Mining educational social networks for student performance classification

Student: Andreu Grimalt Reynes

Supervisor: Matthew Yee-King

Degree: BSc Creative Computing

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. Two studies on the data are presented, they experiment with different feature vectors and Data Mining methods: On one hand, a high level data analysis explores the data and calculates various statistics. Plots and linear regressions are calculated which fail to find any correlations between feedback activity and student performance.
On the other hand an exploratory data analysis experiments with different feature vectors and Data Mining methods. The methods used to analyse the data belong to the Data Mining field and based in a k-nearest neighbour classification, k-means clustering and PCA. The method used with k-nearest neighbour is able to classify students into performance categories with an accuracy of 0.7980.

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" [24]. The process of assessment consists in the methodology used to produce the judgement and involves testing the student knowledge 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 winning 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. As an example of the role of assessment on the learning process, Bigg's proposes Constructive Alignment as a method to ensure that student activity is focused on the course material by aligning course activities and assessments using learning outcomes [14].
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 it requires feedback. Formative and summative assessment are the same process, the only difference being that formative assessment requires feedback. An example of summative assessment would be a test where students get back a mark without any feedback. On the other hand, an example of formative assessment would be a test where students get back a mark and a justification explaining why they got that grade.

Formative assessment is fundamental to education nowadays. It has been adopted

by education curricula in most countries, being the preferred choice by education professionals and students. An example of this is the National Student Survey [6]. It is an annual survey taken by higher education students in their final year, the questions relate to formative assessment which shows how feedback is considered fundamental by educational institutions and students alike.
A fundamental component of formative assessment is feedback, where students receive a justification of the assessment outcome. 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 [20]. 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 [23]. A definition of feedback from an AI perspective is provided by John Hattie [17]: '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. The act of giving feedback requires skills in making informed judgements as well as experience in transmitting suggestions effectively.

1.1 Personal experience on feedback in higher education

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 almost exclusively 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 not usually 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 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 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 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 end 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. Following my experience in my previous degree, the type of assessment has 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 information -which does not automatically involve learning- in order to write it down efficiently on an exam.
My personal experience demonstrates how critical feedback is. When feedback is present I learn, when feedback is not present I disengage and merely memorise. To conclude, 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.

1.2 Online education

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 Berners-Lee in *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor* [13] as a system where humans do the creative work and machines do the administrative part. Such a system is called a *social machine*. In the context of an educational social machine the creative work would be to provide formative assessment whereas the administrative part would be to provide all the technology and infrastructure for the online course to exist.

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 'interest' or engagement is a very interesting phenomena 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.
Youtube is arguably one of the most popular social networks that is used for learning. A good example of how this social network is used in an informal educational context is Periodic Videos [7]. It is a Youtube Channel owned by the University of Nottingham Chemistry Department, it has 2,298 videos about chemistry with a total of 291,658,881 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 and it is constantly growing. As pointed out, most of these channels are targeted to non formal learning and most of their popularity and success is probably due to this reason.

Over the last fifteen years higher education institutions have been transitioning to virtual learning environments (VLE) to support their education services. Universities use systems like Moodle [5] that provide a virtual environment where students and teachers can interact with each other. VLEs typically contain course materials, assignment submission tools and forums where students can engage in discussions among other tools. However, the social potential of the VLEs is often not exploited or used very scarcely. An important detail is that VLEs are usually 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) [9]. MOOCs are courses produced by major educational institutions that have a high number of students and are free of charge. The content of MOOCs usually consists of video lectures, reading lists and exercises but also in-lecture quizzes and interactive forums to support communities of learners. These types of courses are usually distributed in online platforms offering a wide range of courses from different subjects. MOOC platforms were popularised by three companies: Coursera, edX and Udacity. All three appeared in 2012, a fact that prompted the New York Times to coin 2012 as 'The Year of the MOOC' [9]. 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 institutions.
One of the issues MOOCs need to address is that the title achieved after successful completion usually lacks any formal academic recognition. In this regard, there is still a long way to go to be able to compare MOOCs with higher education courses.
MOOCs provide a privileged platform to encourage a more social and interactive educational experience. As explained above, online forums have been traditionally used as platforms for communities of learners. However, recently there has been a shift towards a type of formative assessment called peer feedback. Peer feedback consists in an interchange of feedback between two students. By using an online peer feedback system students can support each other, express concerns or find solutions to problems collectively. More importantly they can teach and learn together, thus creating communities of learners through feedback and social interaction. Online peer feedback systems allow students to formatively assess each other anytime anywhere.

We have seen how important the role of feedback is and how formative assessment can be optimised through peer feedback. Additionally, I presented my personal experience on how summative and formative assessments are needed in order to facilitate learning. An interesting question is whether or not it is possible to establish the relation between these two types of assessment, in other words, whether 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 large amounts of data relating to student behaviour, feedback and student performance. 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 encapsulates multidisciplinary techniques under the name of Educational Data Mining. It consists in the application of Data Mining methods to educational data with the objective of resolving educational research questions.
In this project I attempt 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 3,715 users. Students had access to video lectures where theory and exercises were presented. The exercises were audio-visual programming assignments and once completed they had to be screen recorded, uploaded to Vimeo and then submitted to Music Circle for discussion among peer students. Music Circle logs all the mouse events of the users and from the log data it is possible to infer when feedback takes place. In order to classify students into academic performance from their feedback activities, the data from the MOOC assessments (final grades) and the logs in Music Circle is analysed with Data Mining algorithms.

1.3 Report structure

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 processes to build them.

Section 5 and section 6 contain the two main studies on the data: On one hand, a high level data analysis calculates statistics about the data. The aim of this study was to explore the data and present 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, an exploratory data analysis experiments with different feature vectors and Data Mining algorithms in order to find correlations between feedback activities and student academic performance.

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.

2 Background

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 (DM) 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, these 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 that 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 [16].

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 to 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.

According to Jiawei Han the Data Mining process contains the following steps [16]:

- Data cleaning: Removal of any noise or inconsistent data
- Data integration: Combination of multiple data sources
- 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 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 noisy, 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:

- Remove 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.

2.3 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 educational research [3]. "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" [12]. 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.
C. Romero and S. Ventura identify three main research areas in EDM [22]:

- 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 Adaptive 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 [15],[12],[22]. 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 [19].
Another interesting EDM task is to build a system 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.

2.4 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 raw 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.

Maia and Almeida [18] 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 patterns. 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.

2.5 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.
Baker [11] 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" [11].

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 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, Naïve 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 Naïve Bayes.

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 from [11]

2.6 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 [21] 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 us 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. In general, these publications analyse both quantitative or qualitative features independently but not in conjunction.
On his paper, Romero [21] 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 [5] 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 Maximisation) clustering algorithm provides the best result.

3 Data and databases

3.1 Music Circle

Music Circle is an educational social network developed within the PRAISE research project at the Goldsmiths University Computing Department [25]. 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 saw the dependancies between 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.

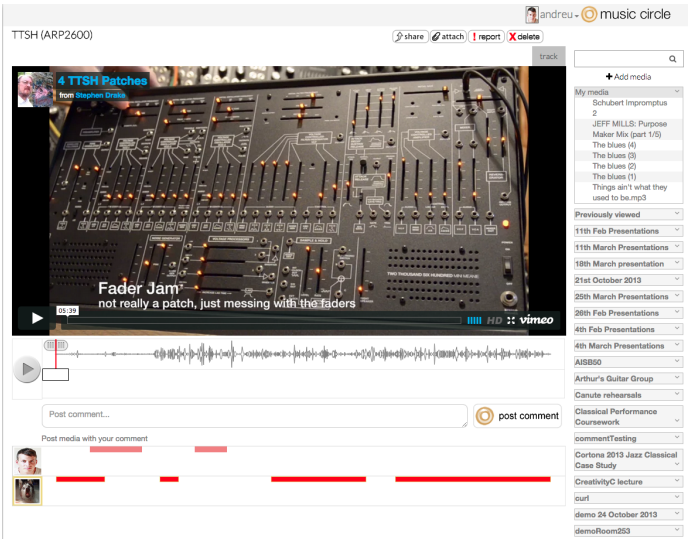


Figure 1: Music Circle web application

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.

3.2 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. Relations between tables are possible by storing the unique keys that relate to other tables. The relational model of data is very popular and many database technologies make use of it. The structure of the database is

defined on creation, therefore, relational databases are not not suitable for data that changes its structure frequently.

MusicCircle uses a MongoDB database. MongoDB is a document oriented database that provides high performance, high availability, and easy scalability [4]. 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 these 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 documents 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.

3.3 Databases

Three different databases were used for this project: Music Circle, Coursera and Research databases. Music Circle contains the data from the Music Circle system, Coursera contains the data from the MOOC that run on Coursera and Research contains a combination of data in Music Circle and Coursera.

3.4 Coursera

The Coursera database is a MySQL database that contains the full Coursera course. Of special interest are the tables that contain grades from students. These tables contain the final grades, as well as the multiple choice questionnaires and peer assessment results. Table 2 show the relevant tables on the Coursera database.

Table	**Description**
course_grades	Students final grades
hg_assessment_evaluation_metadata	Grades of individual assessments
hg_assessment_overall_evaluation_metadata	Summary of all the grades (final and peer grades)
quiz_submission_metadata	Multiple choice questionaries results

Table 2: Coursera database tables that contain students grades

3.5 Music Circle database

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. The database is 19.24 Gb in size and summary of its collections is described in Table 3.

Collection	**Description**
Activity	Replies
ActivityDefinition	Comments
ActivityDefinitionTagLinker	Comment tags
ActivityDefinitionTagLinker	Reply tags
ActivityDefinitionTagLinker	Comments 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
Session	User session data
Tag	Tags
TrackSet	Media items set
UserGroup	Group of users

Table 3: Music Circle collections with a brief description of the data they contain

3.6 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, in 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.
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 three collections: The Actors collection contains users and their actions, the Media collection contains all the media items on the system and the ActorGrades collection contains the users' final grades from the Coursera database.

```
    Actors: [
        {
            _id: element id,
            activities: [
                {
                    at: activity type
                    id: id of the element the action took place on,
                    ts: timestamp
                }
            ],
            sessions:[
             {
    logged_in': 1 if used was previously logged in,
    sessionID: session id,
    sessionStart: date at which the session started,
    user_id: user id
}
            ],
            idx: user id on the Music Circle database,
            name: name oft heuser,

        }
    ],
    ActorsGrades: [
         {
              _id: element id,
```

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

The possible values for 'at' are: 0=upload, 1=view, 2=comment, 3=reply, 4=login, 5=play, 6=log data (mouse events) whereas 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. This process is performed by script [A.1] and described in detail below:
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. Figure 2 illustrates the process.

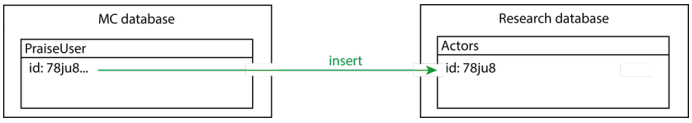


Figure 2: Import users into the research database

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. Figure 3 illustrates this process.

Figure 3: Import media items into the research database

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 item id it belongs to and the text of the comment. The user id of the owner of the comment is known and it is used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the activities array in the correct element on the Actors collection. Figure 4 illustrates this process.

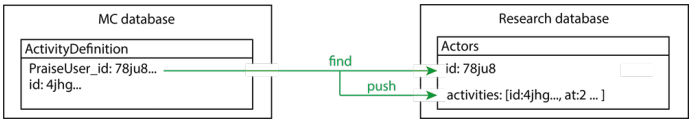


Figure 4: Import comments into the research database

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 it is used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the activities array in the correct element on the Actors collection. Figure 5 illustrates this process.

