University of Sussex

Matching Music with the Semantic Content of the Web

An investigation into appropriate playlist generation using HTML content

Damon Hayhurst

Submitted for the degree of Bachelor of Science

University of Sussex

May 2015

UNIVERSITY OF SUSSEX

Damon Hayhurst, Bachelor of Science

Matching Music with the Semantic Content of the Web
An exploration into using HTML content for the context of automatic playlist generation

Abstract

The popular site, Last.fm contain over 640 million unique track. With so much choice it can be difficult to know where to start. Recommender systems are used to filter through the sheer multitude of items on the Internet. The following dissertation outlines the creation of the recombEAn algorithm. An content based filtering algorithm that uses a webpage as it's context. By extracting keywords from an article and processing them using a variety of natural language techniques, one can use these words in order to determine a, hopefully, appropriate music recommendation that considers both the tone and the context of the webpage. Within the report is an explanation of the algorithm along with processes used to parse and semantically manipulate the extracted keywords. The recombEAn algorithm is then evaluated using human participants in order to determine whether the system returns successful music choices.

Acknowledgements

Thanks to all who encouraged me through my moments of insanity. Special thanks to Sam, Salami, Lyall, Falco and Barnaby for his inspirational moments!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The amount of music on the Internet is overwhelming. The streaming site, Spotify[1], recognises over 30 million songs on it's database(Spotify, 2015). This is only licensed music. Last.fm[2] the social music website, has over 640 million tracks, licensed and unlicensed. With such a lack in constraints, it can be difficult to come to any sort of decision about what to listen to.

User Scenario

Johnny wants to listen to music while he browses the internet but the appearance of a blank search box on a music streaming site, such as Spotify, makes him feel overwhelmed. He is overcome by the sheer amount and variation of music that engulfs the Internet. Johnny feels that none of the music in his library reflects his mood and wants to listen to something new.

The aim of the project is to investigate methods to build a system which uses the semantic content of a webpage to determine an appropriate music recommendation. The system uses the semantic content of the webpages Johnny is browsing in order to determine a suitable music recommendation. It can be described as a content-filtering recommender system. The recombEAn algorithm is introduced which tries to extract keywords, from the content that are important to context and tone, using Natural Language techniques. Matching these keywords with tags found on the social music website, Last.fm. The recombEAn algorithm tries to find a balance between using the most frequent words from an article and using as many tags as possible within a search as to be the most specific and relevant to the article.

Music Information Retrieval (MIR) is a field of research that focused on how music

[1]Spotify: `https://www.spotify.com/`

[2]Last.fm: `http://www.last.fm`

can be analysed and it's data retrieved. Either through tag-based social collaborations systems, such as Last.fm, or by extracting data from the song itself through analysing the audio. One such service, Echonest, does just this. It analyses song data in order to store information such as what key the song is in or the tempo of the song. The database it creates with this can feed into a music discovery system in order to recommend new tracks. One such example is the web application, Going Undercover (Lamere, 2013). This traces a journey from one artist to another through a chain of cover songs. MIR is a growing field that hopes to benefit the classification of music within the digital world.

Recommender systems are algorithms tasked with suggesting appropriate items to a user. There are two main tactics used by recommender systems in order to return appropriate choices. The first being collaborative filtering, which recommends based upon similar users' preferences. A popular example is Amazon's[3] system of recommending products. On every item page, there will be a section offering other products that users bought after viewing the item described on the page. In this way, the recommender system is providing products based on the shared interest that the user browsing and other users have in common. Last.fm does a similar thing and finds the listening habits you and another user have in common and then recommends more music based on their other tastes. Content filtering, the other type of recommender system, recommends based on the descriptions of items. The descriptions usually consist of tags or keywords stored with each item. The user's preferences are noted and then items are recommended based on tags shared between items that the user likes, along with their preference history, and other items. Rotten Tomatoes uses just this in order to recommend films. The user enters the "kind" of films they enjoy and which actors they want to see in order for the system to recommend further films (Reisinger, 2009). Recommender systems are an important part of the growing amount of media found on the Internet because they filter through the sheer multitude of items.

The processing of "human language data" or Natural Language Processing (NLP) is a field that concerns itself with deriving language meaning within a digital system. This is a particularly important field currently due to the rise in social media conversation occurring on the Internet. The site, Twitter[4], is a focal point for NLP because it contains "microblogs" concerning users' opinions and complaints on various topics. An NLP classification algorithm can be built that examines the sentiment of tweets in order to determine whether that user is expressing a positive, negative or neutral sentiment concerning a cer-

[3]Amazon: http://www.amazon.co.uk/

[4]Twitter: https://twitter.com/

tain topic (Agarwal et al., 2011). This process requires a training data set to be utilised in order for words with positive or negative sentiment to be identified. In this way, the classification procedure is said to be supervised. This is useful for businesses who want to find out the opinion of their latest product on a mass scale. A large number of tweets can be analysed in order to determine the ratio of positive to negative responses and thus, the overall reception that product received. Some NLP algorithms are said to be partially or unsupervised. These algorithms are likely to involve machine learning techniques. They often use statistical probability measures created by analysing a large corpus of language rather than keeping a large set of rules, with which to refer back to, like the supervised sentiment classifier explained above. Word2vec is an example of such an algorithm (explained in section 3.3). NLP makes up a large variety of techniques aimed at computing and understanding meaning within human language.

Work with music data retrieval and semantic processing for a recommender system was investigated before by Audio Metaphor. The bulk of the system, being the introduction of the SLiCE algorithm, which takes a brief semantic description and provides a list of sub queries, made up of keywords from the description. The sub queries are used to query the database, Freesound[5]. The search results are sound files which can then be used for soundscape composition (Thorogood et al., 2012). The recombEAn algorithm can be compared with the SLiCE algorithm in the sense that they are both deriving sound recommendations from semantic content. Both systems compute a series of keywords and use them to query a sound database. The algorithmic procedures can be compared because both systems are looking for the most specific result from the semantic content by querying sub list variations of the extracted keywords. The differences lie in the approaches used to find the results and the processes that widen the semantic space in order to search for more similar words.

The following report outlines the processes behind the recombEAn algorithm. First, the methods of extraction are explained. This includes the parsing of HTML content from a webpage and the extraction of keywords from the resultant textual content. It is also explained how the algorithm uses external processes in order to increase the semantic space with which it is searching and extra measures that are considered to make the recommendations more appropriate. The second section explains how the algorithm was derived, using requirements and tests as stimulus. This is followed by a semantic explanation of the algorithm itself, in which, it is explained how keywords are manipulated and

[5]Freesound: https://www.freesound.org/

recombined in order to return a appropriate music choice. The recombEAn algorithm is then evaluated using human participants in order to determine whether the system returns appropriate results. The report is concluded by discussing future work and extensions that can be applied to the system.

Chapter 2

Professional Considerations

The BCS Code of Conduct[1] sets out the ethical considerations for computing professionals. This code must be applied when carrying out any scientific research of any level. In this section, clauses from the Code of Conduct that apply to the research carried out in this dissertation are stated and their applications within the testing done are explained (see chapter **??** for the full explanation of the testing). Screenshots of the questionnaire can be found in appendix A.

Clause 1a) states that one must 'have due regard for public health, privacy, security, and wellbeing of others in the environment'. The testing that took place, was done entirely through the Internet and Google Forms. Participants could fill out the form in privacy, when they wanted to. They were not subject to a testing environment where their well-being could be affected. The public health of the participants is a non-concern because of the nature of the test being carried out via the Internet. The privacy of the subjects was also kept. At no point within the form was the subject asked for their name or any personal information. The form provided information about the system. It also pointed out, that it was not the participant who was being tested, but the system. Prior to being shown any questions, the participant had to select a checkbox which stated that agreed to take part.

Clause 1b) states that one must have 'have due regard for the legitimate rights of Third Parties'. Before being given any questions, it was stated in the information for participants that they are 'welcome to withdraw at any time'. Additionally, email addresses of the researcher and the project supervisor were provided in the case that respondents had any concerns or issues.

In Clause 1c), it is advised that one must conduct 'professional activities without

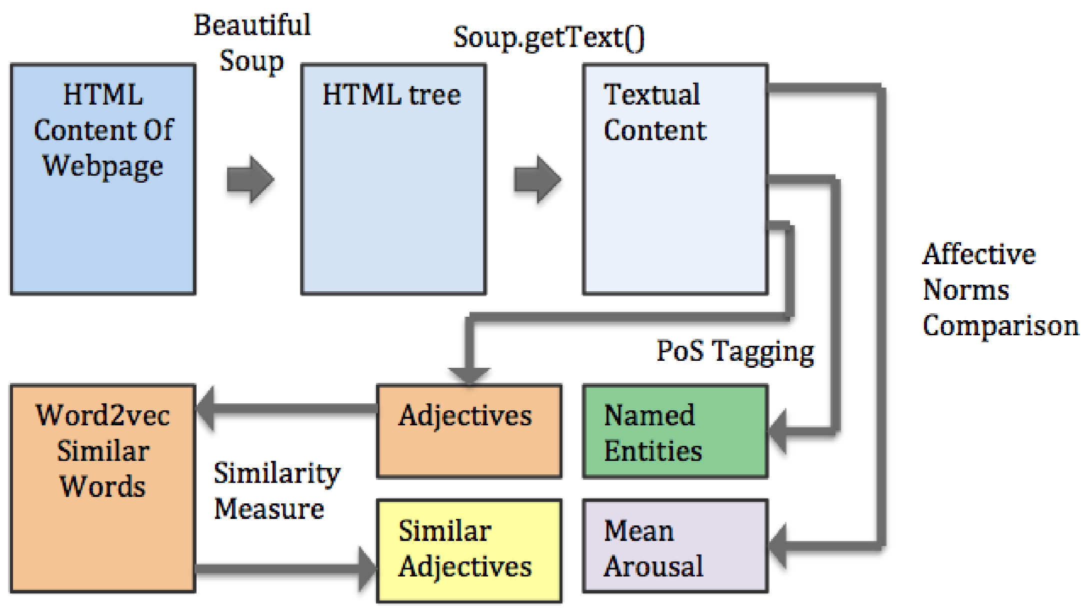
[1]BCS Code of Conduct: `http://www.bcs.org/category/6030`

discrimination'. The testing carried out with this system did not ask for any information that may infer a person's 'sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age or disability'. This information can not be discriminated against because all the data is completely anonymous.

Chapter 3

Methodology

Figure 3.1: Outline of Extraction methods



3.1 Parsing HTML

Processing the main text from an arbitrary webpage is difficult because there is no standardised way for emphasising the main textual content of a webpage. While one could argue that the body HTML tag is most likely to indicate the required passage of text, this is not always the case. When considering social media sites, such as Facebook and Twitter, it is important to note that these sites' main pages are made up of 'micro-blogs' or small passages of text that contain wildly differing thematic content. Yet, these passages of text sit within close proximity to each other and are considered equally when parsing the main textual content of the webpage. It would prove difficult to come to a consensus about a music recommendation if the textual content is made up of multiple short passages of text with wildly differing themes and tones. There is no main textual content for these webpages. For testing, only news articles were used because of the clarity of topic and main textual content.

Two different Python packages were used in order to extract data from HTML pages. These packages were utilised to achieve different extraction tasks.

- Boilerpipe[1] - The extraction of articles
- BeautifulSoup[2] - The extraction of album reviews from Pitchfork[3]

3.1.1 Boilerpipe

Boilerpipe is a Python package that focuses on removing the clutter around the textual content of a webpage. It's approach, uses a heuristic to extract an article from the HTML content. This was the reason it was chosen over BeautifulSoup to extract articles. BeautifulSoup would return the HTML as a dataset with which to iterate over. The layout of a webpage can vary greatly and while an algorithm could be created using BeautifulSoup to find the article text, it would be difficult to refine the algorithm to be one-size-fits-all algorithm. The package does not achieve the correct passage with a hundred percent accuracy but it will acquire the correct passage most of the time. Boilerpipe has several different extractors within it's package. The 'ArticleExtractor' can be used to extract text from news articles.

It was decided that the extraction of the main piece of text from an arbitrary website would be too difficult. There is not a standardised way to markup the main text of a web page within the HTML and, also, due to the fact that the largest portion of text within HTML was not always the main portion of text. The webpages that would be ed would be news articles and blog posts rather than an arbitrary page. Boilerpipe's ArticleExtractor would be used to extract the text from the news articles for interpretation within Python. Any articles that Boilerpipe failed to extract correctly, would be omitted from testing.

[1]Boilerpipe on Github: https://github.com/misja/python-boilerpipe
[2]BeautifulSoup: http://www.crummy.com/software/BeautifulSoup/
[3]Pitchfork: http://pitchfork.com/

3.1.2 BeautifulSoup

BeautifulSoup parses HTML and XML data into a tree which can be traversed in order to scrape information from a webpage. The Python package parses data which can then be navigated by HTML tag. The header of any webpage can be extracted by making a function call to find all texts sandwiched within the head HTML tag.

