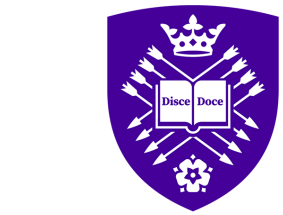


University of Sheffield

# Project Euterpe: A Lyrics-Based Music Recommendation Application

Freddie Butterfield

*Supervisor:* Varvara Papazoglou

*A report submitted in fulfilment of the requirements*
*for the degree of* MEng in Computer Science (Software Engineering) with a Year in Industry

*in the*

Department of Computer Science

January 28, 2026

## Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Frederick Webster Butterfield

Signature: 

Date: January 28, 2026

# Abstract

The lyrics of songs are an unused piece of metadata when it comes to making recommendations on large streaming platforms such as Spotify and Apple Music, and the concept of using lyrics to aid or assist recommendations has been previously shown to work. In this report we showcase *Project Euterpe*: a web application that makes music recommendations based entirely from music lyrics. We start by discussing the technologies surveyed and used, with a focus on a way to capture the content of the lyrics, before moving on to describe how these methods were implemented in the design. We show how the application performed when used in a survey done by 21 participants, and use their feedback to evaluate the performance of the system, as well as discuss the steps that can be taken to improve it.

## Acknowledgements

I would like to express my utmost gratitude to my supervisor Varvara Papazoglou for her great help and substantial contributions to the project, namely the music4all dataset. I would also like to thank my close friends, family, and those that participated in the survey.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the 21st century, the ability to listen to music has been more available and convenient than ever before. Streaming services such as Spotify and Apple Music allow you to listen to music from now to centuries ago, and new releases from artists continue to expand the already vast library of songs that these services provide. The ease of having all your favourite songs and albums on your phone or personal device means that you do not have to pay for outdated or obsolete technology, such as cassettes and CDs.

As well as being able to listen to music, many streaming services will try to recommend you new music. This could be more songs from an artist you already love, or songs from artists you may have never even heard of. Most streaming services such as Spotify currently accomplish this through 2 methods: collaborative filtering - which looks at other users' playlists with the same songs as you and attempts to fill in the gaps, and content-based methods - which look at the songs' content and audio signals to recommend songs with a similar style or energy.

Yet surprisingly the current methods lack the consideration of *lyrics*. Arguably there is a lot to be said about a user's taste in music when it comes to the lyrics and meaning of a song; a powerful property of songs that streaming services do not make use of. Within this report, I will showcase the performance of a web application that recommends songs based purely off of lyrics, and use user feedback to support (or disprove) the use of lyrics as a factor in traditional music recommendation techniques.

## 1.1 Aims and Objectives

The overarching aim of the project is **to develop and evaluate a lyrics-based music recommendation application**. We will then use this application to fulfill three sub-aims: a) **Evaluate the specific vector representation methods used to provide recommendations**, b) **Conduct a survey in which participants evaluate the application and answer questions about lyrics-based music recommendations**, and c) **Evaluate the effectiveness and practicality of lyrics-based recommendations**.

There has been some work done in the NLP research area on lyrics-based recommendations, and these will be discussed later. These papers mostly show that a computer or proposed system can do a good job at giving song recommendations, or correctly classify the genre of the song based on lyrics. This evidence alone may sound promising, however what these papers do not consider is the human element to it: Do people find lyrics important when looking for recommendations? Do people tend to enjoy songs with a similar meaning or lyrical content? Etc. Later in the report, we will look at the results of the survey in which participants have used the application for recommendations themselves which will help us answer some of these questions.

## 1.2 Overview of the Report

The rest of this report will cover:

- The various papers studied to find suitable methods for word and document vector representation, word and vector pre-processing methods, and related work.
- Code libraries that will be used to help make coding the project simpler, as well as the current datasets available.
- An overview of the requirements of the project and their priority.
- A detailed description of the design and implementation of the system.
- The testing phase of development, discussing how the system had evolved during the course of the project.
- Discussion of the results, and what we can infer from them.

# Chapter 2

# Literature & Technology Survey

The process of automatically and accurately analysing the meaning of a song has proven to be a computationally intensive and time-consuming procedure. Many systems have been proposed that build off of the concept of sentiment analysis, which aims to *extract* emotions, sentiments, or opinions from text. One such system was proposed by Erion Çano, who developed a sentiment analyser for song lyrics [1]. This system required a large amount of manually annotated training data, which we unfortunately do not have the time to obtain for this project. Instead of a full sentiment analyser, we may want to look at a weaker and less computationally intensive approach to "representing the meaning" of a song, since we don't want to hinder the performance of the application.

The methodology chosen when conducting the research was entirely literature-based: looking for suitable papers and code libraries that could be potentially used for the project, and in this chapter I evaluate the various technologies that can be used for the application, such as vector representation methods and pre-processing steps. This section starts off by looking at the main methods used to represent text/words in vector space, and we will later look at how they have been used for music recommendation.

## 2.1 Word Vector Representation Algorithms

Arguably the most important part of the system will be the word vector algorithm - we will need a machine learning algorithm that can generate accurate word vectors, or 'embeddings', from our song lyrics. Word vectors aim to encapsulate the semantic context of a word in an n-dimensional vector. For example:

$$\vec{w_{dog}} = \{0.1, -0.2, 0.6, ..., 0.07\} \tag{2.1}$$

Each point in the vector corresponds to an encoding of some semantic or syntactic information about the word, so the more dimensions we have, the more the algorithm can say about the

word. Intuitively, words that are semantically similar e.g. 'dog' and 'wolf', will exist close together in the vector space.

The following sub-sections cover a handful of word vector representation algorithms that I have found particularly interesting and useful to the project.

### 2.1.1 Global Vectors for Word Representation (GloVe) and Word2Vec

From "Text Classification Algorithms: A Survey" [2], there were two algorithms that seemed to be quite useful: 'Word2Vec' [3], and 'GloVe' [4]. Both Word2Vec and GloVe are said to capture the meaning in the words, which is incredibly important for this project. However, one of the major disadvantages to these methods is that they will not be able to handle words outside the vocabulary of the training data - so the dataset will need to be sufficiently large to create a large enough vocabulary to cover the test songs. Datasets for this project are discussed later in section 2.4.

GloVe can be used in three ways: there is a pre-trained model trained on five corpora (these being Wikipedia 2010, Wikipedia 2014 Gigaword 5, Gigaword 5 + Wikipedia 2014, and Common Crawl) with a total of 54.9 billion tokens. The second way involves training GloVe from scratch with custom corpora using its source code. The final way involves combining the first two with a pre-trained model that has been retrained with the additional custom corpora. In the interest of memory and time, it may be best just to stick with training just the lyrics corpora. In section 2.9, we find that the re-trained and the self-trained models perform similarly well.

Looking at the two papers individually shows that Word2Vec looks at the words in a much more local context than GloVe, dubbed a "window-based" approach. This is shown in its two main models: "skip-gram" - which predicts surrounding context words given a target word, and "CBOW" (Continuous Bag of Words) - which predicts a target word given the context words (this window is usually four words before and after the target).

As opposed to a window-based method like Word2Vec, GloVe is an unsupervised matrix factorization method using a global term-term co-occurrence matrix; creating word vector representations with the entire corpus taken into account. With this in mind, it would seem that logically GloVe would be the better choice when it comes to representing the meaning of an entire song, as a sentence or a small section of a song does not always provide meaningful information and thus the overall meaning could possibly be misrepresented with Word2Vec.

### 2.1.2 Bi-Directional Encoder Representations from Transformers (BERT)

One of the most recent and influential advancements in NLP has been the BERT algorithm [5] by Devlin et al. BERT makes use of the concept of Attention [6] - a power concept that allows the algorithm to focus on certain parts of a text and consider the context. This concept is then implemented in a multi-layer bidirectional encoder, and the paper describes the use of two models with 12 and 24 layers.

Comparing this to Word2Vec and GloVe, BERT is a deeper and more complex network that can create embeddings on both a token and sentence level. Despite its fabled high performance, BERT may prove to be tricky to implement and computationally intensive due to its architecture.

## 2.2 Alternative Methods

### 2.2.1 Latent Dirichlet Allocation

Looking at a more probabilistic approach to classifying text, "Latent Dirichlet Allocation" [7] describes an algorithm that will create topics and cluster relevant and useful words into those topics. This could be used to avoid the clusters of genres as seen in section 2.9, and with this algorithm we can perhaps create and analyse semantic fields to see if there are any overlaps between songs. Alternatively, we can just simply get the probability distribution as a vector using libraries discussed later in section 2.6.

### 2.2.2 Parameter-Free Classification Method

Looking towards the other end of complexity, Jiang et al propose a parameter-free text classifier [8]. The classifier uses a simple lossless compression algorithm along with a $k$-nearest neighbour algorithm to classify text. With no parameters or training stage, this is by far the most computationally inexpensive method of representing text in the literature survey, with accuracy results on-par with that of deep neural network methods such as BERT. Rather than a vector representation, this method assigns classes instead, which could be difficult to implement alongside the vector-based methods.

There are some other issues with this method as well. As the size of the vocabulary increases, the speed of the algorithm will start to slow down as the computational complexity of $k$NN is $O(n^2)$. Despite this, the algorithm could be useful if I do not have the computational resources required for neural networks (i.e. GPU).

## 2.3 Vector Weightings

### 2.3.1 TF-IDF

TF-IDF is a product of the **term frequency** (frequency of the word in a document) and the **inverse document frequency** (inverse of the number of documents containing a word). It is computed using this equation:

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i} \tag{2.2}$$

Where:

- $w_{i,j}$ is the TF-IDF weighting of a word $i$ and a document $j$.

- $tf_{i,j}$ is the frequency of the term $i$ in document $j$. Depending on the paper, terms may have been stemmed or lemmatized (or both) so words like 'going', 'gone', 'go', and 'went' will all correspond to 'go'.
- $N$ is the number of documents.
- $df_i$ is the document frequency of a word $i$.

Each word in the vocabulary will have a TF-IDF value (or weight) computed for it, and a TF-IDF matrix can be computed where the rows are usually the document ID and the columns are the terms.

It could be possible to compute a TF-IDF matrix and use it alongside the vector representations as a sort of weighting, allowing words that are more frequent across all documents to be less representative of the overall text and vice versa.

### 2.3.2 Smooth Inverse Frequency

As an alternative to TF-IDF, there is also the Smooth Inverse Frequency (SIF), which is a weighting used by Arora et al for the generation of sentence embeddings [9]. In this paper, they describe SIF as follows:

$$\text{SIF} = \frac{a}{a + p(w)} \tag{2.3}$$

Where $a$ is a changeable parameter and $p(w)$ is the unigram probability of the word in the corpus (i.e. the term frequency). The weighting can be implemented as is, or we can take the log of it too. This weighting is much easier to calculate and could potentially done at the time of the calculation of the song vector, rather than having to pre-compute an entire TF-IDF matrix. In the paper, they show that the sentence embeddings calculated with the SIF weighting perform better on textual similarity tasks than methods such as TF-IDF and baseline GloVe.

## 2.4 Datasets

In order to train the system, the project requires a suitably large dataset of songs and their respective lyrics in order to generate satisfactory vectors. The dataset may also contain metadata such as the artist, album, genre etc.

There are many such datasets available, like the Million Song Dataset [10] which contains audio features and metadata for over a million songs, or a smaller scale 10,000-song sample dataset. More specifically, the musiXmatch dataset is a subset of the MSD that has resolved lyrics for over 77% of the MSD, with 237,662 tracks available in the dataset. Another dataset available is the music4all dataset [11]. This dataset is about half the size of the musiXmatch dataset (109,269 songs) and contains metadata which will help when filtering for language. It can be accessed due to a research agreement with my supervisor Varvara.

When choosing a suitable dataset, we have to consider that more songs means more training time, which could hinder the project's progress. On the other hand, training more songs will result in much more accurate word vectors for the lyrics of the query/test songs. The musiXmatch dataset does contain more songs, but the music4all dataset contains more useful metadata, such as the song's language, genre, tags, name, and artist which will make it easier to filter the songs in the application.

## 2.5 Stemming Methods

Stemming is a part of the text pre-processing stage which, like the vector representations we have seen, has many algorithms that can accomplish this. Stemming aims to remove affixes from words to turn them into a 'root' form to reduce the total number of tokens. For context, the other typical pre-processing steps are:

1. Tokenization - Splitting a text up into 'tokens' (usually words to start with).
2. Capitalisation - Turning every token into upper or lower case.
3. Stop word removal - Removing unwanted words from the text e.g. 'and', 'the'.

In pre-processing, lemmatization can also be performed either before stemming or as an alternative to it. It is similar to stemming in that it reduces a word to its root form. However lemmatization is more analytical, and won't simply just chop off the affix of a word. Unlike the previous, more clear-cut steps, stemming has different methods that we will discuss in this section.

Jivani et al [12] compare a number of stemming algorithms, just like the Text Classification Algorithms survey paper. The paper groups stemmers into three categories: truncating (affix removal), statistical, and inflectional & derivational methods. The stemmer that is said to produce the best output is the Porter stemmer which uses a rule-based approach to remove affixes from words. However, the Porter stemmer is said to be time consuming due to it having many rules and steps.

Statistical methods, such as N-Gram, HMM (Hidden Markov Model) and YASS (Yet Another Suffix Stripper), have a significant advantage of being language independent at a cost for being quite computationally intensive and time inefficient. Seen as though I am only looking at English songs for this project, the use of these stemmers would provide no benefit over using, say, a Porter stemmer.

Inflectional & derivational stemmers require the creation of a lexicon, and thus will not be able to cope with words outside of said lexicon. Overall, it would be best to look at the truncating methods, especially Porter and perhaps its successor Snowball - which is considered to be an improved version of the Porter stemmer.

## 2.6 Libraries

Before I program the project, I will need to use libraries to help with the more complex parts of the project like pre-processing and word vectorization. There are some libraries which are definitely going to be needed:

- NumPy [13], a library for vector and matrix manipulation.
- Flask [14], the framework for which the web application will be built on.
- Pandas [15], a library which contains tools for data analysis and manipulation.
- BeautifulSoup [16], a library which can obtain information from HTML and XML files. It will be needed to obtain lyrics from songs that are not in the dataset.

Regarding word vectorization and pre-processing, there are many libraries which can accomplish this. One of the more commonly used libraries for text processing is the "Natural Language Toolkit" (NLTK) [17], which contains the tools needed to tokenize, lemmatize and stem text. As for generating word vectors, I can use a library called "Gensim" [18] which contains models for LDA and Word2Vec, and all the tools necessary to train and even update them. Unlike Word2Vec and LDA, GloVe has seemingly no corresponding Python library, which would mean either looking for an alternative library or training GloVe manually using the source code provided on the official website [4].

In terms of possible GloVe implementations/libraries, a Python library called "mittens" contains fast implementations of both GloVe and a GloVe variant called Mittens. the Mittens algorithm itself [19] is an extension of GloVe that is more tailored towards domain-specific applications, which could be good for the project. Unfortunately, the library has not been updated since 2019, and can only handle a vocabulary (set of unique words) size of ∼20,000 tokens.

## 2.7 Dimensionality Reduction Methods

In order to avoid extremely large vectors in the computation, it would be good to consider methods that can help reduce the number of dimensions of each vector. Despite its age, the most popular dimensionality reduction technique is "Principal Component Analysis" (PCA) [20]. PCA attempts to define a linear transform $A$ that maps the input vector $x$ to an output vector $y$ that is much smaller in dimension. There are 3 steps to learning the linear transform matrix $A$:

1. Compute the covariance matrix (variance between two classes) for the dataset.
2. Computer the eigenvectors $v_1, ..., v_n$ of the covariance matrix ordered by eigenvalue (largest first).
3. Take the first M eigenvectors and form the matrix $A = (v_1, ..., v_m)$

Then you can reduce an $N$-dimensional vector $x$ to an $M$-dimensional vector $y$ simply by computing $y = Ax$.

Looking at another approach involving a linear transform, there is also "Linear Discriminant Analysis" (LDA). It is an extension and generalisation of "Fisher's Discriminant Analysis" [21] that can be applied to two or more classes. LDA takes into account the classes of the data, unlike PCA, and thus is a supervised approach. Since the dataset has no explicit 'classes' this approach does not seem feasible.

It could be possible to reduce the number of vectors/tokens in the vocabulary and save time by using a stop-list: a pre-defined list of words I would like to exclude from the list of tokens. It is very common to exclude words such as 'the', 'a', 'an', 'and', etc. as they do not really provide much meaning at all and may bloat some metrics due to the large term frequency. In the project, it may be a good idea to use an optional stop-list and see how different the results are. It could even be possible to generate a list of stop-words by using Latent Dirichlet Allocation, as LDA has a quirk in which it will create a topic group containing stop-words.

## 2.8 Performance Metrics

As well as containing information about vector representation methods, Kowsari et al [2] also discuss information about some simple and commonly used performance metrics used in text processing work such as precision, recall, and F1/F$\beta$ score. These scores make use of four numbers:

- True Positives ($TP$): Number of items returned by the system that turn out to be correct.
- False Positives ($FP$): Number of items returned by the system that turn out to be incorrect.
- True Negatives ($TN$): Number of items not returned that turn out to be incorrect.
- False Negatives ($FN$): Number of items not returned that turn out to be correct.

The aforementioned metrics can then be calculated as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.5}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.6}$$

$$\mathrm{F}\beta = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{2.7}$$

(Note that F1 just means $\beta = 1$).

Using these scores should provide an easy and automatic way to assess the performance of the system. However, the recommendations themselves will be subjective, and there may not be an easy way of deciding whether a song recommendation is a "true positive" or a "false negative". The alternative solution would be to conduct a human survey and ask subjects to rank the recommendations the system provides, which is discussed further in chapter 3.

## 2.9 Related Work

This project considers and takes inspiration from the work done by many researchers in this specific area of research. In this subsection we take a look at 3 papers related to the research area linking music and Natural Language Processing.

"Lyrics-based Music Recommendation" by Gossi and Gunes [22] outlines a system that is similar to what I intend to develop, however one big difference is that they try to keep recommendations within a certain genre or "tag category" and focus more on the correct classification of the song, whereas this project will be completely insensitive to the genre of the song. What is interesting here is that they do not use any feature extraction methods on the lyrics like Word2Vec or GloVe. Instead they simply just use the term frequency matrices - perhaps the performance of their algorithm could have been enhanced with the use of these vector representation methods? Nonetheless, their methods were suitable for the classification task. They also used the Snowball (Porter2) stemming method, which seems to have performed quite well.

Another paper linking to the research area is "Lyrics or Audio for Music Recommendation?" by Vystrčilová and Peška [23]. This paper focuses more on comparing lyrics and audio recommendation methods against each other, but I will be focusing on the methods they used for lyric-based recommendations.

In the paper they used 3 text based approaches:

- A TF-IDF Matrix, with PCA applied to it
- Word2Vec
- BERT

For [23], Word2Vec was found to be the most successful lyrics-based method. Which provides me with a potential candidate for vector representation algorithms to use in my project. To automatically evaluate the system, they first performed a 5-fold Monte Carlo Cross Validation with a cosine similarity to construct the list of recommended songs. They then calculated

the recall at the top-10 songs and top-50 songs, and for the top-100 songs they performed a "Normalized Discounted Cumulative Gain" for the top-100 (instead of recall). Here the use of recall is successful, as the cross-validation uses the whole set of data to both test and train on, meaning that the algorithm has an idea of what is considered a "good recommendation".

The third and final paper relating to the research area is "A method of music autotagging based on audio and lyrics" by Wang et al [24]. This one deviates slightly from my objective as it involves using lyrics and audio to automatically generate tags for song, but it was worth analysing to see which text methods they used.

To vectorize the lyrics, they suggested methods such as Word2Vec and GloVe. Afterwards they suggested feeding the word vectors into either a Recurrent or Convoluted Neural Network, and then the classification can be performed.

During the model training, they used a pre-trained model based on Wikipedia pages, a self-trained model based on lyrics, and also a re-trained model trained on lyrics on top of the pre-trained model. In the end, the pre-trained model performed the worst, while the self-trained and retrained model performed similarly. When it comes to programming the project, perhaps it would be best to save time training Word2Vec and GloVe by simply just training it only the lyrics only.

These papers provide a great foundation towards the project, the only drawback however is that the papers lack the practical application. The papers have shown that the concept of lyrics-based recommendations work and that current NLP techniques are indeed well suited to it. However, they lack the human and user aspect of it: "Do people find lyrics important for music recommendations?", "Would we be able to apply these techniques on a web application?" this is the step forward that this project intends to take. All this being said, the papers provide more insight into the feature extraction methods that are commonly used for this task, such as Word2Vec, GloVe, TF-IDF, and BERT.

## 2.10 Summary

In this section, we have identified suitable algorithms to generate vector representations of words and or documents. We have also discussed suitable stemming algorithms, possible performance metrics to evaluate the system, the datasets available for this project, the libraries that will be needed, and work done by others in this research area and evaluating the methods that they used. In terms of suitable vector representation algorithms, we have seen a wide variety of them - each one varying in their methods, complexity and strengths. The related work I have seen has provided a good idea of which algorithms are commonly used, but most importantly: the fact that lyrics-based music recommendation systems can work.

# Chapter 3

# Requirements

The requirements for the application can be split up using a standard "MoSCoW" prioritization technique, a method that splits the requirements into 4 categories: **M**usts, **S**houlds, **C**oulds, and **W**on'ts. The 'Won'ts' are omitted, since there are no requirements that fall into this category and it would seem trivial to discuss them.

## 3.1 Must Haves

Here are the absolute must haves for the project.

1. A GUI that allows for an easy and comfortable interaction with the system. This is particularly important for the evaluation of the application through the survey.
2. A suitably sized dataset with songs and their lyrics, names, artists, and genres.
3. A defined pre-processing algorithm. This could include things like capitalisation and tokenization.
4. At least 2 vector representation algorithms that can be trained on the dataset. 2 is a minimum because it means we can compare them and see which performs the best.
5. An algorithm that generates the word vectors for each song.
6. A way to compare each song vector and provide the recommendations (i.e. most similar) to the user.
7. A user survey that will be used to evaluate the performance of the vector representation algorithms and the application as a whole.

## 3.2 Should Haves

During or after the implementation of the must haves, there are a few requirements that will be worth considering. The "should haves" are requirements that will likely aid the performance of the application, but it is not detrimental if they are omitted.

1. A clustering method to reduce the number of comparisons between song vectors.
2. A dimensionality reduction algorithm that reduces the size of (possibly) large vectors.
3. A "self-updating" application that both improves the vector representation algorithms after each query, and also adds unseen songs into the dataset/collection so that they can be recommended later.
4. A weighting for the word vectors, in order to make the vector a bit more accurate.

## 3.3 Could Haves

These requirements are extra and are not particularly important.

1. An extra vector representation algorithm, perhaps if a library has multiple. If it is not too strenuous then an extra set of results would be good.
2. Quality of life adjustments to the application e.g., a loading screen or aesthetic adjustments.

## 3.4 Professional, Legal, and Ethical Concerns

The main ethical concern for this project will be the survey, as I will be collecting data from real people. An ethics application will have to be filled out before I can start collecting any data, and it is important that relevant data protection laws are followed.

When using the Python libraries, it is important that I adhere to the developer terms. All libraries for this project have a permissive license which allows me to use them, and thanks to Varvara I am able to use the music4all dataset which would have required me to obtain permission otherwise.

# Chapter 4

# Design and Implementation

In this section I will cover the design of the project as it was at the end, and how the requirements have changed along the course of development and why. I have dubbed the application "*Project Euterpe*", named after the ancient Greek muse of lyric poetry. Initially it was meant to be a tentative title, but I stuck with it in the end.

Let us start this section by looking at how the data was trained first, before moving on to how everything works together on the front end. In the source code, a file called `manager.py` was created to act as a hub to perform the training section described below.

## 4.1 Training Stage

This section covers how the vector representation algorithms were trained and how the song vectors were generated. As well as a written explanation, there are also figures E.1, E.2, and E.3 in appendix E that help visualise this process.

### 4.1.1 The Dataset

The dataset used for this project in the end was the "music4all" dataset, kindly lent to me by my supervisor Varvara. As discussed in chapter 2, the music4all dataset contains $109,269$ songs, but only $84,103$ of them ($\sim 77\%$) are in English and are also not instrumental songs. From these $84,103$ songs, there are a total of $102,424,244$ words. When the lyrics are fully pre-processed, we are left with $9,542,412$ tokens. The pre-processing algorithm is discussed in more detail in the next section.

If a queried song is not present in the dataset, then we must 'scrape' the HTML of the lyrics from Genius.com using the BeautifulSoup library. The recommendations provided will always come from the dataset, though.

### 4.1.2 Pre-Processing

Before we can extract any songs from the dataset, we need a suitable pre-processing algorithm. The full pre-processing algorithm (`pre_processor.py`) performs the following:

1. Tokenize the lyrics using NLTK's `RegexpTokenizer`.
2. Convert each token to lowercase and remove stop words at the same time.
3. Lemmatize the tokens using NLTK's `WordNetLemmatizer`.
4. Stem the tokens using NLTK's `PorterStemmer`.

The pre-processing steps can be better explained using an example from Stephen King's "The Gunslinger" [25]:

> *"The man in black fled across the desert, and the gunslinger followed."*

**After tokenization**:

['The', 'man', 'in', 'black', 'fled', 'across', 'the', 'desert', 'and', 'the', 'gunslinger', 'followed']

**After capitalisation (to lower case)**:

['the', 'man', 'in', 'black', 'fled', 'across', 'the', 'desert', 'and', 'the', 'gunslinger', 'followed']

**After stop word removal**:

['man', 'black', 'fled', 'across', 'desert', 'gunslinger', 'followed'].

**After lemmatization**

['man', 'black', 'flee', 'across', 'desert', 'gunslinger', 'follow']

**After stemming**

['man', 'black', 'flee', 'across', 'desert', 'gunsling', 'follow']

Now the text is ready to be fed into the vector algorithms for training.

Stemming algorithms add the additional affix removal for words such as 'gunslinger' or for other words like 'lazy' which it turns into 'lazi'. This may seem trivial, but it simply just adds an extra layer of normalization to the text. It also doesn't matter to the vector algorithms anyway, so long as all occurrences of the words are pre-processed the same way. The stemmer used for the project is the Porter 1 stemmer, included in the NLTK library.

The regex pattern used in the end for tokenization was `\[[^\[\]]*\]|\b\w+\b`, this also removes any square brackets that are used on a Genius lyrics web page to indicate a verse, chorus, pre-chorus, etc. A regex substitution is also needed between each line of the Genius lyrics, as each new line causes tokens to join together. This is just defined as `([a-z])([A-Z])` and then a simple substitution `\1 \2` is performed.

The stop word list was one that I made myself, and it contains all the common stop words such as 'and', 'the', 'a', etc. It also contains ad-libs such as 'yeah', 'uh', 'ah', 'hey', and so on.

The full pre-processor function is multi-functional. It can pre-process both Genius lyrics (that use square brackets) as well as lyrics in the dataset (that do not use square brackets).

### 4.1.3 Extracting the Songs from the Dataset

Now that we have a lyrics pre-processor, we can now begin to extract songs from the dataset using the `data_handler.py` file.

The music4all dataset is split up into 6 CSV files:

1. A genres file.
2. An information file. (artist, song, album name)
3. A language file.
4. A metadata file. (Spotify metadata values such as 'danceability', 'energy', 'tempo', etc.)
5. A tags file. (User generated tag data from last.fm[1])
6. A lyrics file.

For the project we only need the information, language, and lyrics file.

Then a pandas dataframe was created from each of these files. For the lyrics dataframe, I made sure that any instrumental songs (marked 'INSTRUMENTAL') were omitted. I also made sure that from the languages dataframe, only the songs written in English were kept in the dataframe.

We also need to define our own class called `Song`, so we can easily access the song's information. The Song class contains the following information from the dataset:

- ID from the Dataset - String
- Song name - String
- Artist - String
- Album - String
- Lyrics - String
- Main genre - String
- Sub-genre(s) - String array

[1] `https://www.last.fm/`

- Pre-processed song lyrics - String array
- Vector representation (explained in 4.1.5) - NumPy Array

A song is listed with many genres in the CSV file, so we assume the main genre is the first item in that list, and the sub-genres are the rest of the items.

We now loop through the language dataframe and fill in all of the properties except for the vector, which was initially left blank during development to do at another time once the vector generation algorithm had been fully implemented. For safety, we save the list of Song objects to `songs_no_vec.pk`, so that we can import this and use it for both the vector algorithm training and song vector calculation later without having to pre-process the lyrics again.

### 4.1.4 Training the Vector Algorithms

To train the vector algorithms, a separate file called `algorithms_wrappers.py` was created to act as an interface or wrapper between the libraries and my own code.

Each vector representation algorithm requires a list of 'documents' (the songs), and each document is split up into a list of tokens (like in the pre-processing example). We can use the intermediate `songs_no_vec.pk` file to loop through and take all of the pre-processed lyrics into a training data list. For good measure, we also include the normal unprocessed song lyrics, but with tokenization, capitalisation, and stop word removal applied to them.

For GloVe, a .txt corpus file is needed. This file must contain each song on one line as one string, which we can easily do by performing a Python `join()` on the list of tokens of each song and writing it to each line of the file. Then, we can simply run the bash script given in the GloVe source code with some varying parameters (i.e. epochs, vector size). The vocab files and co-occurrence files are made for us too. The `vocab.txt` file tells us that the size of the vocabulary of the training data is $49,805$ words or tokens. The bash script generates a .txt file of the word vectors called `vectors.txt` which we make a copy of (called `glove_vectors.txt`). We can then parse these vectors when we initialise the wrapper.

Due to the GloVe bash script using Unix commands (i.e. zip), I had to perform training separately to the others using MSYS2[2] - a software that provides a Unix-like environment and command line.

For Word2Vec, we turn to the Gensim library model. The model allows the use of both skip-gram and CBOW versions, and for the project I used the CBOW model. The Gensim model simply requires the big list of songs that we made earlier, which we can supply to the model, along with other varying parameters such as the vector size and epochs. Once trained, the model will be saved and then we can simply just tell the wrapper to load the model and vectors when the application starts up, instead of training the entire algorithm again. When

[2] `https://www.msys2.org/`

the model is used in the application, we initialise it without supplying the training data as a parameter in order to skip the training stage and just load the word vectors.

For LDA we use another Gensim model. The wrapper I made requires the training data to be passed in as a parameter, as it must be turned into a dictionary for both training *and* querying. Like with Word2Vec, the model is trained and then saved so that it can be loaded.

### 4.1.5 Generating Vector Representations for the Songs

Now we can go back to `data_handler.py` with our list of vector-less songs. After some trial and error that is discussed further in chapter 5, the Word2Vec and GloVe vectors were calculated using the following method:

Iterate over the song's vocabulary (set of unique words). For each word/token:

1. Take the word vector from the model. If the word vector cannot be found, then pass on to the next word.
2. Multiply the word vector by its Smooth Inverse Frequency weight. We apply the Smooth Inverse Frequency in order to give higher weights to words that do not occur as often in the song, and lower weights to words that occur very often.

Then take all of these vectors and put them into a list, and take the vector mean of this list to produce a single vector. Finally, divide this mean vector by the size of the vocabulary set. Previously in chapter 2, we mentioned that each point in an embedding encapsulates some semantic or syntactic information about the word. So the idea is that by taking an average of the word vectors in the song, those same points should tell us something about the whole song.

In chapter 5 I provide the empirical evidence to justify this method, as well as show the other equations used previously before settling on the final one. That being said, our song vector is quite similar to the one initially described in [9], the only difference being the division by the song's vocabulary length again in my calculation, and the fact that I have chosen to take the term frequency from just the song rather than the whole dataset - as it makes sense for the weight to be representative of the word in just the song.

The description above can be expressed in an equation:

$$\vec{v_s} = \frac{1}{|V| \times N} \sum_{i=0}^{N} \begin{cases} \vec{v_{w_i}} \times \dfrac{a}{a + \text{count}(w_i, s)} & \text{if } w_i \text{ in model vocabulary} \\ \text{pass} & \text{otherwise} \end{cases} \tag{4.1}$$

Here, $N$ is the number of vectors in the resulting list. For songs in the dataset (where encountering a word outside of the vocabulary of the two models is not possible), $N$ is the same as $|V|$, and the equation can now be expressed as:

$$\vec{v_s} = \frac{1}{|V|^2} \sum_{i=0}^{|V|} \begin{cases} \vec{v_{w_i}} \times \dfrac{a}{a + \text{count}(w_i, s)} & \text{if } w_i \text{ in model vocabulary} \\ \text{pass} & \text{otherwise} \end{cases} \tag{4.2}$$

Where:

- $\vec{v_s}$ is the vector representing the whole song $s$, where $s$ has been pre-processed.
- $\vec{v_{w_i}}$ is the word vector for the word $w$ at index $i$ of the vocabulary set.
- "count()" is a function that counts the number of occurrences of $w_i$ in $s$.
- $|V|$ is the size of the vocabulary set.
- $a$ is a changeable parameter of Smooth Inverse Frequency. $a$ was set to 2.

Equation 4.2 will hold true for songs in the dataset, but if a word outside the vocabulary of GloVe or Word2Vec is discovered (i.e. when taking a song from Genius), then the algorithm is told to pass onto the next word since a vector cannot be found for it. This means that the number of vectors in our list (before taking the mean) ($N$) will not be equal to $|V|$ and so what is actually performed is equation 4.1. To cover both bases, we will say that the algorithm is best represented by 4.1.

After calculating the song vector for GloVe and Word2Vec, the LDA topic distribution is calculated on a document level using the Gensim library. Unlike our other two methods, the entire vocabulary set of a song is passed into a function to generate the topic distribution vector, rather than calculating a vector token-by-token. Initially, I believed that the querying of a whole document would prove to be advantageous, but it was quite the opposite, as it meant as I was unable to tweak the calculation of the vector like I did with Word2Vec and GloVe.

After looping through all the songs in `songs_no_vec.pk` and assigning a dictionary containing the 3 different vector representations to the (previously null) vector property, we can now save the complete list to the final file: `songs_vec.pk`. Now that the training stage is complete, we can move on to the design of the application itself.

## 4.2 The Application

This section discusses and explains the implementation of the front end, and how the recommendations are provided with the song vectors. Like section 4.1, figure E.4 helps to visualise the process.

### 4.2.1 The Front End

In `app.py`, the whole project begins to come together in the front end. The application consists of two pages: 1. a home page with some plenary and the user input form, 2. the

results page where recommendations are shown. To create the web application, I used the Flask library as it is very easy to implement for a small and basic web application. As for the style of the program, I used just plain and simple CSS, as the focus of the application is more about the recommendations given, rather than how 'slick' the application looks or how 'fancy' it feels.

When the application starts up, all the song objects are imported from the `songs_vec.pk` file, and the three vector method wrappers are also initialised. GloVe is given the path to its `glove_vectors.txt` file as a parameter, LDA is given all the training data as a matrix of tokens of each song, and Word2Vec needs no parameters as all the loading of the model is done within the wrapper. For the survey, the user is given a random vector representation method, but for the final version they are allowed to choose.

A user can now search for a song. They have the choice of either inputting a song name and an artist name, or entering a Genius URL. If the queried song exists in the dataset, we will simply just take the dictionary of vectors calculated in the training stage. If not, then the application will first need to form a valid Genius URL out of the song name and artist, or just take the URL if supplied. A Genius URL is of the form:

`https://genius.com/<Artist>-<name>-<song>-<name>-lyrics`

which was easily accomplished by some string concatenation and manipulation. Once we have our URL, we simply scrape the lyrics from the HTML, pre-process them, and generate the three vector representations before we can provide the recommendations.

### 4.2.2 Providing Recommendations

The front end `app.py` communicates with another file called `recommender.py`. When the submit button is pressed, we take all the song vectors of the selected (or randomly assigned) model, and compare them all to the query. The similarity metric used in the end was the Euclidean distance metric, however cosine similarity was also considered and tested (results for this are shown in chapter 5). The algorithm will loop through all the songs, and for each song calculate the Euclidean distance between it and the query song's vector. It will then sort the list in ascending order with respect to distance and return the top 5. These top 5 songs are then stored in a session variable and are displayed on the next page with song name, artist, genre, and the distance (or difference) between them.

### 4.2.3 Deploying to the Cloud

For the ease of the participant survey, I uploaded the project to the cloud, in order to avoid confusing participants with a lengthy guide on how to run the application using the source code. The application first had to be containerized using Docker, then it had to be built before uploading to the Google Cloud. The cloud service used for the project was Google Cloud Run, and was straightforward to set up and upload later revisions.

