Predicting Spotify Audio Features from Last.fm Tags

Jaime Ramírez Castillo[1*] and M. Julia Flores[1†]

[1]Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Campus universitario s/n, Albacete, 02071, Spain.

*Corresponding author(s). E-mail(s): Jaime.Ramirez@alu.uclm.es;
Contributing authors: Julia.Flores@uclm.es;
†These authors contributed equally to this work.

Abstract

In this paper, we discuss a number of experiments to analyze the suitability of music label representations to predict certain audio features, such as danceability, loudness, or acousticness ...

Keywords: Music information retrieval, Artificial intelligence

1 Introduction

Music information retrieval (MIR) is an interdisciplinary research field that encompasses the extraction, processing, and knowledge discovery of information contained in music. MIR research covers a wide range of applications and intersects with other areas, such as computer science, signal processing, musicology, and sociology. Examples of MIR applications are recommendation systems, music classification, music source separation, and music generation, among others [1].

MIR applications often attempt to extract information from the music audio signal, although analyzing associated metadata is also a common practice. Audio signals are typically preprocessed and transformed into intermediate formats, such as frequency-based signal representations (e.g. spectrograms), and sets of hand-crafted audio features, which are typically engineered by using domain knowledge (e.g. MFCC, rhythm, or tonal descriptors).

The metadata associated to a music piece is available in multiple formats. Editorial information or lyrics, for example, are mostly available in text format. The ability to process images or videos is also required, for example, for analyzing album artwork, or music videos.

Depending on the specific MIR application, researchers or practitioners expect different output values. Applications that extract audio features typically return audio descriptors, namely values related to the tempo, the key, or the sample rate, among others. Open source libraries such as *Librosa*[1] and *Essentia*[2] offer methods to extract these values. Other applications might produce higher-level values, for example, by using machine learning techniques that estimate the emotion that a track induces, or the music genre of this track.

Among potentially useful input and output values, research has proved Spotify audio features and Last.fm tags to be significant values to characterize music. Spotify audio features capture high-level information about the music signal, such as energy, danceability, or valence, among others. