CA687I - Cloud Systems

A report submitted to Dublin City University by Group A

Colin Hanily, KT Tan, Rana Zeeshan, Darren Rooney, Neal Knauf

Lecturer: Alessandra Mileo

March 2021

School of Computing, Dublin City University, Glasnevin, Dublin 9

Student ID	**Name**	**Email**
20214350	Colin Hanily	colin.hanily2@mail.dcu.ie
21210014	Neal Knauf	neal.knauf2@mail.dcu.ie
20215470	Kay-Ti Tan	kayti.tan3@mail.dcu.ie
20216275	Rana Zeeshan Ali Shahid	rana.shahid2@mail.dcu.ie
20216268	Darren Rooney	darren.rooney37@mail.dcu.ie

Application of Cloud Technologies using data from a Spotify Music Dataset

Introduction

Over the last decade, Spotify has changed the way music is consumed thereby affecting the music creation process. Recent articles have noticed that the average song length has dropped since the 90's as artists are paid per stream if their song is listened to for more than 30 seconds. Similar analysis has shown a phenomenon known as the "Loudness War" in which audio levels in music have been ever increasing. These trends peaked our interest in analysing musical trends over the past century.

Originally, our team goal was to analyse the characteristics that make a song popular, however it quickly became clear that this was unfeasible. Therefore we pivoted our goal to analyse the characteristic trends of music over the past century and how these trends have changed over time. As well as this, we have created a music recommender in which a user can input the characteristics of the type of song they would like to listen to, and the top 3 recommendations are returned to the user, thus providing a novel approach to music searching.

Data Source

The group chose the publicly available Spotify music dataset from Kaggle. The source contains more than 175,000 songs from 1920-2021. Each row represents a single track whilst the 19 columns represent a specific field associated with the track such as popularity, tempo, artists, loudness, explicitly, etc.

The data source is available here on Kaggle.

Analytics

Technologies Used

Apache Hadoop Framework - Used to analyse our data on a google cloud platform instance. Hadoop is a highly compatible big data framework that supports multiple languages which will be useful for our data cleansing, pruning and querying.

Google Cloud Platform DataProc - The dataproc cluster can access and query our data stored on google cloud storage.

Python – Used to create the song recommender using the K-Nearest-Neighbour (KNN) machine learning model included in the Scikit-Learn machine learning library.

Apache Pig - Runs on Hadoop will be used to cleanse and prune our dataset for our data queries.

Apache Hive - Runs on Hadoop in GCP and Hortonworks and was used to run our dataset queries to find data correlation and prepare our findings for visualisation.

Cloudera/Hortonworks Platform with Ambari - Used to run hive queries on the dataset to keep possible billing issues on GCP to a minimum.

Google BigQuery BI Engine - Used BI Engine to integrate the data analysed in Hive and move the results stored in cloud storage HDFS to Tableau.

Tableau - Data visualisation.

Google Drive – Team Reports and task breakdown sharing.

Movavi – Demo video recording.

Slack/Whatsapp/Zoom – Team Communication

Github – Storing project code and documentation

Dataproc & Hortonworks

Data extraction, transformation, load and querying were run on GCP, utilizing dataproc - a fast, relatively easy to use and fully managed cloud service for running Hadoop clusters. In addition some members of the team used Hortonworks Data Platform, powered by Apache Hadoop from a HiveQL querying perspective.

Extract, Transform(Cleanse), Load (ETL) Process

The dataset was cleaned and pruned using Apache Pig on a GCP dataproc instance.
The data originally had 19 columns with 174,389 songs from which 2,159 duplicates entries were removed.

Unused columns from the remaining rows were removed such as release date and ID. Further cleaning was then performed such as rounding song characteristics to 2 decimal places, concatenating the artists column array per row into a single string and converting song duration times from milliseconds to seconds. The data was then saved here as a CSV file using the "|" symbol as a separator due to interpretation issues with Pig as multiple song and artist entries made use of commas. The header fields were then re-inserted for use with our python script. The Pig script for this cleaning can be found here**.**

Observations: A number of indistinguishable artists and name entries were caused by non-english character values. Where artists released the same song in different years, these entries were counted as unique as they had different lengths possibly due to different intros/outros and edits.