## 4.3 Performance Evaluation Survey

The main method for evaluating the system was a participant survey that was divided into 6 sections:

- **Section 0**: Participant and demographic information i.e. preferred gender, age range, first language.
- **Section 1**: Using the application with a specified song, this being *Happy* by Pharrell Williams.
- **Section 2**: Using the application with a song of the participant's choice. Each participant was required to do this once, but they were able to repeat this section another 4 times for a total of 5 possible entries.
- **Section 3**: The participant's listening habits and music tastes.
- **Section 4**: Overall thoughts about the application, and the concept of lyrics-based music recommendation.
- **Section 5**: An optional section to provide any further comments or feedback.

The participants were able to access the application through a web link in the Form[3].

In sections 1 & 2, the questions were more or less the same. A participant would be asked to choose 1 out of the 5 recommendations, and answer the following questions:

- Had they heard of this song before?
- Did they like it?
- Did they think it was similar to the query song in terms of:
  - Meaning, topics or lyrical content?
  - Genre?
  - Style, energy, or vibe?
- Would they add this to a playlist?
- (Optionally) Could they give a better recommendation?

The last question was implemented in order to make a small start on a ground truth for lyrics-based music recommendation. The idea is that we use the "wisdom of the crowd" to come up with recommendations that the system should be coming up with when testing in the future. With enough people, you could be able to bypass the subjectivity of lyrics-based recommendations when multiple people recommend the same song.

[3]Accessible here: `https://project-euterpe-jrsoouufda-nw.a.run.app`

The results of the survey will be discussed in chapter 6. You can also find all the questions that were asked, the ethics application, the project information sheet, and the consent form in appendices A to D.

## 4.4 Changes from Requirements

In the final implementation, there were some requirements from the "should haves" that were attempted but unfortunately were not included in the final version of the application. The first of which is the implementation of a dimensionality reduction method. The implementation of PCA was very easy through the use of scikit-learn, and was successful when reducing the size of all the training song vectors. During testing I found that the speed of calculations was still sufficient even with large vector sizes such as 250, and so I didn't feel it was necessary. Perhaps when doing further work I can use PCA on vectors that are size, say, 750.

A self-updating application was also mentioned in the requirements. However, there are a few issues that prevented me from implementing this. 1) Since I am using the GloVe source code, there is no easy way to 'update' the word vectors like Gensim did for Word2Vec and LDA, and it did not seem fair to only update 2 of the 3 vector methods. 2) The second part of a self-updating application would be to add unseen songs into the pickle file / dataset. This would require metadata such as genre in order to be effective, and Genius cannot really provide any metadata beyond the song name, album name, artist name, and lyrics.

The last requirement that was not included was a clustering algorithm. The K-Means algorithm from scikit-learn was implemented at first, however I found that it was actually quicker to just perform the comparisons over *all* the vectors in the dataset, likely due to the inefficiency of the K-Means clustering algorithm.

# Chapter 5

# Testing

## 5.1 Vector Algorithm Configurations

Over the course of development, many different 'configurations' or variants of the song vector generation algorithm and the recommendation algorithm were tested. These altered different properties of the vector generation algorithm such as:

1. The vector sizes.
2. The distance metric.
3. The training data.
4. How many epochs the algorithms were trained for.
5. How the Word2Vec and GloVe vectors were calculated.

In the end, I settled on a final configuration that is as follows:

### 5.1.1 Word2Vec and GloVe

- Vector size: **250**.
- Distance metric: **Euclidean**.
- Training data: **Fully pre-processed lyrics as well as just the tokenized lyrics**.
- Epochs: **50**.
- Vector calculation: **See equation 4.1**.

The vector sizes initially started at around 50, then 100, and then finally settling on 250. Even with this large vector size, the performance of the distance calculations is still of a sufficient speed. By this point, I was satisfied with the recommendations I was being given, and so kept it at 250.

### 5.1.2 LDA

- Number of topics (distribution vector size): **50**.
- Distance metric: **Euclidean**.
- Training data: **Fully pre-processed lyrics as well as just the tokenized lyrics**.
- Epochs: **50**.
- Vector calculation: **Performed by the Gensim library on the document level.**

For LDA, the vector size had to remain at 50 due to extremely low topic probabilities when testing the vector size at 100. Gensim allows for a `min_probability` parameter that will ignore any probability in the distribution lower than it. This results in different sized probability distribution vectors that will cause errors upon performing similarity metrics because the vector sizes do not match. I found that for a size 100 vector, the probabilities were sometimes so incredibly small that even a `min_probability` of around $1 \times 10^{-10}$ would not make them show up. Nonetheless, 50 is still a suitable number of topics for LDA to cluster as shown in its paper [7]. The general rule is that as the number of topics increases, the 'perplexity' (a measure of uncertainty of a value in our distribution) decreases.

## 5.2 Manual Precision Tests

For some of the configurations I performed some manual precision tests to make sure that the results were satisfactory before conducting the survey. Recall from chapter 2 the formula for precision:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.1}$$

The "True Positives" (i.e. relevant songs) were determined by my own judgement. When testing the application I used these 3 songs:

1. **Happy** by **Pharrell Williams**, a song about being happy and keeping your head up despite the bad things that may inconvenience us[1].
2. **Plastic** by **Loyle Carner**, a song about materialism and superficial affluence.
3. **The Wonder of You** by **Elvis Presley**, a simple love song expressing pure admiration for a partner, and how they cheer us up in times of sadness.

I stuck with these 3 songs in order to be able to compare the different configurations much more easily. Now we introduce the 6 different configurations *A*-*F*. These configurations do not differ in their vector sizes (so $n_{W2V} = 250$; $n_{GloVe} = 250$; $n_{LDA} = 50$ across all configurations). Where they differ is either in the similarity metric, or in the case of Word2Vec and GloVe: in their song vector calculation.

[1] Also used in the survey

In the very early stages of testing, I merely used the top recommendation (i.e. the most similar) as a metric to gauge performance. However I quickly realised that this was a very naive way to measure performance, as songs that may have placed 4th or 5th may actually be relevant, and the 1st one may not be. From then on, I performed these quick precision tests.

### 5.2.1 Configuration *A*

The similarity metric used in configuration $A$ is the cosine similarity, and the songs vectors involving GloVe and Word2Vec word vectors are calculated using this formula:

$$\vec{v_s} = \frac{1}{N}\sum_{i=0}^{N} \vec{v_{w_i}} \tag{5.2}$$

This is a much simpler and earlier version of 4.1 and we will see that we add more parts to it in the later configurations. The LDA vector itself is unchanged throughout all the configurations, what changes for LDA is just the similarity metric. We can see the results for this configuration in table 5.1.

| Song | Vector Method | Number of Relevant Documents Retrieved | Precision |
|---|---|---|---|
| Happy | Word2Vec | 3 out of 5 | 60% |
| Happy | GloVe | 0 out of 5 | 0% |
| Happy | LDA | 0 out of 5 | 0% |
| Plastic | Word2Vec | 1 out of 5 | 20% |
| Plastic | GloVe | 1 out of 5 | 20% |
| Plastic | LDA | 3 out of 5 | 60% |
| The Wonder of You | Word2Vec | 1 out of 5 | 20% |
| The Wonder of You | GloVe | 3 out of 5 | 60% |
| The Wonder of You | LDA | 1 out of 5 | 20% |

Table 5.1: Precision results for all vector methods using configuration $A$

### 5.2.2 Configuration *B*

Configuration $B$ builds off of $A$ and replaces the cosine similarity metric with a Euclidean distance metric instead. Everything else is unchanged. By virtue of all three vector representation methods, the distances between each point in the song vector will match up for our Euclidean distance, since each point in their vector will correspond to the same encoded semantic information (in the case of Word2Vec and GloVe), or their topic probability (in the case of LDA). We can see the results for it below in table 5.2

A noticeable improvement, but we can continue to tweak the algorithm even further.

| Song | Vector Method | Number of Relevant Documents Retrieved | Precision |
|---|---|---|---|
| Happy | Word2Vec | 4 out of 5 | 80% |
| Happy | GloVe | 1 out of 5 | 20% |
| Happy | LDA | 0 out of 5 | 0% |
| Plastic | Word2Vec | 3 out of 5 | 60% |
| Plastic | GloVe | 1 out of 5 | 20% |
| Plastic | LDA | 3 out of 5 | 60% |
| The Wonder of You | Word2Vec | 2 out of 5 | 40% |
| The Wonder of You | GloVe | 2 out of 5 | 40% |
| The Wonder of You | LDA | 3 out of 5 | 60% |

Table 5.2: Precision results for all vector methods using configuration $B$

### 5.2.3 Configuration *C*

Configuration $C$ builds off of $B$ and introduces the Smooth Inverse Frequency into the GloVe and Word2Vec vector calculations:

$$\vec{v_s} = \frac{1}{N}\sum_{i=0}^{N} \vec{v_{w_i}} \times \frac{a}{a + \text{count}(w_i, s)} \tag{5.3}$$

where $a = 2$. This was implemented after testing on specific songs, I found that for *Happy* by Pharrell Williams, the song *pete davidson* by Ariana Grande was being recommended. Both songs contain the repeated use of the word 'happy' and I found the recommendation to be quite dissimilar. The results are shown below in table 5.3.

| Song | Vector Method | Number of Relevant Documents Retrieved | Precision |
|---|---|---|---|
| Happy | Word2Vec | 3 out of 5 | 60% |
| Happy | GloVe | 2 out of 5 | 40% |
| Happy | LDA | 1 out of 5 | 20% |
| Plastic | Word2Vec | 2 out of 5 | 40% |
| Plastic | GloVe | 1 out of 5 | 20% |
| Plastic | LDA | 3 out of 5 | 60% |
| The Wonder of You | Word2Vec | 1 out of 5 | 20% |
| The Wonder of You | GloVe | 3 out of 5 | 60% |
| The Wonder of You | LDA | 3 out of 5 | 60% |

Table 5.3: Precision results for all vector methods using configuration $C$

### 5.2.4 Configuration *D*

For comparative reasons, we define a configuration $D$ which replaces the Euclidean distance metric in $C$ with cosine similarity. The results are shown below in table 5.4.

| Song | Vector Method | Number of Relevant Documents Retrieved | Precision |
|---|---|---|---|
| Happy | Word2Vec | 1 out of 5 | 20% |
| Happy | GloVe | 1 out of 5 | 20% |
| Happy | LDA | 0 out of 5 | 0% |
| Plastic | Word2Vec | 1 out of 5 | 20% |
| Plastic | GloVe | 0 out of 5 | 0% |
| Plastic | LDA | 3 out of 5 | 60% |
| The Wonder of You | Word2Vec | 1 out of 5 | 20% |
| The Wonder of You | GloVe | 4 out of 5 | 80% |
| The Wonder of You | LDA | 1 out of 5 | 20% |

Table 5.4: Precision results for all vector methods using configuration $D$

### 5.2.5 Configuration *E*

This configuration builds off of $C$ and introduces the division by the size of the vocabulary after the mean has been calculated (i.e. equation 4.1). This was implemented in order to try and mitigate the fact that some of the longer songs that had a larger vocabulary were sneaking their way into recommendations when they shouldn't have been. The results are shown below in table 5.5. This feature was implemented because I noticed that songs that were quite short

| Song | Vector Method | Number of Relevant Documents Retrieved | Precision |
|---|---|---|---|
| Happy | Word2Vec | 3 out of 5 | 60% |
| Happy | GloVe | 1 out of 5 | 20% |
| Happy | LDA | 1 out of 5 | 20% |
| Plastic | Word2Vec | 3 out of 5 | 60% |
| Plastic | GloVe | 2 out of 5 | 40% |
| Plastic | LDA | 3 out of 5 | 60% |
| The Wonder of You | Word2Vec | 3 out of 5 | 60% |
| The Wonder of You | GloVe | 2 out of 5 | 40% |
| The Wonder of You | LDA | 3 out of 5 | 60% |

Table 5.5: Precision results for all vector methods using configuration $E$

were given bad recommendations that happened to be very long songs with lots of lyrics. I tried to mitigate this by dividing the song vector by the length of the vocabulary, and I believe it has provided better results.

### 5.2.6 Configuration *F*

Again, for comparative reasons, configuration $F$ is the exact same as $E$ but uses the cosine similarity instead of Euclidean distance. The results are shown below in table 5.6.

### 5.2.7 Findings

As we can see, each configuration tends to improve upon the other, with Euclidean performing just slightly better than cosine for this task. Some precision values are improved by the additions, and some are worsened, making the choice of configuration slightly tricky. In the

| Song | Vector Method | Number of Relevant Documents Retrieved | Precision |
|---|---|---|---|
| Happy | Word2Vec | 1 out of 5 | 20% |
| Happy | GloVe | 1 out of 5 | 20% |
| Happy | LDA | 1 out of 5 | 20% |
| Plastic | Word2Vec | 1 out of 5 | 20% |
| Plastic | GloVe | 0 out of 5 | 0% |
| Plastic | LDA | 3 out of 5 | 60% |
| The Wonder of You | Word2Vec | 1 out of 5 | 20% |
| The Wonder of You | GloVe | 4 out of 5 | 80% |
| The Wonder of You | LDA | 3 out of 5 | 60% |

Table 5.6: Precision results for all vector methods using configuration $F$

end, it was configuration $E$ that was chosen for the final evaluation through the survey, as the precision values it provided were a little bit more consistent than the others, and represented the cumulative improvement of the configurations that it built off.

These tests provide a good baseline for the performance of the system, and we can see that our final chosen configuration $E$ performs well - we could even infer that a user is guaranteed *at least one* relevant song, something we could confirm with more time and tests. However the relevancy of the songs is subject to opinion, and if somebody else were to do the test, these results may have been different.

## 5.3 Front End Tests

From a software engineering point of view, it would be a good idea to include some manual front end tests. The possible user stories for the application are few and simple, so I performed these tests manually. The results are shown in table 5.7.

| Test Case | Result | Notes |
|---|---|---|
| User can search for a song using a name and an artist. | Success | |
| User can search for a song using the Genius URL. | Success | |
| User can search for a song, view the results, and go back to the main page using the "Go back" button. | Success | Vector method changes upon going back. |
| User can search for a song, view the results, and go back to the main page by pressing the browser back button. | Success | Vector method changes upon going back. |
| User can search for a song, view the results, return to the main page, then re-return to the results page where the previous query's results are displayed. | Success | |
| User can search for a song, view the results, refresh the results page, and the recommendations will persist. | Success | |
| User enters an unknown song, is alerted, and encouraged to try again. | Success | |
| A user attempts to try an XSS attack (using a script tag with alert('xss')), script not executed | Success | The default alert pops up, assume success |
| User may have added an extra whitespace at the end, but query should still be sanitised and processed normally. | Failure | Eventually fixed but survey version was updated later than as hoped. |

Table 5.7: Manual System Tests

# Chapter 6

# Results and Discussion

## 6.1 Survey Results

Before we discuss the results of the survey, I would first like to justify the use of a survey as a replacement for automatic evaluation. Normally, in any machine learning project, we would use a development set in order to evaluate the results of the classifier. More specifically, in information retrieval systems, they would use a set containing relevant documents for each query. For the case of *Project Euterpe*, no such "ground truth" exists, therefore automatic evaluation of the project is not possible unless I was to create one myself, which would be very time-consuming. Even if I did create one, the relevant documents would still be subject to criticism, and are not as binary as a regular information retrieval development set. Hence we turn to manual user evaluation through a survey, which allows us to obtain not only feedback on the application, but on this research area as a whole.

To take an example from the survey: two users had chosen the same recommendation from the list to analyse. This song was *Fault Line* by Meeka Kates. One user had said the song was "completely different" lyrically to *Happy* by Pharrell Williams, whilst the other had said it was "slightly similar". This case here provides more evidence for the use of the survey, as it highlights the subjectivity of the system.

### 6.1.1 General Statistics and Demographics

The survey saw 21 participants take part. Each participant was required to do the sections as described in section 4.3 of chapter 4. Section 2 (user's choice of song) had to be done at least once, but there were some participants who did more than one entry:

- 5 participants stopped after one entry.
- 9 participants stopped after two entries.
- 4 participants stopped after three entries.

- 3 participants did all five entries.

Given that I allowed the participants to be able to repeat entries, we actually have 50 results for section 2.

Figures 6.1, 6.2, and 6.3 show the visualisation of the demographic data collected.

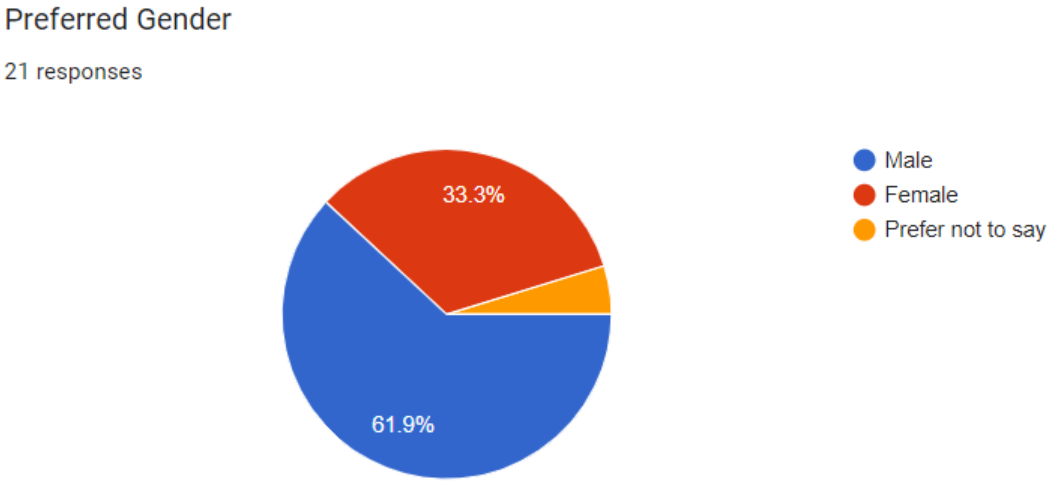


Figure 6.1: Pie chart showing the preferred gender of participants.