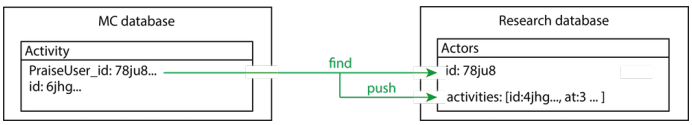


Figure 5: Import replies into the research database

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 it is used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the activities array in the correct element on the Actors collection. Figure 6 illustrates the process.
The Sessions collection is iterated and for each session a new object is built which

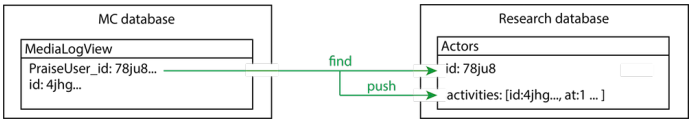


Figure 6: Import media view logs into the research database

contains the type of activity (logging in) and the timestamp when it happened. The user id of the owner of the reply is known and it is used to find the document for that particular user in the Actors collection. The object previously built is then pushed into the activities array in the correct element on the Actors collection. Figure 7 illustrates the process.

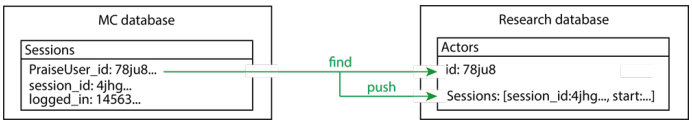


Figure 7: Import sessions into the research database

Finally, the log collection is iterated and its data is imported. Figure 8 illustrates the process.

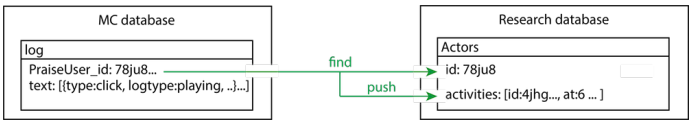


Figure 8: Import logs into the research database

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 it is not possible 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 ran on the machine for two 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 was not constant.
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 arrays and specially pushing elements at the end of them. Unfortunately, batch processing did not increase the computational 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.

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). For this to happen, 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.

3.6.1 Scaling of distributed systems

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.
Mongo response to distributed systems is sharding also known as horizontal scaling. It consists in dividing and distributing 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 contains these elements: Query routers, config servers and shards (see Figure 9).
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 one 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 the 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 is controlled with 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 split 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 built for the Research database had four shards that allowed the four CPUs of the machine to be used simultaneously. Importing the log data using a sharded cluster took 12 hours.

The process to import the log collection is described here: 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 partition of time (usually two weeks). 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 it is used to find the element for that particular user in the Actors collection. The object previously built is then pushed into the activities array in the correct element on the Actors collection.
With this last step the research database is built and ready to use. Scripts A.2, A.3, A.4, A.5 implement the importing methods that build the research database.

Figure 9 shows how the Music Circle and Research databases are connected. It also shows the sharded cluster of the Research database.

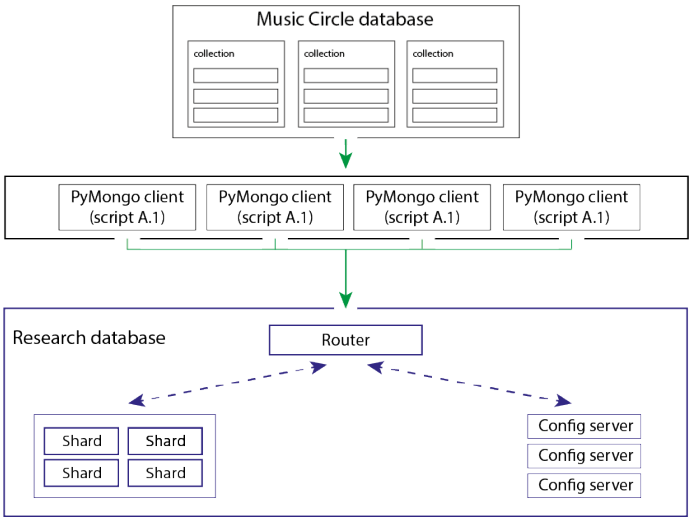


Figure 9: Data flow between the Music Circle and Research database showing a Mongo sharded cluster

Collection	Description	Number of items
Actors	Music Circle users	3715
Sessions	Music Circle users sessions	11350
Media	Media items uploaded	2897
Final Grades	Final grades from students	50300
Final Grades in Research database	Students in the Research database that have grades	1852
Total activities	Total activities in Music Circle	8326972
Comment	Comments posted	7370
Community tracks views	Number of community tracks views	351541
Owned tracks views	Number of views on owned tracks	9975
Comments views	Number of comment views	30351
Plays	Number of plays	36936
Replies	Replies posted	978

Table 4: Summary of the data in the Research database

4 High level data analysis

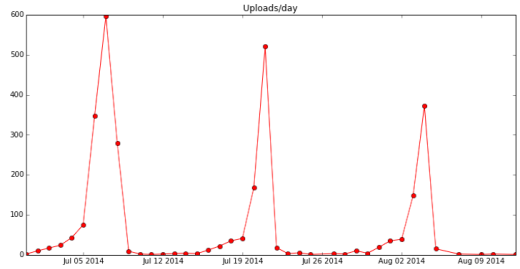
4.1 Aim

The aim of this study was to generate high level statistics. 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.

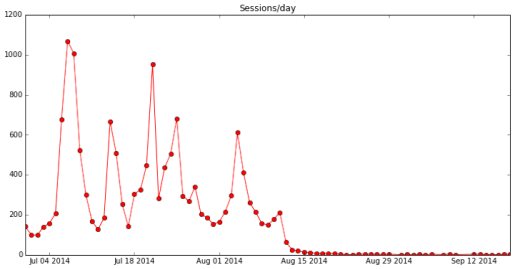
4.2 Methods

The first step was to explore the Music Circle database by showing the activities per day. Script [A.6] iterates through various collections in the Music Circle database and counts how many activities happened per day. Figure 10 show the plots generated. The spikes in the plots correspond to a surge in activity due to the approach of assessment dates.

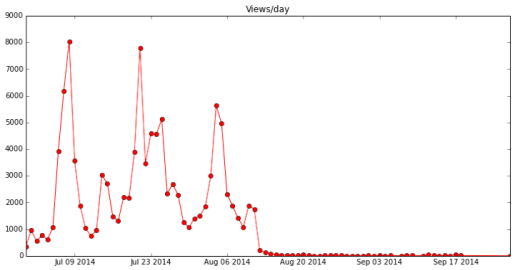
After these initial statistics, the data in the Research database was explored. A necessary step was to clean the data as much as possible. After trying to implement initial statistics I realised that some user events were logged more than once. Those duplicates had to be removed since they would interfere with the results. Script [A.7] 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.



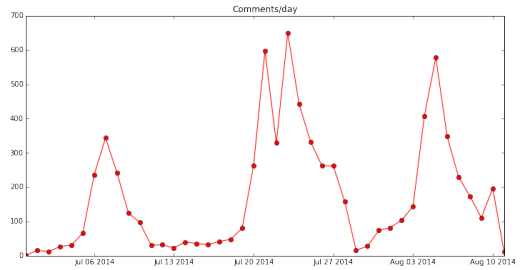
(a) Uploads per day



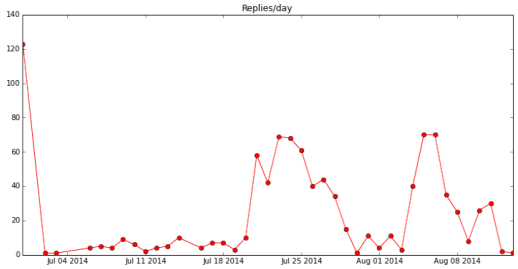
(b) Sessions per day



(c) Views per day



(d) Comments per day



(e) Replies per day

Figure 10: Plots showing activity level over time

As said 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.12] that added some information into the Actors collection elements about the students' level of activity was implemented, specially since Mongo query modifiers can not operate array length queries.

After this data cleaning step it was time to link the Research database with the Coursera database that contains the ground truth (final grades of students). The process of linking the two databases consists in matching user ids in order to connect users in the Research database with final grades in the Coursera database. Figure 11 shows the matching process. 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 being able to link both databases this project could not be completed. The script that solves this issue is [A.15] 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 achieved from the Coursera database and the total number of activities performed by the user.
Not all the users in the Coursera database could be matched to the Research database because some students either did not finish the course or did not upload media to Music Circle. A total of 1,852 students in the Research database have a final grade.

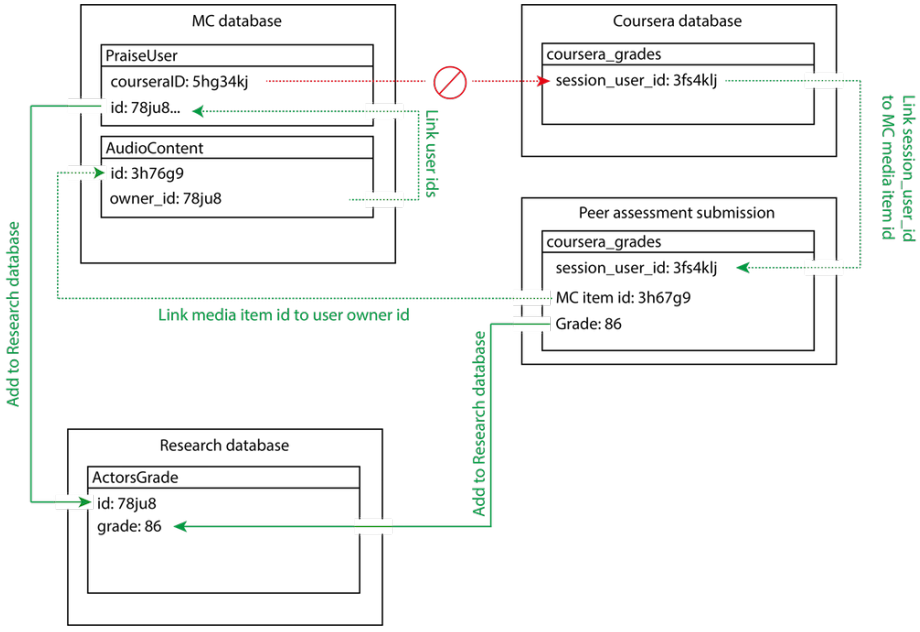
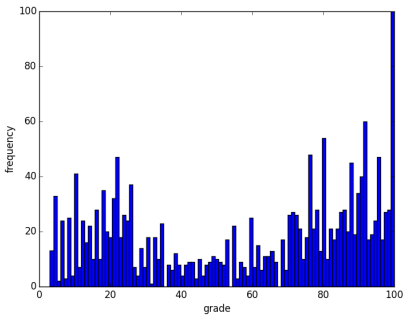
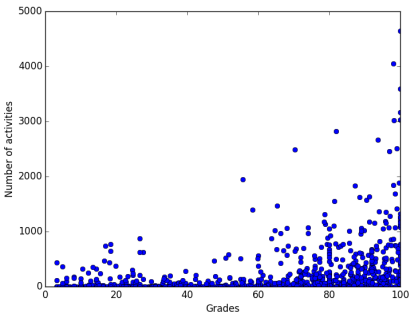


Figure 11: Linking Music Circle, Coursera and Research databases

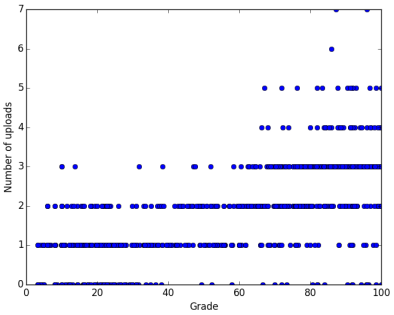
The final step was to generate plots comparing activity levels to final grades. Figure 12 show the plots generated by script [A.16].



