Vec Hub: A Review Based Seed Music Recommendation System (MRS)

Jasper Kirton-Wingate

BSc Music Computing

33414271

Supervised by Prof. Mick Grierson

Department of Computing

Goldsmiths, University of London

15th May 2018

Abstract

Recently, the collaboration and collision of Big Data with advanced Machine Learning algorithms has brought under-appreciated artists success, increased the diversity of music people listen to and has signified a new age in music discovery. The theme of context and semantics has been a small part of a much wider field of research and development in recommendation systems. Here, through applying modern Deep Learning technology to custom Linked Big Open-Source Data, I introduce a novel approach for recommending music by modelling rich semantic context from music reviews. Further, I present it as a simple way to extend and improve upon the capabilities of Collaborative Filtering often used in commercial contexts, and provide direction for future research in the field.

Acknowledgements

Firstly, thank you to my supervisor, Prof. Mick Grierson, for his good ideas and interesting conversations. A big thank you to my parents and family for being so supportive throughout the course of my degree. Thanks to my peers and friends for being supportive and helping me see things from a different perspective. Thank you to all who tested my application, I hope you discovered some great music and artists. Thank you to all the DJs and musicians who inspire me every day!

Table of Contents

Chapter 1: Introduction

1.1 **Project Motivation**

Aside from my studies and passion for technology, science and research, I am a professional DJ (radio and venues) and music producer (commercial media and independent labels). Although I am usually paid for these endeavours, they are also my hobbies, and my need to discover new music is both quintessential for my professional practice and deeply important to me on a personal level. The culture of music in London is as eclectic and diverse as perhaps any city in the world. Since moving here for University I started to regularly go 'record digging' – I find the physical process of searching and discovering music is complementary to the digital one. It was through discussing the advantages of this physical process, the wealthy array of semantic and contextual knowledge contained in the print, that my supervisor and I came up with the idea to try the 'semantic processing' route, and use modern Deep Learning inspired technology – namely Doc2Vec – in order to achieve competitive and gratifying results.

1.2 **Initial Discussion**

The following is our initial conversation on the subject of MRS.

Mick to myself:
"There is a mountain of research in the field, and there are lots of questions. Some basic ones are:

1) Given that this is a well researched field, with lots of good examples, what is the most popular form of music recommendation?

2) Why is it popular? Is that also the best form in your opinion? Could it be improved?

3) What is the relationship between the actual sound and the recommendation? What sort of features make for good recommendations? Is this the right way to think about it?

4) Are recommendations about marketing, or something else?"

My answer:
"Let me have a quick stab at those questions, given my current understanding;

1/2) The most popular platform would most likely be Spotify, given that most people aren't music enthusiasts (sadly), they appreciate just being able to be given simply a playlist of similar songs that they'd like to hear in the background or be able to skip through quickly if they don't like

them. From what I understand Collaborative Filtering is the most popular approach as it is fairly simple to achieve good results based on a load of user information, it also fits the popular music 'chart' model, there are however downsides to this approach especially for music enthusiasts, who wish to seek out music which isn't necessarily popular. The best current approaches in my opinion are hybrid models between collaborative filtering and feature likeness (which is a huge field), contextual approaches are also interesting (time of day, location, activity etc.)

3) Feature likeness in terms of acoustic data usually categorises things like key, tempo, energy, timbre, tonality/atonality etc. which are the results of objective MIR functions (often to answer subjective questions). There are the user 'labels'; like genre and style (can also be reasonably classified from acoustic data) which are very useful. But there are things I've noticed that can be hugely influential to the sound of the record, like the relationship between year and location, the instruments and players or even the engineers who recorded them which I feel is under looked in current recommendation systems, this is a fascinating approach to exploration for me and a lot of other music enthusiasts.

4) Recommendations can be about marketing, and this should be an important factor in the design of such as system, but Spotify monopolises the market and the artists get paid minuscule amounts for the streams. It would be nice to have a recommendation system that linked to places to purchase the music in multiple formats (I'm sure there are some already). What excites me most about recommendations is the opportunity for new methods of exploration, a system has the potential to become an educational entertainment platform if we begin to take connections out of the 'black box' of the learning algorithm and present them to the user in an attractive way, and allow them to interact with the system, evaluating the algorithm for adaptation.

Creatively, I'm thinking about a record store in space where the owner knows his stock inside out and can provide recommendations based on his knowledge -- through this; paths of discovery unfold."

Chapter 2: Background Research

Music Discovery

The culture of DJing and Record Collecting is synonymous with an innate desire for producers, music and art enthusiasts to seek the lesser known. Musical discovery has been associated with colonialism [35]; the fusion of discovered and traditional music has led to new styles or genres; such as Samba and Jazz culminating in Bossa Nova – as a result of European and African colonialism in Brazil. Inevitably, due to my fascination with Ontology (the connection of entities), in this case music releases, I found it intriguing to contemplate the current state of MRS (whereby so much of modern discoveries are made [34]), and assess where such systems could be improved.

The Role of Recommendation and Music Discovery in the Media and Music Industry

In mainstream FM radio, music discovery is mainly fuelled by major record labels. It is however, recently that internet radio has brought the need for more music to be discovered more prevalent than ever, in a broad mathematical function we could estimate this to me far more than the limited number of hours on solely FM radio. With the prominent rise in independent record labels, the need to discover music beyond the scope of the popular (Collaborative Filtering [27]) grows too, some of these record labels (e.g Melodies International) also seek to discover older music that may have been overlooked and/or under distributed at its time of release. With on-demand services, the chance of ‘over-playability’ of tracks grows exponentially. In documentary production, there often arises a need for music to perfectly compliment a situation in context, mood and semantics. One could imagine how user interface based, parameterised recommendation could be extremely useful in quickening and enhancing the selection of soundtracks.

Music on the Web

“Music has been a part of the web almost since the web’s inception. One of the earliest music-centric websites to gain real traction was the Internet Underground Music Archive or IUMA 1 which was started by Rob Lord, Jeff Patterson, and Jon Luini from the University of California, Santa Cruz in 1993 [44]. Originally existing in the form of a Usenet newsgroup, IUMA quickly evolved into a website as its creators recognized the emerging power of the web. IUMA’s goal was to help independent artists use the Internet and the web to connect with fans directly - circumventing the traditional record labels and giving unknown music artists increased exposure.” [9] Recently, in the context of Music Recommendation, particularly on the platform of

Youtube, we have begun to see examples of how Machine Learning can increase relatively unknown artists exposure – pushing bands like Boy Pablo into the mainstream [47].

Web and Music Informatics and The Start of the MRS

As the amount of content on the web extrapolated beyond the amount that could easily be consumed, there emerged new ways to search and discover content. Notably, Google's now famous PageRank algorithm was one of the first ways to deal with the 'Data Overload' problem for Web Ranking, using a network based approach. Whereby; as a page is linked to more often, it appears higher up on a search based on keywords. This is achieved through Markov Models of web pages, their resultant matrix and calculating the Eigenvalues/vectors that exist in the matrix. This type of statistically and mathematically modelled discovery is a core characteristic of the Recommendation algorithm.

It became for music - instead of hyperlinks and text which describe a web page - significantly based upon Music Information Retrieval 'MIR' features as a good way to measure similarity between tracks. Using Digital Signal Processing (DSP) techniques, it is possible to describe the music in terms of it's sonic content, which is quintessential to how similar pieces of music are *sounding*; briefly, temporal frame based spectral features (e.g MFCC) are then aggregated to create a track-level representation. *[5; 6; in 9] [10]*

Alongside developments in search and MIR technologies, the science of recommendation began to evolve. "Recommender systems grew not out of interest in music explicitly but out of the desire to improve the online retail experience. As retailers moved their wares to web-based stores the limitations of the brick-and-mortar physical store no longer applied. As Chris Anderson describes in his popular book (and article) The Long Tail, this meant online retailers could stock a nearly infinite variety of goods - increasing the availability of specialty and niche products [18]. Although in retrospect, Anderson's prediction regarding the demise of mainstream hits seems to run contrary to some more recently observed trends [42], he was correct in many respects. For one, shoppers in the web world have become spoiled for choice and now need assistance finding products they might want to purchase. Of course this emerging trend was evident long before Anderson published his book and several practical solutions had already been developed. The collaborative filtering approach to recommendation was first suggested in the mid 1990s [20] and continues to be the basis of many recommender systems to this day."[9]

Collaborative filtering is based on the presumption that: if person A and B likes track X and person B additionally likes track Y, person A will probably like track Y too. This probability is greater as similarities between N users grows. This seems like a reasonable simplification of how social circles work when recommending music; mutual friends x mutual music = good recommendations. A Pearson correlation coefficient or some other function can be used to apply this intuition to large amounts of data - perhaps user ratings or customer purchases - to create a recommendation system[27].

“As MIR researchers primarily focused on signals and recommender systems engineers focused on ratings data, some researchers began to recognize there is more to music. The cultural context of a piece of music can be extremely important and the nature of this context can be difficult if not impossible to tease out of signals or ratings. Increasingly people began to write about music and publish these writings on the web. The rise of user-generated content on the web meant there was an increasing amount of information about an increasing number of music-related topics. Researchers developed new approaches to music information retrieval by mining the text of these music-related web pages and blogs for key concepts and even matching these concepts with audio signal features [46; 28]. With this web mining approach to music informatics we can, in a sense, determine similar music artists by algorithmically “reading” about music artists on the web - an approach that, arguably, encapsulates some of the cultural context and musical meaning not present in the audio signal itself.” [9] Kurt Jacobson perfectly encapsulates, with relevance to literature, the significance of this project in web based music ontology.

Musical Similarity and User Personality

Musical Similarity is an interdisciplinary subject of study. Musical variance (and therefore metrics of similarity) exist on many different dimensions; e.g. Sonics/Acoustics, Melody & Harmony, Rhythm & Tempo, Composition, Culture and Lyricism. These dimensions are the focus of many MIR algorithms and data retrieval techniques, encapsulated by audio features *and* metadata. It is the goal of projects such as MusicBrainz to gather as much data in these categories about the universe of recorded music as possible. It is this data, then, that gives us exclusive inputs and ‘fuel to the engine’ of MRS; data which - independent from the User - can describe where a piece of music is located within the n-dimensional universe of musical recordings. It is here also where the distinction between *‘user-to-user’* and *‘item-to-item*’ recommendation is drawn (the aforedescribed falling into the latter category) [26]).

It is worth noting that whilst musical similarity is an adequate metric and ‘guide’ for getting recommendations of good quality from input data [34] how much of an influence this should be is hypothetically down to the users preference and taste (eclecticism); some users may want something very similar to a ‘seed’ (a singular starting point), and others may want an increase in the variance of features in the recommendation. [24] proposes a kind of pyramid to distinguish categories of users, each requiring different recommendations: ‘Savants’ (7%), ‘Enthusiasts’ (21%), ‘Casuals’ (32%) and ‘Indifferents’ (40%). The informative study [Jennings, 2007 in 24] is based on the analysis of thousands of subjects, with an age group ranging from 16 through 45.

The Current State of MRS

There has been a great deal of prior research in Music Recommendation Systems (MRS); Schedl et al., in their paper *“Current Challenges and Visions in Music Recommender Systems Research”,* states that “such systems are still far from being perfect and frequently produce

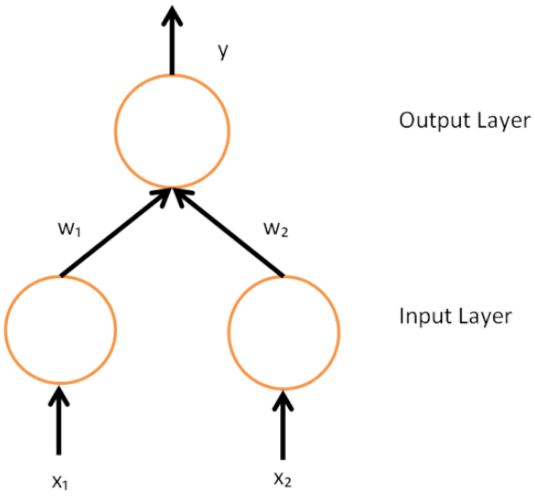
unsatisfactory recommendations. This is partly because of the fact that users' tastes and musical needs are highly dependent on a multitude of factors" []. My plan originally was to "address the issue of personalisation by using algorithms and methods that have a good social and psychological basis"[prelim]. I am well aware that the variance in users tastes of large statistical significance[musical personality ref], therefore I will further evaluate my approach by user testing, after implementation and alteration of the core recommendation system.