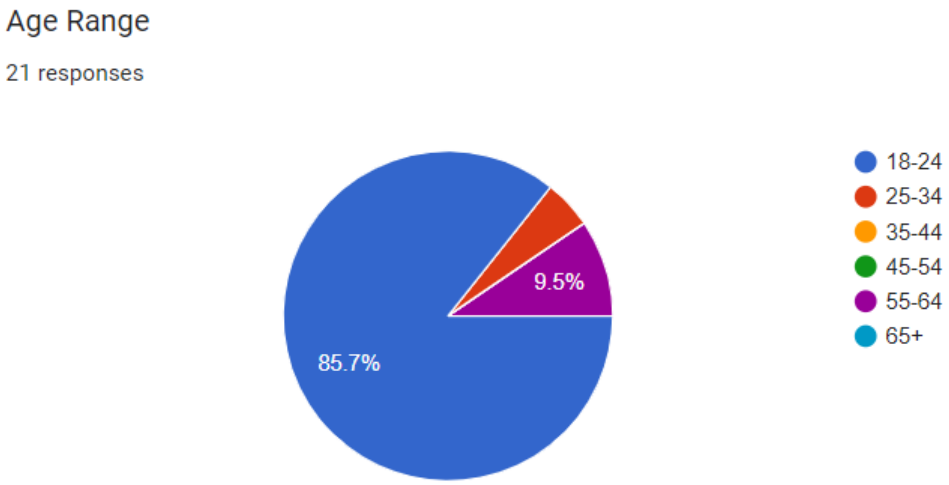


Figure 6.2: Pie chart showing the age range of participants.

### 6.1.2 Section 1: The Participant Uses A Fixed Song

The first section asked the participants to enter *Happy* by *Pharrell Williams* into the application and subsequently pick one of the recommendations given. The purpose of this section was to use the participant's feedback to gauge the general performance of the system, and see whether the results returned were similar in terms of lyrical content, vibe, and genre.

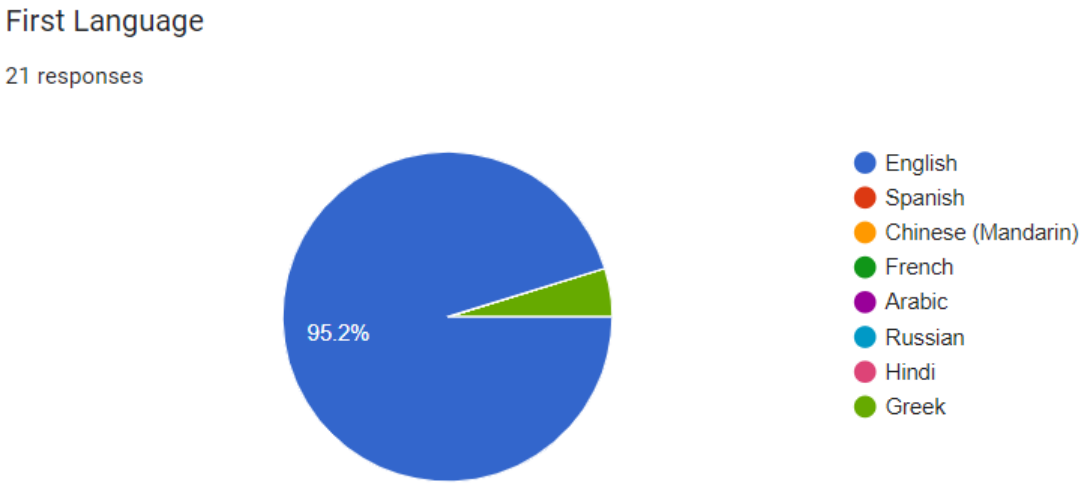


Figure 6.3: Pie chart showing the first language of the participants.

Questions relating to how much they liked the recommendation were also asked, but we will focus on the former set of questions.

The results for this section were generally quite disappointing, around 86% of the participants (18) thought that their chosen recommendation was "slightly different" or worse. Similarly, around 76% (16) of participants thought that their chosen recommendation had a different vibe or style to *Happy*, and around 62% (13) of participants thought that the genre of the recommended song was different.

Figures 6.4, 6.5, and 6.6 show the full visualisation of the distributions of similarity, style, and genre respectively.

Despite this performance, the results line up with the similarly dissatisfying precision results from chapter 5. I had chosen *Happy* to be the song for this section because there is no ambiguity behind the lyrics, and I believed that every participant would correctly identify the meaning and themes of the song (in which all did) and would thus produce more consistent results over a song like *Hotel California* which is known for having multiple interpretations, for example. Perhaps the simplistic and repetitive lyrics found in *Happy* led to these results?

It is also worth noting that the distribution of vector representation methods assigned to the participants was not even. For the purposes of the survey, the application assigns a different and random vector representation for each query. This was favoured over giving the user a choice as many of the participants do not have a background in computer science, and I felt giving them a choice would be intimidating or they would've just chosen the first one in the list. I had hoped that the vector representation methods would have been somewhat evenly distributed across the participants, but the application has a massive skew towards LDA. In section 1, 11 results were done using LDA, 7 of them with GloVe, and only 3 of them with Word2Vec. If I were to repeat this survey again, perhaps it would have been wise to make 3

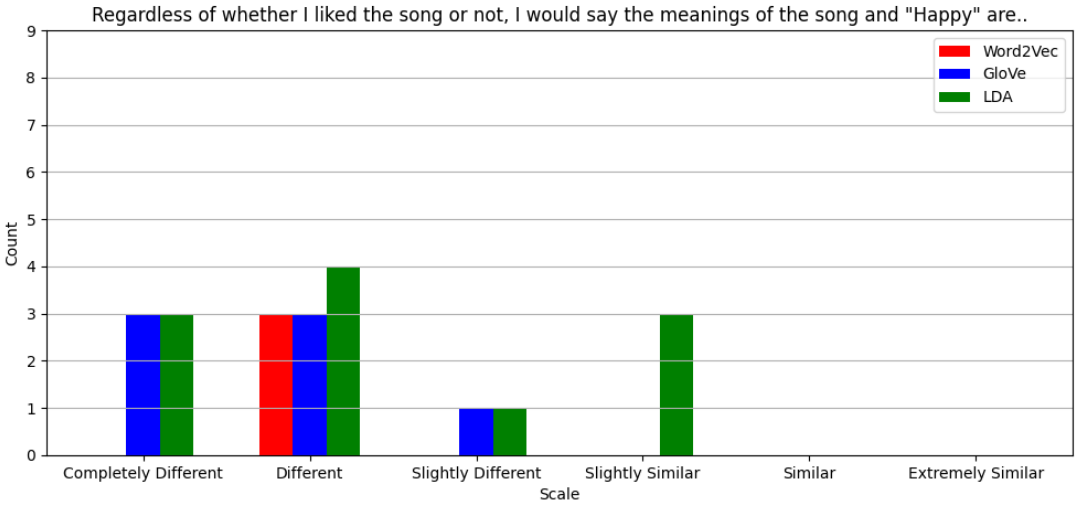


Figure 6.4: Bar chart showing how similar in terms of meaning participants thought *Happy* was compared to their recommendation

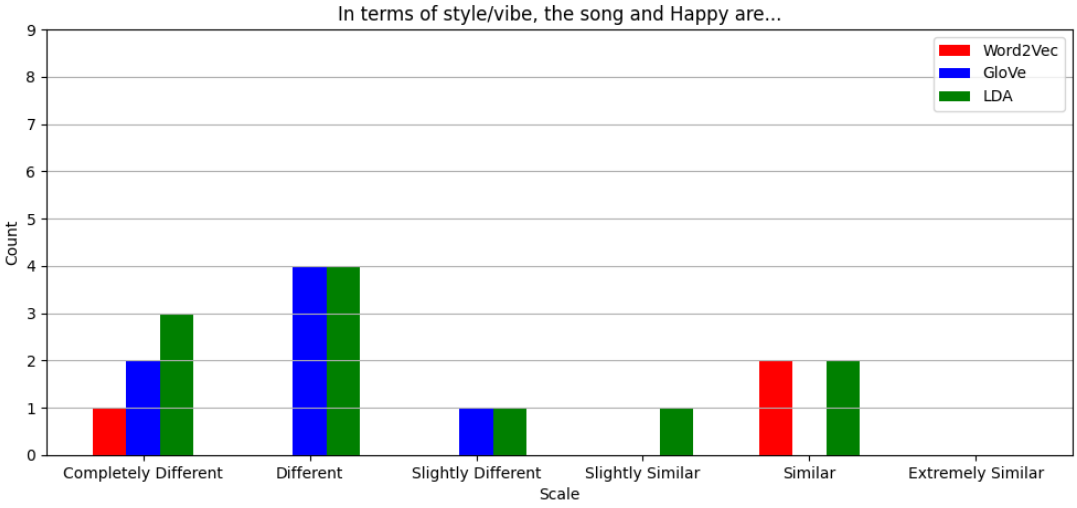


Figure 6.5: Bar chart showing how similar in terms of song style participants thought *Happy* was compared to their recommendation

different copies of the survey for each vector method, and have the participants use the same vector representation throughout the survey, that way I could fabricate an even split for the vector representation methods.

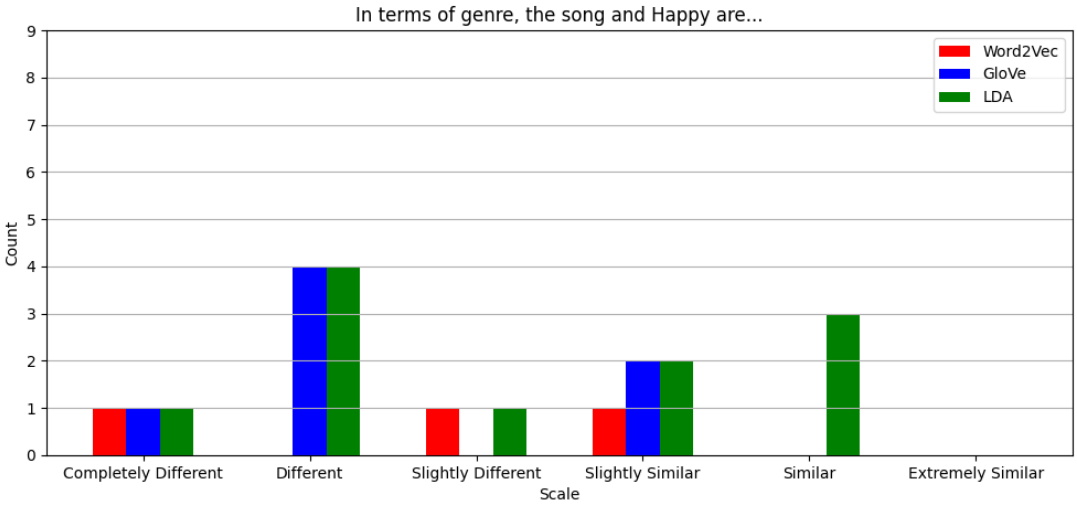


Figure 6.6: Bar chart showing how similar in terms of genre participants thought *Happy* was compared to their recommendation

### 6.1.3 Section 2: The Participant Chooses Their Own Song

Section 2 focuses on the participant's own queries, and we will look at whether the participants actually liked the recommendations they were being given - as well as how similar they thought it was. Here, the vector representation methods are a bit more evenly distributed, and the skew has been weakened due to the repeat entries.

Here the results begin to look a lot better. When asked about whether they liked their recommended song, around 70% (35) of the results were "somewhat agree" or better (shown in figure 6.7). If we were to go one step further in terms of enjoyment and look at whether they would add a song to their playlist, we can see that the results are a bit more polar. There were more people that would strongly disagree to adding the song to their playlist than there were people strongly agreeing, but as before the majority of results (29, or 58%) were "somewhat agree" or better for this question (shown in figure 6.8). Using the like-dislike distribution to assess the performance of the application is a good foundation, but the real testament to performance should be whether they would add a recommendation to a playlist, as it means the participant has (most likely) decided to now actively incorporate the recommendation into their daily listening habits.

The similarity distribution was much more evenly spread, 54% (27) of the entries said that their recommended song was "slightly similar" or better (shown in figure 6.9). However these results come regardless of whether the participant had enjoyed the song or not, so out of the results where the participant had said the recommendation was "slightly similar" or better, 22 were "somewhat agree" or better when asked if they liked the song that they had been given/chosen (shown in figure 6.10). Out of those that had both found the recommendation

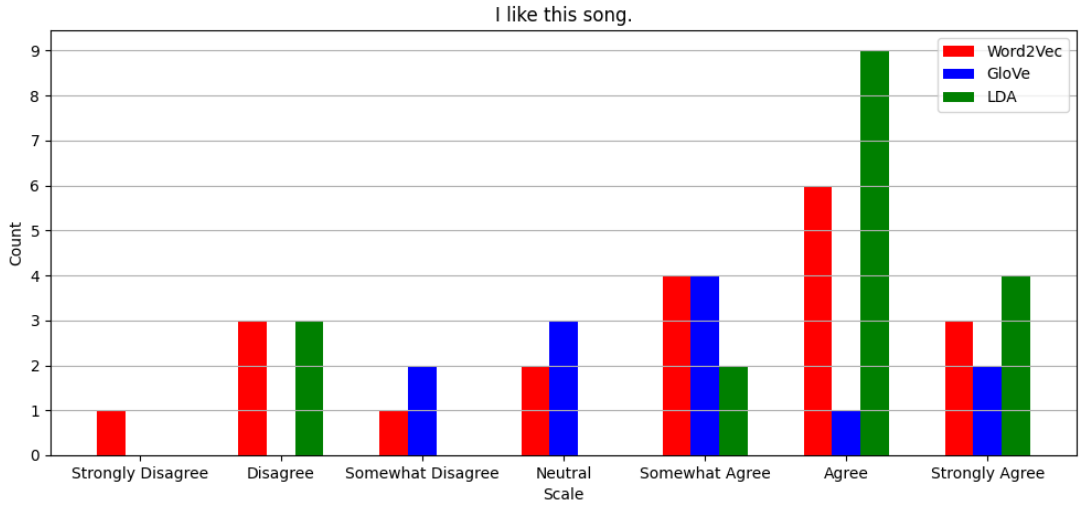


Figure 6.7: Bar chart showing whether the participants liked the recommendation when querying their own song.

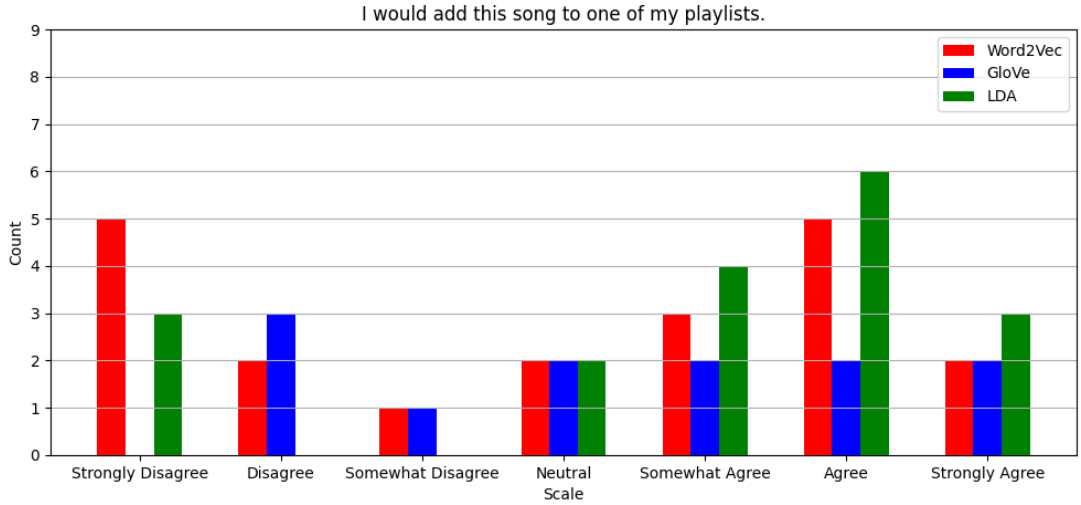


Figure 6.8: Bar chart showing whether the participants would add the recommendation to a playlist when querying their own song.

similar *and* liked it, 100% (14) said that they would add it to a playlist (shown in figure 6.11). Therefore we can infer that there were a total of 14 (out of 50) successful recommendations given by the application. 14 may seem like a low number of successful recommendations, but realistically even the traditional recommendation methods are not 100% guaranteed to give "successful recommendations". We have also defined quite strict criteria for a successful recommendation.

This section of the survey also displays a sudden increase in positive LDA results. During testing, LDA seemed to be the one that under performed quite often. Now we see that LDA has formed the majority of the "strongly agree" responses for figure 6.11.

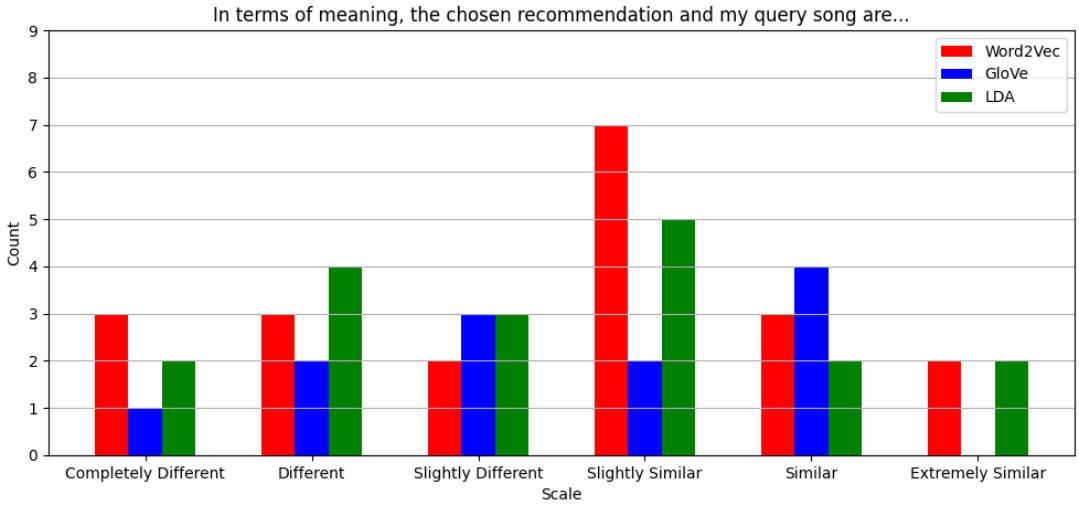


Figure 6.9: Bar chart showing whether the participants thought that the recommendation was similar to their query song.

