering process produces almost a random clustering and for this reason the study could not be completed. After performing PCA the results improved slightly but were still poor. Because of the unsuccessful clustering the study could not go forward. It had to continue by calculating the probability distribution of the vectors

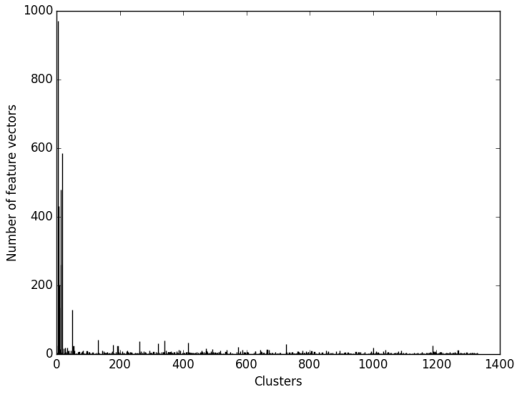

Figure 15: Clustering after PCA

on each cluster in order to be able to compare them.
Finding the right feature vectors is crucial in order to be able to differentiate between users and predict their performance. A possible way to improve the feature vectors is to calculate the frequency of the events rather than the number of them. That way, the feature vectors would not explicitly dependent on the time. In order to calculate the frequencies of events one could consider the data as a signal and calculate its FFT. This way, each user would be characterised by a spectrum, then a clustering process could b applied to the different spectrums thus allowing for differentiation.

7 Reflecion

Working on it... It contains:

- Difference between initial proposal and final project
- Difficult bits
- What went wrong, what went good
- How to improve results

References

[1] Creative Programming MOOC. `https://www.coursera.org/course/digitalmedia`. Accessed: 2015-02-20.

[2] International Educational Data Mining Society. `http://www.educationaldatamining.org/`. Accessed: 2015-04-19.

[3] Mongo Documentation. `http://www.mongodb.org/about/introduction/`. Accessed: 2015-04-22.

[4] Moodle. `https://moodle.org/`. Accessed: 2015-04-20.

[5] Periodic Videos. https://www.youtube.com/user/periodicvideos/. Accessed: 2015-02-20.

[6] SciKit-learn. http://scikit-learn.org/stable/. Accessed: 2015-02-20.

[7] The Year of the MOOC. http://www.nytimes.com/2012/11/04/education/edlife/massive-open-online-courses-are-multiplying-at-a-rapid-pace.html?_r=0. Accessed: 2015-04-20.

[8] Rsj D Baker, Sm Gowda, and Michael Wixon. Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra. *... Educational Data Mining ...*, pages 126–133, 2012.

[9] Rsjd S J D. Baker. Data mining for education. *International Encyclopedia of Education*, 7:112–118, 2010.

[10] Tim Berners-Lee, Mark Fischetti, and Michael L Foreword By-Dertouzos. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. HarperInformation, 2000.

[11] Félix Castro, Alfredo Vellido, Àngela Nebot, and Francisco Mugica. Applying data mining techniques to e-learning problems. *Studies in Computational Intelligence*, 62:183–221, 2007.

[12] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[13] John Timperley Hattie Helen E-Mail Address, John Hattie, and Helen Timperley. The power of feedback. [References]. *Review of Educational Research*, .77:16–7, 2007.

[14] Marcelo Maia, Jussara Almeida, and Virgílio Almeida. Identifying User Behavior in Online Social Networks. *Proceedings of the 1st workshop on Social network systems*, pages 1–6, 2008.

[15] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI Press, 1991.

[16] Cristóbal Romero, Manuel Ignacio López, Jose María Luna, and Sebastián Ventura. Predicting students' final performance from participation in on-line discussion forums. *Computers and Education*, 68:458–472, 2013.

[17] Cristóbal Romero and Sebastián Ventura. Educational data mining: A review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 40(X):601–618, 2010.

[18] Michael Scriven. The Methodology Of Evaluation, 1967.