Neural Networks and Deep Learning

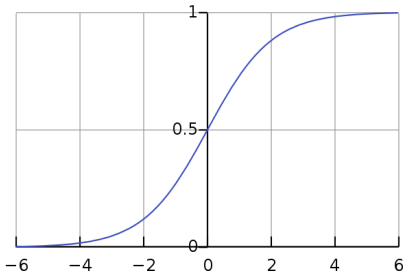
It is worth noting the current success of 'Deep Learning' in this field. In this section I will briefly explain the process of Deep Learning, achieved by training 'Deep Neural Networks' (*DNNs*) and it's comprising algorithms and functions, though skipping over some details, as I am not needing to implement a model from scratch. Then, I will provide some examples of where the concept is currently used in large scale commercial Music Streaming platforms, for the purposes of recommendation.

Neural Networks are biologically inspired Machine Learning models comprised essentially of multiple 'neurons' (which function individually similarly to Logistic Regression described below), connected together, as an architecture fit for the task at hand. The inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen, defines a neural network as:
"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.
[Caudill, 1989 in 3]

A simple 2 layer Neural Network, from a high-level, can be visualised as such:



A neuron takes a numerical input, and gives an output on the basis of an 'activation function', often a Sigmoid function (characterised by the S shape of the curve); typically, Logistic Regression (LR), essentially a 'Single-Layer Perceptron', uses the Logistic function:



Here, in the case of classification, a decision boundary is drawn (e.g class 1 if Y > 0.5, class 0 if <=0.5).

Perceptron function: Y' = B + W X

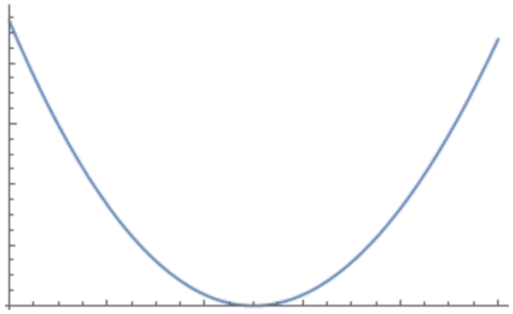
Finding the 'latent' value to pass into the Activation function requires a value of latent variable Y' in function 1. This can then be inputted to the activation function in order to output the classification of the original input. Here, B represents the 'bias' (essentially the Y axis crossover in LR input function), W the weight and X the input value to the perceptron. The activation functions help the Neural Network become a nonlinear function approximate rather than a linear one [3].

Gaining accurate results requires 'training' of the model. This process requires an algorithm called Gradient Descent; called this way because the aim is to literally descend along the gradient towards the global minimum of the error function. A learning rate (α) is applied to adjust to the weights of the function in question (sign dependent on the derivative i.e direction of the gradient). In Neural Networks, this is called *Backpropagation* [7]*;* this combines Gradient Descent with the Chain Rule; computed in a modular fashion to update the weights of the network based upon the values of error and α. More formally in the case of using the sigmoid function in a 'Feedforward' Neural Network: It has a continuous derivative, which allows it to be used in aforementioned Backpropogation.This function is also preferred because its derivative is easily calculated: [17]

$$f'(x) = f(x)(1 - f(x)).$$

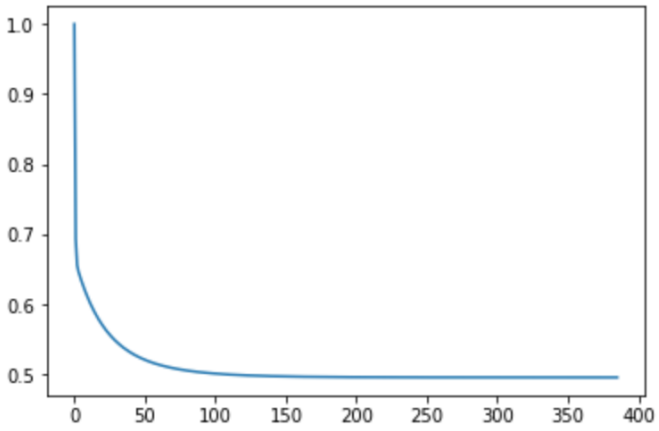
(The fact that the logistic function satisfies the differential equation above can easily be shown by applying the Chain Rule [8].)

The number of times the Backpropagation algorithm is repeated is commonly known as the epoch value. If we perform Backprop/Gradient Descent on an error surface like so (parabola):



with a sufficiently small value of α , it is guaranteed to converge to the global minimum in finite time (this is true for any convex function, as there lies only one minimum), like so:

```
Converged with  385 iterations! 0.496334647201
[[-2.90296148  1.0606745 ]]
```



Deep learning is when multiple neurons that comprise ‘hidden layers’ exist between the input and output layers of neurons, the network can therefore learn complex ‘multi-layer’ problems, where the weights and bias are updated distinctly with each hidden layer. Often, as more complex problems and thus networks are designed, the error ‘*surface*’ (multi-dimensional function) lies on increasingly higher dimensions and with an increasingly complex shape; when this happens, often more adaptive heuristics than ‘vanilla’ gradient descent are needed. An example of one of these heuristics would be *Adaptive Momentum*[11] – it is worth noting that it is not often efficient to find the global optimum on these complex surfaces; there are many ‘*local minima*’ that can be preferential, as long as they are not very shallow, i.e. still a low error.

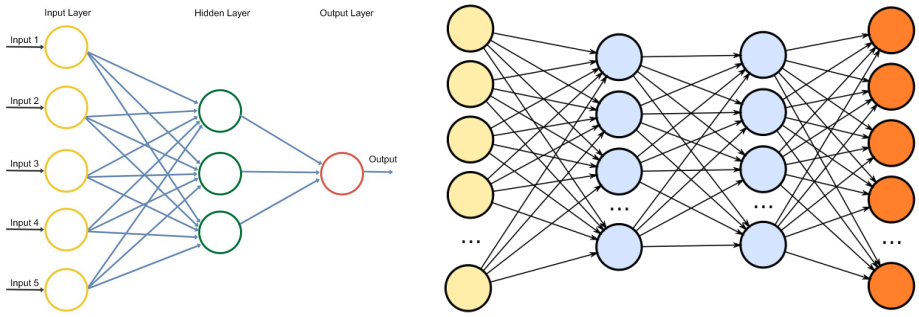


Figure 1: A Deep Feedforward Neural Network architecture, hidden layers can be stacked like in the right image.

Deep learning is used prominently by the multimedia (thus music) streaming website Youtube, which, in my opinion, offers the most interesting and gratifying recommendations of all streaming platforms [54]. Youtube's algorithm uses two Neural Networks, one for *candidate generation* and one for *ranking*. The candidate generation network is for personalised CF that returns a select subset A of videos from the millions that are in the universal set. The ranking network uses video features from extraction and the users relationships to the videos in the subset A. The architecture of both is similar; Rectified Linear Units (ReLUs) are used as the activation instead of the classic Sigmoid function for input data, this is advantageous as they are more efficient for deep networks [53]. SoftMax (essentially logistic regression for multiple classes) is used to train the network and finally Nearest Neighbour to serve the recommendation (similarly to my proposed approach). Importantly, the success of Youtube's recommendations [47], shows how *item-to-item* and *user-to-user* can be combined; a hybrid MRS.

On the semantic side, there has been a good amount of research. Using Neural Network vectorization method Word2Vec has been explored in a playlist context [49], and is also being used by commercial state-of-the-art (SoTA); "[Spotify] Discover Weekly is entirely powered by collaborative filtering, in particular a few extensions to Word2Vec that the machine learning team in NYC built."[21]. My supervisor originally suggested Word2Vec as a way of dealing with semantic information, as financial services had been using it to scrape forums [37]

Word2Vec and Doc2Vec

Historically, '*Bag of Words'* (BOW) models have been used to achieve numeric representations of text documents, but BOW falls short on important aspects such as word ordering; "thus different sentences can have exactly the same representation, as long as the same words are used". It also falls short on semantic meanings of words, "for example, "powerful," "strong" and "Paris" are equally distant… despite the fact that semantically, "powerful" should be closer to "strong" than "Paris." "[13]. Doc2Vec is heavily based on its predecessor – Word2Vec; which generates representation vectors out of words using 2 architectures similar to Feedforward Neural Networks aforementioned [16]; namely Continuous BOW (CBOW) which predicts a word

based on the context, and Skip-gram which predicts the context (surrounding words) given the current word. “Such representations, encapsulate different relations between words, like synonyms, antonyms, or analogies, such as this one:” [4]

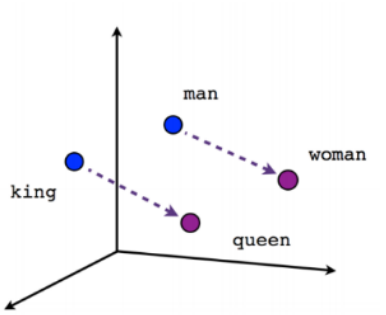


Figure 1: A classic case study in word vectors. King to queen is like man to woman; V(queen-king) ≈ V(woman-man)

Doc2Vec is an extension on Word2Vec - in addition to the set of word vectors, Mikilov and Le propose a ‘Paragraph ID’ vector. In Doc2Vec, like Word2Vec, there are two methods: Distributed Memory (DM) and Distributed BOW (DBOW). DM attempts to predict a word given its previous words and a paragraph vector. Even though the ‘context window’ moves across the text, the paragraph vector does not (hence distributed memory) and allows for some word-order to be captured. DBOW predicts a random group of words in a paragraph given only its paragraph vector (see Figure 2)[13][14]

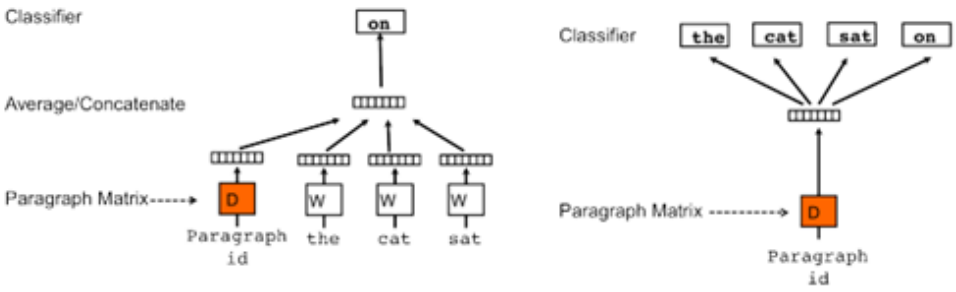


Figure 2. Left to right: classification with DM and DBOW respectively

The workings of Doc2Vec are very similar to a DNN. There are two weight matrices `W1` and `W2` for input and output mapping. In Doc2Vec – vectors, representing words and the paragraph via a feature vector, in both the matrices are updated, via the process of Backpropagation

aforedescribed. The output layers result in a latent vector space to describe the wordings and furthermore paragraph embeddings (taking into account word order and context), in the context of a large document information corpus. The algorithm yields SoTA results as compared to previous Bag-of-words methods in text classification and sentiment analysis[13]. Doc2Vec itself is an unsupervised method, it deals with unclassified data; i.e. there is nothing to prove the accuracies of the resultant vectors – it relies on the quality of the model and architecture itself.

It is worth noting that due to the DNN like architecture, training such a model can be a difficult task. Thus, it uses *Stochastic* Gradient Descent in order to reduce the error in the paragraph and word vectors during training; “At every step of stochastic gradient descent, one can sample a fixed-length context from a random paragraph, compute the error gradient from the network in Figure 2 [referring to left, PV-DM] and use the gradient to update the parameters in our model” [13]. This means that results will almost certainly differ when re-training the network, even when the parameters for the model are the same.

I will be using Doc2Vec here to vectorize music reviews in order to utilise rich semantic insight encoded in resultant vector spaces, and thus use numerical methods to serve a recommendation based upon a ‘seed’ release.

Chapter 3: Specification

3.1 **Hypothesis for review based recommendation**

- Creation of artist networks through resultant word vector similarity (in a similar fashion to some CF models).
- Popularity is much less of an influential factor, with the context and semantics of the reviews being far more influential.

This would also partly address the *‘Cold start’* problem of CF, whereby if a user has listened to only 1 or very few releases, there can be good recommendations served regardless of them fitting in to small enough CF ‘neighbourhoods’ to get recommendations of a reliable quality.[1]

3.2 **Aims**

This project straddles between being a research project and a software project. The web application to be created for the user-testing is a substantial work. The overall aim of the project is testing and evaluating the formulated hypothesis above by means of White-box testing on the Machine Learning model and data collection through the Black-box testing. If this is successful, then the the final, debugged web application can stand on its own as a recommendation and discovery platform.

3.3 **Requirements**

There are some clear requirements in particular for the web application:

- Allow the user to select a ‘seed’ to form the basis for the recommendation to be made.
- Serve the recommendation using the loaded Machine Learning model.
- Allow the recommended release to be played.
- Do this a number of times (3).
- Serve questionnaires to each user to assess their musical background/personality and to get feedback on the recommended releases.

Chapter 4: Web Application Design

This section outlines the design for the web application.

Questionnaire design: likert scale questions (scale of 1-5) and some background variables; Nominal: Name (to keep a unique key for each response – Forename and initials), Gender. Ratio: Age.

4.1 **High-level Structure**

Page 1) *Pre-study survey*. Assesses the users musical background and personality with around 5 questions . i.e. how experienced they are with music and how important music is to them.

Instructions to the user: “Please fill in and submit the questionnaire, then press begin testing”.

Page 2) Start of main *‘App Phase’*;

Instructions to the user: “Please select 3 artists you listen to from this list”. The list is retrieved from UNIQUE artists in the intersection of releases that have a vectorised review in my model, and that have a MusicBrainz - Spotify mapping. Each artist in the list has a hyperlink; this link parses the URI of a release from the artist that is contained in the corpus.

Page 3) *Serve Recommendation*. This is done by using the parsed URI in a Spotify player; loading 30 second previews of each track in the release. The idea is to serve this page and thus a recommendation 3 times.

Page 4) *Post-study survey* for data collection. Finish.

4.2 App Phase

Here I present an overview and strategy for how the recommendations will be served (Pages 2 and 3 aforementioned):
According to the respective JSON file for each review from the corpus

- ➢ Get Spotify URIs,
- ➢ list all *distinct* Artists from the corpus,
- ➢ list the resultant list of artists in alphabetical order to the user,
- ➢ upon selection get the first or perhaps most popular release,
- ➢ get and serve recommendation from 'nearest neighbour*'* in the Doc2Vec *Cosine Similarity* list.

Chapter 5: Methods and Implementation

Given that I have previous programming experience in Python for Machine Learning algorithm implementation, data processing and statistical analysis, I knew that it would be an adequate choice of language for this project. Python is perhaps the most popular language for Data Science. There are many libraries available, the most useful ones for this project include Gensim for the Doc2Vec model, Natural Language Toolkit (Nltk) and Flask for the web application black-box testing, amongst many other libraries that help achieve data processing tasks.

This chapter details the main libraries and how they were beneficial, and the methods used to implement the design and specification described in the previous chapter. It covers most of the implementation process, focusing on the most interesting, challenging and complex tasks. I have included some shortcomings and changes of methods, but I have tried to keep this to a minimum.

5.1 **The Machine Learning Model; 'engine' of the MRS**

Doc2Vec provides an in-built function for calculating the Cosine Similarity (Cos Sim) between 2 latent vector spaces which encapsulate the words of the documents in the corpus. One option was to use a pre-trained model, this would have a rich semantic 'knowledge base' in order to contextualise the input information. In the end, I decided it was better to use a corpus with words which are more relevant to music reviews, this would in effect be more efficient and create a more 'focused' model. This metric of Cos Sim provides an incredibly rich one-dimensional semantic data space, in the study of feature engineering, one could consider this optimal. I can then begin to serve recommendations based on distance in this dimension; a line of semantic similarity between music releases. In the future, if this method proves successful, I can potentially enrich the amount of data available by using other APIs or '*web scraping*' (retrieving text from the web by accessing the HTML) music review sources.

The Cosine Similarity Function

To calculate the Cosine similarity between two latent spaces containing vectors one must first take the dot product -- defined as:

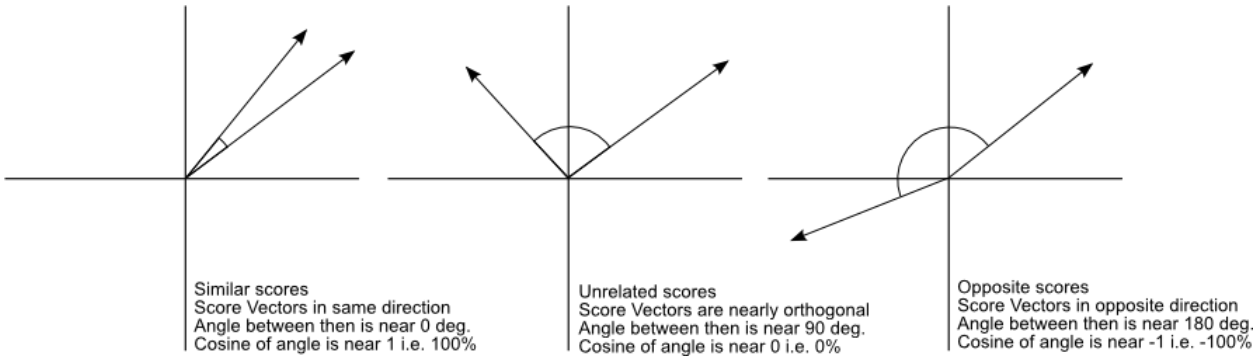
$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Then, we can take the identity of the dot product, and rearrange in terms of cos(theta):

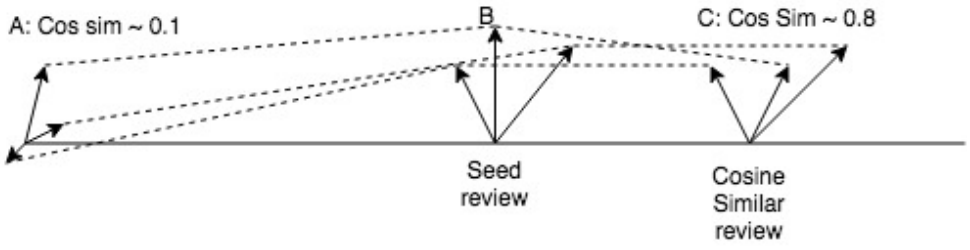
$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

We can see this in visual form like so:



As aforedescribed, Doc2vec uses vectors to represent paragraphs and words in documents, so in this case the cosine similarity could be visualised like this (as aggregate):



NN returns C

“Note that even if we had a vector pointing to a point far from another vector, they still could have an small angle and that is the central point on the use of Cosine Similarity, the measurement tends to ignore the higher term count on documents. Suppose we have a document with the word “sky” appearing 200 times and another document with the word “sky”

appearing 50, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, which is what matters when we are comparing documents." [32]

According to Gordon Mohr – consultant at RaRe Technologies, the Cosine Similarity function by default uses just the 'paragraph vectors', created by the Deep Neural Network like architecture, which by default uses the 'PV-DM' algorithm aforedescribed. [12]

5.2 **Data Retrieval and The Back End of the MRS**

The back-end, also referred to as the *data-access area,* is where the relevant data is served to the application. This section describes the trials, tribulations and successful methods of gathering the right data for the project.

Many Data Scientists will attest to the success of a Machine Learning model being largely dependent on the data which you supply to it, and of course the right model for the data [25]. Rationally, the first step for development of my model was to gain access to the right data for the task; as mentioned in the design specifications, the data would be comprised of music reviews. I found the JSON encoded *'Data Dumps'* of reviews of music releases from *https://critiquebrainz.org/* (an open-source database as part of the *https://musicbrainz.org/* 'family' of sites) and began to research the best way to *'*vectorise*'* the textual (ASCII) review content.

There was some pre-processing to be one; the JSON files for each review contained data such as the superfluous; user data etc. and the essential (for the later 'linking' of the data) *'Unique Identifier'* MBIDs for each release, as well as the review itself.

Initially, knowing that I had some user testing to do, I followed a tutorial which my supervisor recommended [51], in order to set up my own server on the University's network. I have been successful in setting up my own stack on top of Ubuntu 16 and serving HTML via Python and MongoDB.

After this, for some time I was trying to access the CB database by means of the package manager – Docker, in order to get reviews for my Doc2Vec model, I hacked about for 2 days unsuccessfully, only to later find the JSON dumps from late 2016, consisting of a corpus of 8475 reviews. I realise that I was probably not the only one finding the Documentation a bit cryptic and hard to understand, hence the 'pull request' which the developers opened a day prior to my work [55].

After researching tutorials on the internet to help be to get started with Doc2Vec, I managed to successfully train my Doc2Vec model, thanks to [15].

First I import all relevant libraries and dependencies;

```
import gensim
from nltk import RegexpTokenizer
from nltk.corpus import stopwords
from os import listdir
```

Then, we create a list that contains the name of all the text files in the data folder – this will become the corpus:

```
docLabels = []
docLabels = [f for f in
listdir("/Users/jasperkirton/Documents/gSMiths/MajorProj/Train/") if
f.endswith('.txt')]
```

Text cleaning and Natural Language Processing with NLTK

The next task is to clean and *tokenize* the data in order to be inputted correctly into training the Doc2Vec model. We do this using the NLTK; first we split all words into tokens using the following line function (this discards punctuation):

```
tokenizer = RegexpTokenizer(r'\w+')
```

Then, the following set is comprised of '*stopwords',* common words with negligible semantic and contextual meaning like 'to', 'and' 'the' etc.

```
stopword_set = set(stopwords.words('english'))
```

This function does all cleaning of data using the two objects above:

```
def nlp_clean(data):
   new_data = []
   for d in data:
       new_str = d.lower()
       dlist = tokenizer.tokenize(new_str)
       dlist = list(set(dlist).difference(stopword_set))
       new_data.append(dlist)
   return new_data
```

This 'container class' contains 2 functions to iterate over documents and return the relevant file names as the respective label:

```
class LabeledLineSentence(object):
   def __init__(self, doc_list, labels_list):
       self.labels_list = labels_list
       self.doc_list = doc_list
```

```
   def __iter__(self):
       for idx, doc in enumerate(self.doc_list):
           yield gensim.models.doc2vec.LabeledSentence(doc,
[self.labels_list[idx]])
```

Call the function to clean the data before training:

```
data = nlp_clean(data)
```

An iterator is then returned over all the documents/files:

```
it = LabeledLineSentence(data, docLabels)
```

Then begins training. I initiate the Doc2Vec model with fairly small learning rate (alpha), but enough to decrease gradually over a number of iterations, this helps increase 'exploration' by the training method and find a thus good minimum on the error surface. A large vector size helps to increase model accuracy, although I now realise corresponds to the feature vector (see methods) dimensionality, which could be a downfall of this (see '*curse of dimensionality',* where more dimensions make the model much harder to train correctly [56]) :

```
model = gensim.models.Doc2Vec(size=300, min_count=0, alpha=0.025)
model.build_vocab(it)
```

This is actually a naive approach; (around 2000 epochs in total) however it produced some good results:

```
# training of model
for epoch in range(100):
   print('iteration '+ str(epoch+1))
   model.train(it, total_examples=model.corpus_count, epochs=100)
   model.alpha -= 0.002
   model.min_alpha = model.alpha
   model.train(it, total_examples=model.corpus_count, epochs=100)


# saving the created model and confirm
model.save('doc2vec.model')
print('model saved')
```

A challenging task was to extract the review text and exporting to separate, training .txt files in batch. Thanks to reading [38], I was able to semi-successfully implement the script, I then needed to recursively iterate through all review directories within the JSON dump. Part of the challenge is dealing with some fairly odd hexadecimal directory organisation, so there are string conversions with the iteration in order to complete this task efficiently;

```
import json
from os import listdir
docs = []
dirIter = [0, 0]  # stores decimal values for iteration
counter = 1

for i in range(16):  # for each 16 folders
       for j in range(16):  # max 44 files in subfolders?
           dirIter = [i, j]  # current location
           print(dirIter)
           dirIter = ''.join("{:01X}".format(a) for a in dirIter)  # convert
to hex for indexing
           print(dirIter)
           # index into reviews folder using hex
           docs = [f for f in listdir(

'/Users/jasperkirton/Downloads/critiquebrainz-20160611-cc-by-nc-sa-30-json/rev
iews/' + str(dirIter[0]) + '/' + str(dirIter[0]) + str(dirIter[1])) if
f.endswith('.json')]

           # Printing the List of all Json Files loaded into the Memory
           for x in docs:
               print(x)
           for x in docs:
               with
open('/Users/jasperkirton/Downloads/critiquebrainz-20160611-cc-by-nc-sa-30-jso
n/reviews/' + str(dirIter[0]) + '/' + str(dirIter[0]) + str(dirIter[1]) + '/'
+ x) as data_file:
                   data_item = json.load(data_file)
                   b = data_item['reviews']
                   for item in b:
                       name = '' + str(counter) + '.txt'
                       file = open(name, 'wb')
                       output = item['text']  # this field changes from
'entity_id' to 'text for MBID/review retrieval
                       output = " ".join(output.split())  # this concatenates
the paragraphs
                       counter = counter + 1
                       file.write(output.encode('utf-8'))
                       file.close()
```

Upon initial white-box testing of the algorithm and model I had a suspicion that the Doc2Vec algorithm was only vectorising the first paragraph of the review. This led to the task of concatenating the paragraphs with some JSON processing with the additional line seen above:

```
output = " ".join(output.split())
```

Mid implementation - thoughts about text summarisation

My supervisor and I were discussing text summarisation as previously mentioned – In doing some research [43], I think it will not be necessary as key words, such as names of artists, tracks etc. will most likely not be included in the summarisation – which are very important in having a Collaborative Filtering type influence on the model; i.e. by creating resultant artist networks from artist name mentions in different reviews (see hypothesis aforementioned). I am doing some cleaning from NLTK toolkit before training my Doc2vec model, so I think it would be more useful to optimise this instead, which essentially serves a similar purpose as text summarisation.

Pickling the D2V model

For the next stage of development (towards Black-Box testing), my supervisor and I discussed the ways in which I can best user test the Doc2vec based MRS, discussing methods from the study of Computing and beyond; we came to the conclusion that I should '*pickle*' the Doc2vec model, and create a web app around the Python '*Flask*' (*http://flask.pocoo.org/*) framework, that will serve the recommendations, and collect data from survey and user testing. My plan of action was somewhat informed by this useful book [40].

" 'Pickling' is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream is converted back into an object hierarchy. " [2]

Having so far successfully pickled my Doc2vec model (as an object), I implemented a simple Flask web app whereby a user can enter some data via a form, and proposed some pre-study and post-study survey questions. These form a good basis for assessing a users background, and assessing how much they rate the served recommendations [see in appendix].

A question that arose around this time was: *Is pickling the Doc2Vec model really the best way?* For longevity, expansion and Linking of the data it could be more beneficial to do some kind of database – is this realistic with latent vector spaces though? A: No, the latent spaces require a large amount of data to describe them. The beauty about this method, too, is the potential to write some kind of sequential chain of command (perhaps weekly), re-training the model based on new reviews and then repeating the '*pickle, transfer, un-pickle'* pipeline.

One of the biggest obstacles to overcome here was the *'full-stack'* web app development learning curve, whilst I know some HTML, have used Flask briefly prior to this project and am very willing to learn, I have had difficulty working with external servers and SQLite, which I was

trying to use in order to store user data and integrate MusicBrainz data into the web app. I later found solace in the 'Pythonic' way of reading and processing data bases in the form of data sheets and data dictionaries, this allowed me to set my learning of database languages to one side in order to overcome the challenges presented in this stage of development.

As previously mentioned, I originally thought this to be driven by a database language like mongoDB or SQLite. Prior to being sent the Spotify URI mappings, I spent a lot of time trying to figure out the best way to access a select set of MusicBrainz metadata fields for the set of releases in my corpus. I had a data dump of 'Releases' which I uncompressed into a 100GB JSON file, obviously, this is beyond the scope of RAM of every high-end laptop I can think of (I have 16gb of RAM available on my Macbook Pro). This led me to think of how I can parse in the JSON using a buffer, however when trying to access select fields, the JSON object didn't read anything beyond ID; when I tried to use exclusively the 'id' field, I still got a memory error. Fortunately, I overcame these problems by going into the MusicBrainz development Internet Relay Channel (IRC) and describing my issues, as a result of which i received the MB-Spotify URI mappings.

I have written a number of functions comprising '*MappingNetwork.py*', it involves creating multiple Python Dictionaries using the mappings from music releases on CB to Spotify album URIs, then, sorting by MBID's that were exclusive to my corpus used in the Doc2Vec model.

First, I import the relevant libraries to the script: [check all code is up to date in this section]

```
import requests
import json
import time
import os
from pathlib import Path
```

Here, I create an empty dictionary 'artistDict' and list 'MBIDcorpus'

```
artistDict = {}
MBIDcorpus = []
```

Then, I create search indexes from the corpus by iterating through the directory where the MBIDs are stored and respectively indexed (same as in the training directory). This involves encoding the directory and then decoding the name of the file and reading the contents:

```
# Create search indexes from corpus:
directory = os.fsencode("MBIDs")
# iterate through idDict, append:
for file in os.listdir("MBIDs"):
   #if filename.endswith(".txt")
   filename = os.fsdecode(file)
   filenum = filename.split(".txt", 1)
```

```
    cur_id = Path("MBIDs/" + filename).read_text()
    MBIDcorpus.append(cur_id)
```

Here, I read the mappings which were provided to me by the MusicBrainz dev team, and split the middle 2 columns into 2 lists; using the useful ‘zip’ function to combine the 2 lists into a dictionary with ‘MBIDs’ as the key and ‘AlbumIDs’ as value pairs:

```
idList = []
with open('mbspotify-mapping-2018-04-18', 'r') as f:
    idList = f.readlines()

MBIDs = [x.split()[1] for x in idList]
#print(MBIDs)
AlbumIDs = [x.split()[2] for x in idList]
#print(AlbumIDs)
idDict = dict(zip(MBIDs, AlbumIDs))
```

Here, I needed to create 2 new lists in order to save the ‘clean data’, whereby the MBIDs and spotify AlbumID mappings are only stored if they are contained within the MBID corpus. If it is contained; it writes the URI from the relevant point in the file followed followed by ‘\n’ (a new line), if not, it writes unavai, followed by ‘\n’. It is essential to keep what are called ‘separators’, in this case ‘\n’, in order to maintain computational readability in data sheets.

```
MBIDsPost = []
AlbumIDsPost = []

for i in range(len(MBIDcorpus)):
        if idDict.keys().__contains__(MBIDcorpus[i]):  # if MBID from corpus is in the
mapping set
            # save uri from the relevant part of the field
            uri = idDict[MBIDcorpus[i]]
            #decoded_uri = encoded_uri.decode()
            #print(uri)
            MBIDsPost.append(MBIDcorpus[i])
            AlbumIDsPost.append(uri)
            less.write(str(uri) + "\n")
        else:
            less.write("unavai" + "\n")
```

The results of this were disappointing in the fact that only 72% of the releases that comprised my corpus had a completed MusicBrainz - Spotify mapping. The hypothesis for this is that the results will be worsened by the fact that ~¼ of recommendations will not be the most sentimentally / semantically / contextually similar, given the small section of the music release universe that the corpus contains, this is a significant downfall.

Then, I queried the batch of respective URIs to the Spotify API and filled up a dictionary with the artist names. I am ‘sleeping’ the execution of this code by 3 ms, this is in order to keep the ‘GET’

request gate open (i.e not be blocked by Spotify for over-requesting), I then sorted them by distinct values in alphabetical order using Python's '*Ordered Dict*':

```
idDictPost = dict(zip(MBIDsPost, AlbumIDsPost))
f = open("sp_uri:artist_name-mapping.txt", "a")
artists = []
URIs = []
for sp_id in idDictPost.values():
   if sp_id.__contains__('spotify:album:'):  # for the hole in data sheet
       uriPost = sp_id.split('spotify:album:', 1)[1]
   else:
       uriPost = sp_id
   print(uriPost)
   time.sleep(0.03)
   #  avoid over-requesting the API, leave for 45m - leaving ~15m till token expires
   headers = {'Authorization': 'Bearer
BQA-xiIrjHIKcPHixX2eUq55Gul5S9LZtznvc8Ed0pGAf11VbPbaWVemdFB9_eZlP47M1dgvCBpxaCF9ZwQ'}
   r = requests.get('https://api.spotify.com/v1/albums/' + str(uriPost),
headers=headers)
```

There was a particular problem I had with the hexadecimal (hex) input that the Spotify API requires in order to get the Authorization key, seen above after 'Bearer'; a new 'access token' is generated for the hex key every hour, so during the ordeal of obtaining the right data from the API; I had to call this CURL command multiple times:

```
curl -X "POST" -H "Authorization: Basic
NGQxYjQ4MTkwZmNmNDk5MGJmODMwMmMzOWJhOTg2MTA6OGRjYmVjM2M0MThiNDgzM2FjYz
hlMzUyZTgzM2JhZGU=" -d grant_type=client_credentials
```

The hex key here was generated from converting my Client ID and Client ID from [base X] when registering my application, as this: <clientid:clientsecret> – whereby everything contained within < … > is to be hex encoded (something which I found to be unclear in the documentation) [50]

Here, by loading in the JSON formatted data by json.loads(r.text) , we access the 'name' field within 'artists' of the release which is returned, through a 3 dimensional array format. We append the artists name to the list, whilst also appending the URI to another list, in order to write the respective artists the URIs in a file (which can later be uploaded to the server).

```
   sp_data = json.loads(r.text)
   name = sp_data['artists'][0]['name']
   artists.append(name)
   URIs.append(uriPost)
   #artistDict.update({str(uri): name})
   f.write(str(uriPost) + "    " + name + '\n')
```

```
uri_artistDict = dict(zip(artists, URIs))

# Extract the dictionary into a list of (key, value) tuples.
t = [(k, uri_artistDict[k]) for k in uri_artistDict]

# Sort the list -- by default it will sort by the key since it is
# first in the tuple.
t.sort()
# Reset the dictionary so it is ready to hold the new dataset.
d = {}
# Load key-values into the dictionary. Only the first value will be
# stored.
for k, v in t:
   if v in d.values():
       continue
   d[k] = v
```

5.3 **The Front End**

This is the ‘user side’ of the web app, i.e what the user interacts with. Additionally it handles parsing of data from the Back-end. The Front-end uses Flask -- a Python ‘microframework’ for HTML templating and writing functions to serve the web app. The application directory - hosted on ‘myserver’ by Goldsmiths IT - looks like this:

```
.git/
.gitignore
data/ distinct_artists, serveIDs.txt
pkl_objects/ model.pkl
static/ style.css
templates/ pre_study.html, app_phase.html,
release_recommendation.html, post_study.html
app.py
```

In an anatomical analogy; ‘app.py’ is the brain and skeleton of the app, the templates form the body, ‘model.pkl’ (the pickled Doc2Vec model) is the main organ and the data is the fuel in order for the body/app to function. I will start by explaining the main functions of ‘app.py’;

To successfully ‘pickle’ the Doc2Vec model in order to be loaded here and serve recommendations, it required matching the Gensim version on server with the version which I trained the model on locally (version 3.1).

I pickled the model using the Python *pickle* library (instead of Doc2vec.save as it produces multiple files), like so:

```
dest = os.path.join('d2vModel', 'pkl_objects')

if not os.path.exists(dest):
   os.makedirs(dest)

pickle.dump(d2v_model,
           open(os.path.join(dest, 'model.pkl'), 'wb'),
           protocol=4)
```

Moving the pkl object up to the server means first uploading it onto the University's server IGOR (the firewall disables us from being able upload it to '*myserver*' directly) . I used the Terminal (UNIX) command: `scp -r /path/model.pkl jkirt001@doc.gold.ac.uk:`
And then from IGOR: `scp -P 2256 model.pkl jkirt001@myserver:`
The model was then moved to relevant directory using the `mv` command.

It is worth noting that as *myserver* runs on Ubuntu; I had to do all directory management from command line and all coding in the command line text editor *Vim*, which was a learning process in itself.

The '*.pkl'* model is loaded in the Flask web app like so:

```
####### Preparing the d2v model
d2v_model = pickle.load(open(os.path.join(cur_dir,
'pkl_objects/model.pkl'), 'rb'))
with open(os.path.join(cur_dir, 'data/serveIDs.txt'), 'r') as f:
        URIs = [line.replace('\n', '') for line in f]
stage = 0 # initialise our rec counter
artists = []
```

The following code creates an ordered dictionary from the distinct_artists data sheet, this is essential in order to list them in alphabetical order as they appear. Thanks to [39]

```
with open('data/distinct_artists', 'r') as f:
        parsed = ast.parse(f.read()) # security risks?

first_dict = next(node for node in ast.walk(parsed) if
isinstance(node, ast.Dict))
keys = (node.s for node in first_dict.keys)
vals = (node.s for node in first_dict.values)
```

```
od = OrderedDict(zip(keys, vals))
```

Here is the '*app phase'*, i'm passing in the stage variable to keep track of. I initialise artists as the keys to the ordered dictionary created prior, in order to pass them through to the HTML and list them as links to the relevant artist.

```
@app.route('/app_phase/<int:stage>')
def appPhase(stage):
        artists = od.keys()
        return render_template('app_phase.html', artists=artists,
stage=stage)
```

The following code is the most complex function inside of the script, it mainly deals with serving the recommendation. The code deals with the output from the Doc2Vec function call, with the current Artist release as the input; it requires some list comprehension to find the most similar release. Then, we pass in the URI into the HTML template to be loaded by the Spotify player. [full code in appendices]

```
# View specific entry
@app.route('/release_recommendation/<int:stage>/<string:selected_artis
t>')
def release_recommendation(selected_artist, stage):
        check = False
        shift = 1
        cur_uri = od[selected_artist] # query into artist dictionary,
returns URI of corresponding artist release
        cur_index = URIs.index('spotify:album:' + cur_uri) + 1
        reclist =
(d2v_model.docvecs.most_similar(str(cur_index)+'.txt')) #returns a
list, first index is most similar
        rec = [x[0] for x in reclist][0]
        rec_file = str(rec)
        rec_index = int(rec_file.split('.txt')[0])
Stage += 1
return render_template('release_recommendation.html', uri=uri,
stage=stage)
```

The HTML templating is done via the Python templating engine Jinja2 [http://jinja.pocoo.org/]. This allows some Python style programming logic to operate within the HTML code. Here I provide 2 examples using it within the 'release_recommendation.html' file. An example of passing in the recommended URI to the Spotify player is like so -- reading the variable with the 'Expression' {{ … }} syntax:

```
<iframe src='https://open.spotify.com/embed?uri=spotify:album:{{uri}}'
width="300" height="380" frameborder="0" allowtransparency="true"
allow="encrypted-media"></iframe>
```

This following is an example of using conditional logic with the 'Statement' % … % syntax, in this case the purpose is to recognize if it is the last recommendation to be made in the sequence; if so it provides a button that links to the '*post_study.html'* (containing another Google Doc survey), else, it provides a button that links to the list of artists to choose from for the next recommendation.

```
{% if (stage == 3) %}
        <form action=/post_study>
                <input type="submit" value="Finish Study">
        </form>
{% else %}
        <form action=/app_phase/{{stage}}>
                <input type="submit" value="Next">
        </form>
{% endif %}
```

In 'app_phase.html', the following lists the ordered list of artists with a distinct and respective link (href) in HTML using a combination of the 2 syntaxes aforementioned:

```
{% for artists in artists %}
        <li><a href="/release_recommendation/{{stage}}/{{artists}}">{{
artists }}</a></li>
{% endfor %}
```

Post 1st Black-box test phase - Improvements on Implementation

The following section describes implementation to fix the problems discovered and highlighted by the 1st Black-box testing phase.

Text cleaning and NLP

In order to pass the sentences into Doc2Vec, we need to do some cleaning to enhance the training process and resultant vectorisation. My cleaning in the 1st training phase for the black-box was perhaps naive, the following seeks to address the possible issues:

The aims for the pre-training text cleaning are as follows:

- Remove URLs (Achieved in parsing JSON text phase)
- Try a different strategy for dealing withpunctuation in the tokens

The code added to "*json2text.py*" is as follows:

```
output = re.sub(r"http\S+", "", output)  # this removes URLs from the file
```

The following is the new strategy for tokenizing the words and the punctuation separately:

```
dlist = word_tokenize(new_str)
```

Retraining the model

- I was using for the 1st dev phase version 3.1 of gensim, so here I have upgraded on the server and locally to 3.4;
- I have changed min_count from 0 to 2. This parameter is a limit in which any words that occur in the corpus below this number are ignored; there may be some instances where artist/musician names are mentioned once in only 2 reviews (and thus only twice in total); i'd like to keep this connection apparent in the similarities.
- Changed 'size' (also deprecated) to 'vector_size' which works the same.
- The doc2vec.'LabeledSentence' is now deprecated, replaced with 'TaggedDocument' which works the same.

I retrained the model (which took moderately less time than before; ~20 mins rather than ~3 hours for the previous model), did some initial testing which seemed promising, then, I uploaded the new 'pickled' file to the server as '*model2.pkl*'. Finally, I removed some of the code which was dealing with the 'rogue' review (*5133.txt*).

Seed comparison

Spotify has API functions to serve seed recommendations, I could use this and do a 'SoTA' comparison directly, but I decided that it was perhaps unnecessary, due to comparative feedback that I put in my mid-study questionnaire.

Chapter 6: Testing

6.1 **White-Box**

Initial testing - success, thoughts and implications

I have gained some promising results, after training on a large corpus (555 reviews) – for example if I had been listening to 1: https://www.youtube.com/watch?v=9fCIVP03-_M . It would have recommended me 2: https://www.youtube.com/watch?v=naJ472sQnrU as the most cosine similar. This seems like a good recommendation. On the subject of similarity
Additionally it has a lovely capability of analysing sentiment – if we look at the first paragraph from each example:

1:
'This is all about Dvorak's chamber music, and if you doubt me, just listen to the slow movement of the Cello Concerto: Jean-Guihen Queyras and the Prague Philharmonia in rapt communication…we're just eavesdroppers on an intimate conversation. In the outer movements Queyras's instinctive musicianship and ravishingly beautiful tone are bewitching, and Belohlavek follows every imaginative twist and turn in his soloist's phrasing; they're glued together like the finest ballroom dancers.'
2:
'Long-term fans of [Sigur Ros] (http://www.bbc.co.uk/music/artists/f6f2326f-6b25-4170-b89d-e235b25508e8) could be forgiven for feeling a little nervous about Jonsi Birgisson's new project. The quartet he usually fronts possess many admirable qualities, but their international success owes much to a mystique greatly enhanced by lyrics that are gobbledegook to most. Singing in Icelandic has the useful effect of making you sound like the house band from a science fiction film, an in-built benefit Birgisson has decided to eschew with his first solo record, which is sung mostly in English. If the Sigur Ros spell is ever to be broken, this might be the moment.'

You can see here that these paragraphs are imaginative and descriptive. This supports my hypothesis, with semantic and emotive context being preserved and prioritised by the MRS.

White-box testing - Web app perspective

Upon successfully developing my web application. I did some initial testing from a user's perspective myself; whilst mostly being successful and serving some great recommendations there was some work to be done:

The spotify mappings were incomplete (which doesn't allow the system to function as well as it should). This meant putting a 'shifting' system in to check whether the mapping exists, and if not it moves the recommendation along to the next nearest neighbour (next highest Cos

Sim). An example of this would be the recommendation for Claude Debussy; it would have been Mozart (fitting in Genre and style), but as the mapping from the MBID to Spotify URI did not exist in the mappings I was given, it recommends a romantic folk-pop record which makes sense in terms of being ‘easy-listening’, but hardly matches up to the musical complexity and subsequent ‘richness’ of Debussy and Mozart; on a similarity measure of quality, this is not a very good recommendation, although there is some resemblance in instrumentation and sentiment, so it makes sense (perhaps if i compare another review here).

There also seemed to be a rogue review (which is just ‘Yes.’ followed by a URL to an image) which for some reason was throwing high Cosine Similarities to multiple Artist releases; if an empty vector is said to be 0, then the cos(theta) between the 2 vectors would be 1 and hence the latent spaces could be calculated to be very similar. it could have passed through the activation functions of the Neural Network resulting in multiple vectors for whatever reason, or, it could be that it has an empty latent space, and the Cos Sim function is not working properly for this special case. [*check this*] I used an extra conditional with the shifting mechanism to bypass this whenever it was returned by the ‘release_recommendation’ function.

I also notices that some recommendations (around 1 in 5) don’t make much sense at all, I realise that I should have filtered out all URLs before training the model, as they contain a lot of characters which could throw the model off considerably. This is a problem which I could have spotted if I had undergone a thorough data analysis prior to development on the main application; given the initial white-box testing results I was perhaps overly optimistic. A quote which seems fitting for a good lesson to learn here is this from [Feynmann, Cargo Cult Science, 1974] “*It's a kind of scientific integrity, a principle of scientific thought that corresponds to a kind of utter honesty--a kind of leaning over backwards. For example, if you're doing an experiment, you should report everything that you think might make it invalid--not only what you think is right about it: other causes that could possibly explain your results; and things you thought of that you've eliminated by some other experiment, and how they worked--to make sure the other fellow can tell they have been eliminated.”*

Additionally, there is the issue of some Spotify releases not being available in this country (presumably due to copyright issues), I don’t think this is really fixable without each user using a VPN (which is unrealistic).

2nd iteration

Whilst re-training and thus White-box testing the model after the first stage of Black-box testing, an interesting hypothesis arose; on the intersection of similarity and user ‘eclecticness’. This hypothesis came to mind when the ‘similarity’ between seed and recommendation started to increase to the point of artist matching; via new training methodology; such that Mozart would

recommend more Mozart (100.txt -> 1. 5445.txt 0.54 sim, 2. 2667.txt 0.53 sim). I have included the hypothesis in the 'Future Research' section.

After further refining of the training process; I have come to version 2.0 of the model. Some initial tests are as follows: Vinicius Cantuaria a 50s Brazilian singer would have recommended Joyce (another Brazilian singer) 1st, and a soundtrack to a 50s Jazz film 2nd — perfect!
A Mozart string quartet would recommend a Bach suite for solo violin 1st, and a late Mozart symphony 2nd – great!

This was promising, however I discovered that there were still problems similar to that of '*underfitting*' or '*overfitting'* in ML[1] (as certain 'features' were skewing the results too much), it is however very difficult to assimilate which is most applicable as this is not a classical machine learning problem; the recommendations (testing phase) is down to human judgment. A symptom of the problem, that I noticed from inspecting a 'bad' recommendation was: "collection" preceding "." occurring between 2 cosine similar reviews, although this was not the only similarity in the 2 reviews, this is probably an odd occurrence. It seems that my theory to separate punctuation could be a mistake… however with time running out, and the system working quite well overall, I decided to continue with Black-box testing.

6.2 **Black-Box**

For the Black-box testing, I asked many friends, family and colleagues to complete my study and surveys. Also, I put a 'HIT' study up on Amazon Mechanical Turk (MTurk), such that I can pay people from an international background to complete the study, in total across both iterations; just over 30 people completed the study from MTurk alone.

Statistics and Data Visualisations on black-box results

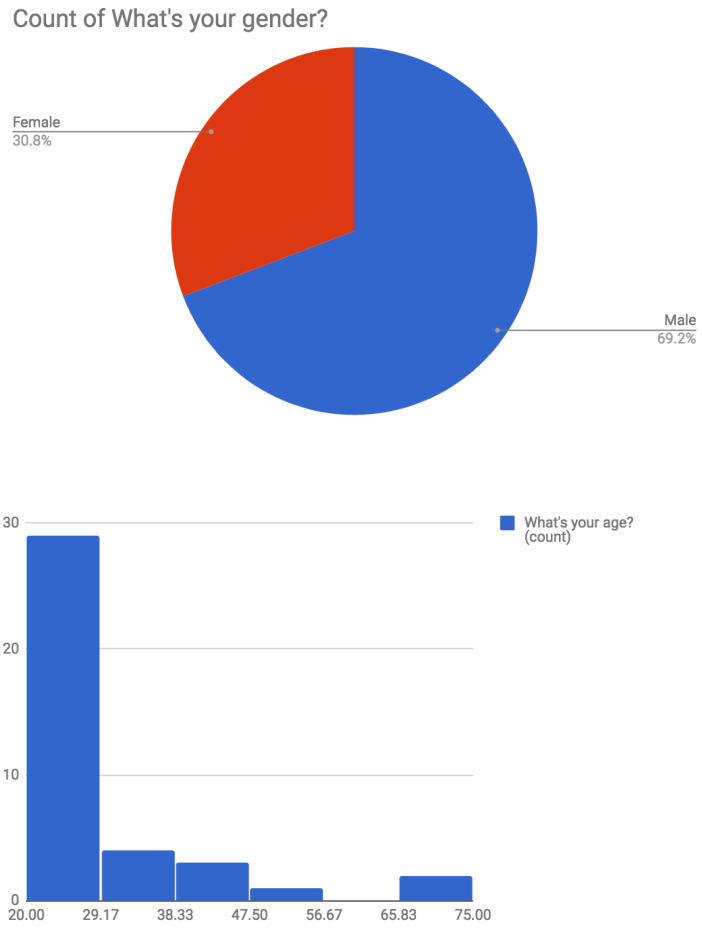
The following is the results of df.describe() in Python's Pandas library, graphs made in Google Sheets, and additional statistics for each questionnaire. There is an ongoing debate in statistics whether Likert scales should be treated as ordinal (order matters, cannot derive a mean [types of data) or interval (distance between values is meaningful[how to use the liekrt scale]), I can see valid arguments on both sides, so I have included 3 values of central tendency – averages: mean, median and mode; all 3 are useful insights given that users may answer likert scales differently. All floating point values are rounded to 1 decimal place.