This package was chosen to extract album reviews from Pitchfork for use within the word2vec model detailed in section 3.3. The album review webpages on Pitchfork have a standardised layout, that is the same on every page. The identity of the HTML tags that contain the textual content of the album review does not vary between reviews. Once the name of this tag was identified, it was simply a case of navigating to that tag name within the tree and extracting the relevant text. BeautifulSoup, was chosen over Boilerpipe because the exact textual content could be extracted. Boilerpipe's extraction process could end up including such features as review score and author because of their proximity to the album review passage. Features that are not relevant within the word2vec model and could end up interfering with the similarity measure. The tree, produced by BeautifulSoup, allows the HTML data to be manipulated in a way that only the main textual content could be extracted. Considering that Boilerpipe uses a heuristic to determine the text for extraction, by manually defining which text should be extracted within BeautifulSoup the performance is faster. Boilerpipe has to consider the full HTML before extraction, BeautifulSoup only has to consider a specific tag name. This is why BeautifulSoup was chosen for the extraction of album reviews, the HTML tags are consistent throughout the review extraction process.

3.2 Natural Language Processing

The Natural Language Toolkit (NLTK) [4] package is an open source platform for the manipulation, extraction and classification of 'human language data'. The package contains a multitude of different modules that can be used to query a passage of text. The modules of focus within the system are as follows:

- Part Of Speech Tagging
- Named Entity Extraction
- Stemming

[4]NLTK: http://www.nltk.org/

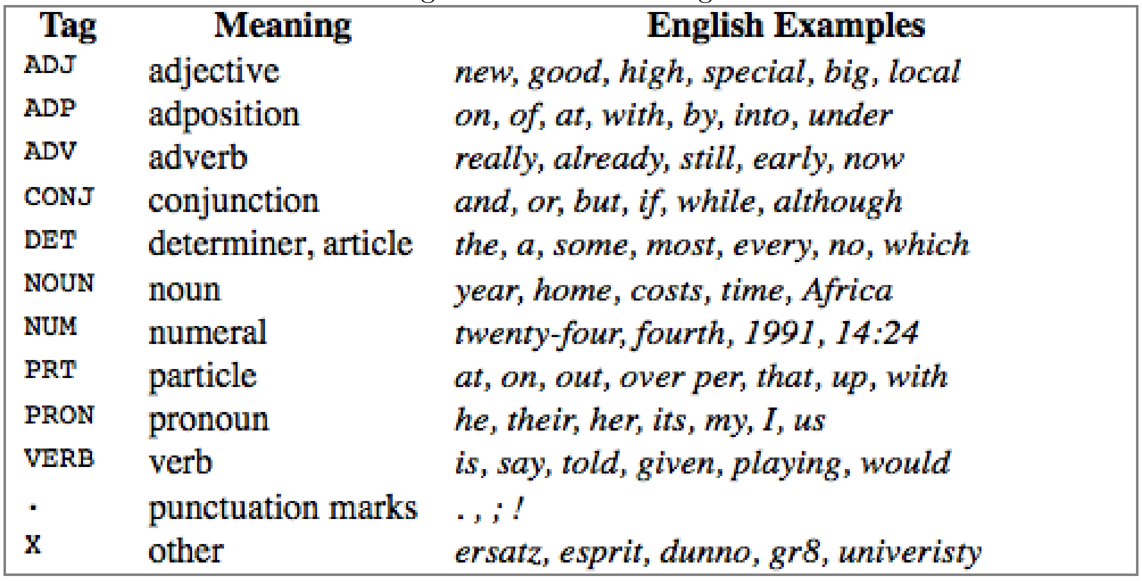
3.2.1 Part Of Speech Extraction

Part of Speech tagging (PoS tagging) is method used to determine a particular word's class. During English lessons, as a child, one is taught that words can be categorised due to how they behave. Some of the most basic categories being nouns, adjectives, verbs and prepositions. The act of PoS tagging, labels words within a sentence as their estimated category. For each individual word and symbol within a sentence the PoS tagger determines the category best suited for it. The classification process considers both the identity of the word and the identity of neighbouring words in order to come to a conclusion. When considering the word *sail*, one could deem it to be a noun or a verb. It is it's position in context that gives it the correct label. PoS tagging uses a set of rules and weighted probabilities to determine each word's part of speech. It can seen that the NLTK PoS tagger is not one hundred percent accurate (Yumusak et al., 2014) due to it's probabilistic nature. Some words can have multiple possible PoS tags but it can be deemed more likely to be one class rather than another due to the number of times it occurs within the english language as that class. PoS taggers are trained on a large dataset to determine the probability that a word is of a certain class. PoS tagging considers these probabilities along with a word's proximal location within a sentence in order to classify a it.

Different parts of speech or classes of words can be extracted by enacting the PoS tagging function upon a passage of text. The NLTK package provides functions with which to prepare a passage of text for PoS tagging. First the passage is tokenized, a passage of text of type string is split into substrings. The boundaries of these substrings are defined by the the whitespace between each word and the punctuation of the sentence. The PoS tagging function can then be called. It returns a list of tuples, with the word represented in the first element of the tuple followed by it's constituent PoS tag. By referring to the second element of each tuple, one can extract all the words or substrings that are considered to be part of a particular class.

The NLTK PoS tagging function labels words with tags defined in the Penn Treebank tag set. A tag set with over 40 different tag classes. In the context of building the system, where one is looking to extract words as singular keywords that could be considered a viable tag word within Last.fm's database, the Treebank tags are too detailed. The tag classes describe their behaviour, i.e.. noun, adjective, and also other attribute such as tense or plurality. A user will define tags for an artist, using Last.fm, completely out of context of a sentence. This is why one needs only to consider the basic behaviour of a given word for tag consideration. The Universal tag set offers a simplified set of class labels shown in figure 3.2.1.

Figure 3.2: Universal tagset



Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Once the textual content has been tokenised and PoS tagged, the amount of words has to be filtered in order to extract only adjectives that could convey tone. To reduce the array of tuples to a smaller number that is easier to consider, the top 100 most common words are removed from the set of PoS tags. The 100 most common words are extracted from a csv (Comma Separated Values) file of the 4000 most common words[5]. Audio Metaphor (Thorogood et al., 2012) also removes the 100 most common words during it's semantic word extraction. Removing the most common english words prevents polluting the extraction of entities and adjectives. Although words like 'He', 'a' and 'and' are not named entities or adjectives, it is important to note that PoS tagging is not entirely accurate and named entity extraction even less so. One mistaken PoS tag can filter a word, such as 'of', to the top of the extracted word list, when ordered by frequency, since the word is so common. On top of the removal of the most common words, a second list created manually removes any adjectives that convey size (i.e biggest) and plurality (ie. many). It was felt that these adjectives do not contain much sentiment that is in common with music. The extracted text can be filtered in order to ignore common and unsuitable words from being considered as keywords.

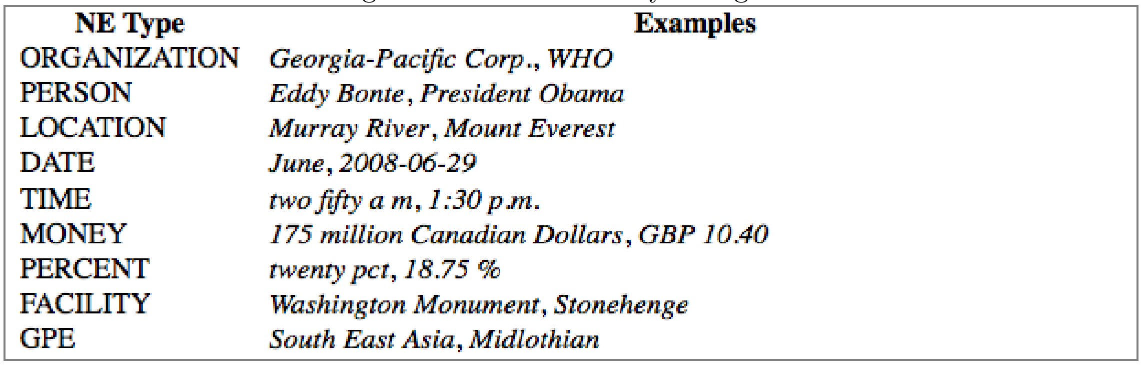
3.2.2 Named Entity Extraction

The NLTK package defines a system whereby proper nouns are identified within a passage of text. This is called Named Entity Recognition (NER). The named entities, extracted

[5]4000 common words corpus: http://www.wordfrequency.info/

using NER, fall into several categories. NER identifies the named entities within a text and describes a category which that entity falls under. The categories are shown in figure 3.3. To extract the named entities, the system refers to a dictionary in order to identify them.

Figure 3.3: Named Entity Categories



NE Type	Examples
ORGANIZATION	*Georgia-Pacific Corp., WHO*
PERSON	*Eddy Bonte, President Obama*
LOCATION	*Murray River, Mount Everest*
DATE	*June, 2008-06-29*
TIME	*two fifty a m, 1:30 p.m.*
MONEY	*175 million Canadian Dollars, GBP 10.40*
PERCENT	*twenty pct, 18.75 %*
FACILITY	*Washington Monument, Stonehenge*
GPE	*South East Asia, Midlothian*

The extraction process suffers from a lot of challenges. Which includes the fact that new organisations are created everyday therefore the dictionary of terms needs to be updated. Location names are a category that does not need regularly updating, so one would expect Locations to correctly recognised. However, NER suffers other short comings that could mean that the system fails to extract a Location or other category. Take the word *Reading* which could refer to the town in the UK or the verb ”to read a book”. If the word starts the sentence and occurs as a verb, in context, then it could be mistakenly extracted as a named entity. Another factor being that the extraction process can very easily parse a word as a named entity even if it might not have been used as a noun in the sentence in context. Also, words like *May* could refer to the date or it could refer to the name of someone. Not only can entities get mistakenly extracted but their entity type can also suffer from confusion (Bird et al., 2009). The latter limitation does not apply within the system because all entities, regardless of type, are extracted from the text. Cross domain entity extraction is shown to be more inaccurate than only extracting entities from one domain (Poibeau and Kosseim, 2001). The Stanford NER extraction system (from NLTK) has been shown to achieve an accuracy of only 73% on webpages (Ratinov and Roth, 2009). The filtering of entities by type and more discussion of this limitation is discussed in chapter 6.

3.2.3 Stemming

Some words, in context, can have suffixes that are added in order to express a certain sentiment, such as, tense, plurality or person. The stemming function removes the suffixes and prefixes in order to regain the root word. a word to something that is not considered part of the english language, instead the single unit that a family of words comes from. For instance, the word *knocked* becomes *knock*. This is definitely suitable in the context of creating tags for Last.fm. Users of the service are unlikely to provide words that are not the root version but the stemming algorithm can be attempted on words that have suffixes pertaining to tense.

3.3 Finding Similar Adjectives

Querying Last.fm with the words found within the passage of text is not guaranteed to return any results. An external system needs to be able to provide new options if the extraction process failed to provide any. Having a provider of additional words outside the system that expresses similar sentiment to the words in the article could aid the query and return more specific results. The Audio Metaphor system looks at tweets that contain matching keywords to the ones from the textual content, and extracts the most frequent words within those tweets in order to create a bigger 'semantic space' (Thorogood et al., 2012). This system looks at obtaining more words from a neural network data structure that measures word similarity by word context and proximity.

Word2vec[6] (Řehůřek and Sojka, 2010) is 'an unsupervised algorithm for learning the meaning behind words' (Řehůřek, 2014). It is a neural networks that use machine learning techniques. The algorithm computes distance vectors between words depending how similar they are, when used in context. The neural network has to be trained on a set of "sentences", which is where the distance measures are computed in the form of vectors. Words are treated to be in close proximity to each other if they are used in similar contexts. Consider the sentence: *The ____ must be fed.* The blank space could be filled with many different words including, cats, dog and baby. These words are considered similar, because they can be used in the same context, and, thus, have similar meaning. The word2vec model determines a distance measure between words depending on how similar the contexts (i.e. the surrounding words) of those words are.

Similar adjectives, that may be music related, can be found using a word2vec model

[6]Gensim's word2vec: https://radimrehurek.com/gensim/models/word2vec.html

that is trained upon album review sentences. Album reviews provide a semantic description of a piece of music. Querying a word2vec model trained on album reviews offers a way to possibly provide new words, using adjectives extracted from an article. Unsuitable adjectives that return no results when queried within Last.fm may have similar words within music related context.