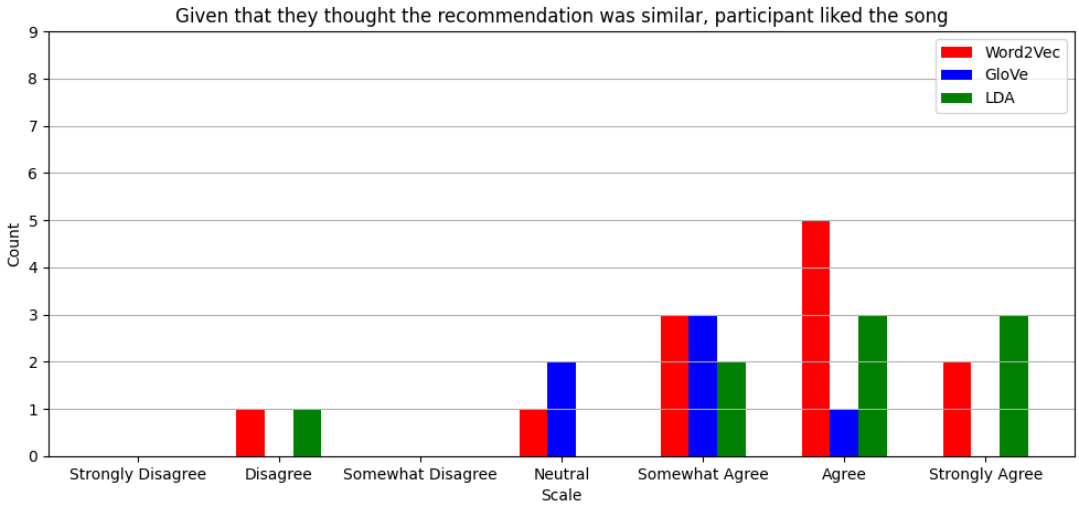


Figure 6.10: Bar chart showing whether a participant liked a song, given that it was similar.

### 6.1.4 Section 3: Participants' Music Tastes and Listening Habits

The third section asked the participants about their listening habits and music tastes. The listening habits of the participants varied from rock to rap to even avant-jazz. 100% of the

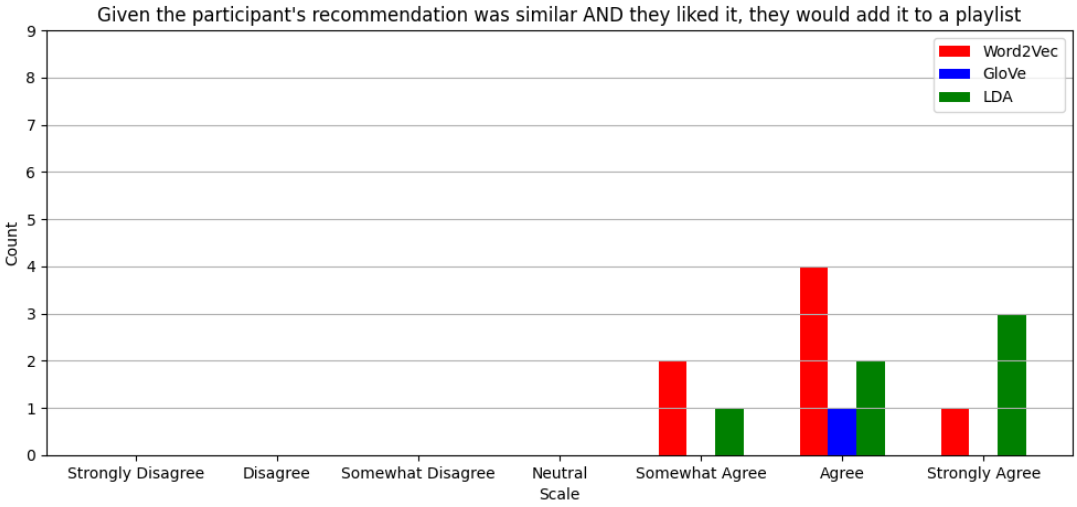


Figure 6.11: Bar chart showing whether the participants would add the recommendation to the playlist, given that they thought the song was similar and they liked it as well. This is what we shall consider a "successful recommendation".

participants used Spotify to stream their music, but some also used other applications and media such as YouTube (10 participants). Figure 6.12 shows the distribution of streaming platforms across the participants.

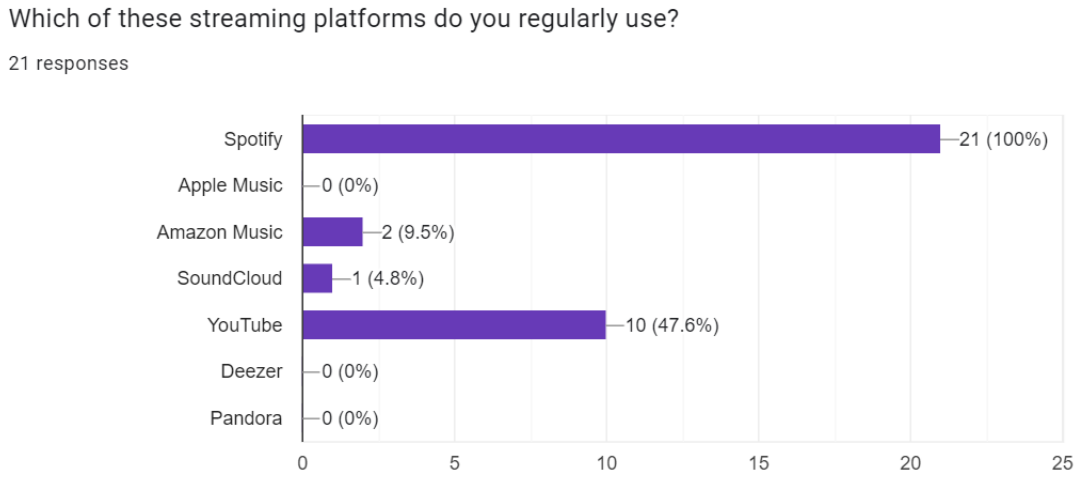


Figure 6.12: Bar chart showing which streaming platforms the participants used regularly.

Participants were also asked about which media they used to listen to music. All participants use some form of streaming service, but others appear to be vinyl and even cassette collectors. This data is shown in figure 6.13

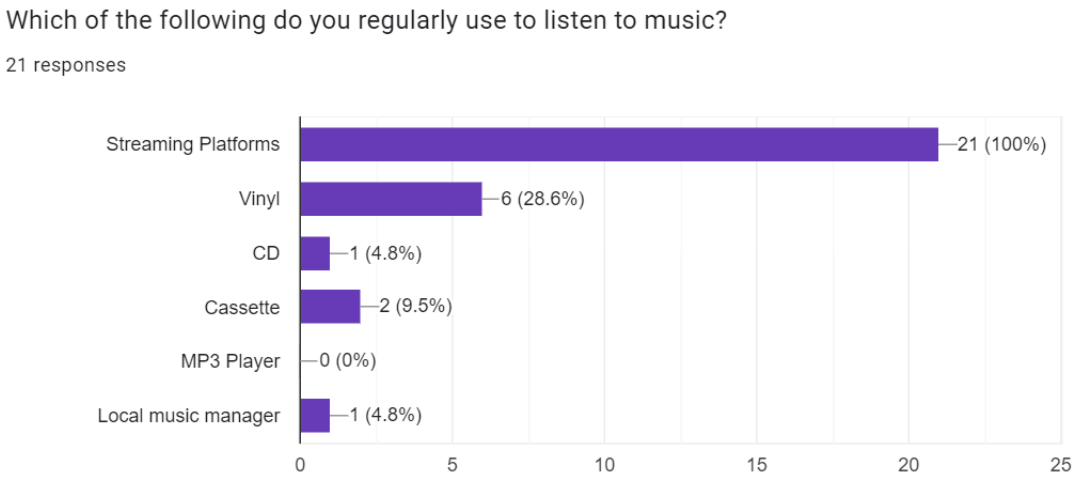


Figure 6.13: Bar chart showing what media the participants use to listen to music.

### 6.1.5 Section 4: Participants' Overall Thoughts

The final section asked the participants about their thoughts on the application and the concept of lyrics-based music recommendation as a whole.

The overall feedback received about the application was very positive. Around 81% of participants answered "slightly agree" or better when asked if they thought the application worked well. Similarly, around 71% of participants would agree that they would use the application again to get more song recommendations. Figures 6.14 and 6.15 show the proportion of the aforementioned statistics respectively.

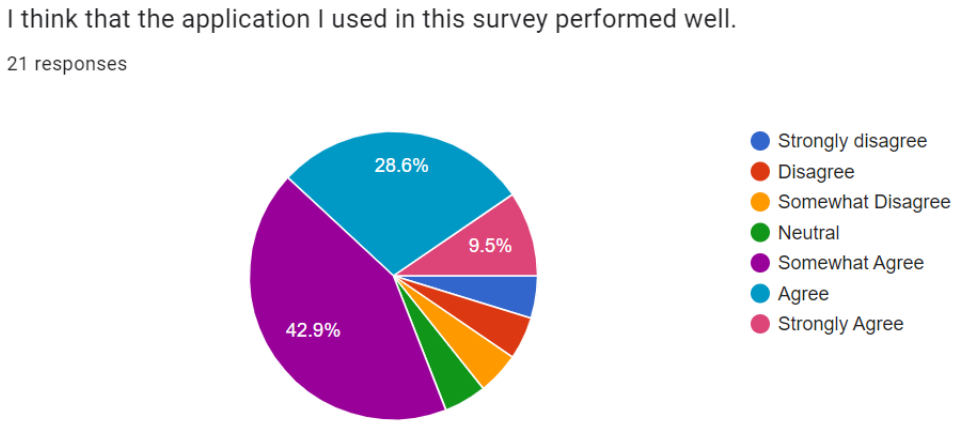


Figure 6.14: Pie chart showing how much participants agreed that the application performed well.

As for lyrics-based song recommendations as a whole, not a single participant seemed disin-

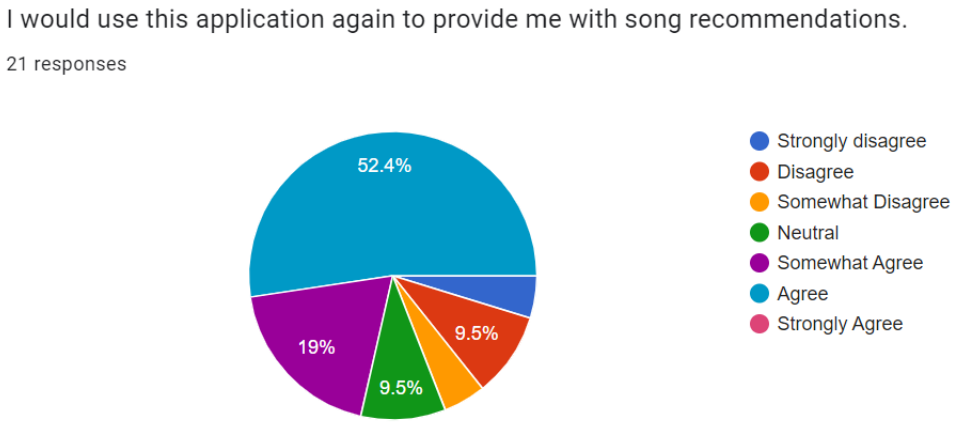


Figure 6.15: Pie chart showing how much participants agreed that they would use the application again.

terested in the concept, and the majority ($\sim 81\%$) also thought it would be a good idea for major streaming platforms to incorporate similar lyrics into their current recommendation methods. This being said, 66.7% of participants do not find lyrics to be all too important when they are looking for music recommendations, suggesting that it could be best to incorporate the project with other recommendation techniques rather than have a standalone algorithm. Figures 6.16, 6.17, and 6.18 show the pie charts for the previous three points that have been discussed, respectively.

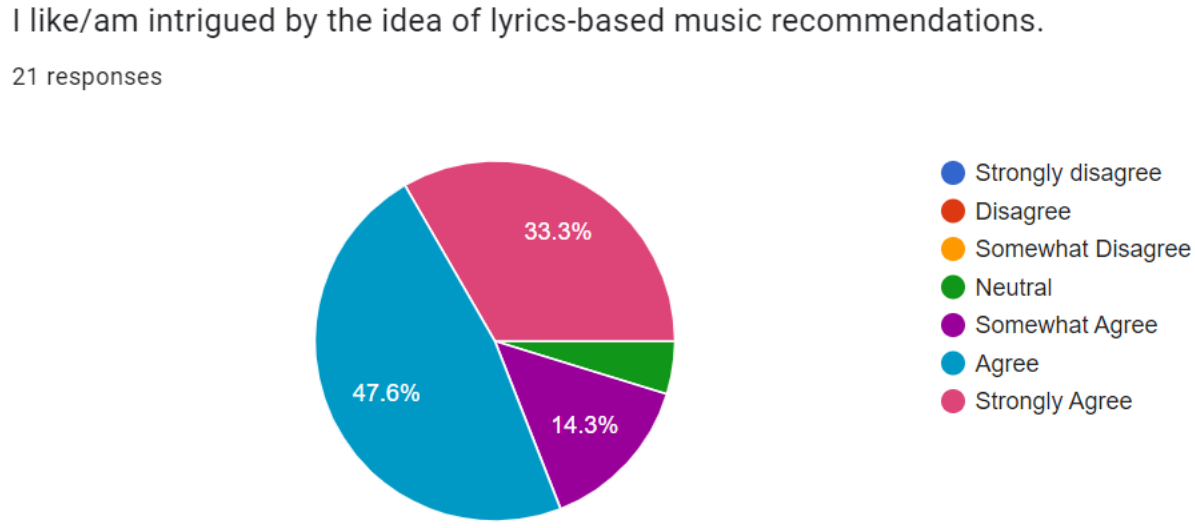


Figure 6.16: Pie chart showing the participants' interest in lyrics-based music recommendation.

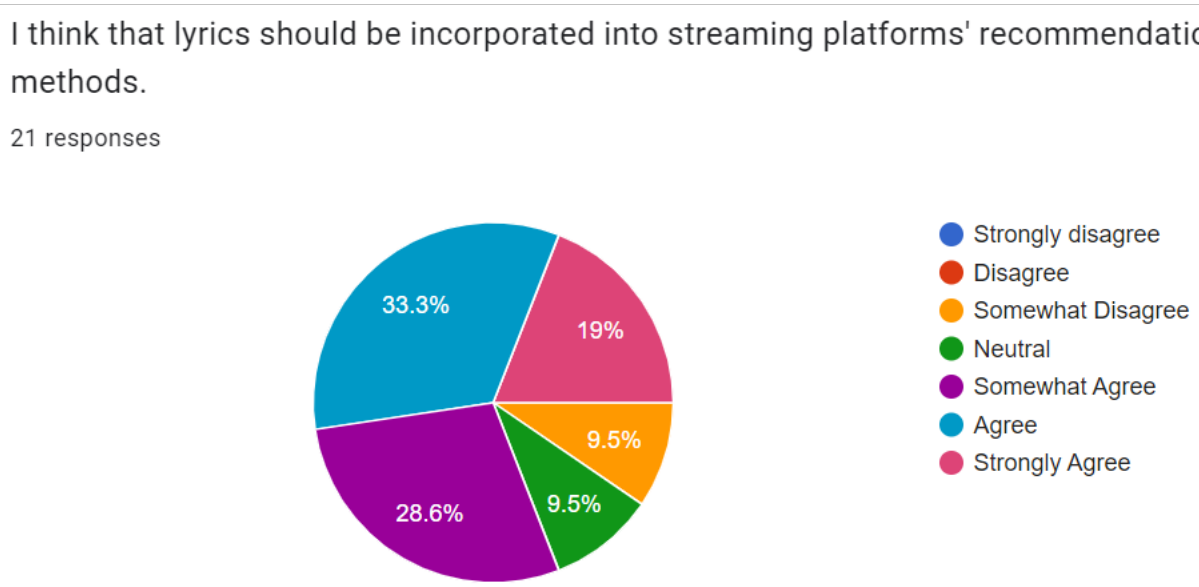


Figure 6.17: Pie chart showing how much the participants agree that this way of recommending songs should be implemented into traditional methods.

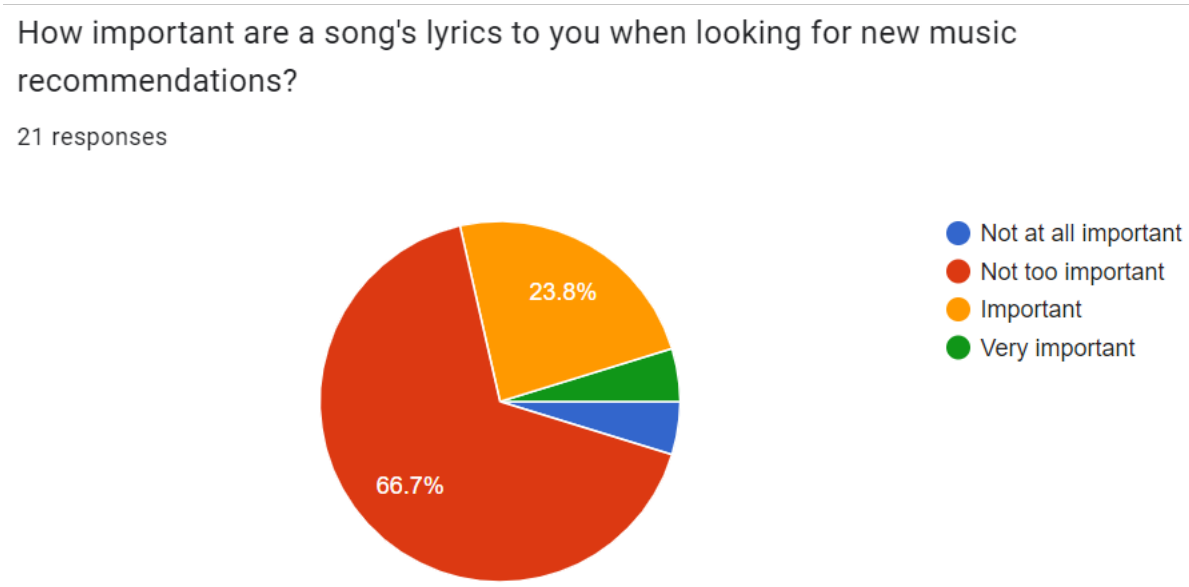


Figure 6.18: Pie chart showing how important lyrics are to the participants when searching for recommendations.