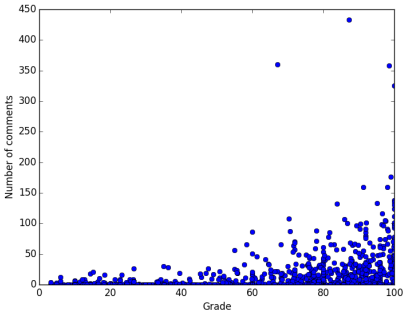
(a) Distribution of grades



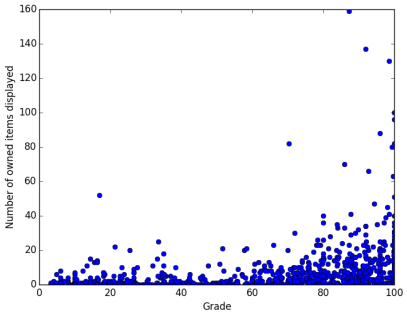
(b) Total activities vs grades



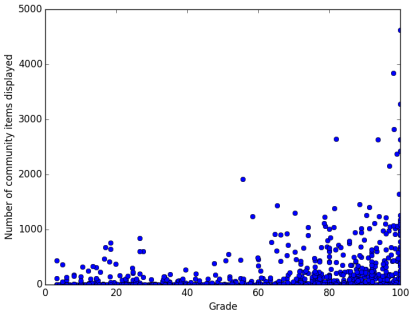
(c) Uploads vs grades



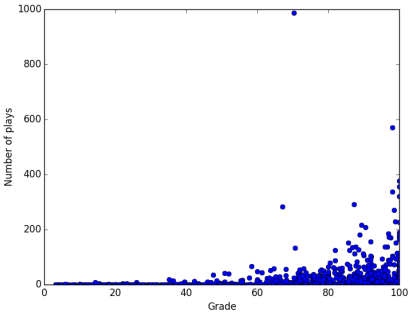
(d) Comment views vs grades



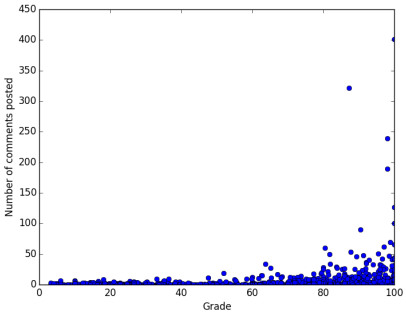
(e) Owned track views vs grades



(f) Community track views vs grades



(g) Number of plays vs grades



(h) Comments posted vs grades

Figure 12: Correlation between activity levels and grades

4.3 Conclusion

This study had different aims: To complete the research database, to explore the data and to present 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 from the linear regressions support this by showing a coefficient of correlation close to 0. The results are presented in Table 5. Trying to fit non linear functions might improve the results. The data does not show any signs of a defined probability distribution, making it difficult to decide what the next steps are.

Activity	r^2
Uploads	0.5067
Comments views	0.1576
Owned tracks views	0.1211
Community tracks views	0.1034
Plays	0.1154
Comments posted	0.1364

Table 5: Results of linear regressions on the different statistics

5 Exploratory data analysis

5.1 Aim

This section presents three studies that explore user behaviour using indictors of feedback activity. Section 5.2 presents a method to construct feature vectors that allow for user differentiation. Section 5.3 presents a method to classify students based on their feedback activity in Music Circle. Section 5.4 contains a higher resolution analysis of the user activity after feedback. Finally, Section 5.5 contains a summary of the results of the three studies.

5.2 Difference on feedback activity between users

The method presented in Section 4 failed at finding correlations between student feedback activity and student performance. This study proposes to consider the relations between the different feedback activities in order to find similarities between users. In order to do this, the first step is to construct a data structure that allows a multidimensional comparison of students considering all the different feedback activities. This is achieved by building a feature vector that contains feedback activity information. Each student has associated a feature vector that contains information about the feedback activities of that particular student. Once built, these vectors allow us to use Euclidian distance to make comparisons between students.

For this study, the feature vector had this form:

$$v_n = [u, c, m, g, p] \tag{1}$$

Where:

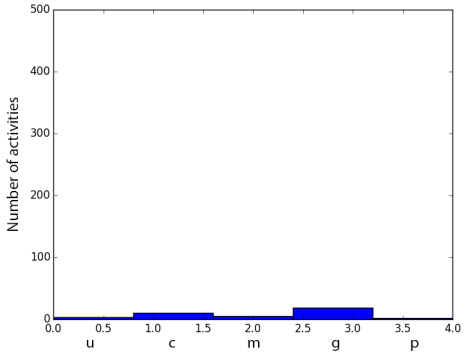
- n = user
- u = total number of uploads for user n
- c = total number of comment views for user n
- m = total number of media views for user n
- g = total number of community media views for user n
- p = total number of plays for user n

The Research database contains the data used to calculate the different components of the feature vectors. Each element in the Actors collection contains information about the activity of a particular student using Music Circle. We consider feedback activities the ones that present information to the students, in Music Circle those activities are triggered by mouse clicks events. Each component of the feature vector contains the total number of clicks on different elements on the Music Circle interface. For example, the p component is calculated counting how many times a user clicked on the play button. The period of time considered is the total length of the case study. The use of cumulative measures of events is suggested by Maia [18]. Moreover, this feature vector is intuitive and easy to calculate.

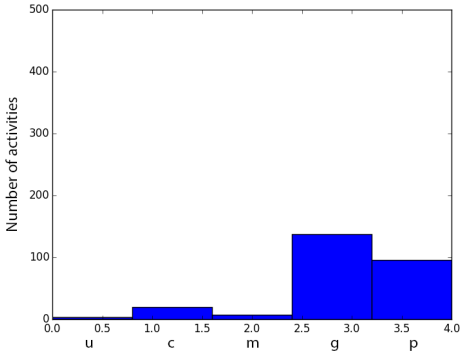
Users	Feature Vectors
A	[2, 25, 30, 89, 23]
B	[0, 7, 0, 737, 7]
C	[1, 0, 1, 102, 19]
D	[0, 77, 7, 0, 59]

Table 6: Examples of feature vectors

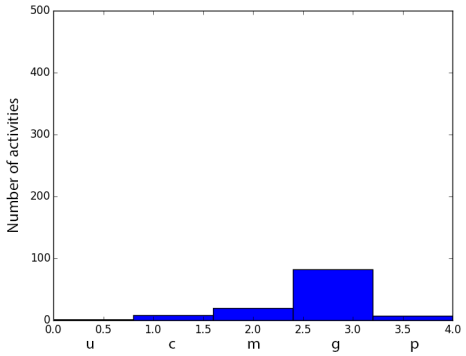
Table 6 contains examples of feature vectors calculated.
Once the feature vectors were built, they were plotted in order to visually inspect them. Script [A.8] calculates the feature vectors, generates histograms for each of the feature vectors and saves them as a .png file. Only users with more than 100 activities logged are considered which produced 664 plots. Once the .png images were generated, a video of them was created using FFMPEG [2]. The video was made in order to be able to watch all the images in a short period of time, the hope was to visually detect similarities in the data which would indicate similar user behaviour. The video can be found here: `http://youtu.be/cnBnJAWI9ro`. Unfortunately, no obvious patterns were detected. Figure 13 shows several histograms corresponding to different users.



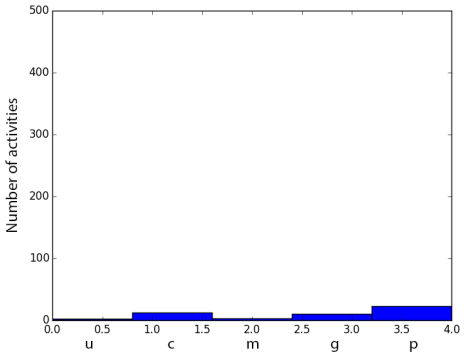
(a) Histogram of the feature vector



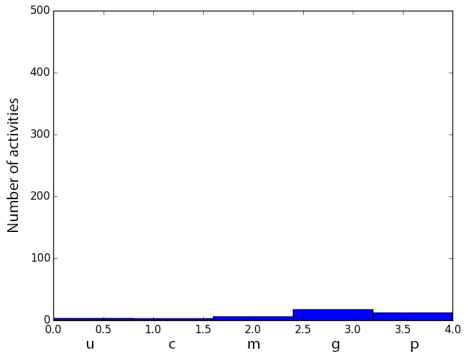
(b) Histogram of the feature vector



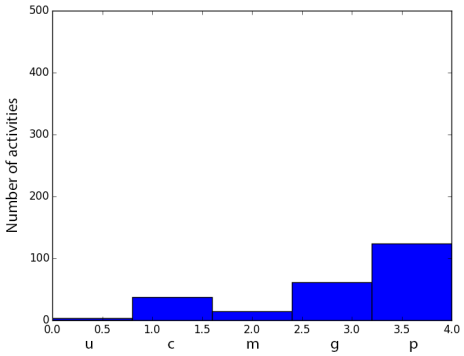
(c) Histogram of the feature vector



(d) Histogram of the feature vector



(e) Histogram of the feature vector



(f) Histogram of the feature vector

Figure 13: Feature vector histograms for different users

The video generated provides a high level overview of the feature vectors. For example, it shows that viewing group media items and playing media items are popular activities between all users.
Something that was unknown at this stage was whether the feature vectors were suited to differentiate users. The video shows how the different feature vectors change, proving that in general different users had different vectors. However, it had to be proven that these differences were enough to differentiate between users.

In order to provide with analytical data about the differences between users a difference matrix was calculated. To do so, the Euclidian distance from each feature vector to all the others is calculated and stored in a matrix. Each row of the matrix contain the distances between a unique feature vector and all the others.
After a close inspection of the data used to construct the feature vectors it was discovered that some of the events logged were very close in time with less than 50ms between them. Those events are the result of 'double clicks' or other artefacts, they do not add any valuable information and it was decided to remove them from the dataset. Additionally, the feature vectors were normalised so that their length would always be 1. This removes dependancies in the number of events performed by the students and adds information about the relationships (and proportions) of each activity with respect to the other activities. Script [A.9] calculates the feature vectors and stores them into a JSON file. There is one feature vector per user. Script [A.10] calculates the Euclidian distance from each feature vector to all the others and it stores it in a matrix. Figure 14 shows a heat map of the distance matrix, it is generated by script [A.11].

In the heat map, clear areas indicate similarity (small distances) whereas dark areas indicate dissimilarity (large distances). It is a proof that the feature vectors calculated can distinguish between users.

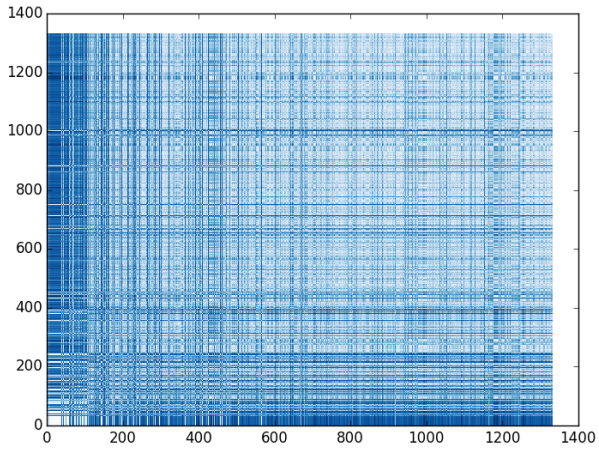


Figure 14: Heat map of the distance matrix

5.3 K-Nearest neighbour classification

The previous study shows how the feature vectors can differentiate between users. This section proposes a method to classify students into performance categories based on the feedback activity. The method chosen for the classification is a k-nearest neighbour algorithm.

Given a dataset D and a point $p \in D$, the k-Nearest Neighbour (KNN) algorithm finds the k closest points to p where $k \in D$. It is a well known data mining method that can be used for classification and regression.
For classification, the algorithm consists in associating a class C to each of the points in D. It calculates the distance from p to all the points in D. The calculated distances are sorted in a decreasing order and the smallest k are chosen. Each of the distances is associated with a point p' and a class C, the majority of classes is chosen as the class of p.
The KNN implementation is provided by *scikit-learn* [8] which is a machine learning library for Python which contains tools for data mining and data analysis. It is built on Numpy, SciPy and matplotlib, making it fully compatible with Python scientific programming tools.

As in the previous study. Feature vectors that describe feedback activity need to be built. Each element in the Actors collection contains information about the activity of a particular student using Music Circle. We consider feedback activities the ones that present information to the students, in Music Circle those activities are triggered by mouse clicks events. Each component of the feature vector contains the total number of clicks on different elements on the Music Circle interface. For example, the u component is calculated counting the total number of uploads of a user. The period of time considered is the total length of the case study. The feature vectors constructed are very similar to the ones used in Section 5.2 but add an extra component that measures the total number of comments posted. For this study, the feature vector had this form:

$$v_n = [u, c, m, g, p, cm] \tag{2}$$

Where:

- n = user
- u = total number of uploads for user n
- c = total number of comment views for user n
- m = total number of media views for user n
- g = total number of community media views for user n
- p = total number of plays for user n
- cm = total number of comments posted for user n

The different components are calculated considering the full length of the case study. Table 7 contains examples of feature vectors calculated.