A comma separated file (CSV) of over 8000 Pitchfork album reviews[7] was used to train the word2vec model. Python's csv module was used to put each row into a dictionary. Each row had an url value associated with it, which were requested sequentially. BeautifulSoup, explained in subsection 3.1.2, was used to navigate to the correct HTML tag that contained the album review sentences. The album review sentences were then scraped, with any stopwords removed, and stored in a list. Any mention of the artist whose album it was in the review was replaced by the string, *ARTIST*. This was done to avoid an artist name mistakenly being returned by the similarity measure. The artist name is not significant in context, except that it is considered the subject within the sentence.

3.4 Calculating Arousal Score

The Affective Norms for English Words (Bradley and Lang, 2010) is a dataset that provides ratings for a large number of English words. The dataset, presented in a text file, contains a list of English words followed by ratings in three categories. These three categories being Happy vs Unhappy (Valence), Excited vs Calm (Arousal) and Controlled vs In-control (Dominance). A group of students were asked to rate each word on a 9 point scale in the three categories. The dataset provides the mean score for each word in each of the three categories. An extract of the dataset can be seen in figure 3.4.

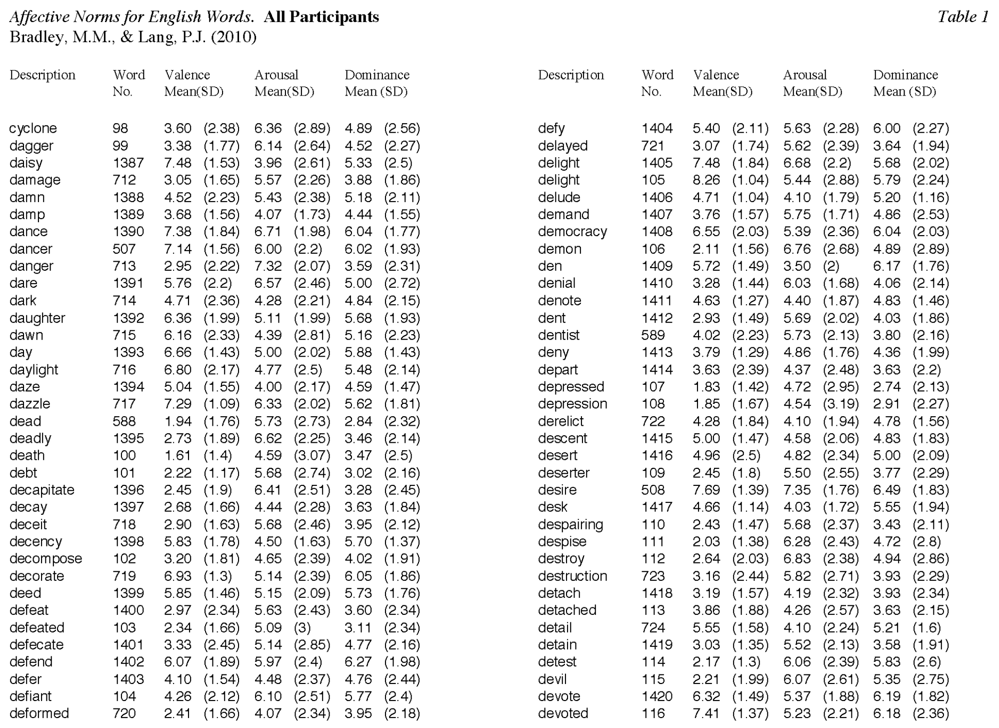
Using the dataset within the system can provide a quantifiable score to aid in the search for appropriate music choices. Echonest[8], provides parameters for it's song search. One of these parameters defines a minimum and maximum energy score with which all songs returned have to be in the boundaries of. The arousal category from the Affective Norms dataset seems like a viable link to the Echonest energy parameter. For a passage of text, the words shared with those in the Affective Norms dataset can have their arousal scores identified. An average arousal score can be computed from the identified words' scores.

The arousal score returned can then be translated to an Echonest energy query para-

[7]pitchfork-reviews.csv by medulla:`https://gist.github.com/meddulla/8277139`

[8]Echonest: `http://the.echonest.com/`

Figure 3.4: Extract from Affective Norms dataset



Affective Norms for English Words. **All Participants** *Table 1*
Bradley, M.M., & Lang, P.J. (2010)

Description	Word No.	Valence Mean(SD)	Arousal Mean(SD)	Dominance Mean (SD)
cyclone	98	3.60 (2.38)	6.36 (2.89)	4.89 (2.56)
dagger	99	3.38 (1.77)	6.14 (2.64)	4.52 (2.27)
daisy	1387	7.48 (1.53)	3.96 (2.61)	5.33 (2.5)
damage	712	3.05 (1.65)	5.57 (2.26)	3.88 (1.86)
damn	1388	4.52 (2.23)	5.43 (2.38)	5.18 (2.11)
damp	1389	3.68 (1.56)	4.07 (1.73)	4.44 (1.55)
dance	1390	7.38 (1.84)	6.71 (1.98)	6.04 (1.77)
dancer	507	7.14 (1.56)	6.00 (2.2)	6.02 (1.93)
danger	713	2.95 (2.22)	7.32 (2.07)	3.59 (2.31)
dare	1391	5.76 (2.2)	6.57 (2.46)	5.00 (2.72)
dark	714	4.71 (2.36)	4.28 (2.21)	4.84 (2.15)
daughter	1392	6.36 (1.99)	5.11 (1.99)	5.68 (1.93)
dawn	715	6.16 (2.33)	4.39 (2.81)	5.16 (2.23)
day	1393	6.66 (1.43)	5.00 (2.02)	5.88 (1.43)
daylight	716	6.80 (2.17)	4.77 (2.5)	5.48 (2.14)
daze	1394	5.04 (1.55)	4.00 (2.17)	4.59 (1.47)
dazzle	717	7.29 (1.09)	6.33 (2.02)	5.62 (1.81)
dead	588	1.94 (1.76)	5.73 (2.73)	2.84 (2.32)
deadly	1395	2.73 (1.89)	6.62 (2.25)	3.46 (2.14)
death	100	1.61 (1.4)	4.59 (3.07)	3.47 (2.5)
debt	101	2.22 (1.17)	5.68 (2.74)	3.02 (2.16)
decapitate	1396	2.45 (1.9)	6.41 (2.51)	3.28 (2.45)
decay	1397	2.68 (1.66)	4.44 (2.28)	3.63 (1.84)
deceit	718	2.90 (1.63)	5.68 (2.46)	3.95 (2.12)
decency	1398	5.83 (1.78)	4.50 (1.63)	5.70 (1.37)
decompose	102	3.20 (1.81)	4.65 (2.39)	4.02 (1.91)
decorate	719	6.93 (1.3)	5.14 (2.39)	6.05 (1.86)
deed	1399	5.85 (1.46)	5.15 (2.09)	5.73 (1.76)
defeat	1400	2.97 (2.34)	5.63 (2.43)	3.60 (2.34)
defeated	103	2.34 (1.66)	5.09 (3)	3.11 (2.34)
defecate	1401	3.33 (2.45)	5.14 (2.85)	4.77 (2.16)
defend	1402	6.07 (1.89)	5.97 (2.4)	6.27 (1.98)
defer	1403	4.10 (1.54)	4.48 (2.37)	4.76 (2.44)
defiant	104	4.26 (2.12)	6.10 (2.51)	5.77 (2.4)
deformed	720	2.41 (1.66)	4.07 (2.34)	3.95 (2.18)
defy	1404	5.40 (2.11)	5.63 (2.28)	6.00 (2.27)
delayed	721	3.07 (1.74)	5.62 (2.39)	3.64 (1.94)
delight	1405	7.48 (1.84)	6.68 (2.2)	5.68 (2.02)
delight	105	8.26 (1.04)	5.44 (2.88)	5.79 (2.24)
delude	1406	4.71 (1.04)	4.10 (1.79)	5.20 (1.16)
demand	1407	3.76 (1.57)	5.75 (1.71)	4.86 (2.53)
democracy	1408	6.55 (2.03)	5.39 (2.36)	6.04 (2.03)
demon	106	2.11 (1.56)	6.76 (2.68)	4.89 (2.89)
den	1409	5.72 (1.49)	3.50 (2)	6.17 (1.76)
denial	1410	3.28 (1.44)	6.03 (1.68)	4.06 (2.14)
denote	1411	4.63 (1.27)	4.40 (1.87)	4.83 (1.46)
dent	1412	2.93 (1.49)	5.69 (2.02)	4.03 (1.86)
dentist	589	4.02 (2.23)	5.73 (2.13)	3.80 (2.16)
deny	1413	3.79 (1.29)	4.86 (1.76)	4.36 (1.99)
depart	1414	3.63 (2.39)	4.37 (2.48)	3.63 (2.2)
depressed	107	1.83 (1.42)	4.72 (2.95)	2.74 (2.13)
depression	108	1.85 (1.67)	4.54 (3.19)	2.91 (2.27)
derelict	722	4.28 (1.84)	4.10 (1.94)	4.78 (1.56)
descent	1415	5.00 (1.47)	4.58 (2.06)	4.83 (1.83)
desert	1416	4.96 (2.5)	4.82 (2.34)	5.00 (2.09)
deserter	109	2.45 (1.8)	5.50 (2.55)	3.77 (2.29)
desire	508	7.69 (1.39)	7.35 (1.76)	6.49 (1.83)
desk	1417	4.66 (1.14)	4.03 (1.72)	5.55 (1.94)
despairing	110	2.43 (1.47)	5.68 (2.37)	3.43 (2.11)
despise	111	2.03 (1.38)	6.28 (2.43)	4.72 (2.8)
destroy	112	2.64 (2.03)	6.83 (2.38)	4.94 (2.86)
destruction	723	3.16 (2.44)	5.82 (2.71)	3.93 (2.29)
detach	1418	3.19 (1.57)	4.19 (2.32)	3.93 (2.34)
detached	113	3.86 (1.88)	4.26 (2.57)	3.63 (2.15)
detail	724	5.55 (1.58)	4.10 (2.24)	5.21 (1.6)
detain	1419	3.03 (1.35)	5.52 (2.13)	3.58 (1.91)
detest	114	2.17 (1.3)	6.06 (2.39)	5.83 (2.6)
devil	115	2.21 (1.99)	6.07 (2.61)	5.35 (2.75)
devote	1420	6.32 (1.49)	5.37 (1.88)	6.19 (1.82)
devoted	116	7.41 (1.37)	5.23 (2.21)	6.18 (2.36)

meter. The score needs to divided by 9 to be mapped from 0.0 - 9.0, that the Affective Norms words are rated out of, to 0.0 - 1.0, the boundaries of the Echonest energy query.

The contributing effect of computing a mean arousal score means that the tone of the article becomes a quantifiable figure. This aids the music recommendation process because it aims to keep the overall tone of the article in line with the music. The arousal measure acts as an extra assurance that the music provided will be exciting, if the language is arousing, and vice versa. In some searches the Last.fm results will return only the artist name because the tags are associated with that artist, rather than a song by that artist. The energy query can then search for a suitable song, by said artist, in order to return it. In the event that a song is returned by the Last.fm search result, the Echonest energy query acts as a check to determine whether the returned song is within the boundaries. The arousal score is an attempt at quantifying tone, in order to have extra parameters that the search query has to attain to and produce a more appropriate musical output.

3.5 Algorithm Design

Linking up the processes of entity extraction, adjective extraction and computing the mean arousal score was required in order to achieve the ideal playlist choices. No algorithm could do this job perfectly since the interpretation of text to music is entirely subjective. The properties associated with Last.fm tags provided the foundations for the algorithm in question. The processes entailed in the above sections need to have their precedence determined, in order to deliver a sensible result from the system. In the following section the process, thought and explanation of the recombEAn algorithm are described.

3.5.1 Requirements

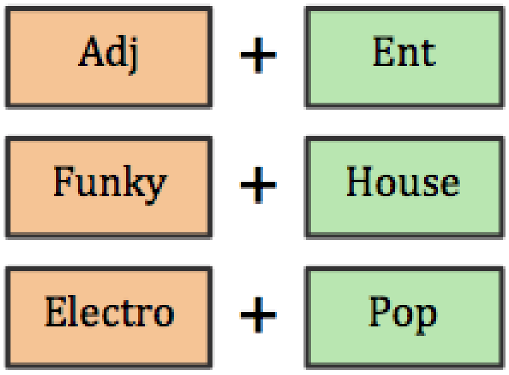
When designing an algorithm, one has to first consider the requirements that the system has to achieve. Recognising the inputs and the appropriate output of the system is useful for determining the requirements. The inputs of the system being two lists, whose items are sorted by frequency in the text. The first being named entities. These words are extracted using the NER method explained in subsection 3.2.2. The second list contains adjectives, extracted from the text using PoS tagging. This is explained in subsection 3.2.1. These lists are sorted in descending order by frequency. Words that occur multiple times within a passage are the most indicative of tone and context. Also, taken as an input, is a mean ”arousal score”. This is used to query EchoNest, explained in subsection 3.4. The required output, being n song recommendations, that abide by the mean arousal score.