A Scripts

A.1 Build the research database

```
from bson import *
from pymongo import *
import time
from datetime import timedelta
from pylab import *


import json
from pymongo.errors import InvalidOperation

#source db
#mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --
    fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/
    musiccircle_latest/musiccircle
cli = MongoClient("localhost:27017")
db = cli.musiccircle

#target instance - intermediate representation
#mongod --port 27018 -dbpath /Users/chris/expts/praise/dataAnalysis
    /mongo/mus0IR --fork --syslog
# mongod --port 27018 --dbpath /Users/matthew/Documents/
    PRAISE_local/data/musiccircle_latest/processedclir = MongoClient
    ("localhost:27018")
clir = MongoClient("localhost:27026")
ir = clir.research

# generate a list of dates
def dateListGenerator(start, end, delta):
    result=[]
    curr = start
    while curr < end:
        curr += delta
        result.append(curr)

    return result
datesList=dateListGenerator(datetime.datetime(2014, 7, 22, 00, 00,
    00, 00),datetime.datetime(2014, 7, 27, 00, 00, 00, 00),timedelta
    (days=1))
datesListLength=len(datesList)
prog=0

for date in datesList:
    prog=prog+1
    foundLogs=db.log.find({"$and":[
                    {"datetime":{"$gte":date}},
                    {"datetime":{"$lt":(date+timedelta(days=1))}}
                    ]
                }
```

```
                )
    print datesListLength-prog," left"

    for logChunk in foundLogs:
        bulk = ir.Actors.initialize_ordered_bulk_op()
        activities=[]
        userFound=bulk.find({'$and':[{'_id':logChunk['uid']}]})
        for logEvent in json.loads(logChunk["text"]):
            if ObjectId.is_valid(logChunk["uid"]):
                if "id" in logEvent and "logtype" in logEvent:
                    tempActivity={'at':6, '_id':logEvent['id'], 'ts
    ':datetime.datetime.fromtimestamp(logEvent['time']/1000.0), '
    type':logEvent['type'], 'logtype':logEvent['logtype']}
                    if(tempActivity not in activities):
                        activities.append(tempActivity)
                        userFound.update({'$push':{'activities':
    tempActivity}})
        if(len(activities)>0):
            try:
                bulk.execute()
            except InvalidOperation as invOpt:
                pass
```

./pythonScripts/importShard1.py

A.2 Build the research database

```
from bson import *
from pymongo import *
import time
from datetime import timedelta
from pylab import *


import json
from pymongo.errors import InvalidOperation

#source db
#mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --
    fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/
    musiccircle_latest/musiccircle
cli = MongoClient("localhost:27017")
db = cli.musiccircle

#target instance - intermediate representation
#mongod --port 27018 -dbpath /Users/chris/expts/praise/dataAnalysis
    /mongo/mus0IR --fork --syslog
```

```
# mongod --port 27018 --dbpath /Users/matthew/Documents/PRAISE_local/data/musiccircle_latest/processedclir = MongoClient("localhost:27018")
clir = MongoClient("localhost:27026")
ir = clir.research

# generate a list of dates
def dateListGenerator(start, end, delta):
    result=[]
    curr = start
    while curr < end:
        curr += delta
        result.append(curr)

    return result
datesList=dateListGenerator(datetime.datetime(2014, 7, 27, 00, 00, 00, 00),datetime.datetime(2014, 7, 28, 00, 00, 00, 00),timedelta(days=1))
datesListLength=len(datesList)
prog=0

for date in datesList:
    prog=prog+1
    foundLogs=db.log.find({"$and":[
                    {"datetime":{"$gte":date}},
                    {"datetime":{"$lt":(date+timedelta(days=1))}}
                    ]
                  }
                 )
    print datesListLength-prog," left"

    for logChunk in foundLogs:
        bulk = ir.Actors.initialize_ordered_bulk_op()
        activities=[]
        userFound=bulk.find({'_id':logChunk['uid']})
        for logEvent in json.loads(logChunk["text"]):
            if ObjectId.is_valid(logChunk["uid"]):
                if "id" in logEvent and "logtype" in logEvent:
                    tempActivity={'at':6, '_id':logEvent['id'], 'ts':datetime.datetime.fromtimestamp(logEvent['time']/1000.0), 'type':logEvent['type'], 'logtype':logEvent['logtype']}
                    if(tempActivity not in activities):
                        activities.append(tempActivity)
                        userFound.update({'$push':{'activities':tempActivity}})
        if(len(activities)>0):
            try:
                bulk.execute()
            except InvalidOperation as invOpt:
                pass
```

./pythonScripts/importShard2.py

A.3 Build the research database

```
from bson import *
from pymongo import *
import time
from datetime import timedelta
from pylab import *


import json
from pymongo.errors import InvalidOperation

#source db
#mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/musiccircle_latest/musiccircle
cli = MongoClient("localhost:27017")
db = cli.musiccircle

#target instance - intermediate representation
#mongod --port 27018 -dbpath /Users/chris/expts/praise/dataAnalysis/mongo/mus0IR --fork --syslog
# mongod --port 27018 --dbpath /Users/matthew/Documents/PRAISE_local/data/musiccircle_latest/processedclir = MongoClient("localhost:27018")
clir = MongoClient("localhost:27026")
ir = clir.research

# generate a list of dates
def dateListGenerator(start, end, delta):
    result=[]
    curr = start
    while curr < end:
        curr += delta
        result.append(curr)

    return result
datesList=dateListGenerator(datetime.datetime(2014, 7, 28, 00, 00, 00, 00),datetime.datetime(2014, 7, 30, 00, 00, 00, 00),timedelta(days=1))
datesListLength=len(datesList)
prog=0

for date in datesList:
```

```
    prog=prog+1
    foundLogs=db.log.find({"$and":[
                    {"datetime":{"$gte":date}},
                    {"datetime":{"$lt":(date+timedelta(days=1))}}
                    ]
                  }
                 )
    print datesListLength-prog," left"

    for logChunk in foundLogs:
        bulk = ir.Actors.initialize_ordered_bulk_op()
        activities=[]
        userFound=bulk.find({'_id':logChunk['uid']})
        for logEvent in json.loads(logChunk["text"]):
            if ObjectId.is_valid(logChunk["uid"]):
                if "id" in logEvent and "logtype" in logEvent:
                    tempActivity={'at':6, '_id':logEvent['id'], 'ts
    ':datetime.datetime.fromtimestamp(logEvent['time']/1000.0), '
    type':logEvent['type'], 'logtype':logEvent['logtype']}
                    if(tempActivity not in activities):
                        activities.append(tempActivity)
                        userFound.update({'$push':{'activities':
    tempActivity}})
        if(len(activities)>0):
            try:
                bulk.execute()
            except InvalidOperation as invOpt:
                pass
```