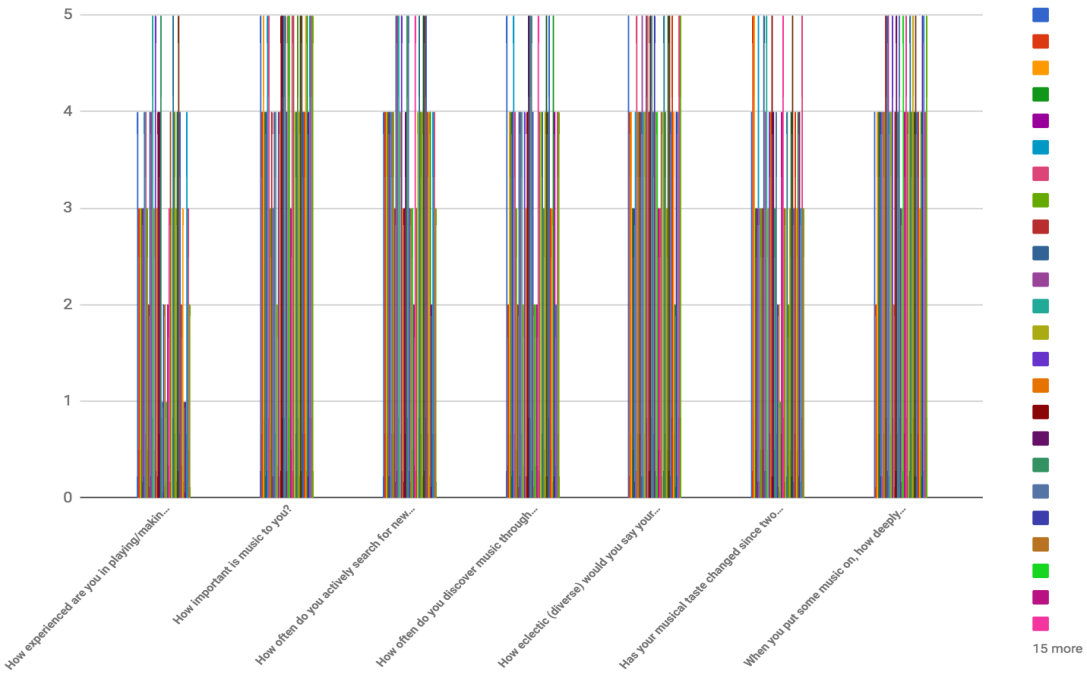
1st Iteration

Pre-Study questionnaire

[1] "Machine Learning Lesson of the Day – Overfitting and Underfitting ...." 19 Mar. 2014, https://chemicalstatistician.wordpress.com/2014/03/19/machine-learning-lesson-of-the-day-overfitting-and-underfitting/. Accessed 14 May. 2018.

I have cleaned the pre-study data; such that it corresponds to the people who successfully took part in the study. Then, I have created a Pie-chart to show the gender distribution, unfortunately it is more skewed towards Males than I would have hoped (just over ⅔). The age too would perhaps be more varied in an ideal study.





Here, each column represents a response, and the columns are grouped by each question. The questions are finished as follows (from left to right; ...“making music?”, “new music?”, “through being recommended tracks/releases?”, “ your music taste is?”, “two years ago?”, “deeply do you listen to music?”)

Some key observations here are:

- High variance of musical experience $\sigma^2 = \frac{\Sigma(X-\mu)^2}{N} = 1.3$
- Music is at least important (3) to the vast majority of people, and is very important (4, 5) to a significant proportion of participants (34/38 = 89%)
- There is not a clear difference in frequency of discovering music through searching and being recommended (both seem fairly common);
- There is quite a high level of eclecticness, similarly to importance.
- The participants generally seem to be quite deep listeners.

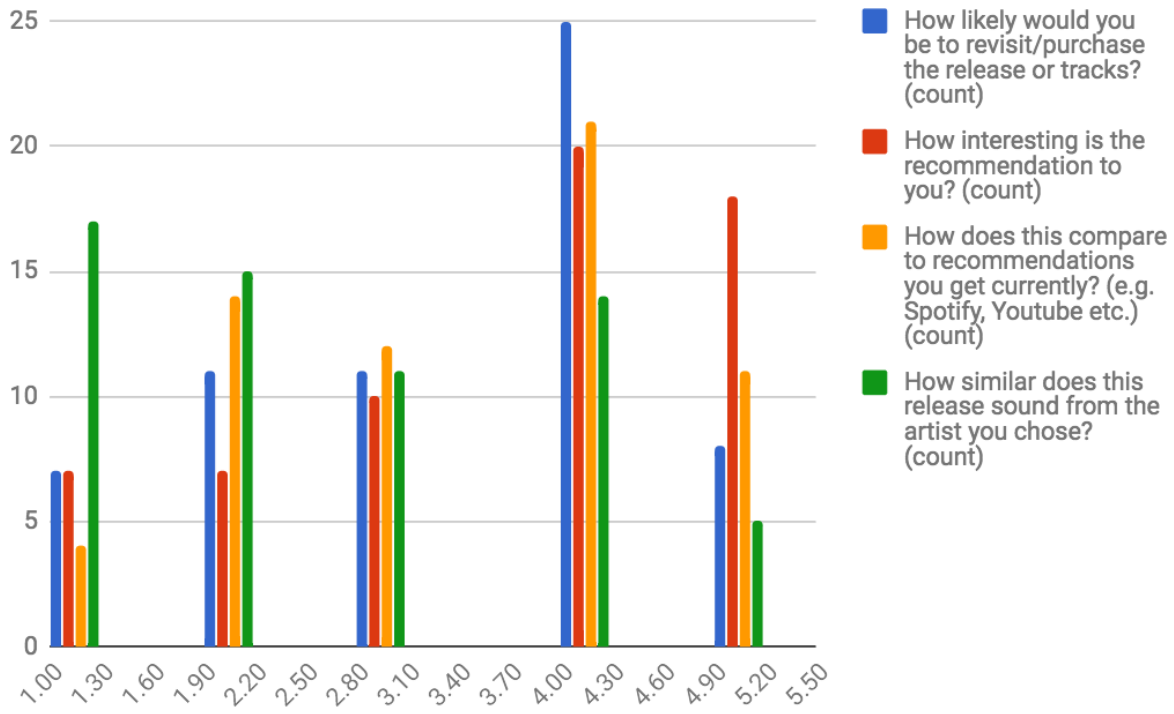
I have highlighted key values which support these observations in the statistics table below:

Responses	Age	Musical Experience	Musical importance	Active Search Frequency	Passive Discovery Frequency	Eclectic Level
Count	38	38	38	38	38	38
Unique	18	5	5	5	5	5
Mode (Freq)	26 (5)	4 (12)	5 (19)	4 (18)	4 (18)	4 (19)
Mean	29.9	3.2	4.4	3.9	3.7	4.2
Median	26	3	4.5	4	4	4

Responses	Musical taste change in past 2 years	Deep listening level
Count	38	38
Unique	5	4
Mode (Freq)	3 (13)	4 (19)
Mean	3.7	4.2
Median	4	4

Singular recommendation evaluation

Responses	Name	Likelihood of revisiting rec	Interest of rec	Compare to sota	Similarity to seed artist
Count	62	62	62	62	62
Unique	38	5	5	5	5
Mode (Freq)	Carlo ES (4)	4 (25)	4 (20)	4 (21)	1 (17)
Mean	-	3.3	3.6	3.4	2.6
Median		4	4	4	2



This histogram shows the frequencies of the responses on the Y Axis for the Likert scale values along the X axis. It highlights the modal distribution as seen in the statistics. We can additionally see that the interest of the recommendation is greatly distributed towards the higher values.

Notable responses | "anything else to add?":
Edmund OE: “Nice suggestion!”
Leon FE: “My music taste is weird and very explicit so it's going to be hard to nail down the stuff I really like.”, “Sorry - I don't like Alicia Keys!!!”, “Much better, I actually would listen to a few of these tracks! :)”
Deepa KG: “NICE SONG”
Gokul: “nice work”
Julie JP: “I did not enjoy the artist, nor did I think it related to my chosen artist. It was an odd choice.”, “I liked the music, but I did not feel that it related to Justin Timberlake.”
Tamil: “the very nice and lovely”
Suresh: “i like this music track very much”, “imitation track is nice”, “sad and lonely is good”
Dillon DM: “The player took a bit for it to actually play, but i got it to work eventually.”
John VJ: “Great tracks, I'm going to check it out.”

Post-study questionnaire

I have not visualised the data here, as there is not a lot to process; I think it is described sufficiently by the central tendencies, esp the Mean. Note that unfortunately not everyone who partook in the study completed the post-study questions.

Responses	Overall quality of recs	Likelihood of re-using service
Count	14	14
Unique	4 (no '1' rating)	5
Mode (Freq)	3 (6)	3 (5)
Mean	3.3	3.1
Median	3	3

"Anything else to add?" | Notable responses:

Edmund OE: "Nice idea for people who like a variety of music"
Leon FE: "I feel like my music taste is incredibly varied, but is very, very specific on a few artists over a wide range of genres, which is going to make me hard to model - there will be a lot of stuff I simply won't like."
Julie JP: "I liked some of the choices but did not feel like they connected to my chosen artists."
Dillon DM: "The service needs some work, most were different than what I picked."
Mohan: "Very good."

1st Iter Evaluation

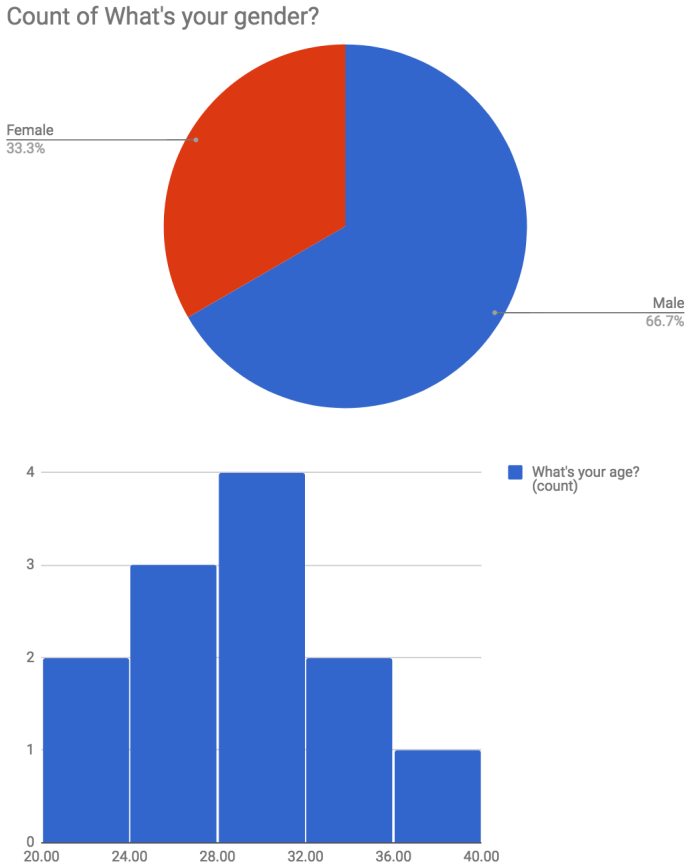
One of the main problems (apart from some seldom 'random' seeming selections) seems to be the issue of similarity, judging by some of the qualitative responses and the quantitative central tendencies in the similarity question (mid-study questionnaire). When I was checking / cleaning the post-study data, I saw the response from a friend who completed the study – Tal AH: I asked about the question: "Overall, how would you rate the quality of your recommendations" – as I knew he really liked his from his personal feedback to me and his mid-study ratings;
Myself to Tal: *"Did you really only rate the quality of the recommendations ⅖ at the end?"*