A desired requirement for the system is that it returns an appropriate music choice. An appropriate choice would be one that abides by the context of the entities and uses adjectives and the mean arousal score to convey tone. The more words considered in the Last.fm query, the greater the chance that a music recommendation is likely to be specific. Use too many words, however, and the query is likely to return no results. The Audio Metaphor (Thorogood et al., 2012) system uses a top-down approach. The list of queries is populated by considering all the keywords first and then each query afterwards contains less and less keywords until all the keywords return results. Audio Metaphor considers a brief semantic description of ‘1000 characters’. Due to this factor, the system has the capacity to query all the keywords and then subtract. The character limit means that the number of combinations of keywords queried will be significantly lower than that of the average article using all the entities and adjectives. The top-down approach of Audio Metaphor would be unsuitable for this system. The amount of queries parsed through Google custom search would be too high. The system needs to consider tags individually first, using the result count from the Google search in order to determine whether concatenating the query's tags with other tags will return a non-zero result count. In this way, the lists of tags, from queries, can be recombined to provide, hopefully, a query with a multitude of tags. Ideally, the algorithm will achieve a level of specificity and ideally, an appropriate music choice.

In the absence of any search queries returning results, the system must still be able to return a music recommendation. The application of similar adjectives within the system injects an external provider of keywords where the article's tag queries return little to no results. Also, the energy parameter provided to Echonest may stop any results from being returned, due to none of songs achieving the correct energy score. A requirement must be that the system still provides a result in this case.

The philosophy behind the algorithm can be seen through the choices in NLP processes explained above. The proper nouns and adjectives of a text passage can be equated to the context of a music piece, i.e.. a country, a date or a name, and the words used to describe the tone of a piece of music. Using a combination of a contextual word and a describing word extracted from the piece of text, or returned through the similarity measure, was the main principle behind how the algorithm would prioritise search queries when determining a conclusive result. The motivation behind this philosophy being, genres of music follow this pattern (see figure 3.5)The combination of an entity and an adjective can be seen as a desired output requirement for the system.

Figure 3.5: Philosophy behind algorithm, an adjective and an entity as the ideal option



3.5.2 Considerations

Last.fm does not provide the API functionality with which to search for multiple tags. It used to provide multi tag search within the service's "playground" webpage[9] until it was removed. A workaround arrived in the form of using Google search's site specific parameter to query only words listed under the "tags" section of "Last.fm/music" webpages. The search query formula used *site:Last.fm/music intitle:"tags for" "[TAG]" "[TAG2]" ...* The square brackets being replaced by the tags in question. The urls returned by the search could then be extracted from Google search using Beautiful Soup. The dataset that this returned could then filtered by HTML tag in order to return each URL address found on the page. The problem with this was that Google would reject requests from an ip address if it were found to be querying the search engine too much. Google custom search[10] allows a user to specify and personalise a search engine. It allows one to focus a search on a specific url or number of urls. Using this feature to focus on "Last.fm/music" allowed one to make multiple searches without fear of being prevented. The results of each search are then crawled with AJAX and imported into python using JSON[11]. The query is added to a dictionary within python along with the number of results and the first 8 result URLs. Using Google custom search and site specific queries, it was possible to overcome the limitations that Last.fm's API service provided.

Verbs are not utilised within the system. The definition of a verb is 'an action, state or occurrence' (Dictionaries, 2015). Songs can be thought of as entities with qualities associated with them. Linguistically, verbs show the relationship between two subjects. While one can use verbs to describe the sentiment the music provided, i.e.. "That music moved me". Last.fm is a social community with music tags that anyone can update which means keywords are likely to be only words which describe or state the piece of music as an item, rather than a verb that conveys the relationship between the music and a singular person. Whereas an adjective describes a sole subject and therefore is likely to be used as a tag.

A large factor that had to be considered within the design was the characteristics of querying Last.fm through Google. When querying multiple tags, the search engine uses the "AND" operator to connect them. Any results returned will have all the tags queried. The more tags queried, the greater the probability that no search results will be shown. So the algorithm has to be concerned with always trying to achieve a search

[9] Last.fm Playground: `http://playground.Last.fm/`

[10] Google Custom Search: `https://cse.google.co.uk/cse/all`

[11] JSON: `http://www.JSON.org/`

result and yet involve as many tags as possible. Audio Metaphor (Thorogood et al., 2012) uses the Freesound search engine to query. Similarly the Freesound search engine uses the 'AND' operator when considering multiple key words. Audio Metaphor searches sub lists of these key words until each keyword features once in the successful set of sublists. All the sound files from the successful sub lists are then returned for the soundscape composer to choose from. Whereas in the required system, one is looking for a single output music recommendation rather than a multitude of files. The algorithm has to be concerned with not being too specific that it does not return any results from Last.fm but also not being too general so that the song choice is appropriate.

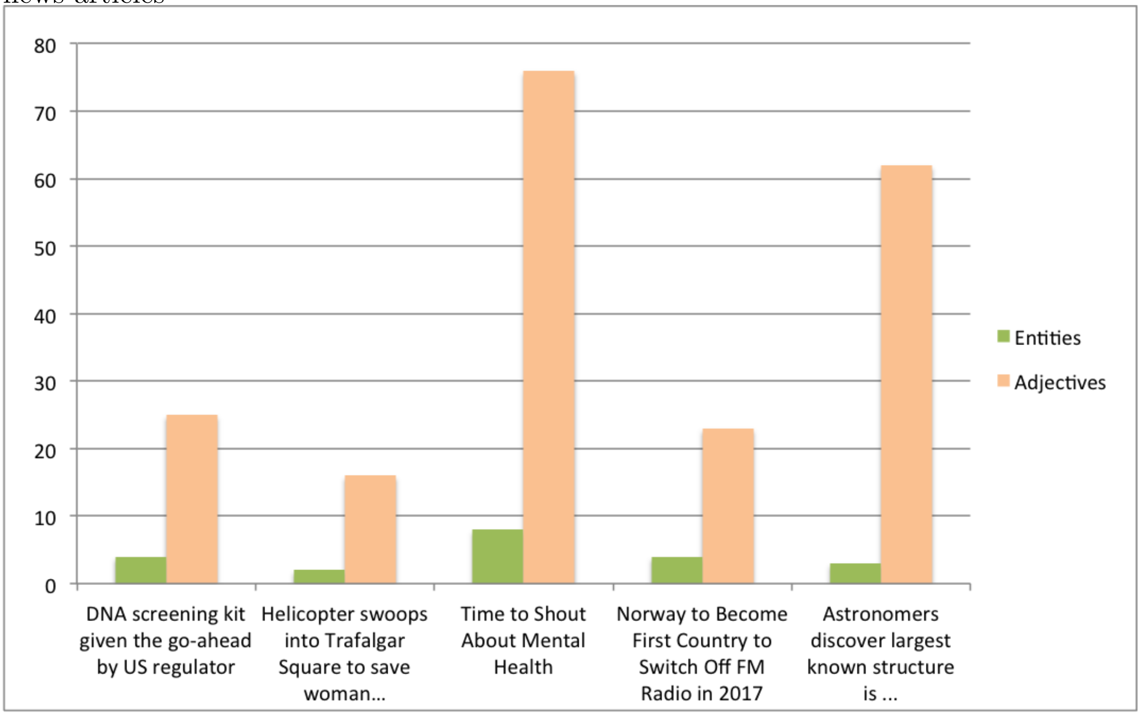
3.5.3 Tests

A series of tests were carried out to determine the relationship between extracted entities and adjectives, and the prospects that the key words will return a search result from Last.fm.

Distribution of Entities to Adjectives

The first test was to done to understand the relationship between the number of entities and the number of adjectives extracted. This did not involve the querying of Last.fm

Figure 3.6: Bar graph showing the ratio of entities to adjectives extracted from 5 different news articles

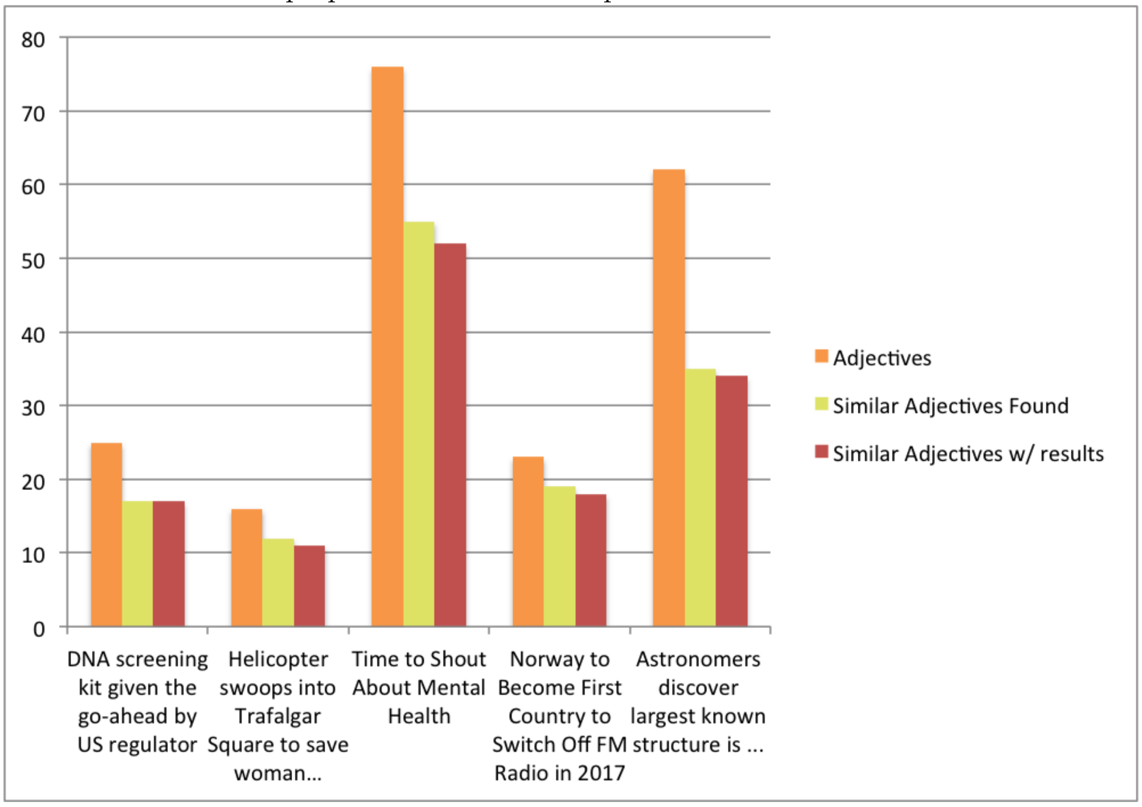


The amount of entities extracted from a passage is far fewer than the amount of adjectives. It can be inferred, using this graph, that entities need to be considered first. If one is to abide by the philosophy of the search query that it should be made up of an entity and an adjective, then it would be far more efficient to search entities first. In this way, one can prioritise which entities provide search results in Last.fm. The graph also conveys the fact that entities were always found within each text, despite all of the shortcomings that the NLTK NER extraction process has (see subsection **??**).

Prospect of Similar Adjectives

Each adjective extracted from the text had their most similar word returned from the word2vec model, a neural network trained on 6000 album reviews (see subsection 3.3). If the model returned a similar adjective then it would then be queried within the context of Last.fm tags.

Figure 3.7: Bar graph showing the ratio of similar adjectives returned from a set of adjectives across 5 news articles. These similar adjectives were then queried with tag search. The resultant proportion of successful queries is shown.