Users	Feature Vectors
A	[0.0435, 0.1159, 0.1014, 0.1884, 0.3043, 0.2464]
B	[0.0857, 0.1429, 0.3714, 0.2571, 0.1143, 0.0286]
C	[0.0385, 0.1154, 0.2115, 0.5384, 0.0000, 0.0962]
D	[0.0789, 0.0000, 0.47367, 0.2632, 0.1579, 0.0263]

Table 7: Examples of feature vectors

Script A.17 trains a KNN algorithm to classify students. The first step is to generate the classes and assign them to the students. Three classes are considered:

- Students that achieved < 50 on their final grade
- Students that achieved between 50 and 70 on their final grade
- Students that achieved $>= 70$ on their final grade

In order to use KKN the value of k needs to be specified. In order to choose the best value for k one must try different k and choose the one that provides better results. Therefore, one needs a method to validate the performance of the algorithm. The method chosen to measure the performance of the classification is commonly referred as cross-validation.

In order to cross-validate the KKN algorithm, the dataset is split into a training and test dataset in a 90/10 ratio. The training dataset is used to train the KKN algorithm whereas the test dataset is used to test its performance. The performance metric used to determine the optimal k is the accuracy of the classification. As seen in section 4, it was not possible to obtain the final grade of all the users. Moreover, both datasets contain a randomised selection of the original dataset, therefore, each time the algorithm is run different results on the accuracy of the algorithm are expected.

The algorithm was run for a range of k varying from k=2 to k=80, for each k the algorithm was run 100 times and the accuracy of the classification was calculated. Figure 15 presents the results of the accuracy versus k, each point consists in the mean accuracy value, the error bars consist in the range of accuracies calculated for a particular k. The greatest accuracy is obtained for k=24 with a value of a=0.7980. Once the optimal k was determined other performance metrics were calculated. Accuracy measures the proportion of corrected classified instances. Precision measures the fraction of instances that are relevant. Recall measures the fraction of relevant instances that are retrieved. Finally, F1 is the harmonic mean between the precision and the recall. Table 16 shows all the performance metrics calculated.

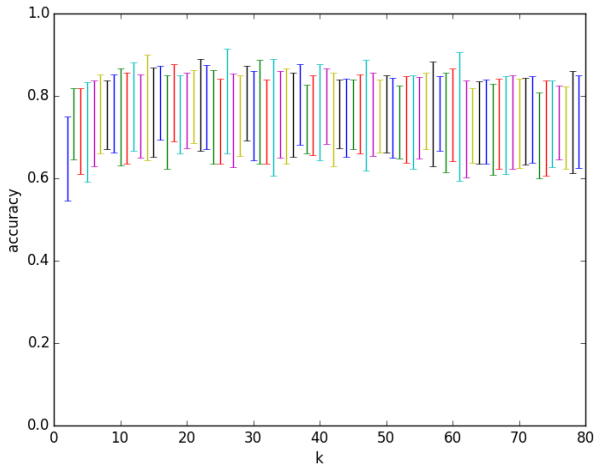


Figure 15: Accuracy of the classification using KNN with cross-validation

Metric	**Value**
Accuracy	0.7980
Precision	0.7300
Recall	0.8000
F1	0.7600

Table 8: Performance metrics for the KNN classification

5.4 In-depth analysis of feedback activity

The aim of this study was to analyse the feedback activity with a higher resolution with respect to the studies described in sections 5.2 and 5.3.
An interesting question is to consider user activity just after feedback. In order to do so, a new method to construct feature vectors is needed. This new method will have to solve the difficult question of determining for how long a particular feedback action has effect.

In this study it is assumed that the effects of feedback last through a whole user session. A session is the period of time between the login of the user and the logout on Music Circle. Script [A.13] 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 session 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. Figure 16 illustrates this process.

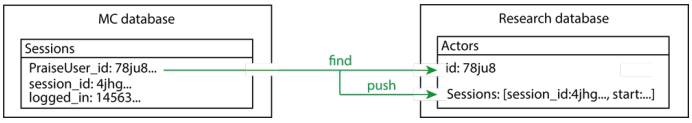


Figure 16: Import sessions into the research database

The process of building the feature vectors is similar to the one followed on sections 5.2 and 5.3. The difference is that on this study each feature vector describes the user behaviour on one session as opposed as the whole case study. Therefore, each user has a set of feature vectors (one per session) instead of just one that considers the whole MOOC.

For this study, the feature vector had this form:

$$v_n = [u, c, m, g, p] \tag{3}$$

Where:

- n = user
- u = total number of uploads for user n
- c = total number of comment views for user n
- m = total number of media views for user n
- g = total number of community media views for user n
- p = total number of plays for user n

The different components are calculated considering the full length of a session. Table 9 contains examples of feature vectors calculated.

Users	Set of feature vectors
A	[0, 5, 0, 958, 5],[1, 1, 0, 0, 4]...
B	[0, 0, 0, 0, 0],[0, 21, 4, 468, 14]...
C	[0, 0, 1, 0, 0], [0, 0, 0, 8, 0]...
D	[0, 82, 1, 530, 23], [0, 25, 1, 123, 16]...

Table 9: Examples of feature vectors

Script [A.14] builds the feature vectors. The first step is to make a list of objects that contain the start and the end date of the 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 clicks on comments are per session. 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.

As in sections 5.2 and 5.3, the feature vectors calculated need to be validated in order to prove that they can differentiate between users. The validation method considers the fact that there is a set of feature vectors for each user and assumes that each of those sets can characterise a particular user. These different sets of feature vectors form the ground truth, one can think of each set as a cluster that describe a particular student behaviour. If the feature vectors describe users appropriately, 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 ground truth. The clustering generated can be compared to the ground truth in order to measure the quality 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" [16].

The k-means algorithm takes k as a 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

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

Script A.18 is responsible for the clustering of the feature vectors. The clustering algorithm is provided by *scikit-learn* [8]. A k-means clustering with k equal to the number of users is performed on all the feature vectors. Figure 17 shows the result of the clustering generated whereas Figure 18 shows the ground truth clustering. From a simple observation it is obvious that the two clusterings are different. However, in order to prove how different they are, similarity metrics are needed.

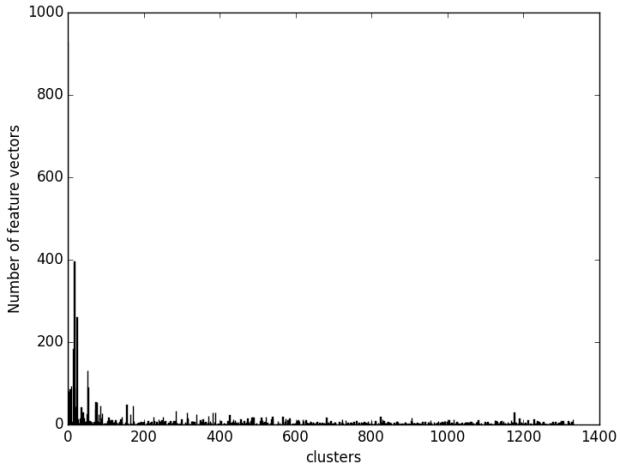


Figure 17: Clustering generated showing number of feature vectors per cluster. Each cluster represents a particular user.

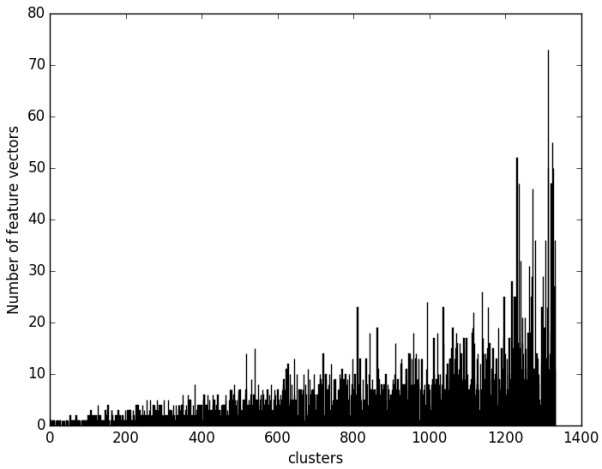


Figure 18: Ground truth clustering. Data structure generated when calculating the feature vectors. Each user contains a set of feature vectors (cluster) that represent their behaviour.

The next step was to compare the ground truth clustering with the generated clustering. Script A.19 performed 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 [8]. 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 whereas 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 calculated clustering produces the results summarised in Table 10.

Metric	Value
Adjusted random index	0.0004
Adjusted mutual Information based scores	0.0045
Homogeneity	0.5380
Completeness	0.6433
V-measure	0.5860

Table 10: Performance metrics for the k-Means clustering

The performance measures show that the clusters produced by the algorithm are very similar to a random clustering process.

The next step was to consider a dimensional reduction technique in order to try to improve the results. Script [A.20] 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. The variances of the components in a decreasing order are: [0.3638, 0.2687, 0.2113, 0.1270, 0.0292]. The feature vectors are transformed into a space with 2 dimensions and the k-Means clustering is performed. Table 11 show the results obtained by using the same measures as before.

Metric	Value
Adjusted random index	0.0012
Adjusted mutual Information based scores	0.0110
Homogeneity	0.4799
Completeness	0.6299
V-measure	0.5436

Table 11: Performance metrics for the k-Means clustering

Figure 19 shows the clustering after performing PCA on the data. The performance measures show that the PCA dimensional reduction does not improve the quality of the clustering.

Given that the performance metrics are poor, the conclusion of this study is that the calculated feature vectors do not allow to differentiate between users.

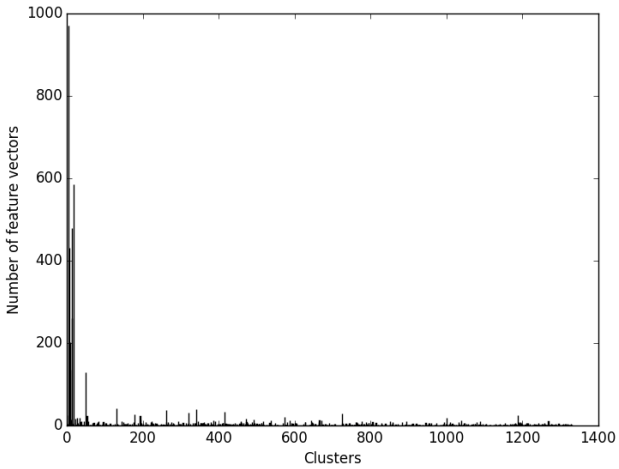


Figure 19: Clustering after PCA

5.5 Conclusion

This section presented different explorations of the data. Feature vectors and data mining methods are explored in order to analyse student feedback activity on Music Circle and their academic performance.

Section 5.2 presents a method to construct feature vectors and visualise them. A five dimensional feature vector that contains feedback activity indicators is constructed. The visualisation in a video of all the feature vectors shows that in general feature vectors from different students are different, meaning that they can be used to differentiate between students.
Additionally, a difference matrix of the feature vectors is calculated. A heat map of the difference matrix is plotted confirming that the feature vectors can be used to make comparisons between students.

Section 5.3 presents a classification of students using k-nearest neighbour (KNN) algorithm. The feature vectors used to train the KNN algorithm are very similar to the previous ones but they add a new component measuring the total number of comments posted per user. In order to determine the optimal value of k, KNN is performed for k between $2 <= k <= 80$ and validated using cross-validation. The optimal performance of the algorithm is achieved using k=24 with an accuracy of 0.7980. Additional performance metrics calculated for $k = 24$ are presented in Table 12.

Metric	Value
Accuracy	0.7980
Precision	0.7300
Recall	0.8000
F1	0.7600

Table 12: Performance metrics for the KNN classification

The KNN performance values are high meaning that this method can accurately classify a student into a performance category using log data collected from an educational social network.