Hive Querying

The cleansed dataset was subsequently picked up in Hive(GCP + Hortonworks) to populate the newly created stats_per_decade table which was used for trend analysis per decade. A number of queries were written in HiveQL to create and extract this table to Tableau allowing the team to visualise the queries. The HiveQL code can be found on GitHub here.

Music Recommender Creation

As the original idea of the cloud application was to investigate the characteristics that make a song popular, a number of models were first made to see if any such correlation did exist. It quickly became apparent after testing logistic machine learning models such as logistic/linear regression and decision tree classification that there was little to no correlation with models producing 39% accuracy.

We quickly pivoted to creating a novel approach to music searching/recommendations. This was achieved by using the K-Nearest-Neighbour (KNN) classifier available in the Scikit-Learn python library. A user is able to enter the characteristics of the type of song which they are searching for such as year of release, energy, speechiness, tempo and the top 3 results are then returned to the user. The python script for this as well as early popularity exploration can be found here.

Connecting Music Recommender to Tableau

The music recommendation script was then connected to Tableau via the python server Tabpy. A user can input the characteristics of the type of song they are looking for into Tableau at which point these characteristics are sent to the Tabpy server via a socket connection and used as arguments in the recommender script. The results are then sent back to Tableau and displayed to the user. The Tableau workbook as well as a setup instructions for connecting the Recommender script to Tableau using Tabpy can be found here.

Data Visualization & Analytics Results

We explored Tableau and Qlik sense for Data Visualization. Although Qlik sense was quite user-friendly, we preferred Tableau because of its extensive features, larger online community and flexible trial version. We used Tableau Public for initial dataset exploration and overall trend analysis w.r.t popularity of songs. Later we moved to Tableau Desktop Professional version for advanced capabilities & compatibility.

We explored Tableau and its features in depth. Starting from basic understanding of different plots, measure values, sheets and dashboards to advanced filters, parameter actions and custom calculated fields etc. Once the Music Recommender script was connected to Tableau, we used parameter actions to restrict the direct query submission, so that users can make all the changes smoothly and then click the "Apply" button to execute those input values. In addition to this, we needed to organize and clean our dataset in multiple ways for different purposes. So we calculated the correlations among all attributes and used a separate data source to populate the heat map.

We wanted our application to be significantly interactive and user-friendly, therefore we gave substantial importance to the interface modification capabilities of Tableau. This enabled us to design a nice looking dark themed application, with white fonts and music related artwork. We structured our application in 4 screens.

1. Introductory Screen

- A cover page to highlight the project and team members

2. Music Browser

- Contains a Music Browser, where users can browse through the music database of 100 years, based on popularity.

3. Music Recommender

- A music recommender GUI which allows input of 15 song attributes used as filters. It will recommend top songs, based on the values of the user's input attributes.

4. Trends & Correlations

- Showcases some important and relevant statistical infographics. This includes characteristic trends, correlation heat map and breakdown of songs based on different parameters such as explicit/non-explicit and major/minor etc. as below.

The figure below clearly shows the trends and breakdown of the musical characteristics over the last century. For example it can be seen that energy levels of music has increased gradually whereas speechiness has been on the decline. This may be due to the constant bettering of music technology and the prevalence of music genres without vocals such as Techno. More in depth breakdown of trends is available in our Tableau Workbook. Surprisingly, average valence(happiness) has dropped dramatically over the last century.