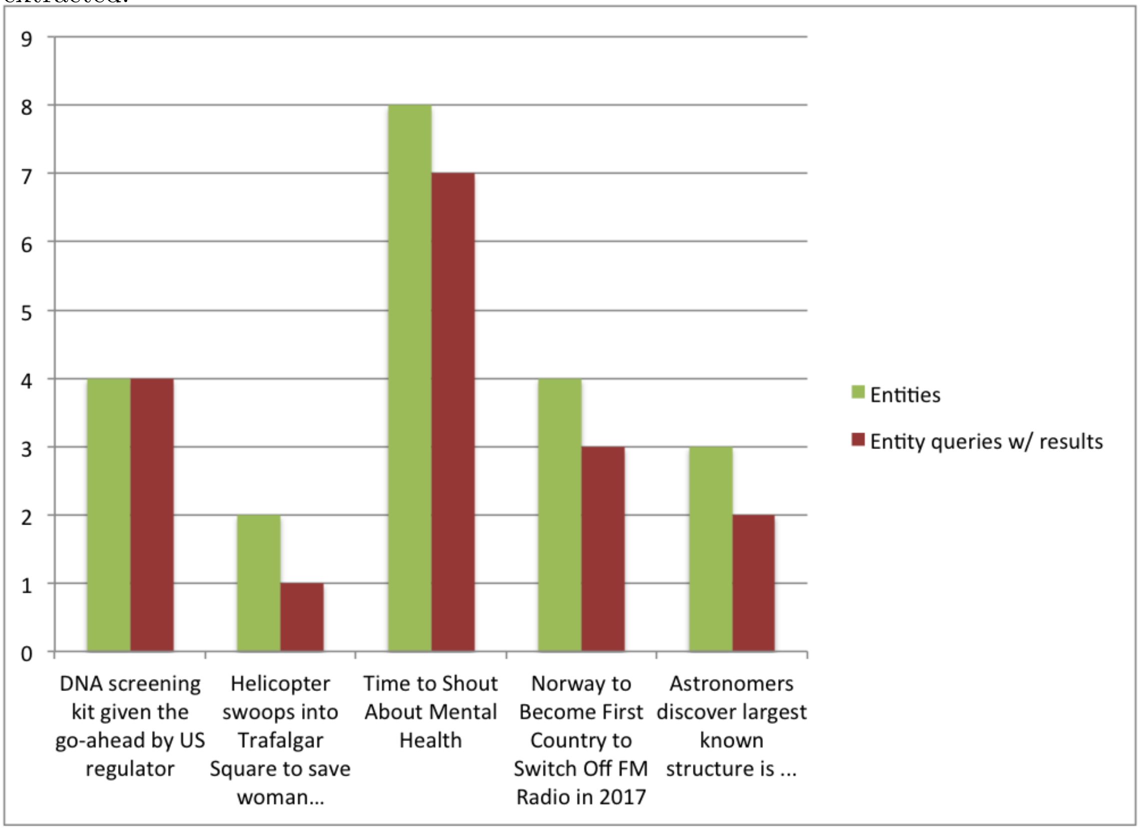
From the graph above, it can be inferred that a similar adjective is very likely to return a result from Last.fm. This evidence bolsters the case that similar adjectives can be used as an external provider when the results returned by adjectives from the text is minimal.

Using similar adjectives can add variation to the system and increase the likelihood of relevant results in the case of texts containing rare or sparse entities.

Prospect of Entities

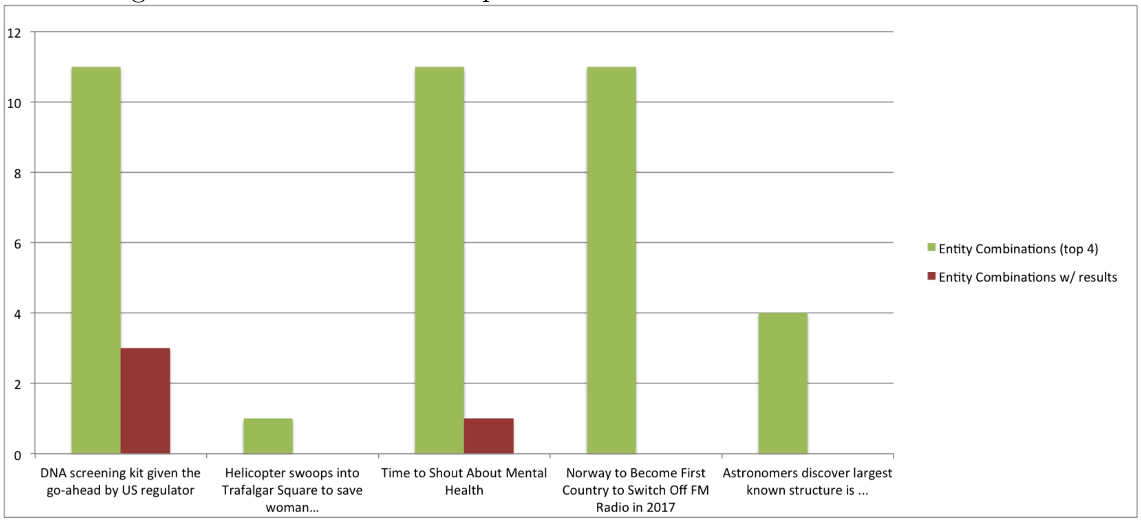
Two final tests were done in order to determine the prospects of the number of results returned for entities queried within the tag search. The first of which, tested the proportion between the number of entities extracted and the number of entities with successful queries (non-zero results from tag search).

Figure 3.8: Bar graph showing the proportion of successful entity queries from the entities extracted.



It can be seen from this graph that at least half of entities searched from each article returned a successful query. Following up from this a second test which considered the scenario of combinations of entities being considered together. From each article, the top 4 entities were combined. Each combination variation, ignoring all combinations with only one element, were queried within the tag search.

Figure 3.9: Bar graph showing the proportion of the number of successful queries when considering all combinations of the top four entities in each of the five articles.



There is little to no prospect of a several entities returning a successful result from what can be understood by the graph. The "Norway" and the "DNA" articles both provided eleven queries, yet only one article returned successful queries. It can be inferred, from the graph, that the likelihood of several entities returning a non-zero rest count is unpredictable and down to the words themselves.

3.6 The recombEAn Algorithm

3.6.1 Tag Classes

A set of named entities and set of adjectives are extracted from the text. Each substring (word) of the original passage extracted becomes a Tag object. Creating an object for each individual substring allows one to store further information about it. A tag object is created with attributes for it's name and it's frequency. The frequency parameter takes the sum of all the string's appearances within the passage. It is used as the key value for sorting a list of tags within the algorithm. Ultimately, it determines consideration priority within the algorithm. Several subclasses, that inherit from Tag, were also created. The subclass chosen for a particular string depends upon which function that word was extracted from. Tag has 3 subclasses, Ent, Adj and SimAdj. Each of these classes inherits the name and frequency attributes that were declared at creation. The Ent objects, are made up of words extracted from Named Entity Recognition. The class has an extra attribute, entity type, in which the category of entity is declared such as location, date or name (see subsection 3.2.2). Adjective strings extracted using PoS tagging, see subsection 3.2.1, are used to create a set of Adj objects. The Adj class contains an additional parameter, an integer, which determines how many similar adjectives to be stored. At initialisation of a new Adj object, a function call to the word2vec model is made. The call returns a set of SimAdj objects whose names are determined by the n most similar words, with n being a variable number of results. The class hierarchy is displayed below in figure 3.10.

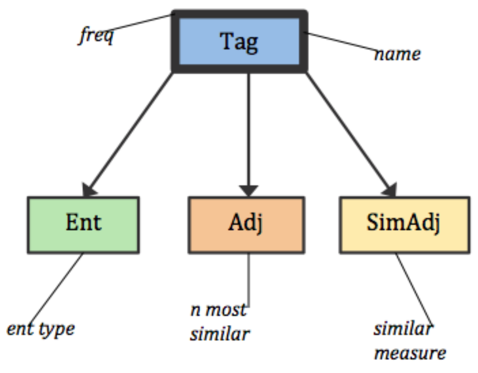
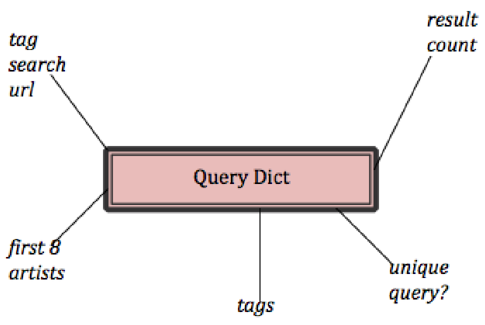


Figure 3.10: Tag Class Hierarchy

3.6.2 Query Dictionaries



When a set of tags is used to query Last.fm, using the *tag search* method, the result is stored in a Python dictionary. The result count, the first 8 artists and a boolean *unique* value are associated with the tags from the query. This *unique* value returns true if the query returns only one result. It was decided that the unique queries are highlighted within the list because they might provide interesting responses for music recommendations. If any of the first 8 artists contain words in common with the tags searched, then they are ignored. This was because it was felt that an artist name was likely to be irrelevant when determining an appropriate music choice.

3.6.3 Algorithm

Entities

The list of returned Ent objects is then sorted into a list. The top n, usually 4, entities are then moved to alternative list. This list is prioritised over the rest of the entities. The top n entities are then queried using Last.fm's tag search. Combinations of entities are not considered at this point, because of the lack of successful entity combination queries from the tests (see subsection 3.5.3). After querying the top n entities, if the top entity query returns more results than the result count minimum, defaulted at 10. Then stop querying entities, otherwise query the rest of the entities.

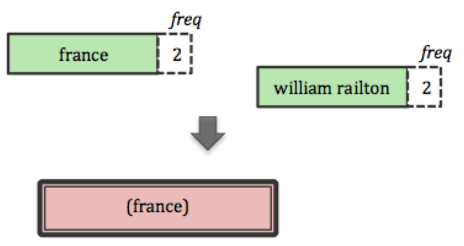


Figure 3.12: Entities and the successful queries

Adjectives

Adj objects are split into two separate lists. If the list of extracted adjectives is above threshold a, within the code this is 8, then split the adjectives into 2 lists determined by their frequency value. If the frequency value is 1 store in a list of unique adjectives. The other list, the rest of the adjectives, has it's number of elements reduced to a.

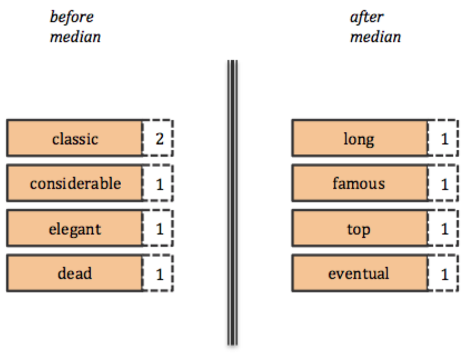


Figure 3.13: Adjectives split by median

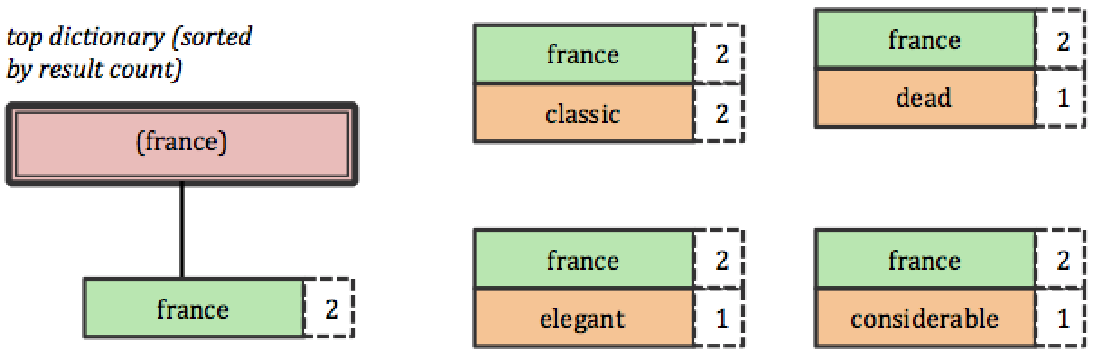
This reduces the amount of queries to be searched. The list of unique adjectives is largely ignored by the system. Unless the boolean parameter, *unique_supply*, is marked true, then the system can consider 5 random unique adjectives. This parameter is defaulted to False, however, because the number of queries greatly increases and thus takes longer to process.

A midpoint for a adjectives is found. This value is used to split the adjectives once again. These two lists are queried differently within the algorithm. The top half is also the adjectives with the highest frequency values because the list still remains sorted after the unique adjectives have been removed.

Looped Processes

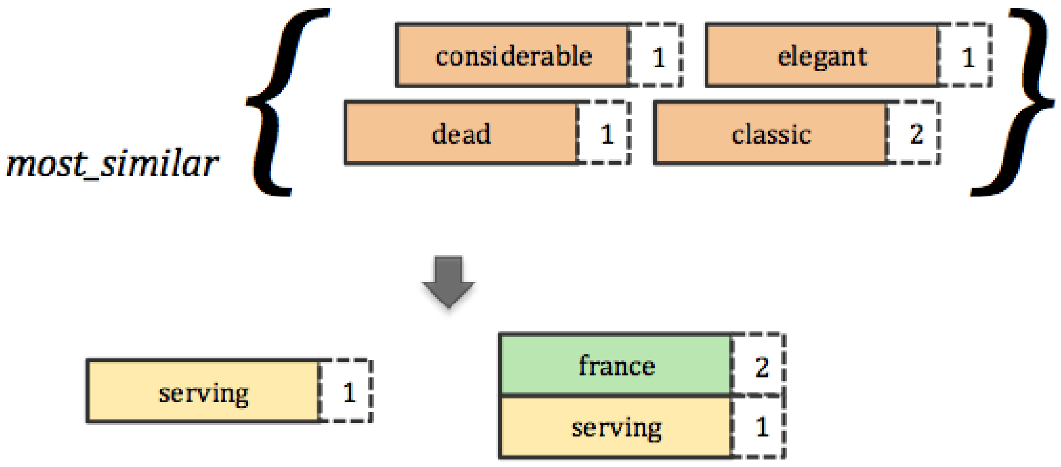
For the remaining portion of the algorithm, each half of the list of adjectives is considered sequentially. If the *unique_supply* parameter is marked true, then an extra cycle including five random unique adjectives occurs. Each ublist of adjectives goes through several processes including querying the most similar adjective and finding new recombinations.