Section 5.4 presents a fine resolution study of activity after feedback. Feature vectors are constructed considering activities on each session, therefore, each student has a set of feature vectors associated.
A data structure is constructed containing sets of feature vectors corresponding to different students and it is considered the ground truth. In order to prove that those sets can characterise students a k-Means clustering is proposed. The hypothesis is that if the sets of feature vectors characterise users a k-Means clustering algorithm with k equal to the number of users will produce a similar clustering as the ground truth calculated. The performance metrics of the clustering process are poor, Table 13 contains a summary of those metrics. The performance of the k-Means clustering is poor.

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

Table 13: Examples of feature vectors

There is no difference between the clustering performed and a random clustering.
In order to try to improve the results a PCA is performed on the feature vectors. Table 14 shows the performance metrics of the clustering after the PCA dimensional reduction. Unfortunately, the results do not improve. The conclusion of this study is that the sets of feature vectors do not represent user behaviour and therefore are not valid to do student comparisons.

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

Table 14: Examples of feature vectors

6 Self-reflection

This section contains a self-reflection on the project as a whole. It contains a discussion about the difficulties encountered, as well as a summary of the results obtained and planned future work.

The initial project proposal considered to explore the fields of Data Mining, Machine Learning and Neural Networks to try to provide answers to questions about education. The data was to be obtained from an educational social network called Music Circle and it consisted in the students events log using the system. Machine learning algorithms were to be applied to this data. Additionally, it proposed to use Digital Signal Processing methods as a data analysis tool. Since the log events depend on time it was possible to construct a signal from the data.

The project explores how Data Mining can be applied to the dataset and it finds a method (k-Nearest Neighbour) that classifies students into academic performance with high accuracy. The Neural Network applications were not explored and the DSP perspective was only considered in the very early stages of the project. The main reason for not fulfilling completely the initial project proposal was a lack of time due to problems encountered setting the environment and also the difficulties in analysing the data. From the very beginning the complexities of the project manifested, these difficulties were both technical and theoretical.

The technical difficulties were related with setting up the development environment. Most of the technologies used were completely new to me, therefore, I had to learn about them. The development environment consisted in Python and Mongo running in a virtual environment [10]. I chose to work in iPython notebooks because of its friendly IDE. However, iPython performance is bad when performing large printing operations which is a technique that I consider crucial for debugging. When iPython needs to print lots of messages on the console, it crashes and corrupts the file. Therefore, all the code is lost. After loosing a few of the scripts I decided to stop using iPython and run the Python scripts from the command line.
Another major difficulty in setting up the environment was to import the Research database. The importing process was slow and it took a lot of time to find a way to increase the computational speed. By using a Mongo distributed system the computa-

tional speed increased by at least a factor of 4. However, the learning and setting up of the Mongo distributed system was time consuming.

Once the environment was set up it was time to start the data analysis. The manipulation of the data was difficult due to their size and it was often complicated to test that the operations performed were correct. In this sense, a test driven development approach would have been helpful. Additionally, as with any real world dataset the data contained noise and inconsistencies that needed to be removed. This caused bugs and anomalous results that were often difficult to debug.

The theoretical problems were difficult to approach and they were related with research being at the core of this project. For example, it was clear to me what was the goal of the project but the exact path to achieve it was unclear. The time nature of the data made it difficult to manage, especially in the case of considering activity after feedback. One of the unresolved questions in this project is to determine when the consequences of feedback start and end. I made the assumption that the activity on each session was the result of an initial feedback event at the start of that session. This assumption simplified the problem but it did not solve it. In order to find a solution I should have found a differentiation between common activities and those that initiate and finalise feedback, this is a difficult task.

Four studies on the data are presented. Section 4 explores the data in the Music Circle and Research database by building plots of the system activity over time and calculating statistics. In particular, linear regressions are fitted into the data, the results of the linear fitting are summarised in Table 15. The value of the coefficient of determination (r^2) is low which indicates that the lines do not fit the data well.

Activity	r^2
Uploads	0.5067
Comments views	0.1576
Owned tracks views	0.1211
Community tracks views	0.1034
Plays	0.1154
Comments posted	0.1364

Table 15: Results of linear regressions on the different statistics

Section 5 presents a deeper analysis of the data by constructing different feature vectors and applying Data Mining techniques to the student classification problem. The method to build the feature vectors consisted in cumulative measures of feedback activity, a technique suggested by Maia's paper [18]. Sections 5.2 and 5.3 consider feature vectors built over the whole length of the case study yielding one feature vector per user. On section 5.2 methods that explore the ability of the feature vectors in differentiating users are explored. On section 5.3 a method to classify students into three classes of performance is explored. It trains a k-Nearest Neighbour algorithm to classify student

performance into three categories from feature vectors that contain feedback activity indicators. The algorithm is validated using cross-validation, producing an accuracy of 0.7980. Table 16 summarises the performance metrics of the algorithm.

Metric	**Value**
Accuracy	0.7980
Precision	0.7300
Recall	0.8000
F1	0.7600

Table 16: Performance metrics for the KNN classification

Section 5.4 considers a higher resolution analysis of the data. The method to build the feature vectors considers the length of sessions yielding one feature vector per session. Therefore, each user has associated a set of feature vectors that contain feedback activity on the different sessions. A method to test how well the sets of feature vectors represents users is presented. All the feature vectors are fed into a k-Means clustering algorithm with k = total number of users. The hypothesis made is that if the sets of feature vectors represent users, then the k-Means algorithm will produce clusters that are similar to the sets of feature vectors calculated. Table 17 presents the results. The clustering produced is similar to a random cluster, therefore, the conclusion is that the set of feature vectors are not useful in representing users.
In order to try to improve the results, a PCA dimensional reduction is performed and the same clustering method applied. The results are shown in Table 18, again the clustering algorithm is not able to produce a good clustering. This results indicate that the feature vectors calculated do not allow the differentiation of users.

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

Table 17: Clustering performance metrics

6.1 Future Work

The future work is focused on fully exploiting the data and the techniques exposed in this report.
The data from the Coursera database was not fully explored. Particularly, the grades from the multiple choice questionnaire were not used. This data provides a rich insight in the learning process of students. Moreover, it can be used to track the learning process and measure the contribution of feedback on that learning.

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

Table 18: Clustering performance metrics after PCA

In this project, only two Data Mining algorithms were used which is a low number given the variety of algorithms in the machine learning field. Neural Networks seem to have a lot of potential because of their ability to model highly complex data.

Feature vectors are the most important element in this project. They distil important information about students and allow the Data Mining algorithms to produce useful results. New feature vectors that consider the frequency of events instead of the number of them are of special interest. Such feature vectors would allow to remove the explicit dependancy on time which might allow for better comparisons between users. Additionally, this approach allows us to think about the spectrum of student behaviour which relates to the DSP approach exposed in the project proposal.

References

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

[2] FFMPEG. https://www.ffmpeg.org/. Accessed: 2015-03-19.

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

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

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

[6] National student survey. http://www.thestudentsurvey.com/. Accessed: 2015-03-19.

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

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

[9] 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.

[10] Virtual Environment. http://docs.python-guide.org/en/latest/dev/virtualenvs/. Accessed: 2015-02-20.

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

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

[13] 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.

[14] John Biggs. Constructive Alignment in Action : Imaginative Curriculum Symposium Monday 4 th November 2002 ALIGNING THE CURRICULUM TO PROMOTE GOOD LEARNING. *Most*, (November):1–7, 2002.

[15] 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.

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

[17] Helen Timperley John Hattie. The power of feedback. [References]. *Review of Educational Research*, .77:16–7, 2007.

[18] 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.

[19] 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.

[20] Arkalgud Ramaprasad. On the definition of feedback. *Behavioral Science*, 28(1):4–13, 1983.

[21] 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.

[22] 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.

[23] D. Royce Sadler. Formative assessment and the design of instructional systems. *Instructional Science*, 5(1):52909, 1998.

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

[25] Matthew Yee-king, Maria Krivenski, and Harry Brenton. Designing educational social machines for effective feedback. pages 1–13.

A Code

A.1 Build the research database (by Chris Kiefer)

```
# All the code by Chris Kiffer

%pylab inline
from pymongo import *
from bson 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

class mediaFormats:
    VIDEO = 0
    AUDIO = 1

class Activities:
    UPLOADED = 0
    VIEWED = 1
    COMMENTED = 2
    REPLIED = 3
    LOGGEDIN=4
    PLAY=5

def importCourseraData():
    actorIdx = 0 #numerical id
    ignoreTheseUsers = [ObjectId('53b16012a66e44472a7e7d72'),ObjectId('53b160c4a66e444cbca4c2f4'),ObjectId('53b16feba66e444dff9c7c8e')]
    users = db.PraiseUser.find({'_id':{'$nin':ignoreTheseUsers}})
    #ir.Actors.remove({})
    for u in users:
        ir.Actors.insert({'_id':u['_id'],"name":u['display_name'],'activities':[], 'idx':actorIdx })
        actorIdx = actorIdx + 1
    print "imported actors"
```

```
39
    media = db.AudioContent.find({'PraiseUser_id':{'$nin':
    ignoreTheseUsers}})
41  ir.Media.remove({})
    for m in media:
43      ir.Media.insert({"_id":m['_id'], 'owner':m['PraiseUser_id'
    ], 'fmt':mediaFormats.VIDEO,'title':m['title'], 'ts':m['datetime
    ']})
        uploadActivity = {'at':Activities.UPLOADED, 'ts':m['
    datetime'], 'id':m['_id']}
45      ir.Actors.update({'_id':m['PraiseUser_id']}, {'$push':{'
    activities':uploadActivity}})
    print "imported media"
47
    comments = db.ActivityDefinition.find({'PraiseUser_id':{'$nin':
    ignoreTheseUsers}})
49  for cm in comments:
        commentActivity = {'at':Activities.COMMENTED, '_id':cm['_id
    '], 'ts':cm['datetime'], 'media_id':cm['AudioContent_id'], 'text
    ':cm['title']}
51      ir.Actors.update({'_id':cm['PraiseUser_id']}, {'$push':{'
    activities':commentActivity}})
    print "imported comments"
53
    replies = db.Activity.find({'PraiseUser_id':{'$nin':
    ignoreTheseUsers}})
55  for rp in replies:
        replyActivity = {'at':Activities.REPLIED, '_id':rp['_id'],
    'ts':rp['datetime'], 'cmt_id':rp['ActivityDefinition_id'], 'text
    ':rp['content']}
57      ir.Actors.update({'_id':cm['PraiseUser_id']}, {'$push':{'
    activities':replyActivity}})
    print "imported replies"
59
    mvlog = db.MediaViewLog.find({'PraiseUser_id':{'$nin':
    ignoreTheseUsers}})
61  for item in mvlog:
        viewActivity = {'at':Activities.VIEWED, 'ts':item['datetime
    '], 'id':item['AudioContent_id']}
63      ir.Actors.update({'_id':item['PraiseUser_id']}, {'$push':{'
    activities':viewActivity}})
    print "Imported media view log"
65
    sessions = db.Session.find({'user_id':{'$nin':ignoreTheseUsers
    }})
67  for item in sessions:
         loginActivity = {'at':Activities.LOGGEDIN, 'ts':item['
    sessionStart']}
69       ir.Actors.update({'_id':item['user_id']}, {'$push':{'
    activities':loginActivity}})
```

```
    print "Imported sessions"

importCourseraData()
print "import complete!"
```

./pythonScripts/importer.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, 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.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, 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):
55                      activities.append(tempActivity)
                        userFound.update({'$push':{'activities':
    tempActivity}})
57          if(len(activities)>0):
                try:
59                  bulk.execute()
                except InvalidOperation as invOpt:
61                  pass
```

./pythonScripts/importShard2.py

A.4 Build the research database

```
1  from bson import *
   from pymongo import *
3  import time
   from datetime import timedelta
5  from pylab import *


7
   import json
9  from pymongo.errors import InvalidOperation

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

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

23 # generate a list of dates
   def dateListGenerator(start, end, delta):
25      result=[]
        curr = start
27      while curr < end:
            curr += delta
29          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.5 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}})
        if(len(activities)>0):
            try:
                bulk.execute()
            except InvalidOperation as invOpt:
                pass
```

./pythonScripts/importShard4.py

A.6 Plot activities per day

```
%pylab inline
%matplotlib inline
from bson import *
from pymongo import *
import time
from datetime import timedelta
from pylab 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:27018")
#ir = clir.mus0IR