./pythonScripts/importShard3.py

A.4 Build the research database

```
from bson import *
from pymongo import *
import time
from datetime import timedelta
from pylab import *


import json
from pymongo.errors import InvalidOperation

#source db
#mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --
    fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/
    musiccircle_latest/musiccircle
cli = MongoClient("localhost:27017")
```

```
db = cli.musiccircle

#target instance - intermediate representation
#mongod --port 27018 -dbpath /Users/chris/expts/praise/dataAnalysis/mongo/mus0IR --fork --syslog
# mongod --port 27018 --dbpath /Users/matthew/Documents/PRAISE_local/data/musiccircle_latest/processedclir = MongoClient("localhost:27018")
clir = MongoClient("localhost:27026")
ir = clir.research

# generate a list of dates
def dateListGenerator(start, end, delta):
    result=[]
    curr = start
    while curr < end:
        curr += delta
        result.append(curr)

    return result
datesList=dateListGenerator(datetime.datetime(2014, 7, 30, 00, 00, 00, 00),datetime.datetime(2014, 7, 31, 00, 00, 00, 00),timedelta(days=1))
datesListLength=len(datesList)
prog=0

for date in datesList:
    prog=prog+1
    foundLogs=db.log.find({"$and":[
                    {"datetime":{"$gte":date}},
                    {"datetime":{"$lt":(date+timedelta(days=1))}}
                    ]
                  }
                 )
    print datesListLength-prog," left"

    for logChunk in foundLogs:
        bulk = ir.Actors.initialize_ordered_bulk_op()
        activities=[]
        userFound=bulk.find({'_id':logChunk['uid']})
        for logEvent in json.loads(logChunk["text"]):
            if ObjectId.is_valid(logChunk["uid"]):
                if "id" in logEvent and "logtype" in logEvent:
                    tempActivity={'at':6, '_id':logEvent['id'], 'ts':datetime.datetime.fromtimestamp(logEvent['time']/1000.0), 'type':logEvent['type'], 'logtype':logEvent['logtype']}
                    if(tempActivity not in activities):
                        activities.append(tempActivity)
                        userFound.update({'$push':{'activities':tempActivity}})
```

```
57          if(len(activities)>0):
                try:
59                  bulk.execute()
                except InvalidOperation as invOpt:
61                  pass
```