### 6.1.6 Further Discussion

We can see that the application is unfortunately quite inconsistent, which is shown clearly both when we look at the results between section 1 and section 2, and even section 2 on its

own. *Happy* had provided consistently negative results, but the reason that the algorithm fails to provide recommendations for such a basic song is up for speculation. One potential reason could be because the lyrics tend to repeat themselves. Let's have a look at a bad recommendation from section 2: *SUCKAPUNCH* by You Me At Six is a song about "hitting rock bottom and coming back from it" (according to the participant). The song does contain the repeated use of the phrase "I rise from the wreckage you left behind" in the refrain, but this phrase is repeated nowhere near as many times as the word "happy" is in *Happy*. The recommendation chosen was *Tell Her About It* by Billy Joel, a song about "encouraging the subject to express their emotions to their love interest" (according to the participant). This song tends to repeat phrases that are some variation of "tell her about it". Two very different songs with not much lyrical similarity, but maybe the short repeated phrases may be affecting the final song vector.

Example 2: *Ocean View* by Easy Life is a song about "enjoying life and relationships". The recommendation chosen was *27* by Fall Out Boy, a song about "enjoying life (but a bit more recklessly)". Two lyrically similar songs, but not in terms of genre. These songs do of course contain a repeated chorus, but these choruses are longer and more varied than the short repeated phrases we saw in the previous example. So perhaps it is the very short phrases that are causing an issue. From chapter 5, I had hoped that the introduction of Smooth Inverse Frequency would help to balance out any issues relating to overly-repeated words in the song, and it would seem that this needs adjusting.

Another pattern we can look for in the application is: Does the algorithm inherently recommend songs of similar genre? Example 2 previously would suggest no, but there were cases in which the genre of the query and the recommendation were similar. For example, a participant had searched for *In The End* by Linkin Park and chose *Better Off This Way* by A Day to Remember. The participant had said that the two meanings were "extremely similar" *and* that the genres were "similar". So we can say that being recommended a song with similar lyrics does not always lead to the songs being the same genre. This is not necessarily a bad thing, as the application was written with no consideration for genre in order to attempt to broaden the music tastes of the listener, otherwise our recommendation algorithm would be not that much different than the usual content-based methods. Writing the algorithm with no regard for genre (or any other musical property) on purpose also allows us to see if there are any interesting properties or patterns like what we have discussed in this paragraph.

Finally, we must discuss which vector representation method performed the best. The results were much more evenly split than I had thought, and there is no stand-out algorithm. One good measure of performance for a vector representation would be the proportion of chosen songs that were ranked "slightly similar" or better. Table 6.1.6 shows the proportions of similar songs, as well as the split between each section.

Going off of this table alone, we can see that Word2Vec has statistically performed the best. This also lines up with our precision results in table 5.5, in which Word2Vec consistently gave

| | | Number of similar songs | | |
|---|---|---|---|---|
| **Vector Method** | **No. Results** | **Section 1** | **Section 2** | **Percentage** |
| Word2Vec | 23 | 0 | 12 | 52.1% |
| GloVe | 19 | 0 | 6 | 31.6% |
| LDA | 29 | 3 | 9 | 41.4% |

Table 6.1: Table showing the proportion of similar songs for each vector representation method.

us a precision of 60% for each song. If we go by proportion of "successful recommendations" we can see that Word2Vec is responsible for 7 out of the 14 successful recommendations, with LDA coming in a close second with 6.

### 6.1.7 Limitations

The survey is not without its limitations, and is not a perfect way of evaluating the application. To start, the participants are subject to the precision of the application, since they are asked to pick 1 song out of the 5 given to them. This perhaps has affected our results in some way, as there may have been multiple instances where the participant had selected an irrelevant song, but there may have been relevant songs within the results. This could have affected their overall opinion on the application as expressed in section 4. We can perhaps propose an alternate form of the survey, where the participants are asked to briefly look over the meaning of all the songs, and ask how many of them were similar. This would give us some more precision results, but it would also mean we won't be able to ask the more specific questions about the song i.e. would they add it to a playlist, etc. without it becoming too tedious. We could also ask them to listen to and look at the lyrics thoroughly for all 5 songs, but this would be more time-consuming and ultimately drive away potential participants.

As mentioned previously, we were also affected by the random vector representation algorithm. There was an inexplicable bias towards the application providing participants with LDA in section 1, and has affected the results for that section. Since the majority of participants were completely unfamiliar with the 3 algorithms, I thought that this was a necessary implementation. As described in section 6.1.2, there could have been 3 different surveys, each telling the user to select a specific vector representation method for the whole duration of the survey. This would lead to a more even spread of results for each vector representation method.

## 6.2 Goals Achieved

Let us now return to the aim (and sub-aims) of the project, as stated in chapter 1:

- Develop and evaluate a lyrics-based music recommendation application. We will use

this application to:

1. Evaluate the vector representation methods used to provide the recommendations.
2. Conduct a survey in which participants evaluate the application and answer questions about lyrics-based music recommendations.
3. Evaluate the effectiveness and practicality of lyrics-based music recommendations.

We have certainly met all the aims here. We have developed a functioning web application that is not too computationally intensive and runs quite well on both a local device and the Google Cloud, showing that lyrics-based recommendations can be done practically. The web application has the capability to provide users with song recommendations that reflect the lyrical content of a song they search for. Although, it seems that more work can be done on the recommendation algorithm/vector generation in order to make it more accurate, but I believe that the system at its current stage is quite good (albeit inconsistent) at what it is meant to do.

We have evaluated the performance of 3 vector representation algorithms, of which Word2Vec has performed the best. This being said, from my view I would say that all 3 algorithms performed more or less equally as well.

We have also conducted a successful survey, which has allowed us to evaluate the performance of the system as well as gain insight into people's thoughts on lyrics-based song recommendations. We have found that although most people did not consider lyrics to be too important when looking for recommendations, they were still interested by it, and seemed keen to have it implemented along with traditional methods.

## 6.3 Further Work

The system definitely has room for improvement, the most obvious improvement we can make is increasing the number of epochs on the training algorithms. This will undeniably increase the performance by a noticeable but not greatly considerable amount. As well as increasing epochs, we could increase the vector sizes to large amounts, and then perform a PCA transform on the song vectors.

Now that we have decided the best performing vector method, we can narrow down the number of vector representations to just Word2Vec and develop a more bespoke vector generation algorithm. This also allows us to maybe implement the self-updating application if we discard GloVe.

Instead of a SIF weighting to the Word2Vec and GloVe word vectors, we could also perhaps use TF-IDF weightings instead and compare the results to see if it is an improvement. This weighting would require the construction of the TF-IDF matrix, which may be a disadvantage compared to the swiftness of SIF.

Since no ground truth exists as mentioned in section 6.1, it is possible that a social media-esque 'likes' system that can be implemented into the application in order to build one. When a user queries their song, they could have the ability to simply say which recommendation was the best, or perhaps leave a like for each recommendation they thought was similar. We can then generate a ground truth for a song using the number of likes on its recommendations. The main problem however is that it will require a large amount of visitors to the website in order for even one song to have a sufficient amount of likes.

We do actually have a (incredibly minuscule) start on a ground truth. In the survey, participants were asked to provide what they thought would have been a good recommendation instead of the one they were given. This could contribute towards the start of one, but the generation of a ground truth like this would require thousands of participants.

# Chapter 7

# Conclusions

## 7.1 Immediate Criticisms

Over the course of development, there were a few flaws that I thought I may be criticised for. I would like to justify some of these.

At a glance, one may comment that I have perhaps over-relied on libraries such as Gensim, scikit-learn, or NLTK. Yes, this project has used a great deal of external code libraries as well as the GloVe source code, and has formed a backbone for the project. However, the use of libraries has given me the opportunity to compare and contrast these different vector representation methods, and if I were to implement them from scratch I may not have had time to implement all of them to compare. There were also certain roadblocks and limitations faced when tailoring the algorithms to the project that have made the integration of these libraries less straightforward than one might think, such as having to run GloVe separately in MSYS2.

The other regret I had is that I should have probably increased the number of epochs when training the data. Increasing the epochs will undoubtedly result in better vectors, and is something that can easily be done as described in section 6.3. Similarly for the vector sizes, where I could've increased them and then applied some dimensionality reduction technique. As suggestions for the future, they sound promising, yet I think for the survey edition this could have also been done. In the end I was worried about time, and chose to sacrifice these many features to get the survey out earlier and collect more results.

## 7.2 Summary and Final Words

In this report, we have discovered the lyrics-shaped gap in existing recommendation technologies that can be filled using current NLP techniques. We have explored the ways we can efficiently represent the meaning of a song, as well as the other components necessary for

this project to work. We have discussed how all this research came together in the design of the project, and how this design changed over the course of development through testing. In our results, we discover that our application performs quite well, which is an encouraging sign for the future of both the project and lyrics-based music recommendations as a whole. We have shown, once again, that Word2Vec remains one of the best methods for NLP tasks such as this one. We have also introduced the use of LDA into this research area, as this method has not been seen in the related work. The potential improvement for this project is boundless, and I do also regret not implementing some of the ideas discussed in 6.3 into the current version of the project. This being said, I am pleased with the project's performance.

This project marks an important chapter during my studies at the University of Sheffield, it has allowed me to use concepts that were taught to me as well as allowed me to further my knowledge in text processing beyond that of what was taught to me. My hope for this project is that in the future, music streaming platforms will eventually incorporate lyrics into their recommendation systems to make more meaningful recommendations to users. The system does function well on its own, however it could be even stronger when combined with the other metadata that Spotify uses. As mentioned previously, there is definitely work to be done to enhance the application, and I hope that I can continue to work on this project in the near future.

# Bibliography


[1] E. Çano, "Text-based sentiment analysis and music emotion recognition," *CoRR*, vol. abs/1810.03031, 2018.

[2] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, p. 993–1022, mar 2003.

[8] Z. Jiang, M. Yang, M. Tsirlin, R. Tang, Y. Dai, and J. Lin, ""low-resource" text classification: A parameter-free classification method with compressors," in *Findings of the Association for Computational Linguistics: ACL 2023*, (Toronto, Canada), pp. 6810–6828, Association for Computational Linguistics, July 2023.

[9] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," Jan. 2019. 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.

[10] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[11] I. A. P. Santana, F. Pinhelli, J. Donini, L. Catharin, R. B. Mangolin, Y. M. e Gomes da Costa, V. D. Feltrim, and M. A. Domingues, "Music4all: A new music database and its

applications," in *27th International Conference on Systems, Signals and Image Processing (IWSSIP 2020)*, 2020.

[12] A. G. Jivani *et al.*, "A comparative study of stemming algorithms," *Int. J. Comp. Tech. Appl*, vol. 2, no. 6, pp. 1930–1938, 2011.

[13] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[14] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[15] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[16] L. Richardson, "Beautiful soup documentation," *April*, 2007.

[17] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[18] R. Rehurek and P. Sojka, "Gensim–python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[19] N. Dingwall and C. Potts, "Mittens: an extension of GloVe for learning domain-specialized representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (M. Walker, H. Ji, and A. Stent, eds.), (New Orleans, Louisiana), pp. 212–217, Association for Computational Linguistics, June 2018.

[20] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[21] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[22] D. Gossi and M. H. Gunes, "Lyric-based music recommendation," in *Workshop on Complex Networks*, 2016.

[23] M. Vystrčilová and L. Peška, "Lyrics or audio for music recommendation?," in *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*, WIMS 2020, (New York, NY, USA), p. 190–194, Association for Computing Machinery, 2020.

[24] H.-C. Wang, S.-W. Syu, and P. Wongchaisuwat, "A method of music autotagging based on audio and lyrics," *Multimedia Tools and Applications*, vol. 80, pp. 15511–15539, 2021.

[25] S. King, *The Gunslinger*. New York, NY: Signet, 1982.

# Appendices

# Appendix A

# Survey Questions

This appendix shows the exact structure and wording of the questions in the survey.

## A.1 Participant Information

This section simply asked the participant the following questions:

1. Preferred Gender (Male, Female, Prefer not to say, [Other])
2. Age Range (18-24, 25-34, 35-44, 45-54, 55-64, 65+)
3. First Language (English, Spanish, Chinese (Mandarin), French, Arabic, Russian, Hindi, [Other])

## A.2 Section 1: Using the Application with a given song.

In this section the participant was first given a link to the application in order to answer the following questions. Note that these are taken verbatim from the survey form.

In the survey, there were two types of scales. The scale from 0-6 was labelled as "Strongly Disagree" (0), "Disagree", "Slightly Disagree", "Neutral", "Slightly Agree", "Agree", and "Strongly Agree" (6). The scale of 0-5 was labelled as "Completely Different" (0), "Different", "Slightly Different", "Slightly Similar", "Similar", "Extremely Similar" (5). We avoid the term "exactly the same" because this is a nigh impossible occurrence for this project. Similarly with a middle-ground or "neutral" for the second scale.

The following questions were asked, in this order:

- To make the experience less intimidating for those unfamiliar with word vectorization algorithms, the word vector type has be randomly selected for you and should be displayed at the bottom of the page. What vector type has been given to you? (Word2Vec, GloVe, LDA).

- Please enter **"Happy"** into the song name box and **"Pharrell Williams"** into the artist box, press submit, and answer the next question.
- Have you heard of **Happy** by **Pharrell Williams** before? (Yes, No, Unsure).
- Briefly, what do you think the meaning/topics of the song is/are? You can have a look at the lyrics by clicking on this link here.
- What are the songs given to you by the application? Please list the song names and the artists below.
- Please take a look at the 5 songs given to you, and select ONE of the 5 songs.
- What song did you pick?
- If you can, justify your choice (e.g. maybe you've heard of the song or artist before or if it's a genre you like), or if you just picked at random.
- Have you heard of this song before? (Yes, No, Unsure).
- If you can, please listen to the song you chose and have a look at its lyrics (you can use Genius like before or some other website if you wish). Please answer the following questions.
- I like this song (Scale of 0-6).
- Regardless of whether I liked the song or not, I would say the meanings of the song and **Happy** are... (Scale of 0-5).
- In terms of style/vibe, the song and **Happy** are... (Scale of 0-5).
- In terms of genre, the song and **Happy** are... (Scale of 0-5).
- I would add this song to one of my playlists. (Scale of 0-6).
- Briefly, what do you think the meaning/topics of the chosen recommendation is/are?
- **[OPTIONAL]** If you think that the two songs weren't similar lyrics-wise, what do you think would have been a better recommendation? You may use other songs from the recommendation list if you wish, but be sure to mention this in your answer.

## A.3 Section 2: Using the application using a song of your own choice.

The questions here are similar to section 1's, but with a few changes.

- The vector type may have changed because the page has been reloaded (i.e. you came back to it from the previous results page). What vector type have you been given now? (Word2Vec, GloVe, LDA).

- What was the song you searched for?
- What was the artist you searched for?
- Briefly, what do you think the meaning/topics of your queried song is/are?
- What are the songs given to you by the application? Please list the song names and the artists below.
- Please take a look at the 5 songs given to you, and select ONE of the 5 songs.
- What song did you pick?
- If you can, justify your choice (e.g. maybe you've heard of the song or artist before or if it's a genre you like), or if you just picked at random.
- Have you heard of this song before? (Yes, No, Unsure).
- If you can, please listen to the chosen recommended song and answer the following questions.
- I like this song (Scale of 0-6).
- In terms of meaning, the chosen recommendation and my query song are... (Scale of 0-5).
- In terms of style/vibe, the chosen recommendation and my query song are... (Scale of 0-5).
- In terms of genre, the chosen recommendation and my query song are... (Scale of 0-5).
- I would add this song to one of my playlists. (Scale of 0-6).
- Briefly, what do you think the meaning/topics of the chosen recommendation is/are?
- **[OPTIONAL]** If you think that the two songs weren't similar lyrics-wise, what do you think would have been a better recommendation? You may use other songs from the recommendation list if you wish, but be sure to mention this in your answer.

The first entry of this section was compulsory, but there was the option to do up to 4 additional entries (5 total for section 2).

## A.4 Section 3: Your music tastes.

The following questions were asked:

- What is/are your favourite genre(s) of music?
- Who is/are your favourite artist(s)?
- Which of these streaming platforms do you regularly use? (Spotify, Apple Music, Amazon Music, SoundCloud, YouTube, Deezer, Pandora, [Other]).

- Which of the following do you regularly use to listen to music? (Streaming Platforms, Vinyl, CD, Cassette, MP3 Player, [Other]).
- How often would you say you listen to music, on a weekly basis?

## A.5 Section 4: Thoughts about lyrics-based music recommendation

The following questions were asked:

- I think that the application I used in this survey performed well (Scale of 0-6).
- I would use this application again to provide me with song recommendations (Scale of 0-6).
- I like/am intrigued by the idea of lyrics-based music recommendations (Scale of 0-6).
- How important are a song's lyrics to you when looking for new music recommendations? (Not at all important, Not too important, Important, Very Important).

## A.6 Section 5: Further Comments

The participants were able to use this section to provide any thoughts or extra feedback about the application.

# Appendix B

# Ethics Application

This appendix includes the ethics application for the survey, as well as the approval letter.


# Application 057178

## Section A: Applicant details

Date application started:
Fri 20 October 2023 at 11:20

First name:
Freddie

Last name:
Butterfield

Email:
fbutterfield1@sheffield.ac.uk

Programme name:
Software Engineering (Computer Science) with Industrial Placement

Module name:
COM3610
Last updated:
04/02/2024

Department:
Computer Science

Applying as:
Undergraduate / Postgraduate taught

Research project title:
A Lyrics-Based Music Recommendation Web Application

Has your research project undergone academic review, in accordance with the appropriate process?
Yes

Similar applications:
*- not entered -*

## Section B: Basic information

### Supervisor

| Name | Email |
|---|---|
| Varvara Papazoglou | v.papazoglou@sheffield.ac.uk |

### Proposed project duration

Start date (of data collection):
Thu 11 April 2024

Anticipated end date (of project)
Wed 8 May 2024

### 3: Project code (where applicable)

Project externally funded?
No

Project code
*- not entered -*

### Suitability

Takes place outside UK?
No

Involves NHS?
No

Health and/or social care human-interventional study?
No

ESRC funded?
No

Likely to lead to publication in a peer-reviewed journal?
No

Led by another UK institution?
No

Involves human tissue?
No

Clinical trial or a medical device study?
No

Involves social care services provided by a local authority?
No

Is social care research requiring review via the University Research Ethics Procedure
No

Involves adults who lack the capacity to consent?
No

Involves research on groups that are on the Home Office list of 'Proscribed terrorist groups or organisations?
No

### Indicators of risk

Involves potentially vulnerable participants?
No
Involves potentially highly sensitive topics?
Yes

## Section C: Summary of research

### 1. Aims & Objectives

The project I am designing is a web application that provides song recommendations based on lyrics. Since song lyrics can be quite subjective, we have no "gold standard" of recommendations on which to perform automatic evaluation. Instead, I am choosing to conduct a survey which will ask users whether they enjoyed listening to the songs that were recommended to them, in order to get a more human evaluation of the system. The main aims from this project will be 1. Support the concept of lyrics-based music recommendation, 2. build an application that will be able to provide lyrics-based song recommendations to a user.