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

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):
        count=0
        for item in self.preparedList:
            count=count+item["freq"]
        if count==self.data.count():
            print "data is OK"
        else:
            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.7 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.8 Histogram of the feature vectors for all users

```
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
#%pylab inline
#from IPython import display
```

```
15
#source db
17 #mongod --dbpath /Users/chris/expts/praise/dataAnalysis/mongo/db --
    fork --syslog
#mongod --dbpath /Users/matthew/Documents/PRAISE_local/data/
    musiccircle_latest/musiccircle
19 #cli = MongoClient("localhost:27017")
#db = cli.musiccircle
21
#target instance - intermediate representation
23 #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")
25 clir = MongoClient("localhost:27026")
ir = clir.research
27
# get first of very active users
29 #tenVeryActiveUsers=ir.Actors.find({"$and":[{"$where":"this.
    activities.length >49000"},{"activities.at":{"$eq":6}}]})
tenVeryActiveUsers=ir.Actors.find({"$and":[{"$where":"this.
    activities.length >100"},{"activities.at":{"$eq":6}},{"
    activities.at":{"$eq":0}}]},timeout=False)
31
# this finds feedback triggers (clicks)
33 feedbackTriggers=[]
#fig = plt.figure()
35 theHistogram=[]
#plt.ion()
37 #plt.show()
nUser=0
39 numberOfUsers=tenVeryActiveUsers.count()
for user in tenVeryActiveUsers:
41     feedbackTriggers=[]
    theHistogram=[]
43     del feedbackTriggers[:]
    del theHistogram[:]
45     nUser=nUser+1
    print numberOfUsers-nUser," left"
47
    for activity in user["activities"]:
49             if activity['at']==0:
                theHistogram.append(0)
51             # get click events ignore playing logs
             if 'type' in activity.keys() and activity['type']=='
    click' and activity["logtype"]!="playing":
53                 feedbackTriggers.append(activity)
    print "feedbackTriggers has length = ",len(feedbackTriggers)
```

```
    # sort feedbackTriggers array by time
    feedbackTriggers.sort(key=operator.itemgetter('ts'))
    iterator=np.arange(0,len(feedbackTriggers)-1)


    for i in iterator:
        if feedbackTriggers[i+1]['ts']-feedbackTriggers[i]['ts']>datetime.timedelta(milliseconds=200):
            #print feedbackTriggers[i]
            if feedbackTriggers[i]['logtype']=='region_block':
                theHistogram.append(1)
            if feedbackTriggers[i]['logtype']=='my_track_nav_item':
                theHistogram.append(2)
            if feedbackTriggers[i]['logtype']=='community_title_nav_item':
                theHistogram.append(3)
            if feedbackTriggers[i]['logtype']=='play':
                theHistogram.append(4)
                #feedbackTriggersFiltered.append(feedbackTriggers[i])
    #fig = plt.figure()
    if 0 in theHistogram and 1 in theHistogram and 2 in theHistogram and 3 in theHistogram and 4 in theHistogram:
        plt.axis([0, 4, 0, 500])
        p.hist(theHistogram,5)
        #display.clear_output(wait=True)
        #display.display(plt.gcf())
        #time.sleep(0.01)
        plt.savefig('./png/'+str(nUser)+'_'+str(int(time.time()))+'.png')
        plt.clf()

#print "feedbackTriggers has ", len(feedbackTriggers)," elements"
```

./pythonScripts/generateHistogramsNoSessions.py

A.9 Build 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
```

```
12
14
clir = MongoClient("localhost:27026")
16 ir = clir.research

18 # get users

20 tenVeryActiveUsers=ir.Actors.find({"$and":[{"activities.at":{"$eq"
    :0}},{"activities.at":{"$eq":6}}]},timeout=False)

22 totalUsers=tenVeryActiveUsers.count()

24 feedbackTriggers=[]
featureMatrix=[]
26 featureVector=[]
nUsers=0
28
outfile=open("featureMatrix.json", "w")
30
for user in tenVeryActiveUsers:
32
    uploads=0
34  region_block=0
    my_track_nav_item=0
36  community_title_nav_item=0
    play=0
38  comments=0
    nUsers=nUsers+1
40
    featureVector=[]
42  del featureVector[:]
    feedbackTriggers=[]
44  del feedbackTriggers[:]

46  for activity in user["activities"]:
            # get click events ignore playing logs
48      if 'type' in activity.keys() and activity['type']=='click'
    and activity["logtype"]!="playing":
            feedbackTriggers.append(activity)
50
        if activity['at']==0:
52          uploads=uploads+1
        if activity['at']==2:
54          comments=comments+1
    print "feedbackTriggers has length = ",len(feedbackTriggers)
56  # sort feedbackTriggers array by time
    feedbackTriggers.sort(key=operator.itemgetter('ts'))
58  iterator=np.arange(0,len(feedbackTriggers)-1)
```

```
60      for i in iterator:
            if feedbackTriggers[i+1]['ts']-feedbackTriggers[i]['ts']>
        datetime.timedelta(milliseconds=200):
62              #print feedbackTriggers[i]
                if feedbackTriggers[i]['logtype']=='region_block':
64                  region_block=region_block+1
                if feedbackTriggers[i]['logtype']=='my_track_nav_item':
66                  my_track_nav_item=my_track_nav_item+1
                if feedbackTriggers[i]['logtype']=='
        community_title_nav_item':
68                  community_title_nav_item=community_title_nav_item+1
                if feedbackTriggers[i]['logtype']=='play':
70                  play=play+1
                    #feedbackTriggersFiltered.append(feedbackTriggers[i
        ])
72      featureVector.append(uploads)
        featureVector.append(region_block)
74      featureVector.append(my_track_nav_item)
        featureVector.append(community_title_nav_item)
76      featureVector.append(play)
        featureVector.append(comments)
78
        norm = [float(i)/sum(featureVector) for i in featureVector]
80      grade=ir.ActorsGrade.find_one({'id':user['_id']})
        if grade is None:
82          grade=None
        else:
84          grade=grade['grade'][0]
        featureMatrix.append({'id':str(user['_id']), 'grade':grade,'
        featureVector':norm})
86
        print nUsers
88


90  json.dump({'featureMatrix':featureMatrix}, outfile, indent=4)
    outfile.close()
92  print "feature matrix has ",len(featureMatrix)," vectors"


94



96
    #print "feedbackTriggers has ", len(feedbackTriggers)," elements"
```

./pythonScripts/featureMatrix.py

A.10 Build the distance matrix

```
1   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
#%pylab inline
#from IPython import display


#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('featureMatrix.json')
featureMatrix = json.load(json_data)


distFile=open('distanceMatrix.json','w')


def calculateDistance(vector1,vector2):
    iterator=np.arange(0,len(vector1))
    sum=0
    for i in iterator:
        sum=sum+math.pow(vector1[i]-vector2[i],2)
    return math.sqrt(sum)


featureMatrixDimension=len(featureMatrix['featureMatrix'])
print featureMatrixDimension
iterator1=np.arange(0,featureMatrixDimension)
```

```
distanceMatrix=[]
tempDistance=[]
for i in iterator1:
    print featureMatrixDimension-i, 'left'
    iterator2=np.arange(0,featureMatrixDimension)
    tempDistance=[]
    del tempDistance[:]
    for j in iterator2:
        tempDistance.append(calculateDistance(featureMatrix['featureMatrix'][i],featureMatrix['featureMatrix'][j]))
        if(i==j):
            print calculateDistance(featureMatrix['featureMatrix'][i],featureMatrix['featureMatrix'][j])

    distanceMatrix.append(tempDistance)
json.dump({'distanceMatrix':distanceMatrix}, distFile, indent=4)
distFile.close()
```

./pythonScripts/distanceMatrix.py

A.11 Build a heat plot of the distance matrix

```
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
#%pylab inline
#from IPython import display


#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('distanceMatrix.json')
distanceMatrix = json.load(json_data)

distanceMatrix_=np.matrix(distanceMatrix['distanceMatrix'])
print distanceMatrix


heatmap = plt.pcolor(np.array(distanceMatrix_),cmap=plt.cm.Blues)
plt.savefig('./heatmap2.png')
```

./pythonScripts/heatPlot.py

A.12 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']:
    if activity['at']==0 or activity['at']==6:
      uploadsAndActivities6=uploadsAndActivities6+1
  # Add a field to the actor withe the total numper of activities and uploads
```

```
26    ir.Actors.update({'_id':actor['_id']},{'$push':{'
        uploadsAndLogActivities':uploadsAndActivities6}})
```

./pythonScripts/countUploadsAndActivities.py

A.13 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.14 Build the feature vectors

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

3  from bson import *
   from pymongo import *
5  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
```

```
53
def makeActivitiesList(actor):
55  # we put the activities in a list so we can sort by timestamp
    activitiesList=[]
57
    for activity in actor['activities']:
59      if activity['at']==0:
          activitiesList.append(activity)
61      if activity['at']==6:
          # get the 'click'
63        if checkForKey('logtype',activity) and activity['logtype']!='
      playing' and checkForKey('type',activity) and activity['type']==
      'click':
            activitiesList.append(activity)
65
    activitiesList.sort(key=operator.itemgetter('ts'))
67
    return activitiesList
69
def makeActivityChunks(sessionsObjectList,activitiesList):
71  # we want to devide the activities list respect to the sessions
    activitiesSet=[]
73
    for session in sessionsObjectList:
75    #print session['sessionStart']
      tempAct=[]
77    del tempAct[:]

79    for activity in activitiesList:

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

87  return activitiesSet

89def checkForString(dictionary,key,check):
    if key in dictionary and dictionary[key]==check:
91    return True
    else:
93    return False

95def makeFeatureVectors(activitiesSet):
    n=0
97
  #print len(activitiesSet)
99
```

```
  featureVectors=[]
  del featureVectors[:]
  for activitySet in activitiesSet:
    n=n+1
    upload=0
    region_block=0
    my_track_nav_item=0
    community_media_nav_item=0
    play=0
    tempVector=[]
    for activity in activitySet:
      if activity['at']==0:
        upload=upload+1
      if checkForString(activity,'logtype','region_block'):
        region_block=region_block+1
      if checkForString(activity,'logtype','my_track_nav_item'):
        my_track_nav_item=my_track_nav_item+1
      if checkForString(activity,'logtype','
    community_media_nav_item'):
        community_media_nav_item=community_media_nav_item+1
      if checkForString(activity,'logtype','play'):
        play=play+1

    tempVector.append(upload)
    tempVector.append(region_block)
    tempVector.append(my_track_nav_item)
    tempVector.append(community_media_nav_item)
    tempVector.append(play)
    featureVectors.append(tempVector)

  return featureVectors


actorsList=sortActors()

outfile=open("histogramFeatureVectors.json", "w")

featureList=[]
for actor in actorsList:
  featureObject={}
  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.15 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>'
        sessionId=re.search(re.escape(startSessionId)+"(.*?)"+re.escape(endSessionId),content).group(1)
        sessionId=sessionId.split('session_user_id: ')
        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
  def linkMCUserIdsWithGrades():
64  data=[]
  for el in getMCUserIds():
66    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():
80    if 'activitiesLength' in actor:
      ir.ActorsGrade.update({'id':actor['_id']},{'$push':{'
    activitiesLength':actor['activitiesLength']}})
82

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.16 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
    allActivities=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]
      allActivities=allActivities+featureVector[0]+featureVector
    [1]+featureVector[2]+featureVector[3]+featureVector[4]
    cummulative.append({'actor_id':actorId,'features':[uploads,
    region_block,my_tracks,community_tracks,play,allActivities]})

  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])

      # all activities
      y.append(actor['features'][5])
      grades.append(grade['grade'])


  regr = linear_model.LinearRegression()
```

```
77    regr.fit(grades, y)
    # The coefficients
79    print('Coefficients: \n', regr.coef_)
    print('R^2', regr.score(grades,y))
81
    plt.xlabel('Grades')
83    plt.ylabel('Number of activities')
    plt.plot(grades,y,'o')
85    plt.savefig('allActivities.png')
    plt.show()
87

89 generatePlot()
```

./pythonScripts/groundTruth.py

A.17 Train and classify students with KNN