./pythonScripts/importShard4.py

A.5 Plot activities per day

```
1  %pylab inline
   %matplotlib inline
3  from bson import *
   from pymongo import *
5  import time
   from datetime import timedelta
7  from pylab import *

9  #source db
   #mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --
       fork --syslog
11 #mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/
       musiccircle_latest/musiccircle
   cli = MongoClient("localhost:27017")
13 db = cli.musiccircle

15 #target instance - intermediate representation
   #mongod --port 27018 -dbpath /Users/chris/expts/praise/dataAnalysis
       /mongo/mus0IR --fork --syslog
17 # mongod --port 27018 --dbpath /Users/matthew/Documents/
       PRAISE_local/data/musiccircle_latest/processedclir = MongoClient
       ("localhost:27018")
   #clir = MongoClient("localhost:27018")
19 #ir = clir.mus0IR

21 # Class that plots frequency of items in a collection vs time

23 class PlotItemsVsTime:

25     def __init__(self,title,data,startDate,resolutionInDays,
       datefieldName):
           self.data = data.find().sort(datefieldName,1)
27         self.title=title
           self.startDate=startDate
29         self.resolutionInDays=resolutionInDays
           self.datefieldName=datefieldName
31
       def binData(self):
33         mainList=[]
```

```
        tempList=[]
35      self.startDate=self.startDate.replace(hour=0, minute=0,
    second=0, microsecond=0)
        deltaDays=self.resolutionInDays
37      nextDate=self.startDate+timedelta(days=deltaDays)
        print "items in input collection",self.data.count()
39      print "start date=",self.startDate
        for item in self.data:
41          currentDate=item[self.datefieldName]
            if currentDate != None:
43              if nextDate>=currentDate:
                    tempList.append(item)
45              elif currentDate>=nextDate:
                        mainList.append({"date":self.startDate,"
    items":tempList})
47                      self.startDate=currentDate.replace(hour=0,
    minute=0, second=0, microsecond=0)
                        nextDate=self.startDate+timedelta(days=
    deltaDays)
49                      tempList=[]
                        #there is an item that will go in the next
    group, so add it now
51                      tempList.append(item)
        mainList.append({"date":self.startDate,"items":tempList})
53      print "end date",currentDate
        print "bins=",len(mainList)
55      self.binList=mainList
        return mainList
57
    def countFreqOfBins(self):
59      preparedList=[]
        count=0
61      for i in self.binList:
            preparedList.append({"date":i["date"],"freq":len(i["
    items"])})
63          count=count+len(i["items"])
        print "total items=",count
65      self.preparedList=preparedList
        return preparedList
67
    def checkForLostItems(self):
69      count=0
        for item in self.preparedList:
71          count=count+item["freq"]
        if count==self.data.count():
73          print "data is OK"
        else:
75          print "data might be wrong, diff=",count-self.data.
    count()
```

```
    def doThePlot(self):
        self.binData()
        self.countFreqOfBins()
        self.checkForLostItems()
        plt.figure(figsize=(12,6))
        plt.plot([element["date"] for element in self.preparedList
    ],[element["freq"] for element in self.preparedList],"ro-")
        plt.title(self.title)
        plt.show()

# Uploads per day
uploadsPerDay=PlotItemsVsTime("Uploads/day",db.AudioContent,
    datetime.datetime(2014, 6, 30, 13, 18, 42, 824000),1,"datetime")
uploadsPerDay.doThePlot()

# Sessions per day
sessionsPerDay=PlotItemsVsTime("Sessions/day",db.Session,datetime.
    datetime(2014, 6, 30, 13, 18, 42, 824000),1,"sessionStart")
sessionsPerDay.doThePlot()

# Views per day
viewsPerDay=PlotItemsVsTime("Views/day",db.MediaViewLog,datetime.
    datetime(2014, 6, 30, 15, 11, 32, 585000),1,"datetime")
viewsPerDay.doThePlot()

# Comments per day
commentsPerDay=PlotItemsVsTime("Comments/day",db.ActivityDefinition
    ,datetime.datetime(2014, 6, 30, 19, 48, 45, 511000),1,"datetime"
    )
commentsPerDay.doThePlot()

# Replies per day
repliesPerDay=PlotItemsVsTime("Replies/day",db.Activity,datetime.
    datetime(2014, 6, 30, 19, 48, 45, 511000),1,"datetime")
repliesPerDay.doThePlot()
```

./pythonScripts/initialPerDayPlots.py

A.6 Remove duplicates from activities array

```
# This script remove the duplicates from the activities array

from bson import *
from pymongo import *
import time
from datetime import timedelta

# Connect to mongo query router
clir = MongoClient("localhost:27026")
```

```
ir = clir.research

# Find all the actors
Actors=ir.Actors.find()

# iterate through actors
for actor in Actors:
  # for each actor make an empty array
  tempActivities=[]
  # for each actor activity
  for activity in actor["activities"]:
    # put the activity in the tempActivities if it's not there
    if activity not in tempActivities:
      tempActivities.append(activity['ts'])
    # if it is there, then remove from the database
    elif activity in tempActivities:
      #print activity
      # remove from the database
      ir.Actors.update({'_id':actor["_id"]},{'$pull':{'activities':
    activity}})
  print "checked ",len(actor["activities"]), "activities"
```

./pythonScripts/removeDuplicates.py

A.7 Add total number of activities and uploads for each user

```
# Count the uploads and activities for each user

from bson import *
from pymongo import *
import time
from datetime import timedelta

# Connect to the research database
clir = MongoClient("localhost:27026")
ir = clir.research

# Find the actors that have activities and uploads
actors=ir.Actors.find({"$and":[{"activities.at":{"$eq":0}},{"
    activities.at":{"$eq":6}}]},timeout=False)
# Just to keep track of the progress (script takes a few seconds to
    execute)
totalActors=actors.count()
n=0
# For each actor, count the activities and uploads
for actor in actors:
  print totalActors-n," left"
  n=n+1
  uploadsAndActivities6=0
```

```
    for activity in actor['activities']:
23      if activity['at']==0 or activity['at']==6:
            uploadsAndActivities6=uploadsAndActivities6+1
25  # Add a field to the actor withe the total numper of activities
      and uploads
    ir.Actors.update({'_id':actor['_id']},{'$push':{'
      uploadsAndLogActivities':uploadsAndActivities6}})
```

./pythonScripts/countUploadsAndActivities.py

A.8 Import sessions for each user

```
1 # Extract the sessions for each users and put them into an array in
      the Actors collection

3 from bson import *
  from pymongo import *
5 import time
  from datetime import timedelta
7
  # Music Circle database
9 cli = MongoClient("localhost:27017")
  db = cli.musiccircle
11 # Research database
  clir = MongoClient("localhost:27026")
13 ir = clir.research

15 # Find all users with uploads and activities
  actors=ir.Actors.find({"$and":[{"activities.at":{"$eq":0}},{"
      activities.at":{"$eq":6}}]},timeout=False)
17 # To keep track of the progress (script takes a while to execute)
  totalActors=actors.count()
19 n=0
  # For each actor find all the sessions that correspond to the actor
21 for actor in actors:
    print totalActors-n," left"
23  sessions=db.Session.find({'user_id':actor['_id']})
    # For all the sessions found, put them into the right actor in
      the Actors collection inside a new field called sessions
25  for session in sessions:
      ir.Actors.update({'_id':actor['_id']},{'$push':{'sessions':
      session}})
```

./pythonScripts/makeSessions.py

A.9 Build the feature vectors

```
# Build feature vectors and store them in a json file

from bson import *
from pymongo import *
import time
from datetime import timedelta
import numpy as np
import json
import operator
import os

# Connect to the research database
clir = MongoClient("localhost:27026")
ir = clir.research

# get actors that have uploaded and have log activities
actors=ir.Actors.find({"$and":[{"activities.at":{"$eq":0}},{"
    activities.at":{"$eq":6}}]},timeout=False)
totalActors=actors.count()

print totalActors

# Sort the actors by their number of activities
def sortActors():
  actorsList=[]
  n=0
  # make a list of actors so we can sort it
  for actor in actors:
    n=n+1
    print totalActors-n," left"
    actorsList.append(actor)
  actorsList.sort(key=operator.itemgetter('uploadsAndLogActivities'
    ))
  return actorsList

# Check if a key is in a dictionary
def checkForKey(key,dictionary):
  if key in dictionary:
    return True
  else:
    return False

def makeSessionObjectList(actor):
# make an object that will store session start and end timestamp
  sessionsObjectList=[]
  iterator=np.arange(0,len(actor['sessions'])-1)

  for i in iterator:
    tempSession={}
```

```
    tempSession['sessionStart']=actor['sessions'][i]['sessionStart']
    tempSession['nextSession']=actor['sessions'][i+1]['sessionStart']
    sessionsObjectList.append(tempSession)
    #print tempSession
  return sessionsObjectList

def makeActivitiesList(actor):
  # we put the activities in a list so we can sort by timestamp
  activitiesList=[]

  for activity in actor['activities']:
    if activity['at']==0:
      activitiesList.append(activity)
    if activity['at']==6:
      # get the 'click'
      if checkForKey('logtype',activity) and activity['logtype']!='playing' and checkForKey('type',activity) and activity['type']=='click':
        activitiesList.append(activity)

  activitiesList.sort(key=operator.itemgetter('ts'))

  return activitiesList

def makeActivityChunks(sessionsObjectList,activitiesList):
  # we want to devide the activities list respect to the sessions
  activitiesSet=[]

  for session in sessionsObjectList:
    #print session['sessionStart']
    tempAct=[]
    del tempAct[:]

    for activity in activitiesList:

      if activity['ts']>=session['sessionStart'] and activity['ts']<session['nextSession']:
        tempAct.append(activity)
        #print activity
    if len(tempAct)!=0:
      activitiesSet.append(tempAct)

  return activitiesSet

def checkForString(dictionary,key,check):
  if key in dictionary and dictionary[key]==check:
    return True
  else:
```

```
93      return False

95  def makeFeatureVectors(activitiesSet):
      n=0
97
      #print len(activitiesSet)
99
      featureVectors=[]
101   del featureVectors[:]
      for activitySet in activitiesSet:
103     n=n+1
        upload=0
105     region_block=0
        my_track_nav_item=0
107     community_media_nav_item=0
        play=0
109     tempVector=[]
        for activity in activitySet:
111       if activity['at']==0:
            upload=upload+1
113       if checkForString(activity,'logtype','region_block'):
            region_block=region_block+1
115       if checkForString(activity,'logtype','my_track_nav_item'):
            my_track_nav_item=my_track_nav_item+1
117       if checkForString(activity,'logtype','
        community_media_nav_item'):
            community_media_nav_item=community_media_nav_item+1
119       if checkForString(activity,'logtype','play'):
            play=play+1
121
        tempVector.append(upload)
123     tempVector.append(region_block)
        tempVector.append(my_track_nav_item)
125     tempVector.append(community_media_nav_item)
        tempVector.append(play)
127     featureVectors.append(tempVector)

129   return featureVectors


131
actorsList=sortActors()
133
outfile=open("histogramFeatureVectors.json", "w")
135
featureList=[]
137 for actor in actorsList:
      featureObject={}
139   activitiesSet=makeActivityChunks(makeSessionObjectList(actor),
        makeActivitiesList(actor))
      #print len(activitiesSet)
```

```
  featureObject['actor_id']=str(actor['_id'])
  featureObject['featureVectors']=makeFeatureVectors(activitiesSet)
  featureList.append(featureObject)

json.dump({'features':featureList}, outfile, indent=4)
outfile.close()
```

./pythonScripts/featureVectorsToJson.py

A.10 Link between Research and Coursera databases

```
import os
import re
from pymongo import *
import numpy as np
import matplotlib.pyplot as plt
import pylab as p
import math
from bson import *
import pymysql

#source db
#mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --
    fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/
    musiccircle_latest/musiccircle
#cli = MongoClient("localhost:27017")
#db = cli.musiccircle

clir = MongoClient("localhost:27026")
ir = clir.research

mysqldb = pymysql.connect(db='coursera', user='root', passwd='root'
    , host='localhost', port=8889  )

mysqldb.select_db('coursera')
mysql_cur = mysqldb.cursor()


def getSessionIdsFromDirectory(dir):
  sessionIdsList=[]
  for subdir in os.walk(dir):
    for el in subdir[2]:
      if el=='fields.html':
        path=subdir[0]+'/'+el
        f = open(path, 'r')
        content = f.read()
        startSessionId='<title>'
        endSessionId='</title>'
```

```
36          sessionId=re.search(re.escape(startSessionId)+"(.*?)"+re.
    escape(endSessionId),content).group(1)
            sessionId=sessionId.split('session_user_id: ')
38          sessionId=sessionId[1].replace(')','')
            startMediaId='href="https://coursera.musiccircleproject.com
    /?media/'
40          endMediaId='">'
            mediaId=re.search(re.escape(startMediaId)+"(.*?)"+re.escape
    (endMediaId),content)
42          if mediaId!=None:
              mediaId=mediaId.group(1)
44            mediaId=mediaId.split('" title="')
              #mediaIdsList.append(mediaId[0])
46            sessionIdsList.append([{'sessionId':sessionId,'mediaId':
    mediaId[0]}])
  return sessionIdsList
48


50 def getMCUserIds():
  data=[]
52  for el in getSessionIdsFromDirectory('peer1'):
    oid=el[0]['mediaId']
54  dataTemp={'sessionId':el[0]['sessionId'],'userId':oid}
    if ObjectId.is_valid(oid):
56    userIdCursor=ir.Media.find({'_id':ObjectId(oid)})
      for i in userIdCursor:
58      userId=i['owner']
        dataTemp['mcUserId']=userId
60  data.append(dataTemp)
  return data
62

64 def linkMCUserIdsWithGrades():
  data=[]
66  for el in getMCUserIds():
    dataTemp={'userId':el['userId']}
    query="SELECT normal_grade FROM course_grades WHERE
    session_user_id=\'"+str(el['sessionId'])+"\'"
68  #query="select normal_grade from course_grades where
    session_user_id='3bb0d5a68482a648611c7533844c89ccf733ab1b'"
    #print query
70  mysql_cur.execute(query)
    row=mysql_cur.fetchall()
72  dataTemp['grade']=row[0]
    data.append(dataTemp)
74  if 'mcUserId' in el:
      ir.ActorsGrade.insert({'id':ObjectId(el['mcUserId']),'grade':
    row[0]})
76 #print data

78 def addActivitiesLength():
```

```
  for actor in ir.Actors.find():
    if 'activitiesLength' in actor:
      ir.ActorsGrade.update({'id':actor['_id']},{'$push':{'
    activitiesLength':actor['activitiesLength']}})

count=0
x=[]
y=[]
i=0
for actor in ir.ActorsGrade.find():
  if 'activitiesLength' in actor:
    a=actor['activitiesLength'][0][0]
    x.append(a)
    y.append(actor['grade'])
    i=i+1
plt.plot(y,x,'ro')
plt.show()
#linkMCUserIdsWithGrades()
```

./pythonScripts/dbLink.py

A.11 Generate plots that compare general activity to grades achieved

```
from numpy import genfromtxt
from bson import *
from pymongo import *
import time
from datetime import timedelta
import numpy as np
import sys
import json
from pymongo.errors import InvalidOperation
import operator
import matplotlib.pyplot as plt
import pylab as p
import math
import os
from sklearn.cluster import KMeans
from sklearn import datasets, linear_model
import random


clir = MongoClient("localhost:27026")
ir = clir.research


def generateCummulativeListFromFeatureVectorsBySession(json_data):
  featureListFromFile = json.load(json_data)
```

```
  cummulative=[]

  for actor in featureListFromFile['features']:
    actorId=actor['actor_id']
    uploads=0
    region_block=0
    my_tracks=0
    community_tracks=0
    play=0


    for featureVector in actor['featureVectors']:
      uploads=uploads+featureVector[0]
      region_block=region_block+featureVector[1]
      my_tracks=my_tracks+featureVector[2]
      community_tracks=community_tracks+featureVector[3]
      play=play+featureVector[4]

    cummulative.append({'actor_id':actorId,'features':[uploads,
    region_block,my_tracks,community_tracks,play]})

  return cummulative

def generatePlot():
  cummulativeList=
    generateCummulativeListFromFeatureVectorsBySession(open('
    histogramFeatureVectors.json'))

  y=[]
  grades=[]
  for actor in cummulativeList:
    grade=ir.ActorsGrade.find_one({'id':ObjectId(actor['actor_id'])
    })
    if grade != None:
      # uploads
      y.append(actor['features'][0])

      # region block
      #y.append(actor['features'][1])

      # my tracks
      #y.append(actor['features'][2])

      # community tracks
      #y.append(actor['features'][3])

      # play
      #y.append(actor['features'][4])

      grades.append(grade['grade'])
```

```
    regr = linear_model.LinearRegression()
    regr.fit(grades, y)
    # The coefficients
    print('Coefficients: \n', regr.coef_)
    print('R^2', regr.score(grades,y))

    plt.plot(grades,y,'o')
    plt.savefig('uploads.png')
    plt.show()


generatePlot()
```

./pythonScripts/groundTruth.py

A.12 Clustering the feature vectors

```
from bson import *
from pymongo import *
import time
from datetime import timedelta
import numpy as np
import sys
import json
from pymongo.errors import InvalidOperation
import operator
import matplotlib.pyplot as plt
import pylab as p
import math
import os
from sklearn.cluster import KMeans
import random
from collections import *

#source db
#mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/musiccircle_latest/musiccircle
#cli = MongoClient("localhost:27017")
#db = cli.musiccircle

#target instance - intermediate representation
#mongod --port 27018 -dbpath /Users/chris/expts/praise/dataAnalysis/mongo/mus0IR --fork --syslog
# mongod --port 27018 --dbpath /Users/matthew/Documents/PRAISE_local/data/musiccircle_latest/processedclir = MongoClient("localhost:27018")
```

```
#clir = MongoClient("localhost:27026")
#ir = clir.research


json_data=open('histogramFeatureVectors.json')
featureListFromFile = json.load(json_data)


numerOfUsers=len(featureListFromFile['features'])

groundTruthCuster=[]
i=0
for actor in featureListFromFile['features']:
  for feature in actor['featureVectors']:
    groundTruthCuster.append(i)
  i=i+1

k_means = KMeans(n_clusters=numerOfUsers, init='random')

featureList=[]
for actor in featureListFromFile['features']:
  for feature in actor['featureVectors']:
    featureList.append(feature)

k_means.fit(featureList)

clusters={}
iterator=np.arange(0,numerOfUsers)
for i in iterator:
  clusters[str(i)]=[]

clusterListForHist=[]

for actor in featureListFromFile['features']:
  for feature in actor['featureVectors']:
    cluster=k_means.predict(feature)
    for c in cluster:
      clusters[str(c)].append({'feature':feature,'actor_id':actor['
    actor_id']})
      clusterListForHist.append(c)

#plt.axis([0,s 4, 0, 800])

p.hist(groundTruthCuster,numerOfUsers,color='r')
plt.xlabel('clusters')
plt.ylabel('Number of feature vectors')
plt.show()
plt.savefig('histGroundTruth.png')
plt.clf()
p.hist(clusterListForHist,numerOfUsers,color='b')
```

```
plt.xlabel('clusters')
plt.ylabel('Number of feature vectors')
plt.show()
plt.savefig('histClusterGenerated.png')
```

./pythonScripts/clusteringFeatureSessionVectors.py

A.13 Comparison of two clusters

```
from bson import *
from pymongo import *
import time
from datetime import timedelta
import numpy as np
import sys
import json
from pymongo.errors import InvalidOperation
import operator
import matplotlib.pyplot as plt
import pylab as p
import math
import os
from sklearn.cluster import KMeans
import random
from collections import *
from sklearn import metrics


json_data=open('histogramFeatureVectors.json')
featureListFromFile = json.load(json_data)


numerOfUsers=len(featureListFromFile['features'])

k_means = KMeans(n_clusters=numerOfUsers, init='random')

featureList=[]
for actor in featureListFromFile['features']:
  for feature in actor['featureVectors']:
    featureList.append(feature)

k_means.fit(featureList)

groundTruthCuster=[]
i=0
for actor in featureListFromFile['features']:
  for feature in actor['featureVectors']:
    groundTruthCuster.append(i)
  i=i+1
```

```
41
modelCluster=[]
43
for actor in featureListFromFile['features']:
45  for feature in actor['featureVectors']:
      cluster=k_means.predict(feature)
47    for c in cluster:
        modelCluster.append(c)
49
print "adjusted random index (different from assigning random
    classes?)= ",metrics.adjusted_rand_score(groundTruthCuster,
    modelCluster)
51 print "adjusted mutual Information based scores (tends to increase
    with number of clusters)= ",metrics.adjusted_mutual_info_score(
    groundTruthCuster,modelCluster)
print "homogenity, completeness, v-measure scores = ",metrics.
    homogeneity_completeness_v_measure(groundTruthCuster,
    modelCluster)
```