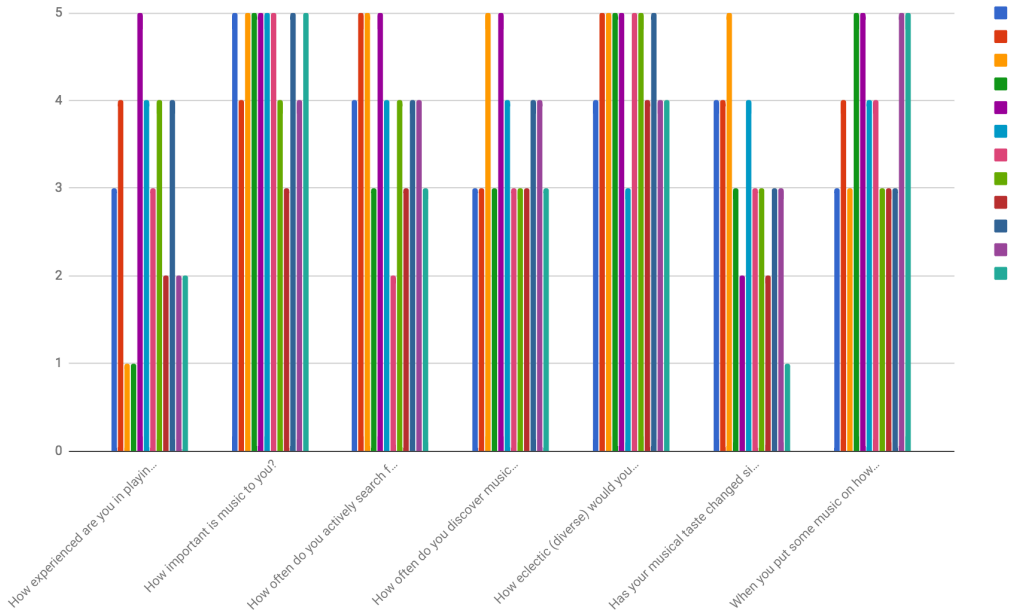
To which he replied (paraphrased); *"Yeah they were really good I just get just as good ones from spotify… oh yeah I thought I put a 3, definitely correct that."*

This is an interesting point to note that a measure of quality may be comparative against what a user considers to be the SoTA. To eliminate this ambiguity, in future I will rephrase the question more specifically. The questions in the mid study questionnaire(s) seem to be more specific in order to draw more definitive conclusions about the responses.

2nd Iteration

Pie, Histogram, Column charts for 12 users.





Here we can see a similar distribution to the 1st pre-study column chart.

Singular recommendation evaluation

Responses	name	1. Likelihood of revisiting rec	2.Interest of rec	3.Compare to SoTA	4.Similarity to seed artist
count	32	‘’	‘’	‘’	‘’
unique	12	5	5	5	5
mode(freq)	Rajavarman P (4)	3 (8)	3 (11)	4 (11)	3 (10)
mean	-	2.7	2.8	2.9	2.8
median	-	3	3	3	3

We can see that the central tendencies mode and mean of ‘similarity to seed artist’ are significantly higher than the previous testing phase. However, the other 3 Likert fields have significantly lower ratings. We could conclude that this is a less successful iteration of the MRS given the other quality indicating factors, and further hypothesise that similarity may not the most important factor in the quality of recommendations.

Notable responses | “Any other feedback you’d like to give?” :
Tal AH: “V Similar :O”
John JJ: “The recommendations were very poor, I despise this artist and would never listen to ANYTHING by him.”, “I would never listen to anything by this artist. “, “Guns N' Roses for Devo? Really?”
Arun SH: “Good and best”
Rajavarman P: “That is my favorite song.” “Good”, “Nice”

Post-Study

Responses	Overall quality	How likely to reuse service
count	8	8
Mode (freq)	4 (3)	4 (3)
Mean	3.1	3.1
Median	3.5	3.5

Chapter 7: Evaluation

Between the 2 iterations of the MRS, it’s quite clear that the 1st was the more successful due to the higher averages on the questions of quality (during study and post-study). However, the 2nd was much quicker to train as aforementioned in the implementation section, with some further improvement, I think it is possible to achieve at least the quality of recommendations served by the 1st system, whilst maintaining the training efficiency of the 2nd system. I have realised that even in trying to address the issues in the 2nd iteration, due to the lack of time, I did not give enough attention to detail in order to fully address ML issues, such as the dimensionality of the model: ‘*size’, ‘vector_size = 300’*. For completeness, I will continue to work on perfecting the system after the submission of this project.

There is an overall issue of data and thus recommendation quality, some of the recommendations are linked in the genre and style mappings (e.g hip-hop to jazz) but the music *itself* is not necessarily interesting or ‘good’. We need an extra metric to ensure quality, possibilities include (ordered by priority in my opinion);

a. Getting user data in order to provide some Collaborative Filtering influence and insurance of quality from ‘social’ evidence, this could be; [27]
    - Implicit feedback (about listening behaviour)
    - Explicit feedback (typically user ratings)

b. Taking audio features to increase sonic similarity, though this is surprisingly well and interestingly captured by the analytic abilities of Doc2Vec. There is evidence of acoustic similarities between seed and recommended releases (E.g Bonobo recommending '*Transient v Resident - Medulla*' in 1st system and '*Kelpe - Cambio Wechsel'* in 2nd – there are clear timbral similarities in the combination of acoustic and electronic textures).

There should also be a way of ensuring the quality of music reviews, this is achieved in CB by means of the 'positive', 'negative' ratings for each review. Most review platforms offer some kind of positive rating feature, which in further development should control which reviews are contained in the corpus.

The web application satisfies all aims outlined. Although there is the small issue of users not completing the study and their results still being submitted for all questionnaires completed prior to them leaving; or a small issue where users complete 4 reviews instead of 3, possibly due to them going back instead of clicking 'next'. Both could be addressed with some more complex HTML templating and design, but this would take a significant amount of time and is perhaps not worth it. Some clearer instructions may help, although the current ones are well thought out.

In regards to original hypothesis, I think it holds true, although the "Creation of artist networks through resultant word vector similarity", is hard to prove, I think it does well at keeping true to geographical location and era however, which means there is definitely some respect to artist ontology. (Ghostface Killah leads to recommending fellow Wu-Tang Clan member GZA in 1st system).

As for: "popularity is much less of an influential factor, with the context and semantics of the reviews being far more influential", this definitely holds true, there have been multiple cases where users (myself included) have found new, more unknown artists as quality recommendations from known artists. In regards to the Bonobo recommendations aforementioned; Bonobo is very popular with over 6 million hits for his album 'Sweetness' on Youtube – 'Kelpe' has ~20k for his most popular album, and 'Transient V Resident' less than 20 views for most of their tracks.

Chapter 8: Future Research

Given a more modern day ‘holistic’ set of requirements for a recommendation system; there are a number of possibilities to take this work further, partly as a result of new forms of data.

The progression toward Big Linked Data

The fore founder of the internet; Sir Tim Berners-Lee, has been celebrating and advocating the potential of a ‘Linked Data’ web model; Berners-Lee outlined four principles of linked data in his "Linked Data" note of 2006 [48], paraphrased in the following [30, in 29]:

1. Use URIs to name (identify) things.
2. Use HTTP URIs so that these things can be looked up (interpreted, "dereferenced").
3. Provide useful information about what a name identifies when it's looked up, using open standards such as RDF, SPARQL, etc.
4. Refer to other things using their HTTP URI-based names when publishing data on the Web.

This movement towards a semantically linked and easily accessible knowledge universe has reached integration with my database of choice; MusicBrainz and its affiliated - ‘sister’ - metadata and associated sources of data; they call the project LinkedBrainz [31]. For this project I have already been utilising the existing Linked Data features of MusicBrainz; the URIs (MBIDs) in order to create unique mappings such that I can get artist names and release names from the original JSON data containing the reviews, this was however a difficult task in order to achieve my goals; e.g. it was less complicated to query the Spotify URI to gain access to the artist names for releases. There are currently efforts to do this by means of an API such as [http://musicontology.com/].

In the future perhaps all major sources of data will be linked in such a way that Music Recommendation and Discovery will fulfill its maximum potential; so as to utilise much as the information which is currently available on the web, and thus allow users to successfully discover it for their wants and needs (by means of search or recommendation). The main data provision issue, then, is making sure all music releases, old and new, are sufficiently described in Linked Data, such as to add to a well structured ontology.

Integration into the SoTA - Creating a Hybrid MRS

Here I will outline where I would take this work next in order to integrate the benefits of my work into a SoTA like model, by combining this work with other methods aforedescribed such as CF – with input features: user ratings and some listening behaviour. I will keep to Music Release Recommendation, although it could be just as useful for Track recommendation.

The input to the Doc2Vec or similar model could be any review or even metadata describing the release, i.e location of recording, instruments played, musician information etc. I'll take inspiration from Youtube's approach aforementioned where they divide the problem into 2 tasks; *candidate generation* and *ranking*. My work here highlights that in candidate generation it is very useful, however the ranking, whist often not optimal, is still useful. I therefore propose that the model could be useful inside a weighted function, to influence both tasks as the developer sees necessary. As aforementioned it addresses well the common '*Cold start'* problem in CF systems.

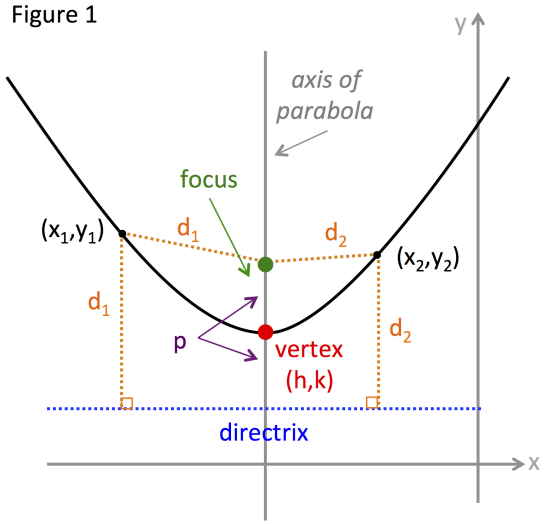
Personalisation and Music Discovery

A large issue at hand is on the topic of personalisation. In this project we have seen the success of a recommendation from a selection of releases, and perhaps whilst this may be adequate for playlisting and 'vanilla' recommendation applications, there is much more research to be done on the best ways to Discover new music by AI assisted search (augmented?). It is logical that the kind of research done here is useful in development for this; I could simply take a user input of N-nearest neighbours, and allow some kind of network to unfold so the user can really explore the universe of latent space. Collab filtering is of course a great way to achieve personalisation.

- User-centric, context sensitive recommendation;
    - Routinely situational, strongly associated with the playlisting problem (e.g. to fit what a user is doing in their day to day routine), enabled by locational and date-and-time data. [reference for research here]. This could additionally be a commercial area-of-interest in the case of a shopping mall; play music to aid the current situational objectives (e.g play faster music before closing time).
    - Professional / creative roles:
        - Media producers (Documentary, advertisement etc.). This is a particularly in depth and interesting topic. This could fall into the area of audio-visual processing research; using video analysis techniques to gain relevant 'search' parameters for a recommendation/discovery/sorting algorithm. The biggest problem here, I think, would be creating effective parameterisation for the user to control the discovery algorithm. Using Doc2Vec could be a great way to analyse synopsis written for pieces of music to serve this purpose.
        - Radio DJs; The main objective of the system here would be to facilitate/inspire interesting and captivating playlisting by the DJ, whilst recommending items previously unknown to the DJ; e.g 'joining the dots' [Gilles Peterson, BBC 6 Music]. This is something my recommendation system seems to excel at. (artist networks, sonic and musical similarities).