Figure 3.14: First half of adjectives with all queries created involving top entity query

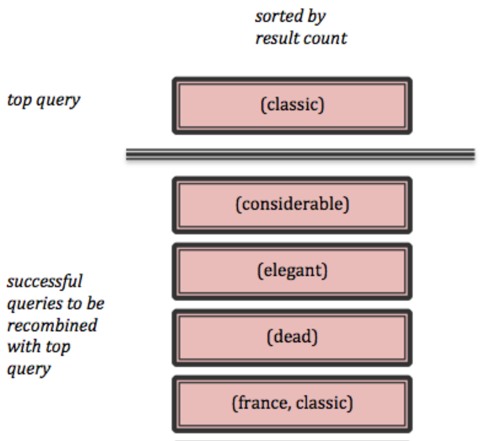


Primary Adjective Queries The first of these processes groups the top entity and each adjective in the list sequentially. Additionally if the top half is involved, the adjectives will be queried uniquely as well. Querying each adjective uniquely means that these adjectives can be recombined with tags from other queries in order to make longer and more specific tag sets. There is a likelihood that a singular adjective could become the top suggested query at the end of the algorithm function, going against the original philosophy that the top result should contain tags of an entity and an adjective. This is prevented by the final list of queries being sorted by their total tag frequency. The resultant list is likely to have gone through several recombination cycles so it is unlikely that a query with a singular tag will be highest on the list.

Figure 3.15: Processing more sets of tags created using the word2vec similarity measure



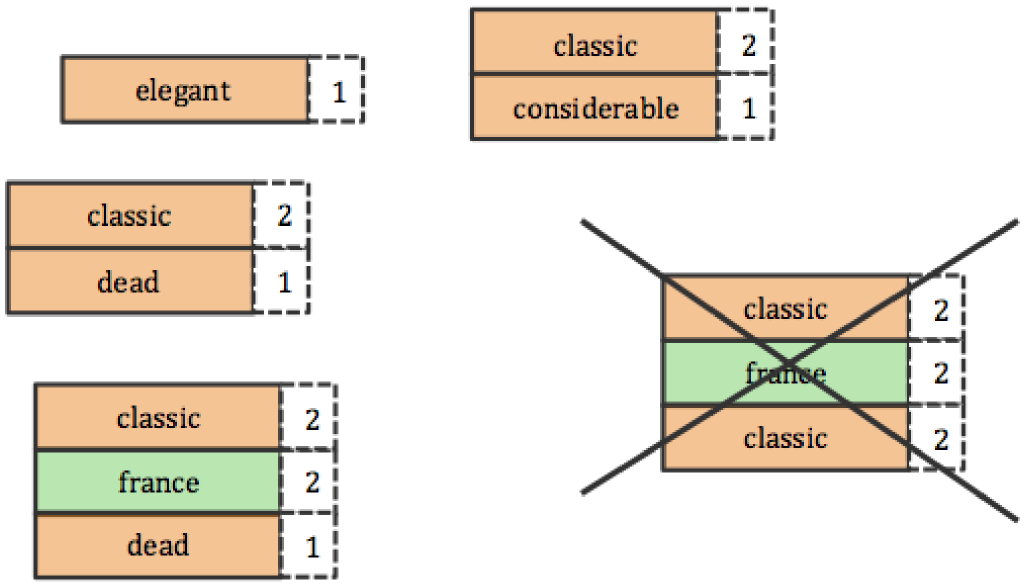
Most Similar Adjective All of the adjectives from the half of the list in consideration are placed in a similarity function call to the word2vec model (see section 3.3). The most_similar function call takes an arbitrary length array. Each adjective is called by this function separately. Any adjectives that returns a successful result, rather than an error is added to an array. The most_similar function call then takes the entire array as it's input. The test results given in subsection 3.5.3, show that a similar adjective is likely to return a result from Last.fm. One similar adjective is enough, too many and the query list will be polluted with lots of queries and some possibly may not be appropriate due to the computer accuracy. Entering all the words for one similarity function call is likely to return a more relevant word than getting a similar word for each adjective. The resultant SimAdj object returned from this call has it's name added to the query list.



Each query from the list is now called by the tag search function, any non zero results are added to the searched list. The searched list is sorted by the result count from the tag search in descending order.

Recombination The recombination of searched queries is the final process to take place. This is where the tag queries are concatenated in order to aim for queries that involve as many tags as possible and thus, are more specific. For a given number of recombination cycles (defaulted at 2), the top query dictionary is evaluated from the list of searched queries. The rest of the queries have they're tags concatenated with the tags of the top dictionary, any duplicate tags are removed. A check is done to see if the new recombined tag query has not already been considered the algorithm. Every list of tags that passes this check is then queried using the tag search function. All successful queries are added to the list of searched queries, which is then sorted by result count again. There is a possibility that the top query from the previous cycle will be the top query again. To provide more variation, the top dictionary is appended to the end of the list of searched queries so that, on the next cycle of recombination, the new queries are concatenated from the second query dictionary's tags and every other dictionary's tags. On the third cycle of recombination, the third from top query dictionary is used and so on and so forth.

Figure 3.17: Tag recombinations considered from figure 3.16

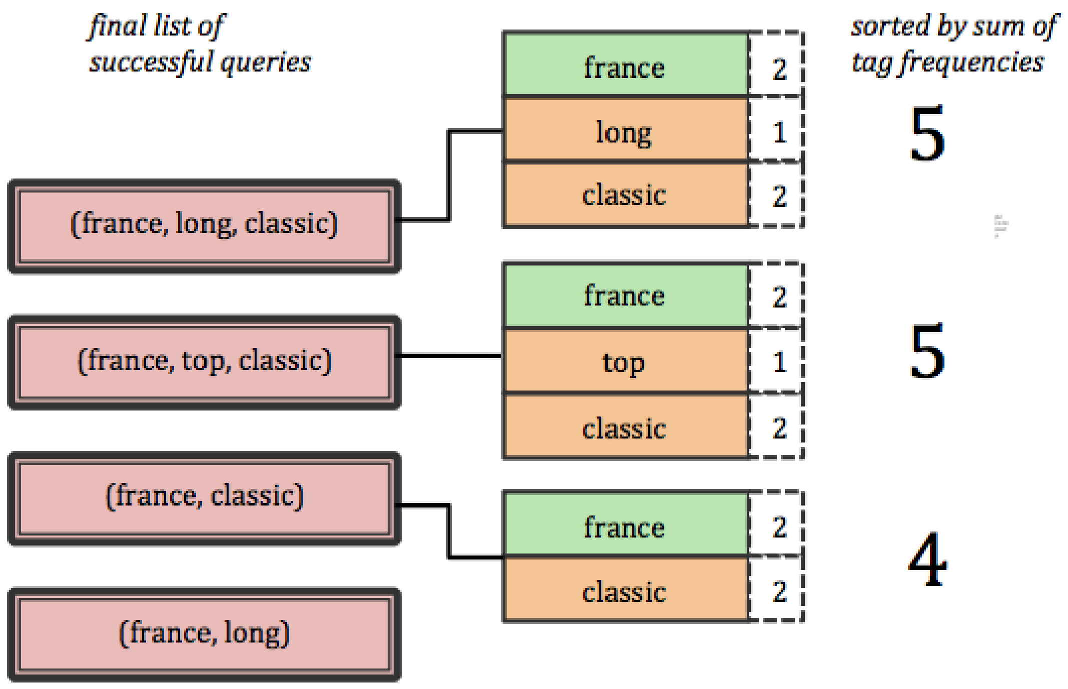


After several cycles of recombination, any adjectives that have not been considered yet, i.e.. the bottom half, go through the the processes of singular querying, similar adjective querying and recombination querying. If one has evaluated the parameter, *unique_supply*, to be true, then the system performs the process cycle one more time using up to 5 unique adjectives.

Searched Queries

The list of successful searched queries returned is sorted by the sum of their tags' frequencies. The reason for the summation is because it strikes a balance between the number of tag elements and the frequency of the tags. If the top query contains only one tag then that passage of text is likely to mention that word many times and thus be appropriate. Equally, if the query's tags are made up of several tags then it is more specific and thus likely to be appropriate The original philosophy of providing a resultant query with an entity and an adjective is maintained. All adjectives are considered with the top entity concatenated to them so their frequency count is likely to be larger from the beginning. If the top query only contains entities or adjectives then it is likely to be appropriate because the tag words work well together and provide lots of results or the words are mentioned a frequent amount of times and thus still appropriate.

Figure 3.18: Extract from top of searched query list after all adjectives considered



Querying Echonest

Finally the searched query list, sorted by total sum of tag frequency, is iterated over. Each query's search results, consisting of a set of artist names and, if returned by the search, a song name, are tested with the Echonest energy query derived from the mean arousal score (see section 3.4). Sequentially the artist, and possibly a song name, are queried using the Echonest song search API. The returned artist name, and possible song, are added as parameters to a url with domain name *developer.echonest.com*. Also within the url's parameters are the minimum energy and the maximum energy. To determine these, the Echonest energy query is given a 0.25 leeway. 0.125 is added and subtracted from the original value in order to create the maximum and minimum energy parameters. The url is requested and the results parsed with JSON into an array. If the resultant array has values then one is chosen at random. In the case that a song is being searched, then this still applies, except the array will return one value which will be the song, if it is within the energy parameters.

3.7 Testing

?? A test was carried out in order to determine whether the recombEAn algorithm was returning appropriate and relevant song recommendations. 20 participants filled out a questionnaire via Google Forms, a screenshot of which, can be found in appendix A. It featured an introductory page where the test's nature and the system were explained. Before any questions were presented to the participant, it was required that they agree to the terms of the test. Further explanation about what was explained in the introductory page and how it agrees with the BCS code of conduct can be found in chapter 2. The questionnaire consisted of 8 articles each coupled with a piece of music. 8 was chosen as an appropriate number because any more and the participant might suffer from boredom. 4 of the couplings were chosen by the system and 4 were chosen at random. The random generation process involved generating songs for each article and then shuffling the couplings so that each song was paired with a different article. The URL links to both were included within the questions. All the pieces of music were linked via the video sharing site, Youtube[12]. The articles were collected from a variety of different news sites, the full rundown of the article URLs can is shown in table 3.1. The question order was randomised for each participant. This was to ensure that no biases could arise out of each participant

[12]Youtube:`https://www.youtube.com/`

hearing the songs in the same order. Mood choice is likely to be affected dependant on the song heard prior.

Table 3.1: Articles with respective URLs

Shortened Article Title	**Short URL**
Simon Cowell's Ultimate DJ...	http://bit.ly/1bE2X39
Female, over 65 and Danish...	http://bit.ly/1DT9ca0
NASA May Have Accidentally...	http://bit.ly/1FjWCpJ
New York state to dim...	http://bbc.in/1zhBLCx
Nepal earthquake...	http://bbc.in/1EqvctX
Binge drinking raises...	http://dailym.ai/1Eqvdhu
Bruce Jenner's iconic Wheaties...	http://dailym.ai/1QziDnE
Rabies still kills...	http://bit.ly/1APxPUJ

Table 3.2: System generated music recommendations

Shortened Article Title	**Song (Artist - Track)**
Simon Cowell's Ultimate DJ...	Anoop Desai - Lost & Found
Female, over 65 and Danish...	Bodebrixen - Unhappy Country Song
NASA May Have Accidentally...	Parliament - Theme From The Black Hole
New York state to dim...	Masta Ace - Beautiful

Table 3.3: Randomly generated music recommendations

Shortened Article Title	**Song (Artist - Track)**
Nepal earthquake...	Wire - Reuters
Binge drinking raises...	Katy Perry - Legendary Lovers
Bruce Jenner's iconic Wheaties...	Jackie Wilson - Your Love Keeps Lifting Higher
Rabies still kills...	Laura Michelle Kelly - Somewhere Only We Know

For each article-song couple, participants were asked to mark a rating of appropriateness on a 6 point Likert scale. The range of pre determined responses to choose from were:

- 1 - Extremely Inappropriate
- 2 - Very Inappropriate
- 3 - Inappropriate

- 4 - Appropriate
- 5 - Very Appropriate
- 6 - Extremely Appropriate

A 6-point Likert scale has no definable neutral midpoint and thus some researchers may argue that without this neutrality a bias would be introduced. A participant may feel totally neutral about a question but a 6-point scale forces a non-neutral response which is likely to be accentuated in the negative direction (Gwinner, 2006). On the other hand, having 6 points forces the participant to come to a decision. In the context of the test, the decision concerning mood and thematic content is entirely subjective and it was felt that a neutral response would be an 'easy way out' for the participant. Arguably, in this context, a participant who has neutral feelings towards an article-song coupling is unlikely to consider the pairing a good match and thus, appropriately, should mark negative. An even number of ratings means that the respondent has to commit to a positive or negative response (Gwinner, 2006).