./pythonScripts/validateClusteringFeatureSessionVectors.py

A.14 Comparison of two clusters

```
1 import numpy as np
from sklearn.decomposition import PCA
3 import json
import matplotlib.pyplot as plt
5 import math
from sklearn.cluster import KMeans
7 from sklearn import metrics


9


11 pca = PCA(n_components=2)

13 def getJSONDataAsList(json_data):
    featureListFromFile = json.load(json_data)
15  featureList=[]
    for actor in featureListFromFile['features']:
17    for feature in actor['featureVectors']:
        featureList.append(feature)
19  print "features list has = ",len(featureList)," values"
    return featureList
21
def getJSONData(json_data):
23  return json.load(json_data)

25 def performPCA(featureList):
    pca.fit(featureList)
```

```
27    print "variances = ",pca.explained_variance_ratio_
      transformedVectors=pca.transform(featureList)
29    return transformedVectors

31  def normaliseList(featureList):
      featureListNorm=[]
33    for feature in featureList:
        norm=math.sqrt(sum([j*j for j in feature]))
35      if norm !=0:
          normalisedVector=[float(i)/math.sqrt(sum([j*j for j in
        feature])) for i in feature]
37      else:
          normalisedVector=[0 for i in feature]
39      featureListNorm.append(normalisedVector)
      return featureListNorm
41
    def preapareForPlot(featureList):
43    X=[x[0] for x in featureList]
      Y=[y[1] for y in featureList]
45    return[X,Y]

47  def clusterKMeans(featureList,nClusters):
      k_means = KMeans(n_clusters=nClusters,init='random')
49    k_means.fit(featureList)
      clusterClasses=[]
51    for feature in featureList:
        cluster=k_means.predict(feature)
53      for c in cluster:
          clusterClasses.append(c)
55    return clusterClasses

57  def calculateGroundTruthClustering(json_data):
      featureListFromFile = json.load(json_data)
59    groundTruthCuster=[]
      i=0
61    for actor in featureListFromFile['features']:
        for feature in actor['featureVectors']:
63        groundTruthCuster.append(i)
        i=i+1
65    return groundTruthCuster

67  def compareClustering(groundTruth,modelCluster):
      print "adjusted random score (different from assigning random
        classes?)= ",metrics.adjusted_rand_score(groundTruthCustering,
        modelCluster)
69    print "adjusted mutual Information based scores (tends to
        increase with number of clusters)= ",metrics.
        adjusted_mutual_info_score(groundTruthCustering,modelCluster)
      print "homogenity, completeness, v-measure scores = ",metrics.
        homogeneity_completeness_v_measure(groundTruthCustering,
```

```
    modelCluster)

figure=plt.figure()

featureList=getJSONDataAsList(open('histogramFeatureVectors.json'))
#normalise
normalisedList=normaliseList(featureList)

# plot first 2 components of normalised data
#doTheplot(normalisedList,'b')

#cluster featureList
print 'clustering'
#featureListClustering=clusterKMeans(featureList,len(featureList))

# feature clustering vs ground truth clustering
print "cluster metrics for featureList"
groundTruthCustering=calculateGroundTruthClustering(open('
    histogramFeatureVectors.json'))
#nUsers = len(json.load(open('histogramFeatureVectors.json'))['
    features'])
#plt.hist(groundTruthCustering,nUsers)
#plt.show()
#compareClustering(groundTruthCustering,featureListClustering)

#normalised feature clustering vs ground truth clustering
#print "cluster metrics for normalised feature list (I checked it's
     exactly the same as with non normlised)"
#compareClustering(normalisedList,featureListClustering)

# do PCA


print "PCA"
pcaFeatureList=performPCA(normalisedList)
print 'clustering....'
print len(pcaFeatureList)

pcaClustering = clusterKMeans(pcaFeatureList,1332)
print 'comparing...'
compareClustering(pcaFeatureList,pcaClustering)

print 'making hist ...'
plt.hist(pcaClustering,len(pcaClustering))
plt.savefig('pcaClustering.png')

'''
featureListData=preapareForPlot(featureList)
featureListFigure = plt.figure(0)
```

```
117 ax1 = featureListFigure.add_subplot(111)
    ax1.plot(featureListData[0], featureListData[1],'ro')
119
    normFeatureListData=preapareForPlot(normalisedList)
121 normFeatureListFigure = plt.figure(1)
    ax2 = normFeatureListFigure.add_subplot(111)
123 ax2.plot(normFeatureListData[0], normFeatureListData[1],'ro')

125 plt.show()

127 '''
    #doTheplot(pcaFeatureList,'r')
```

./pythonScripts/pca.py