### 2. Methodology

The survey will be created using Google Forms, and will involve a series of questions. The format is as follows:

Section 1: Participant/demographic information
a. Preferred gender
b. Age range
c. Native language

Section 2: The user will input a song chosen by me (currently this is "Happy" by Pharrell Williams) into the system and ask questions related to the first song that is recommended to them.
a. Did you like the song? (Strongly disagree - strongly agree)
b. Do you think the two songs are similar in terms of their meaning? (Completely different - Extremely similar)
c. Do you think they have a similar style/key/vibe? (Completely different - Extremely similar)
d. Would you add it to a playlist? (Strongly disagree - strongly agree)
e. What do you think the meaning of the two songs are? (As a short-answer textbox)
f. (Optional) What do you think would have been a better recommendation? (Short-answer textbox)
g. (For research purposes) What word vector method was given to you? - When the user uses the application a random word vector method (3 different types) will be given to them, they just have to read off the screen which one was given to them.

Section 3: Same as section 1 but the user now has the chance to enter a song of their choice, they may do this up to 5 times (minimum once).

Section 4: A user will be asked about their music tastes and habits.

Section 5: A user will be asked about their thoughts on the application and lyrics-based music recommendation as a whole.

I will then use this data to then evaluate how well the application has performed, and also ask the question whether lyrics-based recommendations are a good idea or not. The data can also be visualised in a graphic or tabular format.

#### 3. Personal Safety

Have you completed your departmental risk assessment procedures, if appropriate?

Not applicable

Raises personal safety issues?

No

The users will be able to perform this in the comfort of their own home, and they are simply just listening to music at an advised low volume.

### Section D: About the participants

#### 1. Potential Participants

The participants can be anyone with access to a music streaming service e.g., Spotify, Apple Music, Amazon Music, YouTube, etc.

#### 2. Recruiting Potential Participants

If all goes well, I aim to be able to deploy my project to a web server. That way, I can simply provide a link to it in the description of the Google Form. I will then share this Google form with my peers on messaging apps (Facebook, Discord, WhatsApp, etc.). If I am unable to deploy the project, then I will have to provide the participants with the source code as a .zip file and give them detailed instructions on how to unzip and run the application.

#### 2.1. Advertising methods

Will the study be advertised using the volunteer lists for staff or students maintained by IT Services? No

*- not entered -*

#### 3. Consent

Will informed consent be obtained from the participants? (i.e. the proposed process) Yes

Consent will be obtained within the Google Form, or the participant can request or download a copy of it from either myself or from within the Google Form.

#### 4. Payment

Will financial/in kind payments be offered to participants? No

#### 5. Potential Harm to Participants

What is the potential for physical and/or psychological harm/distress to the participants?

There is no considerable physical or psychological harm to the participants. If we consider a worst-case, then we could technically say

that loud music could constitute as physical harm, but that would be the fault of the participant. As for psychological harm, this research involves reading and interpreting creative work; it is possible that some song lyrics can be related to or associated with personal, religious, political, social, and economic views and events that can cause some distress or can be reminiscent of negative experiences and memories.

How will this be managed to ensure appropriate protection and well-being of the participants?

We will recommend that the participants select a volume level on their headphones that they are comfortable with. We can also advise that the participant partakes in the research in a comfortable environment e.g. their bedroom, to reduce the amount of distress if the song or song lyrics trigger something unpleasant. If a participant has become upset due to a song that was recommended to them, they will be able to contact us and we can provide website links to help them (e.g. https://www.mentalhealth.org.uk/your-mental-health , https://www.sheffield.ac.uk/ssid/mental-health/index). If they did not like a song, they will be able to express this in the form.

### 6. Potential harm to others who may be affected by the research activities

Which other people, if any, may be affected by the research activities, beyond the participants and the research team?

I do not believe that any other people besides the participants will be affected by my research.

What is the potential for harm to these people?

None.

How will this be managed to ensure appropriate safeguarding of these people?

None required.

### 7. Reporting of safeguarding concerns or incidents

What arrangements will be in place for participants, and any other people external to the University who are involved in, or affected by, the research, to enable reporting of incidents or concerns?

If participants or any other people are dissatisfied with any aspect of the research and wish to make a complaint, they can contact the researcher (Freddie Butterfield: fbutterfield1@sheffield.ac.uk) or the project supervisor (Varvara Papazoglou: v.papazoglou@sheffield.ac.uk) in the first instance. If they feel their complaint has not been handled in a satisfactory way then they can contact the Head of the Department of Computer Science, Prof. Heidi Christensen: heidi.christensen@sheffield.ac.uk.

Who will be the Designated Safeguarding Contact(s)?

The researcher, the project supervisor, and the Head of the Department of Computer Science, as specified in the previous question.

How will reported incidents or concerns be handled and escalated?

The DSC (Designated Safeguarding Contact) should try to obtain as much information as possible regarding the reported incidents or concerns, whilst also acting sensitively and providing reassurance regarding the process that will be undertaken for handling the report. The Research Ethics & Integrity Manager in Research Services will also be notified so that advice can be sought regarding the appropriate next steps. If necessary, advice will be sought from one of more the following, depending on the nature of the concern:
- The University's Safeguarding Panel
- Human Resources
- University Research Ethics Committee
- Vice-President for Research and/or Faculty Directors of Research & Innovation

The victim, and other relevant parties should be kept informed regarding progress and key decisions in dealing with the matter.

## Section E: Personal data

### 1. Use of personal data

Will any personal data be processed or accessed as part of the project?
Yes

Will any 'special category' personal data be processed or accessed as part of the project?
No

Provide the number of people whose personal data you expect to process or access.
30

### 2. Managing personal data

Which organisation(s) will act as data controller(s) of the personal data?

University of Sheffield only

Who will have access to the personal data?
Myself, my supervisor, and the computer science department.

What measures, processes and/or agreements will be put in place to manage the personal data?
Participants will not be identified in the final report, only the combined results and statistics will be shown.

Will all identifiable personal data in digital or physical format be destroyed within a defined period after the project has ended?
Yes

When will the identifiable personal data be destroyed?
Once the dissertation project has been marked and I have received a grade (August 2024).

### 3. Third-party services

Will any external third-party services not provided by the University be used to process or access personal data during the project?
No

### 4. Security of computers, devices and software

Will personal data be processed or accessed on any computers or devices that are not managed by the University of Sheffield?
Yes

Will all computers and devices that are not managed by the University of Sheffield be secured in accordance with the IT Code of Connection?
Yes

Will any software not approved by the University of Sheffield be used to process or access data?
No

Will any software be written or developed in order to process or access the personal data?
No

## Section F: Supporting documentation

### Information & Consent

Participant information sheets relevant to project?
Yes

Document 1132236 (Version 1) All versions

Consent forms relevant to project?
Yes

Document 1132237 (Version 1) All versions

### Additional Documentation

### External Documentation

N/A.

## Section G: Declaration

Signed by:
Freddie Butterfield

Date signed:
Mon 22 January 2024 at 19:05

Offical notes

*- not entered -*


Downloaded: 12/04/2024
Approved: 04/02/2024

Freddie Butterfield
Registration number: 200337708
Computer Science
Programme: Software Engineering (Computer Science) with Industrial Placement

Dear Freddie

**PROJECT TITLE:** A Lyrics-Based Music Recommendation Web Application
**APPLICATION:** Reference Number 057178

On behalf of the University ethics reviewers who reviewed your project, I am pleased to inform you that on 04/02/2024 the above-named project was **approved** on ethics grounds, on the basis that you will adhere to the following documentation that you submitted for ethics review:

- University research ethics application form 057178 (form submission date: 22/01/2024); (expected project end date: 08/05/2024).
- Participant information sheet 1132236 version 1 (22/01/2024).
- Participant consent form 1132237 version 1 (22/01/2024).

If during the course of the project you need to deviate significantly from the above-approved documentation please inform me since written approval will be required.

Your responsibilities in delivering this research project are set out at the end of this letter.

Yours sincerely

Jon Barker
Ethics Administrator
Computer Science

Please note the following responsibilities of the researcher in delivering the research project:

- The project must abide by the University's Research Ethics Policy: https://www.sheffield.ac.uk/research-services/ethics-integrity/policy
- The project must abide by the University's Good Research & Innovation Practices Policy: https://www.sheffield.ac.uk/polopoly_fs/1.671066!/file/GRIPPolicy.pdf
- The researcher must inform their supervisor (in the case of a student) or Ethics Administrator (in the case of a member of staff) of any significant changes to the project or the approved documentation.
- The researcher must comply with the requirements of the law and relevant guidelines relating to security and confidentiality of personal data.
- The researcher is responsible for effectively managing the data collected both during and after the end of the project in line with best practice, and any relevant legislative, regulatory or contractual requirements.

# Appendix C

# Project Information Sheet

This appendix shows the downloadable version of the information sheet that participants were presented with before starting the survey. This information was also included in the Google Form.

# Information Sheet

16/01/2024

## Project Title: A Lyrics-Based Music Recommendation Application

You are being invited to take part in a research project. Before you decide whether or not to participate, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part. Thank you for reading this.

**1. What is the project's purpose?**
The purpose of this project is to reinforce and provide evidence to support (or disprove) the practicality and effectiveness of lyrics-based song recommendations, and to also build an application that uses purely lyrics to make recommendations as a medium to provide this evidence.

**2. Why have I been chosen?**
You have been chosen because you are a person who listens to music and is able to listen to music (through Spotify, YouTube, Apple Music, etc.), are a part of the general public, are interested in this research area, or all three. Due to the subjective nature of song lyrics, it is difficult for a computer to automatically evaluate the results of the application, therefore the evaluation of it relies entirely upon user feedback.

**3. Do I have to take part?**
Your participation in the project is completely voluntary. If you do decide to take part you will be asked to complete a consent form within the Google Form (and you can also request a copy of this information sheet by contacting Freddie Butterfield at fbutterfield1@sheffield.ac.uk) and you will be automatically directed to the study which will involve answering questions about the recommendations that the application gives to you, as well as questions about your own listening habits and your thoughts on lyrics-based music recommendation. It is preferred that you complete this survey on a computer or laptop. Even if you give consent, you are free to withdraw from the study at any time with no consequences and without giving a reason by simply closing the web browser on your smartphone, tablet or computer, and your responses will not be recorded. Please note that by choosing to participate in this research, this will not create a legally binding agreement,

nor is it intended to create an employment relationship between you and the University of Sheffield.

**4. What will happen to me if I take part? What do I have to do?**

Completion of the survey should take, as a minimum, approximately 20-30 minutes (with no time limit). However, this may take longer (you should allow up to an hour) should you decide to (optionally) test the system multiple times. You will be asked to provide some demographic information such as gender and age range, and then you will be required to answer some questions. The questions are split into sections as follows:

1. Participant demographic information (no names or personal information taken)
2. Using the application with a given song (same for all participants)
3. Using the application with a song of your choice (you may repeat this section up to 5 times, with a minimum of once)
4. Your listening habits and tastes
5. Your thoughts on lyrics-based music recommendations, and the application you will be using

Even if you give consent, you will still be free to withdraw at any time without giving a reason, by simply closing the browser, and none of your responses will be saved.

**5. What are the possible disadvantages and risks of taking part?**

There is a risk that a song you have been asked to listen to (which will be one of the recommendations) is a song that may cause you to become upset or emotional. You are also advised to keep your device volume at a safe level.

**6. What are the possible benefits of taking part?**

Whilst there are no immediate benefits for those people participating in the project, it is hoped that this work will provide some useful and interesting insights on lyrics-based music recommendation. If you choose to suggest a better recommendation(s) in the survey, you will be contributing to the development of a ground truth/labels for automatic evaluation in the future. You may also walk away from the survey with some potentially new music recommendations.

**7. Will my taking part in this project be kept confidential?**

Your responses will be anonymised but there may be some comments about demographics in the final report. You will not be identifiable in any reports or publications, and no personal details will not be shared with other researchers. The data is anonymised from the beginning, i.e. first and last names will not be asked for.

**8. What is the legal basis for processing my personal data?**

According to the data protection legislation, we are required to inform you that the legal basis we are applying in order to process your personal data is that 'processing is necessary for the performance of a task carried out in the public interest' (Article 6(1)(e)).
Further information can be found in the University's Privacy Practice Notice https://www.sheffield.ac.uk/govern/data-protection/privacy/general

**9. What will happen to the data collected, and the results of the research project?**
The data will be analysed by Freddie Butterfield. During data collection, your responses will be stored fully anonymised, without any identifiable personal data. Once the report has been submitted and a grade has been given for the project in August, the data will be deleted.

**10. Who is organising and funding the research?**
The research is organised by Freddie Butterfield and is not funded.

**11. Who is the Data Controller?**
The University of Sheffield will act as the Data Controller for this study. This means that the University is responsible for looking after your information and using it properly.

**12. Who has ethically reviewed the project?**
This project has been ethically approved via the University of Sheffield's Ethics Review Procedure, as administered by the Computer Science department. The ethics approval number is 057178.

**13. What if something goes wrong and I wish to complain about the research or report a concern or incident?**
If you are dissatisfied with any aspect of the research and wish to make a complaint, please contact the researcher (Freddie Butterfield: fbutterfield1@sheffield.ac.uk) or the project supervisor (Varvara Papazoglou: v.papazoglou@sheffield.ac.uk) in the first instance. If you feel your complaint has not been handled in a satisfactory way, you can contact the Head of the Department of Computer Science, Prof. Heidi Christensen: heidi.christensen@sheffield.ac.uk. If the complaint relates to how your personal data has been handled, you can find information about how to raise a complaint in the University's Privacy Notice: https://www.sheffield.ac.uk/govern/data-protection/privacy/general .

**14. Contact for further information**
If you have any questions either ahead of participation or relating to your data and the study, please contact the researcher (Freddie Butterfield: fbutterfield1@sheffield.ac.uk). An electronic copy of this information sheet and consent form can be provided on request by contacting the researcher (Freddie Butterfield: fbutterfield1@sheffield.ac.uk).

Thank you for taking part in the project!

# Appendix D

# Participant Consent Form

This appendix shows the downloadable version of the consent form for the survey. The consent form was also implemented in the Google Form as well as having this downloadable copy available.

# Participant Consent Form

A Lyrics-Based Music Recommendation Application

| ***Please tick the appropriate boxes*** | **Yes** | **No** |
|---|---|---|
| **Taking part in the project** | | |
| I have read and understood the project information sheet dated 16/01/2024 or the project has been fully explained to me. (If you will answer No to this question please do not proceed with this consent form until you are fully aware of what your participation in the project will mean.) | | |
| I have been given the opportunity to ask questions about the project. | | |
| I agree to take part in the project. I understand that taking part in the project will include my opinions, as well as some demographic information. | | |
| I understand that by choosing to participate as a volunteer in this research, this does not create a legally binding agreement nor is it intended to create an employment relationship with the University of Sheffield. | | |
| I understand that my taking part is voluntary and that I can withdraw from the study at any time; I do not have to give any reasons for why I no longer want to take part and there will be no adverse consequences if I choose to withdraw. | | |
| **How my information will be used during and after the project** | | |
| I understand that all the information provided will be anonymised. | | |
| I understand and agree that the authorised researchers will have access to this data and may use the data in publications, reports, web pages, and other research outputs only if they agree to preserve the confidentiality of the information as requested in this form. | | |
| **So that the information you provide can be used legally by the researchers** | | |
| I agree to assign the copyright I hold in any materials generated as part of this project to The University of Sheffield. | | |

# Appendix E

# Project Diagrams

## E.1 Training Stage Flow

This section shows the flow of operations for the training stage. Due to its size, it has been split into three figures, the last of which has been rotated to fit the page.

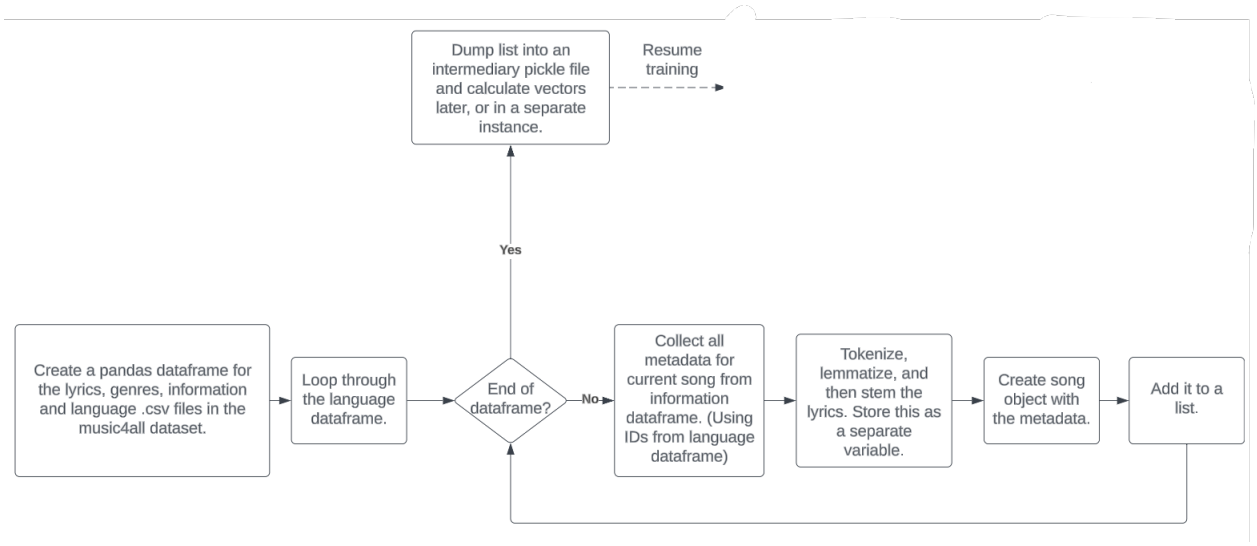


Figure E.1: A diagram of a section of the training stage, up until section 4.1.4 as described in chapter 4.

## E.2 Application Flow

This section shows the order of operations of the web application. Due to its size, it has had to be rotated to fit the page.