The following is a hypothesis for personalisation based on work done in this project. It can be explained perhaps by means of the projecting a vector, measuring similarity, onto the theorised error surface that the training phase of the DNN (ideally) descends. Suppose a parabola (within a much more complex error surface) looks somewhat like this [57]:

Figure 1



Using the 'axis of parabola' as an approximate measure of the similarity level of resultant recommendations from the trained model, where (x1,y1) and (x2, y2) are the upper limit, adequate values of the weights for the DNN or similar architecture, i.e where the recommendations are 'mostly satisfactory', I propose 2 cases to satisfy maximum quality Q, based on eclecticness (e), measured somewhat in question 7 (corresponding field J1) in Pre-study survey.csv:

$$max(Q \mid min(e))\ occurs\ at\ (h,\ k)$$
$$max(Q \mid max(e))\ occurs\ at\ (x1,\ y1)$$

Further, it is possible to use a parametric equation to define a range of $max(Q \mid e)$.

This hypothesis could be tested in a personalised recommendation application during the training and testing phase of a ML model – i.e use the testing results to tune the training parameters accordingly.

Models of cognition for recommendation

Given so many types of Music Information and data; e.g reviews, acoustic descriptors, user listening history, it seems only appropriate to find the most efficient and successful methods to comprise an 'intelligent' system; such that it becomes a hybrid Machine Learning / Artificial Intelligence 'model of cognition'. We can see evidence for this in my own project; we can receive

mostly good recommendations based on one metric resultant from Deep Learning (Cos Sim of latent vector spaces), but this is not the entire story of context, quality and similarity; we need an extra metric to ensure quality. If we take inspiration from neuroscience here (as Visual AI is currently doing[58]), we can learn that different parts of our brain process information in different ways, perhaps then we can draw a parallel from brain development, to advancements in 'Big Data' problems, similar to those aforedescribed in this report; particularly of 'Data Overload' nature.

Chapter 9: Conclusion

Initially, I was as ambitious as my passion would permit; I wanted to create a Music Discovery application with many features; in this work I have refined my original thought process into a much more focused work. Due to time limitations, I was not able to fully realise my original ambitions (described in my project proposal and preliminary report), leaving much future research and development to be done. Given that this is my first Data Science project, I have made some mistakes; but therefore learnt a tremendous amount about how to approach an MRS in an innovative way. Overall, I've really enjoyed the research and development process in this project.

Whilst not being perfect, the resultant system satisfies my original aims laid out at the beginning of development; it produces very interesting recommendations, and good results overall. So I am pleased, and most of my users are too.

Bibliography

[1] '(16) Can anyone explain what is cold start problem in recommender...', *ResearchGate*. [Online]. Available: https://www.researchgate.net/post/Can_anyone_explain_what_is_cold_start_problem_in_recommender_system. [Accessed: 14-May-2018].
[2] '11.1. pickle — Python object serialization — Python 2.7.15 documentation'. [Online]. Available: https://docs.python.org/2/library/pickle.html. [Accessed: 14-May-2018].
[3] 'A Basic Introduction To Neural Networks'. [Online]. Available: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html. [Accessed: 14-May-2018].
[4] G. Shperber, 'A gentle introduction to Doc2Vec', *ScaleAbout*, 26-Jul-2017. .
[5] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman, 'A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures', *Computer Music Journal*, vol. 28, no. 2, pp. 63–76, Jun. 2004.
[6] E. Pampalk, 'Audio-Based Music Similarity and Retrieval: Combining a Spectral Similarity Model with Information Extracted from Fluctuation Patterns', p. 10.
[7] 'Backpropagation', *Wikipedia*. 14-May-2018.
[8] 'Chain rule', *Wikipedia*. 29-Apr-2018.
[9] K. Jacobson, 'Connections in Music', p. 33-35.
[10] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, 'Content-Based Music Information Retrieval: Current Directions and Future Challenges', *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, Apr. 2008.
[11] H. Shao, D. Xu, and G. Zheng, 'Convergence of a Batch Gradient Algorithm with Adaptive Momentum for Neural Networks', *Neural Process Lett*, vol. 34, no. 3, p. 221, Dec. 2011.
[12] 'Cosine Similarity - which vectors? Thoughts on training process? - Google Groups'. [Online]. Available: https://groups.google.com/forum/#!topic/gensim/8h2ej9air1s. [Accessed: 14-May-2018].
[13] Q. Le and T. Mikolov, 'Distributed Representations of Sentences and Documents', p. 9.
[14] D. D. Labs, 'District Data Labs - Modern Methods for Sentiment Analysis'. [Online]. Available: https://districtdatalabs.silvrback.com/modern-methods-for-sentiment-analysis. [Accessed: 14-May-2018].
[15] D. Mishra, 'Doc2vec in a simple way', *Deepak Mishra*, 08-Feb-2017. .
[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, 'Efficient Estimation of Word Representations in Vector Space', *arXiv:1301.3781 [cs]*, Jan. 2013.
[17] 'Feedforward neural network', *Wikipedia*. 06-May-2018.
[18] C. Anderson, 'Forget squeezing millions from a few megahits at the top of the charts. The future of entertainment is in the millions of niche markets at the shallow end of the bitstream. continued >', p. 30.
[19] M. Denning, 'From Port to Port', *Foreign Affairs*, 21-Aug-2015.
[20] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, 'GroupLens: An Open Architecture for Collaborative Filtering of Netnews', in *Proceedings of the 1994 ACM*

Conference on Computer Supported Cooperative Work, New York, NY, USA, 1994, pp. 175–186.
[21] E. Bernhardsson, ‘How did Spotify get so good at machine learning? Was machine learning important from the start, or did they catch up over time? - Quora’. [Online]. Available: https://www.quora.com/How-did-Spotify-get-so-good-at-machine-learning-Was-machine-learning-important-from-the-start-or-did-they-catch-up-over-time. [Accessed: 14-May-2018].
[22] ‘How to Use the Likert Scale in Statistical Analysis ~ Statistics Café’. [Online]. Available: http://statisticscafe.blogspot.co.uk/2011/05/how-to-use-likert-scale-in-statistical.html. [Accessed: 14-May-2018].
[23] ‘How to Use the Likert Scale in Statistical Analysis ~ Statistics Café’. .
[24] O. Celma and P. Lamere, ‘If You Like Radiohead, You Might Like This Article’, *AI Magazine*, vol. 32, no. 3, pp. 57–66, 2011.
[25] ‘In AI, is data more important than algorithms? - Quora’. [Online]. Available: https://www.quora.com/In-AI-is-data-more-important-than-algorithms. [Accessed: 14-May-2018].
[26] F. Ricci, L. Rokach, and B. Shapira, ‘Introduction to Recommender Systems Handbook’, in *Recommender Systems Handbook*, Springer, Boston, MA, 2011, pp. 1–35.
[27] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, ‘Item-based collaborative filtering recommendation algorithms’, 2001, pp. 285–295.
[28] B. Whitman, D. Roy, and B. Vercoe, ‘Learning word meanings and descriptive parameter spaces from music’, 2003, vol. 6, pp. 92–99.
[29] ‘Linked data’, *Wikipedia*. 22-Mar-2018.
[30] ‘Linked Data - Design Issues’. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 14-May-2018].
[31] ‘LinkedBrainz - MusicBrainz Wiki’. [Online]. Available: https://wiki.musicbrainz.org/LinkedBrainz. [Accessed: 14-May-2018].
[32] ‘Machine Learning :: Cosine Similarity for Vector Space Models (Part III) | Terra Incognita’. [Online]. Available: http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/. [Accessed: 14-May-2018].
[33] ‘Machine Learning Lesson of the Day – Overfitting and Underfitting’, *The Chemical Statistician*, 20-Mar-2014. .
[34] Ò. Celma, *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Berlin Heidelberg: Springer-Verlag, 2010.
[35] G. Carfoot, ‘Musical discovery, colonialism, and the possibilities of intercultural communication through music’, *Popular Communication*, vol. 14, no. 3, pp. 178–186, Jul. 2016.
[36] ‘My Market Research Methods - Types of data measurement scales: nominal, ordinal, interval’. [Online]. Available: http://www.mymarketresearchmethods.com/types-of-data-nominal-ordinal-interval-ratio/. [Accessed: 14-May-2018].
[37] G. Kamita, *nlp-forex-predict: Foreign exchange market prediction with nlp*. 2017.
[38] A. Kar, ‘Power of Doc2Vec in Paragraph Embedding’, *Arunava Kar*, 13-Feb-2018. .

[39] ‘python - How to eval a dict py file Ordered’, *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/25634080/how-to-eval-a-dict-py-file-ordered. [Accessed: 14-May-2018].
[40] S. Raschka, *Python Machine Learning*, 1 edition. Packt Publishing, 2015.
[41] ‘Amazon.com Link’. .
[42] A. Elberse, ‘Should You Invest in the Long Tail?’, *Harvard Business Review*, 01-Jul-2008. [Online]. Available: https://hbr.org/2008/07/should-you-invest-in-the-long-tail. [Accessed: 14-May-2018].
[43] ‘Text Summarization in Python: Extractive vs. Abstractive techniques revisited | RARE Technologies’. [Online]. Available: https://rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/. [Accessed: 14-May-2018].
[44] W. Maurer, ‘THE DYNAMICS OF MUSIC DISTRIBUTION’. [Online]. Available: http://www.chimeinteractive.com/about/press/iris_online-9501.shtml. [Accessed: 14-May-2018].
[45] ‘The Music Ontology’. [Online]. Available: http://musicontology.com/. [Accessed: 14-May-2018].
[46] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence, ‘The Quest for Ground Truth in Musical Artist Similarity’, p. 23.
[47] J. Blattmann, ‘The secret behind YouTube’s algorithm is you’, *Elevate by Lateral View*, 30-Nov-2017. .
[48] TED, *Tim Berners-Lee: The next Web of open, linked data*. .
[49] R. Karam, ‘Using Word2vec for Music Recommendations’, *Towards Data Science*, 07-Dec-2017. [Online]. Available: https://towardsdatascience.com/using-word2vec-for-music-recommendations-bb9649ac2484. [Accessed: 14-May-2018].
[50] ‘Web API | Spotify for Developers’. [Online]. Available: https://beta.developer.spotify.com/documentation/web-api/. [Accessed: 14-May-2018].
[51] ‘Web server · Wiki · data-networks-web / lab-exercises’, *GitLab*. [Online]. Available: http://gitlab.doc.gold.ac.uk/data-networks-web/lab-exercises/wikis/web-server. [Accessed: 14-May-2018].
[52] ‘Welcome | Jinja2 (The Python Template Engine)’. [Online]. Available: http://jinja.pocoo.org/. [Accessed: 14-May-2018].
[53] ‘What are the benefits of using rectified linear units vs the typical sigmoid activation function? - Quora’. [Online]. Available: https://www.quora.com/What-are-the-benefits-of-using-rectified-linear-units-vs-the-typical-sigmoid-activation-function. [Accessed: 14-May-2018].
[54] P. Covington, J. Adams, and E. Sargin, ‘Deep Neural Networks for YouTube Recommendations’, 2016, pp. 191–198.
[55] ‘CB-277: Update the installation guidelines by dpmittal · Pull Request #180 · metabrainz/critiquebrainz’, *GitHub*. [Online]. Available: https://github.com/metabrainz/critiquebrainz/pull/180. [Accessed: 14-May-2018].
[56] ‘Curse of dimensionality’, *Wikipedia*. 14-May-2018.

[57] ‘parabola_standard_equation_1.jpg (1558×1500)’. [Online]. Available: https://www.softschools.com/math/pre_calculus/images/parabola_standard_equation_1.jpg. [Accessed: 14-May-2018].
[58] D. Bear, ‘How machine learning is helping neuroscientists understand the brain’, *Massive*. [Online]. Available: https://massivesci.com//articles/neuroscience-machine-learning-metaphors/. [Accessed: 15-May-2018].

Appendices

All the code is contained at;

Backend:
http://gitlab.doc.gold.ac.uk/jkirt001/Major_Project_localCode

Server side:
http://gitlab.doc.gold.ac.uk/jkirt001/Major_Project

Google Forms questionnaires:

Pre-study:
https://docs.google.com/forms/d/1B9nxHroGSFPF4Gm9dC9BP77pAKsPJJN0se8SxgA8SUI/

Rate recs:
https://docs.google.com/forms/d/17qV9jAx7KPrOIUfnZd4vV8w-c5dC29j9wt4rlReoqqQ/

Post-study:
https://docs.google.com/forms/d/1new3jAIH2ohH_xQgj-gWta0GikzOeBilhozGWrh5-Q8