For each question, the participant had to make 2 judgements of appropriateness, these were:

- Thematic Content
- Mood

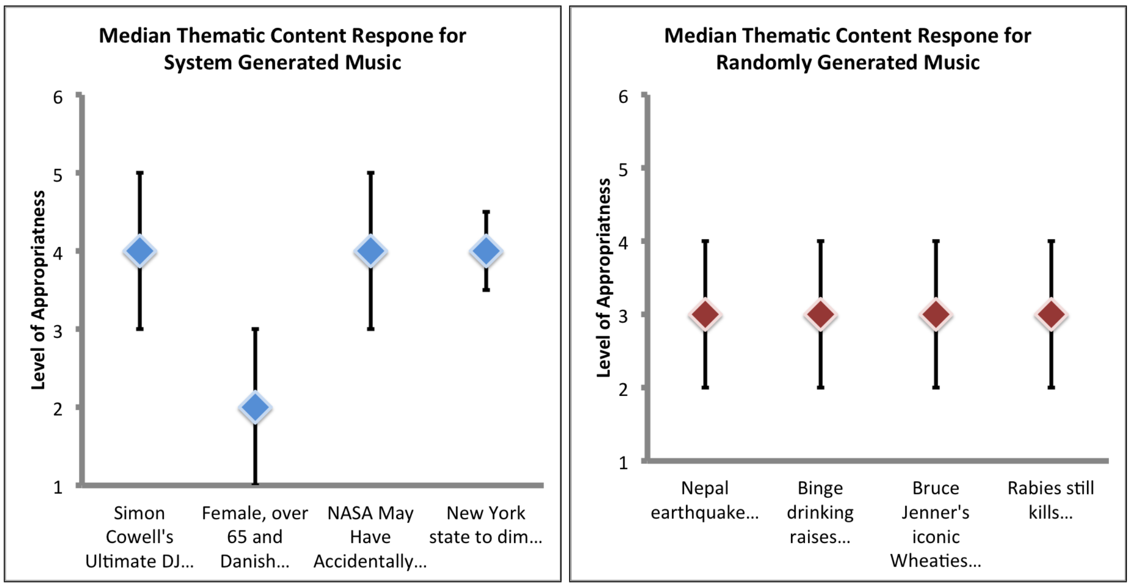
In this way, the system can be tested by to determine the nature of success within it's ability to extract context from an article (thematic content) but also it's ability to judge tone (mood), from the arousal measure and considered adjectives.

Chapter 4

Results

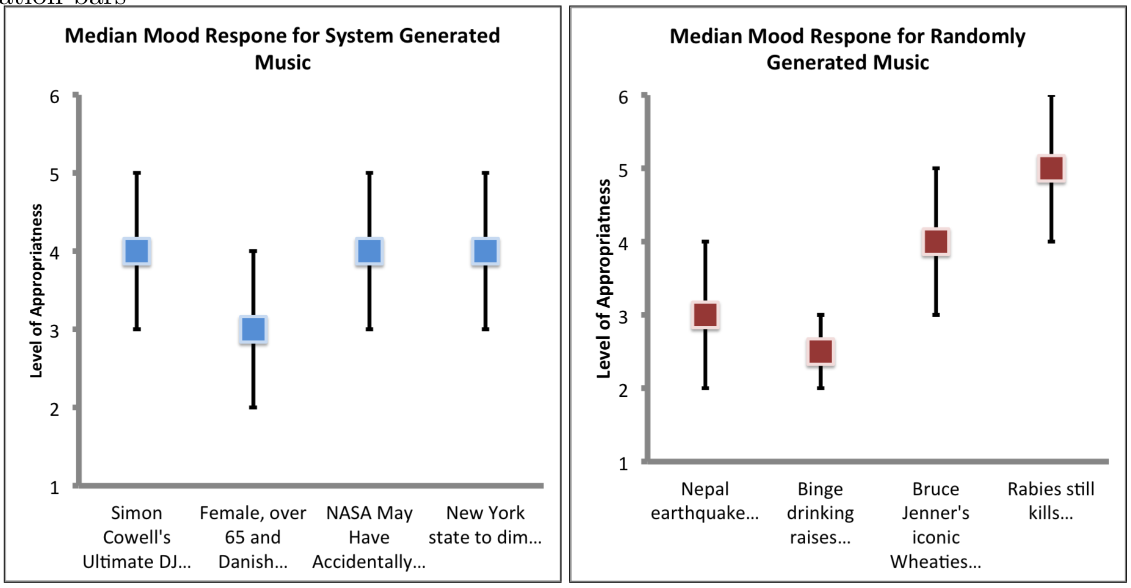
The results from this test were placed in a spreadsheet and the median mark from the 20 participants was plotted on a graph. The median absolute deviation (MAD) for each result was also plotted (the error bars). The median was chosen, rather than the mean, because a Likert scale is ordinal data rather than interval data. There is no clear scalar measurement between the points as they represent responses rather than quantifiable values. Calculating the mean relies on the assumption that the ‘psychological distance’ between the points on the Likert scale is the same (Kostoulas, 2013). The median is a better use of average because it finds the middle response rather than using the values as measurements.

Figure 4.1: Median graph plots for thematic content appropriateness responses with absolute deviation bars



Comparing the two graphs in figure 4, it can be seen that the system-generated music provided a more varied response than that of the randomly generated music. Most respondents seemed to agree that the majority of the system generated music was appropriate. Particularly article 4 which has a very minimal deviation that does not stretch into inappropriate territory. However, article number 2 did not portray the same feelings with its median and deviation firmly lying in inappropriate. The results from the random generation are consistent with what would be expected, inappropriate, and the dispersion of results shows that there is no obvious outliers. Although the dispersion also portrays the fact that some of the respondents found these articles to be inappropriate with their music couplings.

Figure 4.2: Median graph plots for mood appropriateness responses with absolute deviation bars



The mood responses (see figure 4) shows less consistency than that of thematic content. The respondents again seemed to agree that articles 1, 3 and 4 were appropriate. However their dispersion shows that there are outliers which seem to disagree. Article 2's median show that most respondents tended to agree that it's mood was inappropriate. The randomly generated articles provided not very consistent results with what was to be expected. Most respondents agreed that article 4 from random generation was appropriate with it's deviation showing that there are no obvious outliers that would disagree. Article 3 many believed to be appropriate however it's deviation shows that some disagreed with this response. The other articles, most respondents marked as inappropriate. Article 2 has very little variability and it can be said that the majority found it's mood to be inappropriate.

Chapter 5

Analysis

While the data conveys some messages it is difficult to say anything with absolute certainty. Many of the results have dispersion plots that lie within both appropriate and inappropriate. It was decided to perform further analysis on the data

5.1 Mann Whitney U

The Mann Whitney U test compares 2 populations of samples against each other. It is a non-parametric test which means it makes no assumptions about the distribution of scores ie. it does not concern itself with the measurement of distance between the intervals. In this context, there is no distribution because there is no scalar measure of appropriateness. The ranks have no real numerical value and are ordinal. A null hypothesis needs to be created for the Mann Whitney U test in order to interpret the resultant value from the test. The null hypothesis is based on the assumption that there is no difference between the randomly generated music and the system generated music.

Null Hypothesis *The probability of one value from the group of system generated data being greater than than any value from the group of randomly generated data is the same as the probability of a value from the random data being greater than any value from the system data.*

In short terms, the test involves ranking every sample in both groups as one continuous rank. Each sample is given a new rank with the smallest value getting the rank 1 and so on. Any values that have ranks that tie are given the average rank of that tie. For each group the ranks of the values are averaged and if the values are found to be very different, 'the P value will be small' (GraphPad, 2015)

The Mann Whitney U test was carried out upon both the thematic content and the mood results using the module the appropriate contained within the scipy library[1].

Aspect	***p* value**
Thematic Content	0.0460
Mood	0.3331

The resulting p values conveys whether the null hypothesis can be rejected or not. A small p value of less than 0.05 warrants the ability to reject the null hypothesis. In this case, the thematic content p value is significantly small enough to reject the null hypothesis. The p value for Mood is above 0.05 and thus, the null hypothesis has to be accepted for this aspect.

5.2 Analysing Method

It is important to remember that there are often biases affecting the testing. One of which would be the fact that the experiment did not take place in a testing environment. Rather, it took place over the internet. Each participant filled out the form in his/her own time rather than under the watchful eye of a researcher. This means that the results could be skewed due to the participant possibly becoming bored with the test and rushing. Without test conditions, it is difficult to perceive whether participants filled in the sheet correctly.

The private environment allowed the participant to take as much time as they needed to fill out the sheet. On the one hand, the participant may have filled in only some of the answers and then taken a break. This break may have affected his/her mood at the time and when said participant returned to fill in the rest, the mood that they were in skewed their judgement. Although, it is difficult to make a valid case for this because judgement of mood is so subjective. This might explain the sparse results. The judgement of thematic content is much easier to make because it involves looking at the track name, artist name and listening to the lyrics. Mood is such an undefinable thing, it changes from person to person depending on the music and can't be watered down to a 6 point scale. The varied results with no implicit inferences definitely portrays this subjective bias.

The song itself may have caused a bias within the participant. If the person had a strong distaste for a certain song, their judgement of appropriateness may have been affected. This also might explain the sparse results for mood. On the other hand, the

[1]scipy.stats.mannwhitneyu: http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.mannwhitneyu.html

act of participants filling out the questionnaire in their own time allowed them to feel more comfortable than asking them to operate under test conditions where they could be subject to pressure.

The test may have suffered from characteristic bias where a participant subconsciously changes their behaviour for the purpose of the test. The participants were not aware that 4 of the questions were randomly generated, therefore the participant may have been seeking appropriateness in order to, in their minds, ”solve” the questionnaire. Participants would have been ”seeking appropriateness” rather than making a judgment. For each song, they were looking out for anything that matched as appropriate no matter how minuscule. Another issue with the test was the small sample size. With a sample size as small as 20 it is difficult to come to some certainty about whether the test was successful. For more viable result a larger sample size would need to be considered.

Chapter 6

Future Work

The recombEAn algorithm can consider many other factors of a webpage when determining a music decision. For future consideration, the happiness dimension from the affective norms database (see section 3.4) could be considered in conjunction with the Echonest *mood* parameter. Calculating the mean happiness provides another boundary for the music recommendation to fulfil and hopefully be more relevant. Alternatively, the colour scheme for the webpage extracted from the CSS content could also be used. The colours of the webpage could become the determiner for the *mood* parameter i.e. the colour blue is considered less "happy" than red or orange. The colour could also be used be mapped to the *key* parameter within Echonest. Adapting the algorithm to match verbs is another extension that could occur. Used in conjunction with the stemming algorithm, a verb could be stemmed in order to find variations of the word that exist within Last.fm's tag database. Those words could then be added into the algorithm's recombination cycle to widen the semantic space further.

The amount of words on the page could also be a determiner for the music recommendation. Longer word counts would lean the algorithm towards more ambient suggestions in order to aid reading. An internet browser plugin could be designed that implements the recombEAn algorithm. This would sit on the web browser's interface cueing up songs found by querying the web pages currently being browsed. A genetic algorithm could also be applied that sways the algorithm towards music that the user prefers. Thumbs up/thumbs down buttons would be added to the plug-in, that, when pressed store the songs tags from Last.fm. Any new recommendations would be more lenient toward music that contained tags shared in common with those that the user had liked before. A preferential list of songs could be saved and loaded when the user is browsing "micro-blogging" sites such as Facebook and Twitter where the semantic content wildly differs multiple times throughout the webpage.

Further optimisation would be investigated in which the number of queries carried out by the Google custom search would be minimised. In order to conserve the amount of queries carried out overall along with increasing the efficiency of the overall algorithm.