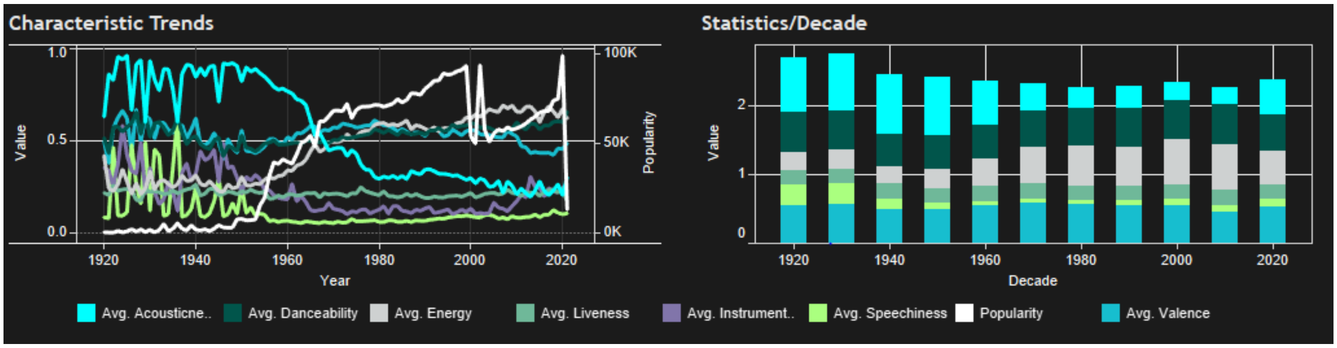
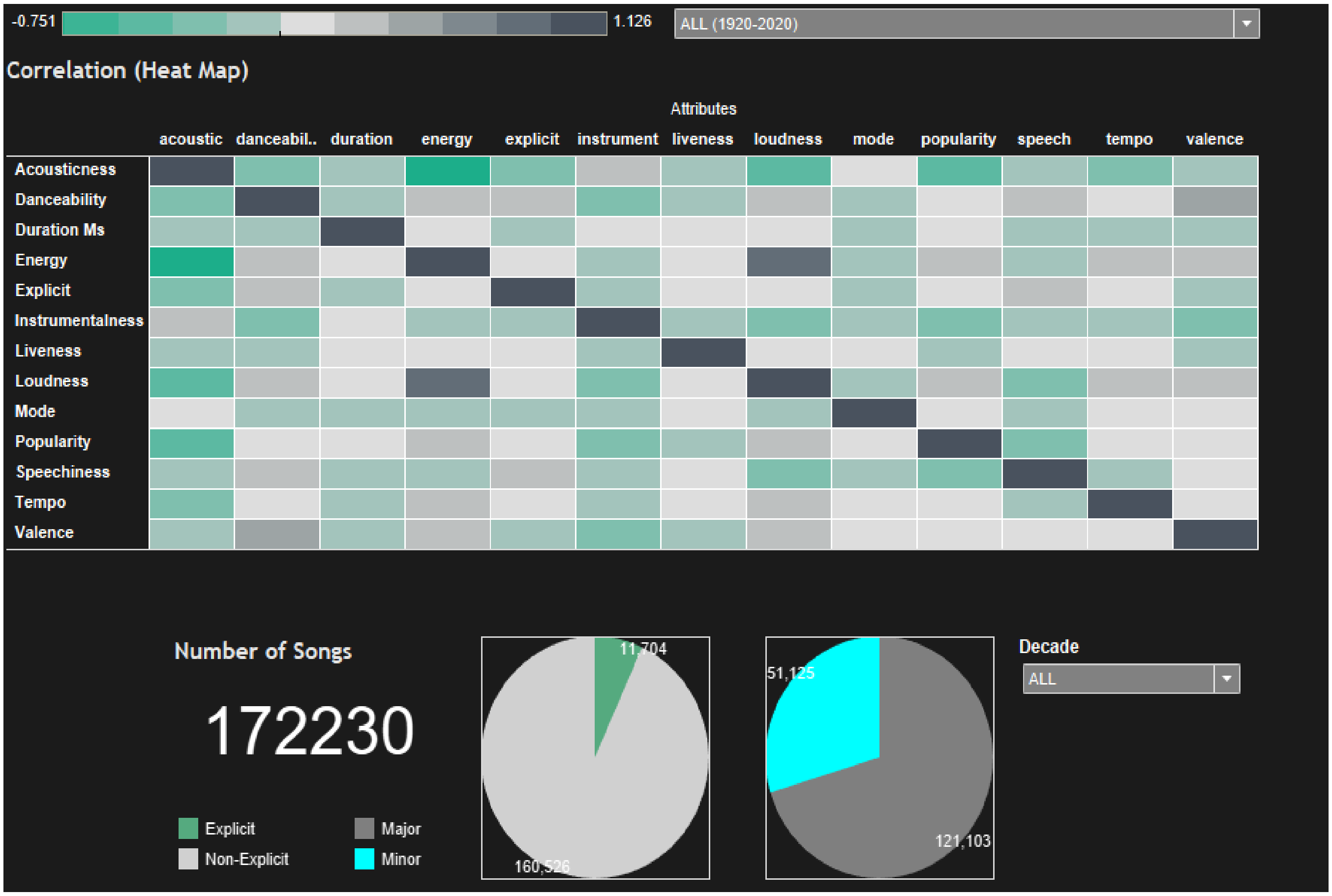