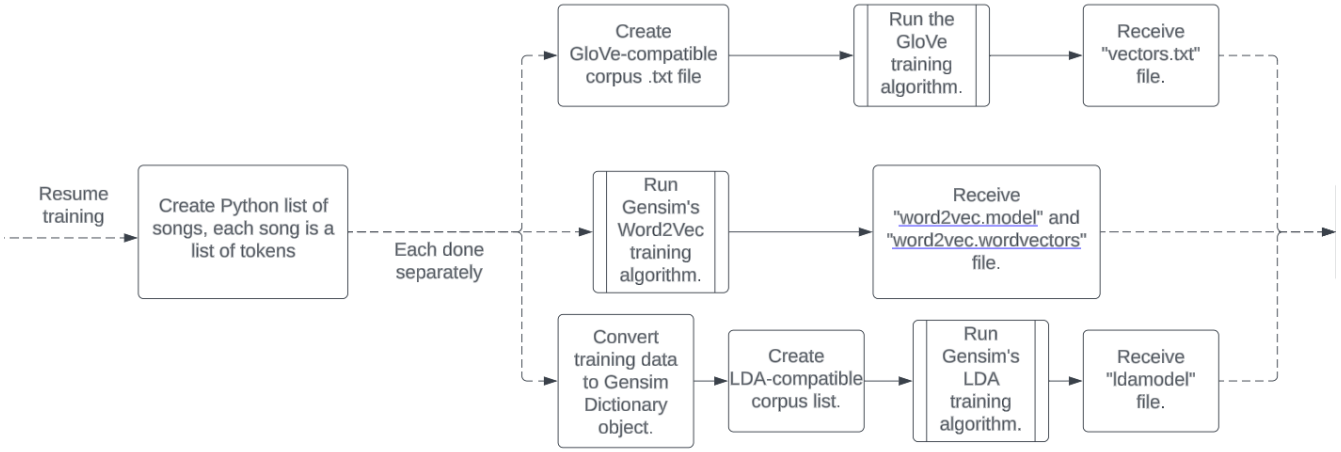


Figure E.2: A diagram of a section of the training stage, up until section 4.1.5 as described in chapter 4.

## E.3 File Dependencies

This section shows the file and library dependencies of the files in the project.

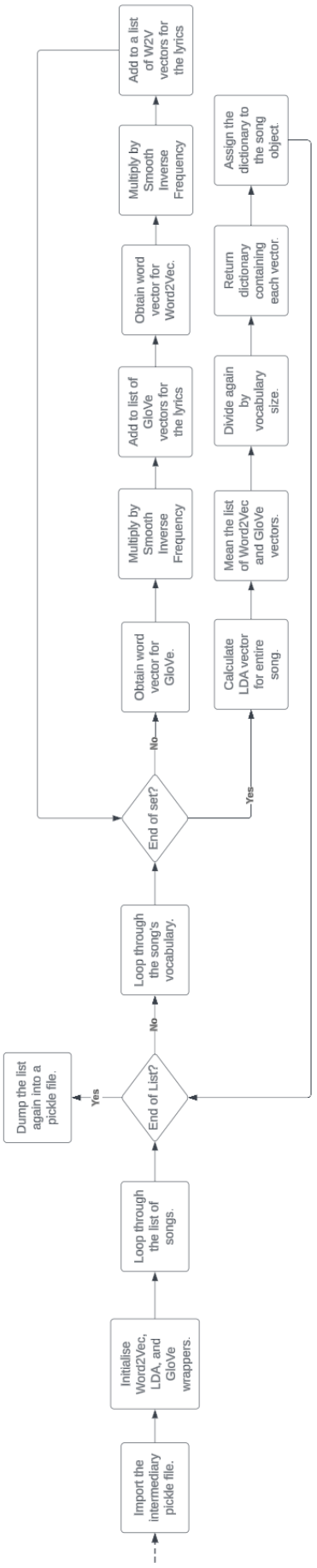


Figure E.3: A diagram of a section of the training stage, detailing 4.1.5 as described in chapter 4.

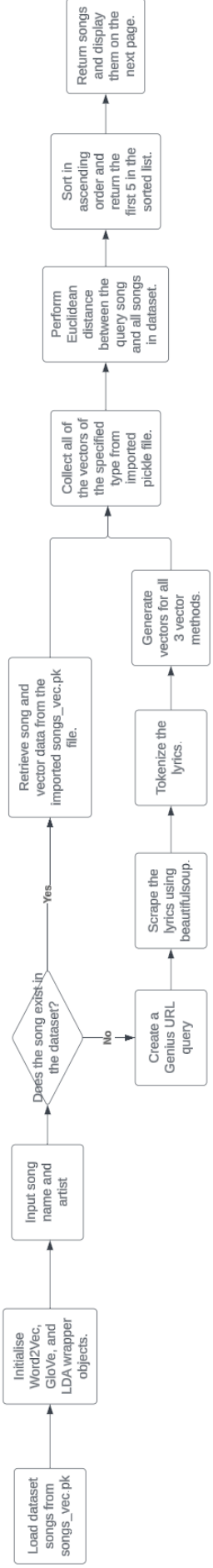


Figure E.4: A diagram showing how the web application flows.

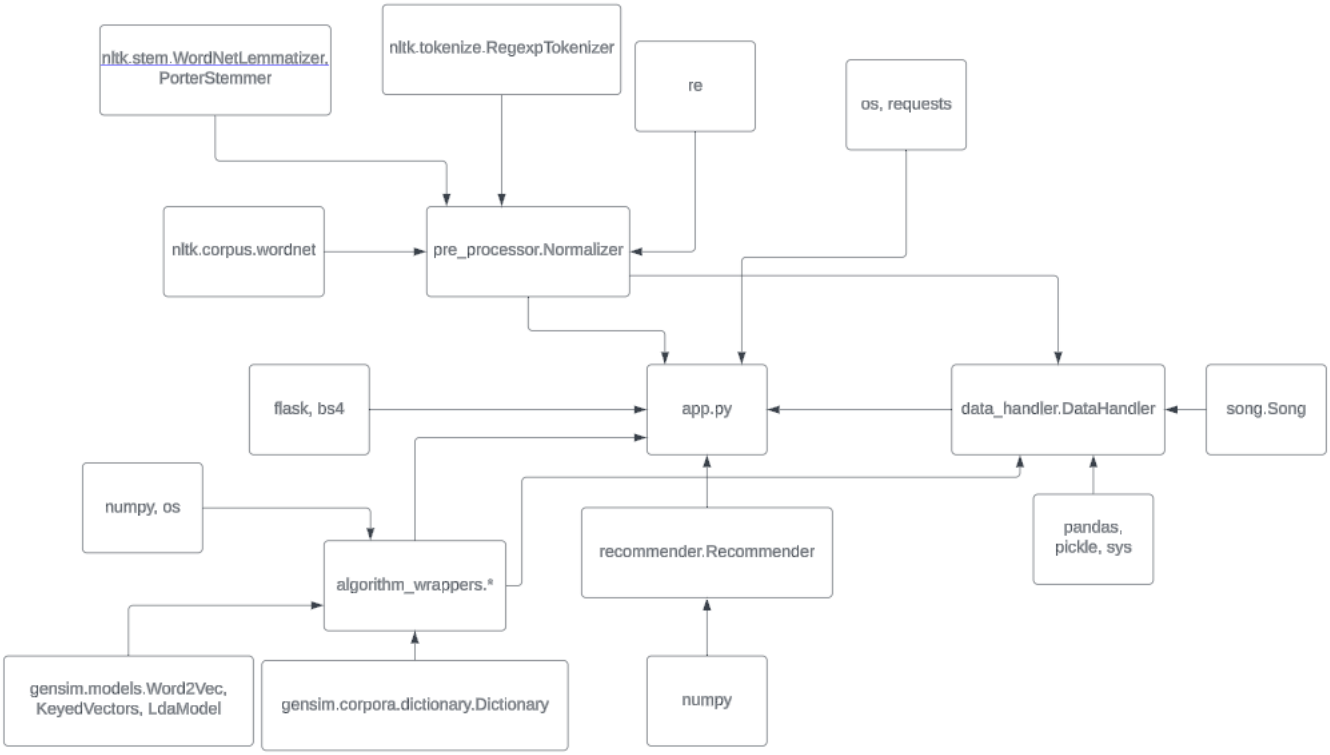


Figure E.5: A diagram showing the file and library dependencies of the project.

# Appendix F

# Front End Snapshots

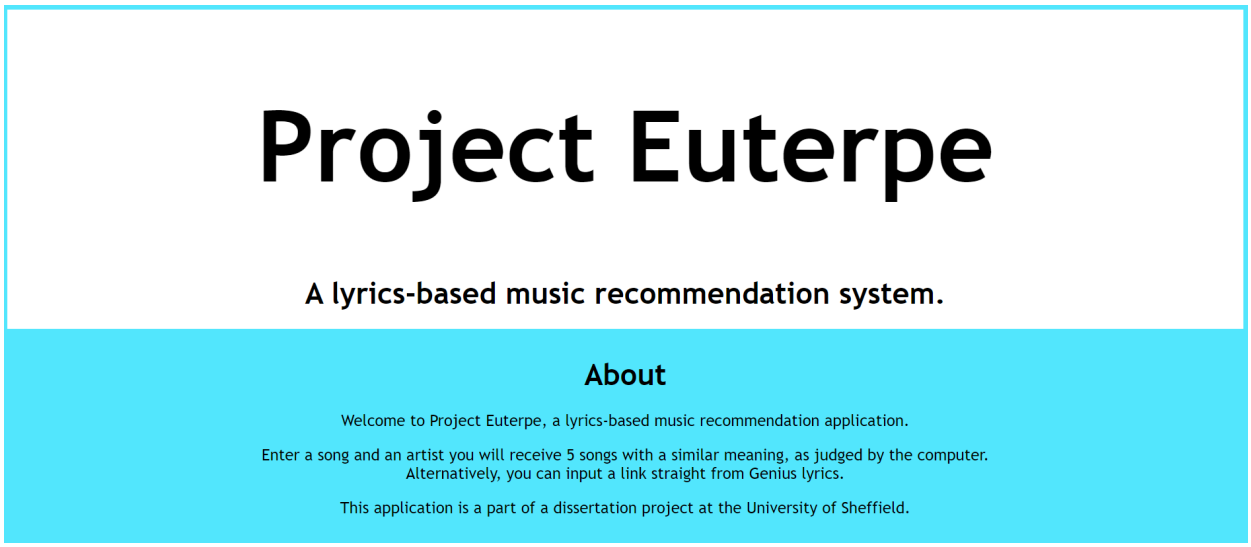


Figure F.1: The top half of the front page of the application.

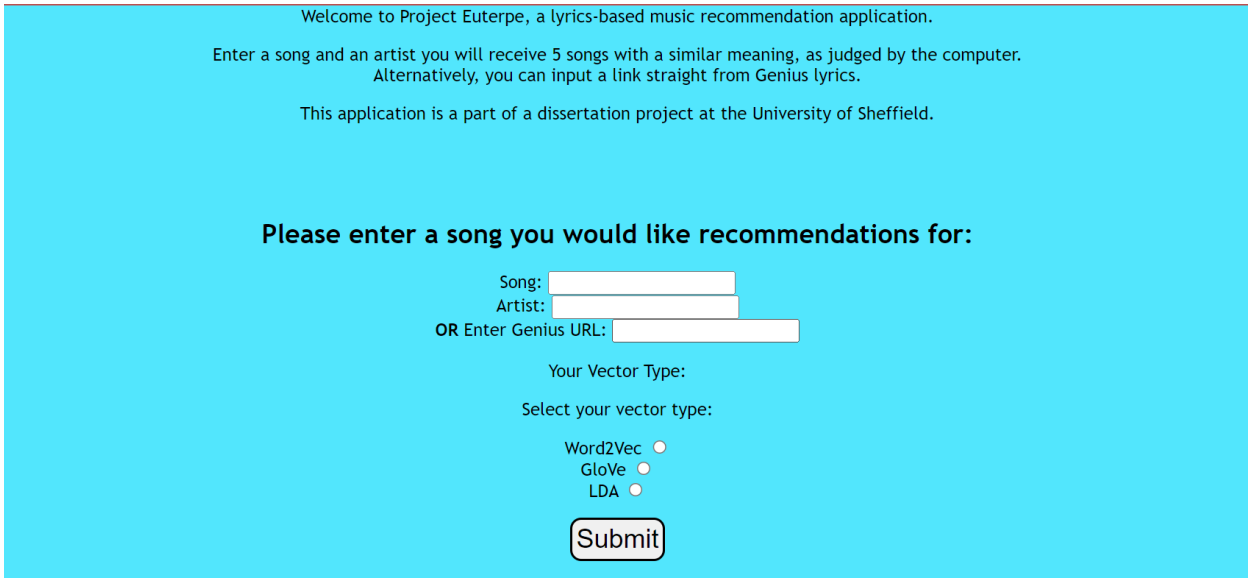


Figure F.2: The bottom half of the front page of the application.

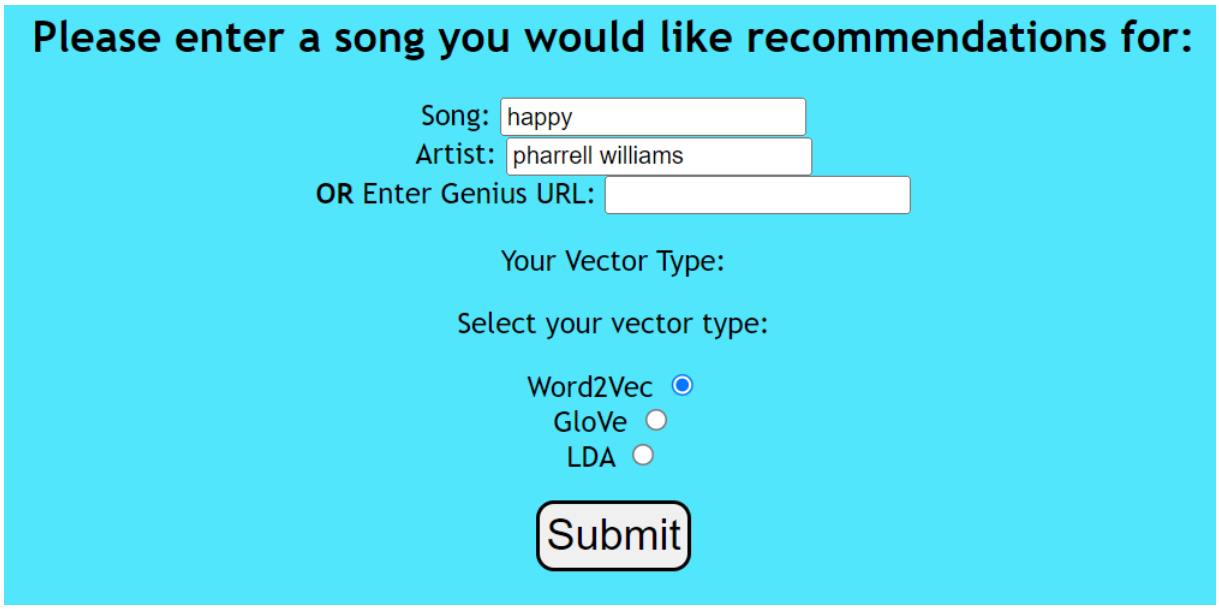


Figure F.3: Example of a search query.

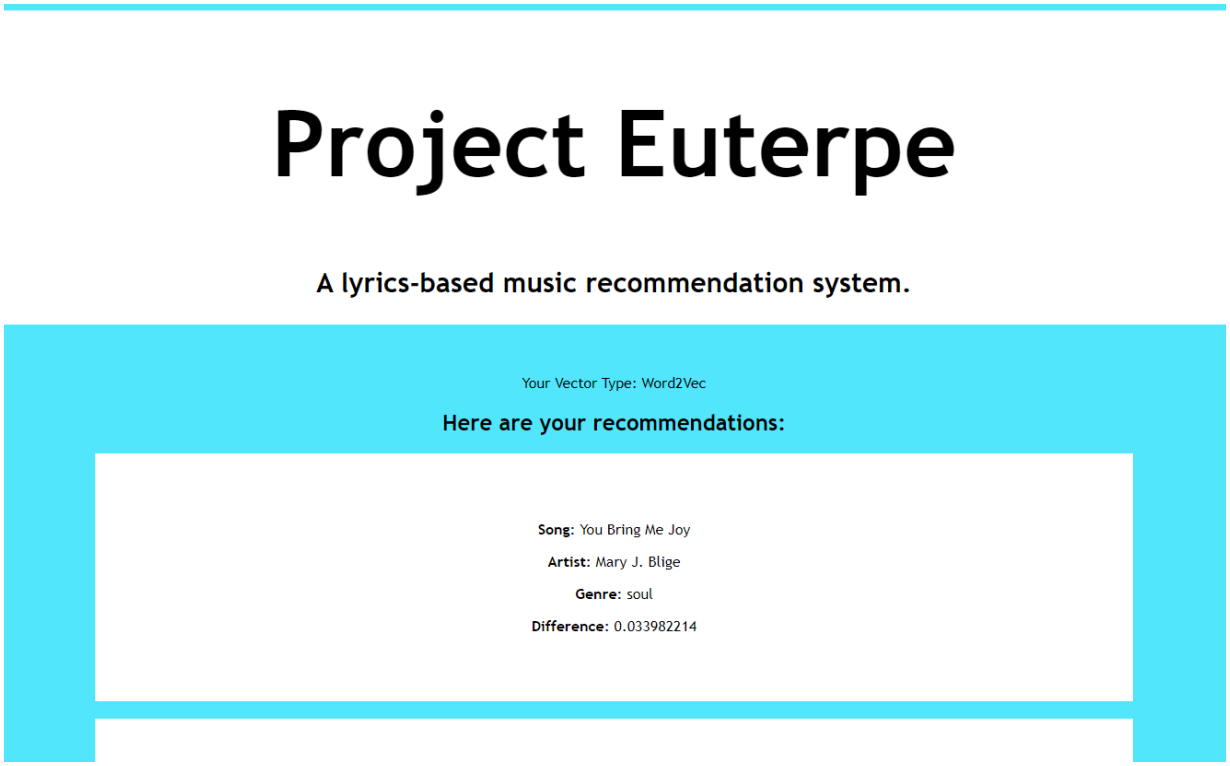


Figure F.4: The results page, using the query from F.3.