```
1  from numpy import genfromtxt
   from bson import *
3  from pymongo import *
   import time
5  from datetime import timedelta
   import numpy as np
7  import sys
   import json
9  from pymongo.errors import InvalidOperation
   import operator
11 import matplotlib.pyplot as plt
   import pylab as p
13 import math
   import random as random
15 import os
   from sklearn.cluster import KMeans
17 from sklearn import datasets, linear_model
   from sklearn.neighbors import NearestNeighbors
19 from sklearn.neighbors import KNeighborsClassifier

21 clir = MongoClient("localhost:27026")
   ir = clir.research
23
   featureListFromFile = json.load(open('featureMatrix.json'))
25
   outfile=open("kNeighbourScore.json", "w")
27


29 # Train data
   def makeTrainData():
31   trainData=[]
```

```
  while(len(trainData)<int(0.9*len(featureListFromFile['
    featureMatrix'])) ):
    random.seed(time.time())
    index=random.randint(0, len(featureListFromFile['featureMatrix'
    ])-1)
    if featureListFromFile['featureMatrix'][index] not in trainData
    :
      trainData.append(featureListFromFile['featureMatrix'][index])
  return trainData

# Test data
def makeTestData():
  testData=[]
  for featureVector in featureListFromFile['featureMatrix']:
    if featureVector not in trainData:
      testData.append(featureVector)
  return testData



# k neighbours classifier
# build feaure vectors and classes for train
def buildFeatureVectorsAndClassesForTrain():
  featureVectorsTrain=[]
  classesTrain=[]
  for user in trainData:

    grade=user['grade']
    if grade is not None:
      if grade<50:
        classesTrain.append(1)
      if grade>=50 and grade<70:
        classesTrain.append(2)
      if grade>=70:
        classesTrain.append(3)

      featureVectorsTrain.append(user['featureVector'])
  return [featureVectorsTrain,classesTrain]

# build feaure vectors and classes for test
def buildFeatureVectorsAndClassesForTest():
  featureVectorsTest=[]
  classesTest=[]
  for user in testData:
    grade=user['grade']
    if grade is not None:
      if grade<50:
        classesTest.append(1)
      if grade>=50 and grade<70:
        classesTest.append(2)
```

```
        if grade>=70:
          classesTest.append(3)
        featureVectorsTest.append(user['featureVector'])
    return [featureVectorsTest,classesTest]

iterator=np.arange(0,100)
iterator2=np.arange(2,80)
kVsAccuracy=[]

for j in iterator2:
    print j
    temp=[]
    del temp[:]
    for i in iterator:
        trainData=makeTrainData()
        testData=makeTestData()
        print 'train set = ',len(trainData)
        print 'train test = ',len(testData)
        featureVectorsTrain_=buildFeatureVectorsAndClassesForTrain()[0]
        classesTrain_=buildFeatureVectorsAndClassesForTrain()[1]
        print 'trainV=',len(featureVectorsTrain_)
        print 'trainC=',len(classesTrain_)

        featureVectorsTest_=buildFeatureVectorsAndClassesForTest()[0]
        classesTest_=buildFeatureVectorsAndClassesForTest()[1]

        print 'testV=',len(featureVectorsTest_)
        print 'testC=',len(classesTest_)

        classifier=KNeighborsClassifier(n_neighbors=j)
        classifier.fit(featureVectorsTrain_,classesTrain_)
        score=classifier.score(featureVectorsTest_,classesTest_)
        temp.append(score)
        print score

    maxVal=max(temp)
    minVal=min(temp)
    kVsAccuracy.append([j,maxVal,minVal])

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

./pythonScripts/kNearestNeighbour.py

A.18 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']:
```

```
49      featureList.append(feature)

51  k_means.fit(featureList)

53  clusters={}
    iterator=np.arange(0,numerOfUsers)
55  for i in iterator:
      clusters[str(i)]=[]
57
    clusterListForHist=[]
59
    for actor in featureListFromFile['features']:
61    for feature in actor['featureVectors']:
        cluster=k_means.predict(feature)
63      for c in cluster:
          clusters[str(c)].append({'feature':feature,'actor_id':actor['
        actor_id']})
65        clusterListForHist.append(c)

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

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

./pythonScripts/clusteringFeatureSessionVectors.py

A.19 Comparison of two clusters

```
1   from bson import *
    from pymongo import *
3   import time
    from datetime import timedelta
5   import numpy as np
    import sys
7   import json
    from pymongo.errors import InvalidOperation
9   import operator
    import matplotlib.pyplot as plt
11  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

modelCluster=[]

for actor in featureListFromFile['features']:
  for feature in actor['featureVectors']:
    cluster=k_means.predict(feature)
    for c in cluster:
      modelCluster.append(c)

print "adjusted random index (different from assigning random
    classes?)= ",metrics.adjusted_rand_score(groundTruthCuster,
    modelCluster)
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.20 Comparison of two clusters

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



pca = PCA(n_components=2)

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

def getJSONData(json_data):
  return json.load(json_data)

def performPCA(featureList):
  pca.fit(featureList)
  print "variances = ",pca.explained_variance_ratio_
  transformedVectors=pca.transform(featureList)
  return transformedVectors

def normaliseList(featureList):
  featureListNorm=[]
  for feature in featureList:
    norm=math.sqrt(sum([j*j for j in feature]))
    if norm !=0:
      normalisedVector=[float(i)/math.sqrt(sum([j*j for j in
    feature])) for i in feature]
    else:
      normalisedVector=[0 for i in feature]
    featureListNorm.append(normalisedVector)
  return featureListNorm

def preapareForPlot(featureList):
  X=[x[0] for x in featureList]
  Y=[y[1] for y in featureList]
  return[X,Y]

def clusterKMeans(featureList,nClusters):
```

```
    k_means = KMeans(n_clusters=nClusters, init='random')
    k_means.fit(featureList)
    clusterClasses=[]
    for feature in featureList:
        cluster=k_means.predict(feature)
        for c in cluster:
            clusterClasses.append(c)
    return clusterClasses

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

def compareClustering(groundTruth,modelCluster):
    print "adjusted random score (different from assigning random
        classes?)= ",metrics.adjusted_rand_score(groundTruthCustering,
        modelCluster)
    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)
ax1 = featureListFigure.add_subplot(111)
ax1.plot(featureListData[0], featureListData[1],'ro')
plt.savefig('pca2D.png')
'''
normFeatureListData=preapareForPlot(normalisedList)
normFeatureListFigure = plt.figure(1)
ax2 = normFeatureListFigure.add_subplot(111)
ax2.plot(normFeatureListData[0], normFeatureListData[1],'ro')
'''
plt.show()

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

./pythonScripts/pca.py

B Project proposal

Data mining and DSP analysis of social media for student performance prediction

Student: Andreu Grimalt Reynes

Supervisor: Matthew Yee-King

1 Final project proposal

In my final project I would like to explore the fields of Data Mining, Machine Learning and Neural Networks to try to provide answers to questions about education.

The growth in the use of VLEs (Virtual learning environments), MOOCs (Massive Online Open Courses) and other online education systems provides an increasing stream of data regarding student behaviour. It is now possible to use data mining algorithms to analyse student habits and try to find correlations between student behaviour and academic performance.

Teachers and researchers have begun to explore the possibility of using social media in teaching and learning. In general, the use of social networks generate data sets that accurately describe their users behaviour. When social media is used for learning, the data set generated can be used to parametrise the learning experience and therefore to optimise it. This enables us to think about social networks, education and student behaviour as a classification problem. That is to say, to use the data generated in social media and apply data mining algorithms to it in order to classify students in groups according to their academic performance. Once the algorithms are trained in recognising these groups, they can predict future performance and allow the optimisation of the learning.

Code Circle is a social network for sharing and discussing programming code. Its aim is to make it easy to share code with a community of users and to provide tools that encourage feedback and the discussion of work in progress. Therefore, being able to give specific feedback on the code (i.e being able to select particular bits on the code and attach comments to them) and being able to modify the code and run it on the browser are key features. Code circle was developed at Goldsmiths as part of a group summer project, partly using software components developed within the PRAISE research project. It is aimed at programming students, it can run Processing sketches on the browser and display

the code of a particular sketch. It is currently used in a case study involving approximately 200 undergraduate students from an introductory programming course.

In Code Circle all the user interaction is logged. This data set contains information about the user, the time, the type and the element that received the user action. The data has this structure: {*user_id: Number, time: Number, type: String, id: Number, logotype: String*}

- user_id: The user id that performed the action
- time: Timestamp when the action occurred
- type: The type of action (click, mouseover, etc...)
- id: Id of the element that received the action
- logtype: Text description of the element that received the action

In my project I would like to apply data mining techniques to the analysis of the data set generated within Code Circle.
The data analysis will address the following research questions:

- Can data mining and neural networks algorithms classify student academic performance?
  Try to train a system to recognise "good" and "bad" behaviour in students. This is a supervised learning problem which will use the grades of the students as ground truth. Several data mining algorithms can be used and compared using performance measures.
- Can DSP be used to pre-process the data and perform feature reduction on it? How does it perform compared to other dimensional reduction techniques?
  Code Circle's log data depends on the time and it can be treated as a signal. I would like to explore how DSP can be used with data mining in order to pre-process the data and perform dimensional reduction. Also, I would like to measure how it performs compared to other feature reduction techniques.

2 References

Ian Witten, Eibe Frank, Mark Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Elsevier Science & Technology, 2011.

Romero, Cristóbal and López, Manuel-Ignacio and Luna, Jose-María and Ventura, Sebastián, *Predicting students' final performance from participation in on-line discussion*

forums, Computers & Education, vol. 68, p458-472, Elsevier Ltd, 2013.

Hundhausen, Christopher D and Carter, Adam S, *Facebook me about your code: An empirical study of the use of activity streams in early computing courses*, Journal of Computing Sciences in Colleges, vol. 30, p151-160, 2014.

Kulkarni, Chinmay and Wei, Koh Pang and Le, Huy and Chia, Daniel and Papadopoulos, Kathryn and Cheng, Justin and Koller, Daphne and Klemmer, Scott R., *Peer and self assessment in massive online classes*, ACM Transactions on Computer-Human Interaction (TOCHI), vol. 20, 2013.

J.P. Vandammea, N. Meskensa, J.F. Superbya, *Predicting Academic Performance by Data Mining Methods*, Education Economics, p405-419, 2007.

C Project preliminary report

Mining educational social networks for student performance classification

Andreu Grimalt Reynes

Abstract

This document is a preliminary report on my final project in computing. The project is concerned with the analysis of log data generated by an online learning platform in order to extract knowledge about student performance. The first section of this report contains a general description of the project as well as an overview of the techniques used. It is followed by a description of the aims and objectives. The next sections discuss the methods employed, the project plan, the progress to date and the planned work. Finally, there is an appendix which contains software code.

Introduction

The growth in the use of VLEs (Virtual learning environments), MOOCs (Massive Online Open Courses) and other online education systems provides an increasing stream of data regarding student behaviour. This new paradigm makes it possible to use data mining algorithms to analyse student habits and try to find correlations between student behaviour and academic performance. Teachers and researchers have begun to explore the possibility of using social media in teaching and learning [10]. In general, the use of social networks generate data sets that accurately describe their users behaviour. When social media is used for learning, the data set generated can be used to parametrise the learning experience and therefore to optimise it. This enables us to think about social networks, education and student behaviour as a classification problem. That is to say, to take the data generated in social media and apply data mining algorithms to it in order to classify students in groups according to their academic performance. Once the algorithms are trained in recognising these groups, they can predict future performance and allow the optimisation of the learning experience.

The data used on this project was generated in a case study involving Music Circle, a social network for sharing and discussing media [4]. Music Circle makes it easy to share media with a community of users and it provides tools that encourage feedback and the discussion of work in progress. Therefore, being able to give specific feedback on the media item (i.e being able to select particular bits on the media and attach comments to them) are key features. Music Circle was developed within the PRAISE research project at the computing department at Goldsmiths [5].

Aims and objectives

Aim

The aim of this project is to use the activity patterns of Music Circle users in order to classify them by their academic performance.

The activity pattern of a user is fully described by mouse events which are logged into the Music Circle database. The log data taken into account is the response after feedback, that is to say what the user does after receiving feedback from other students in the system. The actions a user can perform on the system are the following: Viewing a media item, reading or writing an annotation on a media item and reading or writing a reply on an annotation. These actions can be deduced from the log data which has the following structure:

- user id: The id of the user that performed the action
- time: The time where the user performed the action in milliseconds since the epoch time
- type: The type of mouse action (click, mouse over, mouse out)
- id: The id of the element the action was performed on
- logtype: A custom name used to identify the element the action was performed on

The categories in which the students are classified are simply "successful students" and "struggling students". This implies the assumption that good students have a different behaviour from struggling students and that this difference is represented in the log data. Hopefully, the research will show if this assumption is valid.

Objectives

Setting up the development environment: Virtual Environment, Python scientific tools, MongoDB and PyMongo

The programming language used in this project is Python. The main reason to use Python is that it has a rich library ecosystem in scientific programming and machine learning. It provides tools for doing mathematical operations and plotting as well as a friendly IDE (iPython notebooks), all conveniently sandboxed in a virtual environment.

Music Circle data is stored in a Mongo database. The data is processed in Python using PyMongo which provide the tools to work with MongoDB databases.

Build a research database

The data from Music Circle is stored in a way that makes it efficient for the server to retrieve it. Unfortunately, this way of storing the data is not the most efficient in order to analyse it. Therefore, a database with a new structure needs to be built.

Feature vector construction from the log data

The feature vector describes the behaviour of a particular student at a particular interval of time. There are many ways to construct such a vector but one must take into account that not all possible

feature vector representations will be useful for the classification problem. Therefore, the selection of a suitable feature vector is critical for the success of the project.

Clustering of the feature vectors

Once the feature vectors are constructed they need to be validated. One needs to demonstrate that the vectors describe user behaviours and that they can be used for classification. In order to do this, the first approach is to try to cluster a set of feature vectors. If the vectors represent the behaviour of users correctly and that behaviour is consistent through tscalingime, then they will cluster in n clusters where n is the number of users.

Feature reduction

Not all the components of the feature vectors will contribute the same amount to differentiate users. The objective at this stage is to find which components of the vectors contribute the most to differentiate between users and discard the components that are not relevant.

Statistical distribution comparison

Once the principal components are found, one can construct a statistical distribution of the feature vectors for each user and then compare those distributions. The hypothesis is that students from different categories will have different statistical distributions, by associating the distributions to the ground truth data one should be able to classify the students.

The ground truth data consists of the grades from quizzes and peer assessments that students completed during the case study.

Methods

This section describes the methods used to implement the objectives.

Virtual Environment, Python scientific tools, MongoDB and PyMongo

The software stack used in this project consists in Virtual Environment, iPython notebook, SciPy, PyMongo and Scikit-learn.

Virtual Environment is a tool to keep the dependencies required by different projects in separate places, by creating virtual Python environments for them [9]. iPython notebooks are a web-based interactive computational environment where you can combine code execution, text, mathematics, plots and rich media into a single document [2]. SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering [8]. PyMongo is a Python distribution containing tools for working with MongoDB [6]. Finally, Scikit-learn is a machine learning library in Python [7].

Build a research database

The Music Circle database is approximately 19Gb in size with some collections containing an order of 10^6 elements. In order to process the data in a reasonable amount of time, one needs to program in a way that maximises the CPU use.

MongoDB response to large database processing is sharding [3]. Sharding main idea is to split the database across different machines or processor cores in order to allow scaling as well as parallel computing. Such a system has the name of sharded cluster.

A sharded cluster has three main components: Shards, configuration servers and routers.
Shards contain the data in the database (each shard contains a subset of the data in the database). The configuration servers contain metadata about the cluster which map to data in the shards. Finally, the routers route the reads and writes to the shards.

There are some parameters that need to be chosen carefully in order to maximise the benefits of sharding. These parameters are: Number of shards, number of routers and chunk size.
Ideally, the number of shards should match the number of CPUs available. This way MongoDB can access the database using more than one CPU thus allowing parallel computing. Technically only one router is needed, however, in real world scenarios it is recommended to use at least two routers in the case one fails. The size of the chunks will determine how the data gets distributed in the shards, one must aim for an equal distribution of data between the shards.

Feature vector selection

The feature vector selection is crucial for the success of the project. In general, the feature vectors will be determined by the log data available. At the same time, there are many ways of combining the log data in order to construct the vectors, this means that once built the feature vectors need to be validated in order to demonstrate that they characterise correctly user behaviour. These validation process consists in clustering the vectors. To do so, a set of feature vectors corresponding to different users and different periods of time is selected. If these vectors successfully represent users' behaviour and this behaviour is consistent over time, the vectors corresponding to each user should cluster together. In ideal conditions, the clustering process would produce n clusters where n is the number of users.

Feature reduction and statistical comparison

The previous step will hopefully find a set of users with consistent behaviour through time. Some of the components of the feature vector will contribute more than others to differentiate between user behaviour.
Feature reduction techniques allow to find the principal components of the feature vector. This dimensional reduction process is necessary in order to ease the analysis and avoid redundant information that could contribute to false results [11].

In such scenario, one can build a statistical distribution of the feature vectors for each user and then compare the distributions using Kullback-Leibler divergence [12]. Making the hypothesis that the statistical distribution of feature vectors between successful and non successful students are different and linking the distributions to the ground truth data, this method should classify students in binary categories of performance. With this last step the project can be validated by trying to cluster different distributions. The number of clusters should be equal to two with one cluster representing successful students and the other representing struggling students. In order to evaluate the project, the clustering can be compared to the ground truth data and measure how accurate the classification is.

Project plan

The software development method used in this project is agile development [1]. It consists in iterative implementations, face-to-face communication between members of the team, short feedback loop and quality focus.

The project is divided into the objectives described in the "Aims and Objectives" section. Each of the objectives is further divided into small incremental tasks which are set at a meetings held weekly. Every week a set of tasks have to be implemented and on the following meeting those implemented tasks are reviewed.
Supposing that the implementation has been successful a new set of tasks is set. Otherwise, the failures on the implementation are analysed and solutions are proposed.

There is not a long term planning of the project but a succession of incremental and adaptive iterations over short time frames which get set in weekly meetings.

Progress to date

The progress up to date relates to setting up the development environment, building the research database and finding a suitable feature vector.

Most of the time was spent in building the research database. Because of the size of the Music Circle database, finding a method to import the log data efficiently (described in "Build a research database" section) was essential.
The research database contains two collections: Actors and Media.
Each element in the Actors collection contains all the activities of a particular user on the system. The Media collection contains data about the media items uploaded to the system.

This is the schema of the research database:

```
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 Chris Kiefer who previously did research on the same dataset.

With the research database running on a shared cluster, the Music Circle data was imported in parallel thus reducing the computation time by a factor of 4.

At the moment all the time is invested in constructing a useful feature vector. The main difficulty encountered is in dealing with the time nature of the data. This dependancy makes it difficult to unambiguously establish the causes of a given action on the system. The problem is that the user actions after feedback depend on how the data is quantised in time. In general, one can not be sure for how long a particular action affects the subsequent user behaviour in the system. At the moment feedback actions are considered to be characterised by user clicks on a particular element. The quantisation of the data is made by establishing a time threshold after a user action (a click). The hypothesis is that all the log events under the threshold are caused by that particular click. This approach is very subjective and another alternative should be found. The success of the project depends heavily on how well this issue can be solved.

Planned work

On a chronological order the planned work consists in the feature vector construction, the clustering of the feature vectors, the principal component analysis of feature vectors and the probability distribution comparison.

As described in the previous section, the construction of the feature vector presents some challenging problems.

The clustering of the feature vectors will consist in the application of different clustering algorithms and the selection of the most accurate result.

The feature vectors dimensionality is expected to be low (5 dimensions approximately). In order to find the relevant components (feature reduction), the first approach is to calculate all the possible combinations of components in the vector ($n^2 - 1$ where n is the dimensionality of the vector). With these new set of feature vectors one can build a probability distribution of them for each user. The hypothesis considers that the distributions between successful and struggling students are different. Therefore, one can select the combination of components that maximise the distance between two distributions belonging to users from different categories (a successful student and a struggling student). This metric should allow the selection of the optimal

components of the feature vectors.
Once the principal components are selected and the distributions calculated one can proceed to evaluate the project as described in the "Methods" section.

References

[1] Agile manifesto. http://agilemanifesto.org/. Accessed: 2015-02-20.

[2] iPhython. http://ipython.org/. Accessed: 2015-02-20.

[3] MongoDB sharding. http://docs.mongodb.org/manual/sharding/. Accessed: 2015-02-20.

[4] Music Circle. https://goldsmiths.musiccircleproject.com. Accessed: 2015-02-20.

[5] PRAISE project. http://www.iiia.csic.es/praise. Accessed: 2015-02-20.

[6] PyMongo. http://api.mongodb.org/python/current/. Accessed: 2015-02-20.

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

[8] SciPy. http://www.scipy.org/. Accessed: 2015-02-20.

[9] Virtual Environment. http://docs.python-guide.org/en/latest/dev/virtualenvs/. Accessed: 2015-02-20.

[10] Félix Castro, Alfredo Vellido, Àngela Nebot, and Francisco Mugica. Applying data mining techniques to e-learning problems. In *Evolution of teaching and learning paradigms in intelligent environment*, pages 183–221. Springer, 2007.

[11] Imola K Fodor. A survey of dimension reduction techniques, 2002.

[12] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

D Weekly logs

D.1 19 January - 23 January

Task	Notes
Set up GIT	Done
Set up python env	Done
Find if Basecamp is appropriate tool for printing all the tasks as I want to put that on the report	Will do the job as a list of todos. Ask Matthew if I can have a separate project on Basecamp
Start Python tests. Check Pymongo, Lists, Pylab (plots)	Done
Plot some simple statistics	Done
Check machine learning library	Planning to use scikit-learn
Did some FFT tests	Just to clarify concepts
Started thinking about DSP approach	Problems: are signals periodic? Check DSP approach on stock markets/economics
Arranged a meeting with Ben (PhD doing research on social networks)	Wednesday 28, 5:30, 329

D.2 2 February - 8 February

Task	Notes	Hours
Check the log data generated	The data is valid from 30th of June	24
Problem: Log collection is huge	Made a cluster to speed up	48
Check how other's use feature vectors	Read Baker's paper. They use feature vectors, summary here: https://basecamp.com/2065604/projects/1534650/todos/151387616	4
Read 'Towards sensor free affect detection in cognitive tutor algebra'	Summary here: https://basecamp.com/2065604/projects/1534650/todos/154618637	4

D.3 9 February - 15 February

Task	Notes	Hours
Import research database	Set up sharded cluster, quite difficult	40
Try clustering on feature vectors	First you need to make them!	16

D.4 16 February - 22 February

Task	Notes	Hours
Wrote the preliminary report	Done	40

D.5 23 February - 1 March

Task	Notes	Hours
I did research on how people analyse social network data	I found papers that use feature vectors to find student academic performance from their activity in forums	8
I started to build the bibliography	Done	2
Coding	No progress so far. Finding it difficult to build the feature vectors. Also, iPython is really bad at printing and crashes a lot	30

D.6 2 March - 8 March

Task	Notes	Hours
Coding	Built feature vectors	8
Coding	Made histograms of feature vectors	8
Coding	Made a video with the histograms of the feature vectors, wrote a bit about it	8
Coding	Made a difference matrix with the feature vector, wrote a bit about it	8

D.7 16 March - 22 March

Task	Notes	Hours
Coding	Built new feature vectors that consider user sessions	10
Thought about clustering the feature vectors	Started trying clustering on feature vectors	20

D.8 23 March - 29 March

Task	Notes	Hours
Coding	Clustered the feature vectors	10
Research in how can I compare two clusters	Started trying clustering on feature vectors	4
Research in how to proceed once the clusters had been validated	I can calculate vector distribution in the clusters and then compare using Earth's moving distance	16

D.9 30 March - 12 April

Task	Notes	Hours
Write project report	-	-

D.10 12 April - 26 April

Task	Notes	Hours
Methods section on the report looking a bit weak, need to do another study on the data	Thinking about applying another DM algorithm to the data	-