Figure 1: Characteristic Trends Breakdown

Figure 2 below show the correlation heat map of the characteristics. This can be broken down and viewed per decade with the dropdown menu. From the below heat map we can see that energy levels have a strong correlation with loudness and popularity whereas acousticness is inversely correlated with many characteristics such as energy, danceability and loudness. The Pie charts included also clearly shows the breakdown of music explicitness and tonality.



ALL (1920-2020)

Correlation (Heat Map)

Number of Songs

172230

Decade

ALL

Figure 2: Characteristic Correlations and Pie Chart breakdown

Not included in our final Tableau worksheet, the below graphs show the trends of tempo, loudness and duration (seconds).

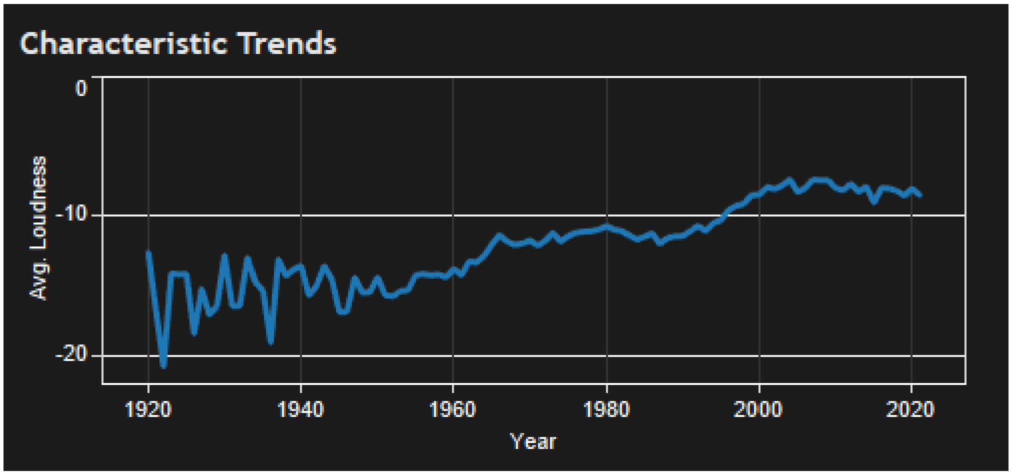


Figure 3: Loudness vs. Year

From the figure 3 above, it is clear that music originally varied greatly in loudness (Decibels), this may have been due to lack of technology and advanced music compression techniques. However loudness steadily increased between 1942 and 2004 but seems to have levelled out over the last decade.

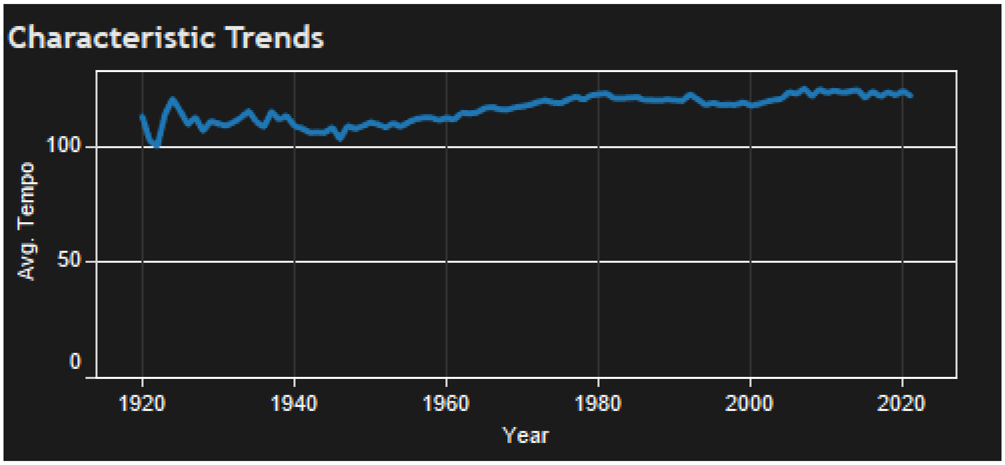


Figure 4: Tempo vs. Year

From figure 4 above, it can be seen that the average tempo has been increasing gradually over the past century although staying steady at around 123 BPM in the last decade. This coincides nicely with the gradual increase in energy levels of music.