Chapter 7

Conclusion

It can be concluded that the recombEAn algorithm does provide promising results in terms of it's song recommendations in relation to thematic content. Although the small sample size means it's difficult to say anything for sure. From the test results and the Mann Whitney U evaluation, it can be shown that the system did achieve a level of appropriateness in terms of thematic content. The process of extracting adjectives and entities from textual content and finding matching tags can provide a viable music choice. The system, however, failed to provide a suitable music recommendation to fit the mood of an article, it can be inferred from the test. The failure of mood can be narrowed down to several factors. Some, of which, are not by fault of the algorithm but the subjective nature of mood and it's variability from person to person. Everyone has their own individual music preferences and the differences between these, means its difficult to come to a consensus.

Arguably there is a strong case that the failure of mood is down to the algorithm. The arousal measure was a single quantifiable measure that could define the chosen song if only an artist name was extracted from the tag search. Articles tend to contain a lot of words from both ends of the affective norms dataset. This factor means that most web pages provided similar arousal measure results. Although different news article sites provided noticeably different arousal scores, most scores would end up in the threshold between 0.4 and 0.6 because the super arousing words would be balanced with less arousing ones. Optimising the algorithm in the future, one would focus on further testing and remapping the relationship between arousal score and the Echonest energy parameter so that a small value change in the arousal score would warrant a bigger change in the energy parameter. Further parameters, such as the appropriately labelled attribute, *mood*, would be brought in to define a more specific Echonest query and hopefully return better results for mood in future testing. Further discussion of future work can be seen in chapter 6.

The recombEAn algorithm is the first step towards providing a content filtering recommender system that uses Internet web pages as it's context. The test shows that the recombination of keywords extracted from the textual content of the article is a viable technique for providing an appropriate music response, particularly in terms of thematic content. Further work needs to be put in to mood recognition within the algorithm, in order for the system to achieve the requirement of returning an appropriate music choice.

Bibliography

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. (2011). Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. Association for Computational Linguistics. 3

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media. 12

Bradley, M. M. and Lang, P. J. (2010). Affective norms for english words (anew): Stimuli instruction manual and affective ratings. Technical report, Technical Report C-2, The Center for Research in Psychophysiology, University of Florida. 14

Dictionaries, O. (2015). verb - definition of verb in english from the oxford dictionary. 18

GraphPad (2015). Mann-whitney u test. 34

Gwinner, C. (2006). 5-point vs. 6-point likert scales. Technical report, Infosurv. 31

Kostoulas, A. (2013). On likert scales, ordinal data and mean values. 32

Lamere, P. (2013). Going undercover. 2

Poibeau, T. and Kosseim, L. (2001). Proper name extraction from non-journalistic texts. In *Computational Linguistics in the Netherlands. Selected Papers from the Eleventh CLIN Meeting*, pages 11–12. 12

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics. 12

Řehůřek, R. (2014). Making sense of word2vec. 13

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP*

Frameworks, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en. 13

Reisinger, D. (2009). Top 10 movie recommendation engines. 2

Spotify (2015). Information — spotify press. 1

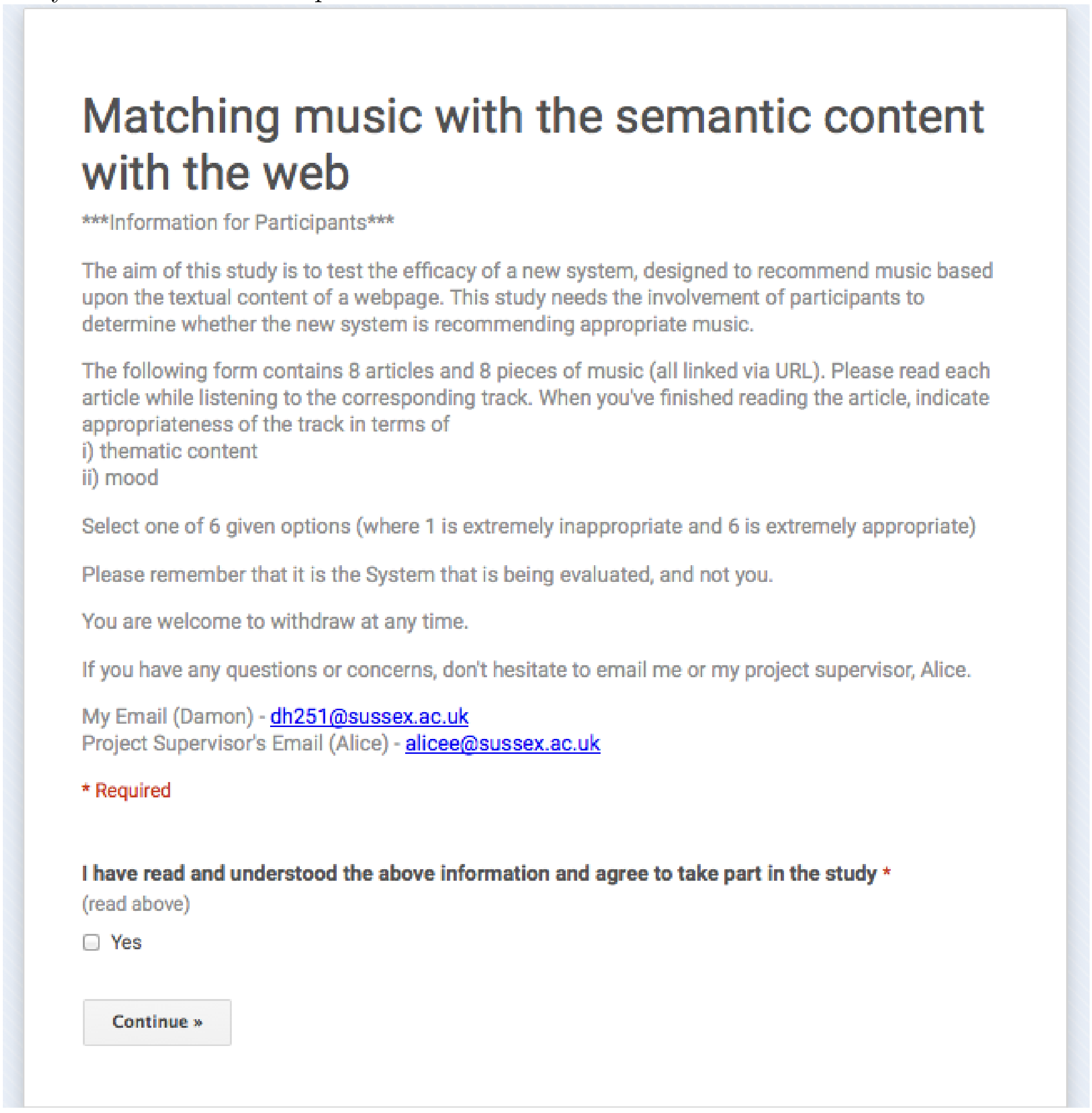
Thorogood, M., Pasquier, P., and Eigenfeldt, A. (2012). Audio metaphor: Audio information retrieval for soundscape composition. *Proc. of the Sound and Music Computing Cong.(SMC)*. 3, 11, 13, 16, 19

Yumusak, S., Dogdu, E., and Kodaz, H. (2014). Tagging accuracy analysis on part-of-speech taggers. *Journal of Computer and Communications*, 2(04):161. 10

Appendix A

Google Form Questionnaire

Figure A.1: First Page, mandatory checkbox given that had to be agree to in order to carry out the rest of the questionnaire



Matching music with the semantic content with the web

Information for Participants

The aim of this study is to test the efficacy of a new system, designed to recommend music based upon the textual content of a webpage. This study needs the involvement of participants to determine whether the new system is recommending appropriate music.

The following form contains 8 articles and 8 pieces of music (all linked via URL). Please read each article while listening to the corresponding track. When you've finished reading the article, indicate appropriateness of the track in terms of
i) thematic content
ii) mood

Select one of 6 given options (where 1 is extremely inappropriate and 6 is extremely appropriate)

Please remember that it is the System that is being evaluated, and not you.

You are welcome to withdraw at any time.

If you have any questions or concerns, don't hesitate to email me or my project supervisor, Alice.

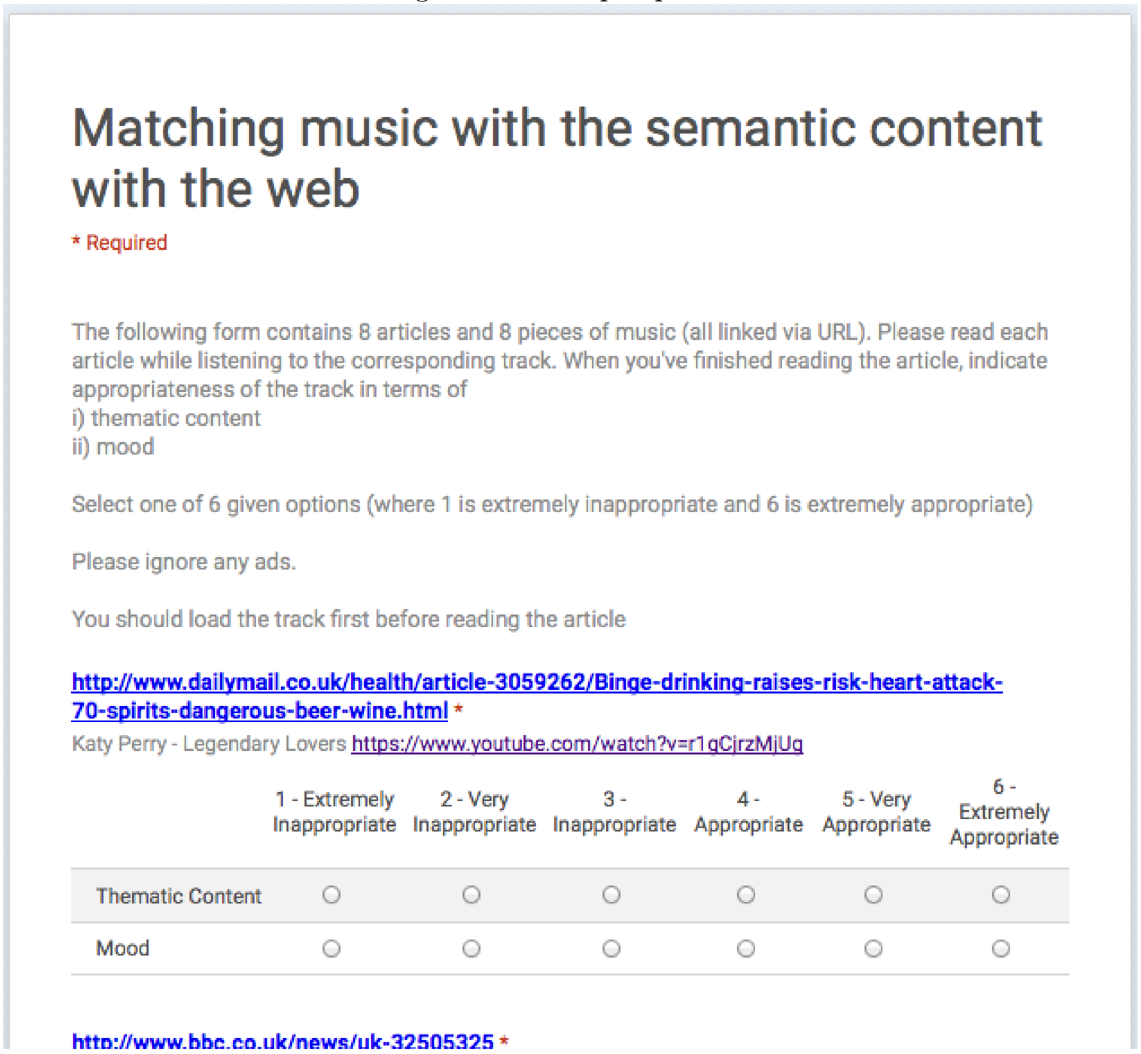
My Email (Damon) - dh251@sussex.ac.uk
Project Supervisor's Email (Alice) - alicee@sussex.ac.uk

* Required

I have read and understood the above information and agree to take part in the study *
(read above)

☐ Yes

Figure A.2: Sample question



Matching music with the semantic content with the web

* Required

The following form contains 8 articles and 8 pieces of music (all linked via URL). Please read each article while listening to the corresponding track. When you've finished reading the article, indicate appropriateness of the track in terms of
i) thematic content
ii) mood

Select one of 6 given options (where 1 is extremely inappropriate and 6 is extremely appropriate)

Please ignore any ads.

You should load the track first before reading the article

http://www.dailymail.co.uk/health/article-3059262/Binge-drinking-raises-risk-heart-attack-70-spirits-dangerous-beer-wine.html *

Katy Perry - Legendary Lovers https://www.youtube.com/watch?v=r1qCjrzMjUg

	1 - Extremely Inappropriate	2 - Very Inappropriate	3 - Inappropriate	4 - Appropriate	5 - Very Appropriate	6 - Extremely Appropriate
Thematic Content	○	○	○	○	○	○
Mood	○	○	○	○	○	○

http://www.bbc.co.uk/news/uk-32505325 *