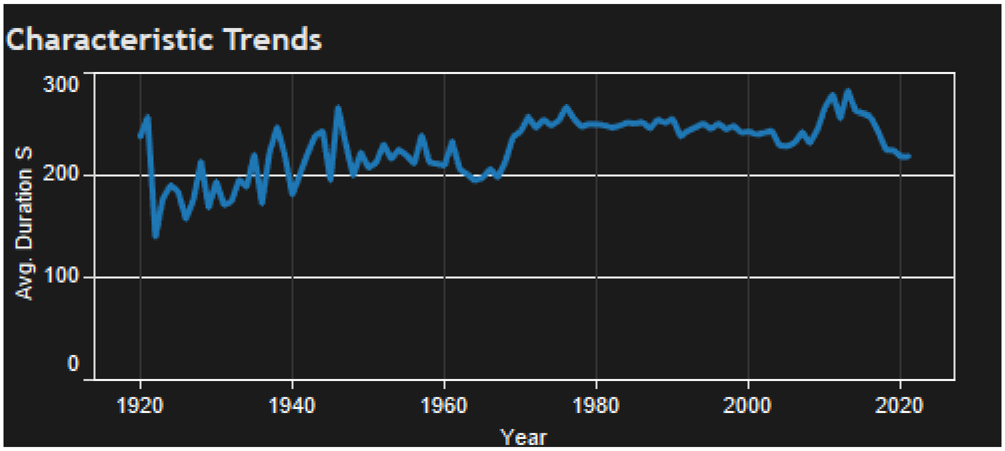


Figure 5: Duration vs. Year

From figure 5 above it was quite interesting to see that music duration in seconds was gradually increasing up until 2010 before decreasing sharply. This visualizes nicely the hypothesis that song length has been decreasing as artists are now paid per stream that is longer than 30 seconds on services such as Spotify.

Related Work

There are many examples of analysis of the Spotify music dataset available on Kaggle such as the analysis available here, as well as examples made as part of Kaggle's "Million Song Dataset Challenge" as well as an extensive song recommender available here. These datasets acted as inspiration after our pivot from popularity analysis to create a simple approach to music recommendations/searching by allowing users to search for recommended songs using the characteristics provided by the dataset.

Challenges and Lessons Learned

There were limitations and lessons learned when we tried to use Cloudera/Hortonworks to analyse some of the data with HiveQL. Being able to access Ambari and Hive on a local machine was useful for visual and learning purposes. Unfortunately, the processing requirements alongside the paywall issues with no response from the Cloudera customer support or sales teams proved to make this experience less than optimal in comparison to the GCP solution. We also attempted to use the Cloudera ODBC Driver to integrate the data extract into Tableau. Unfortunately, some of the features are behind a paywall making this option prohibitive to use for the project. In this case we tested BI Engine for a smooth and fast integration into Tableau. In the end we opted to load our datasets as .csv files into Tableau for ease of use with our Demo however these technologies and methods could be used for future projects

Connecting the recommender script to Tableau was not as straightforward as originally thought and took quite some time to figure out as there is a steep learning curve to connecting external python script to tableau. For example, it took some time to learn that a python script called by Tableau must return either 1 result or the same number of results as arguments. Creation of a button was needed in order to ensure tableau would not call the script every time a parameter is moved. Creation of a button for this purpose is not well documented but after some trial and error was eventually created bettering the user experience of the application.

Attempts were then made to launch the music recommender with Apache Spark by using the pyspark python library however after some research it became clear that this would be difficult as KNN is not supported by pyspark. A thesis was also discovered that made it clear that Spark would provide little to no advantages in terms of speed, this is due to KNN being a lazy learner algorithm. It also became clear that the original idea of a popularity predictor would have been easily supported by pyspark and would have greatly increased speeds as Logistic/Linear Regression and other such Machine Learning algorithms are supported by pyspark and are straightforward to adapt with Scikit-Learn. However the results would have been highly inaccurate.

During data visualization of Music recommender, we noticed that the query was being submitted instantly for each change in parameter by the user. This was affecting the user-experience and was slowing down the overall process. We resolved this challenge by using parameter actions. We connected the original parameters with their duplicate versions, and displayed the duplicates on the screen for the user. Then we involved an Apply Button to trigger query submission once a user is done with all the changes. This improved the user-experience to quite an extent.

Responsibility statement

Name	Individual Mark
Colin Hanily	25
Neal Knauf	12.5
Kay-Ti Tan	12.5
Rana Zeeshan Ali Shahid	25
Darren Rooney	25

Colin - Satisfactory

- Idea Generation
- Created Team Github Repo/OneDrive
- Midway Report Preparation
- Data Cleansing and Pruning with Apache Pig
- Wrote HiveQL queries to create reduced table of song stats per decade
- Created song popularity predictor before being replaced with song recommender
- Created song recommender and connected to Tableau
- Created Initial Draft of Data Visualisation with Tableau
- Assisted with the final report

Darren - Satisfactory

- Idea Generation with Neal
- Created HDFS in Cloudera/Hortonworks
- Wrote HiveQL queries to create reduced tables of popular songs per decade
- Recorded the final video demonstrating the application

Rana

- Basic exploration of Hadoop, Hive, Pig & Spark
- Initial analysis of dataset and trends w.r.t songs popularity
- Data cleansing & Pruning to target Correlations (R-Script)
- Plotted multiple graphs to showcase annual trends of songs' attributes (Tableau)
- Connected the Song Recommender with Tableau
- Plotted the correlations of song attributes for each decade (HeatMap - Tableau)
- Designed & executed the final application (User Interface) for Data Visualization on Tableau
- Assisted with the final report

KT

- Data Cleansing and Pruning
- Set up GCP Dataproc Cluster
- Created HDFS in Hortonworks
- Assisted with the final report

Neal

- Created HDFS in GCP Dataproc
- Created HDFS Cloudera/Hortonworks
- Tested Cloudera HIVE and ODBC driver integration with Tableau
- Tested BigQuery BI engine integration with Tableau
- Set up bi-weekly Zoom meetings
- Assisted with the final report

Response to Peer Feedback

We found the peer feedback to be quite helpful in providing a slightly different viewpoint from our initial plan. We used this feedback to change our initial idea of analysing the criteria of song popularity and creating a song popularity prediction model to instead focusing on using the dataset to analyse the characteristic trends of music over the last decade and create a song recommender based on characteristics inputted by the user. The feedback was very helpful and coincided with our own findings that song popularity analysis/prediction would be unfeasible.

Feedback was also received to look into using Apache Spark for song recommendation model creation with pointers given here. As was previously stated in the Challenges section of this report, after some research, we found that our novel approach of using the KNN classifier to recommend songs based on user inputted characteristics was not supported by Python Spark libraries and that Apache Spark speeds would not be much faster using as was apparent from the thesis available here. Had we chosen to continue with our popularity prediction model, spark could have easily been integrated using pyspark and the multitude of tutorials available online.

The peer feedback also mentioned that https://tunebat.com/ coil be used as it has a library of over 40 million songs. However we felt that 175,000 songs from our database on Kaggle and which provided 19 characteristic fields for each song would be more than suitable/granular.

Appendix 1

Task Breakdown as per Midway Report

Task	Main Contributor
Decide on Dataset	Team
Decide on analysis to undertake	Team
Decide on technologies to use	Team
Midway Report	Colin
Data Trends Exploration	Darren / KT
Cleanse/Prune data	KT
Create HDFS	Neal
Upload HDFS to GCP	Neal
Create relevant queries for data analysis	Darren / Neal
ML model to see how popular user inputted song would be	Colin
Connect HDFS to Data Visualisation	Neal
Connect ML model to Data Visualisation	Colin
Data Visualisation	Rana
Demo Video	Darren
Final Report	Team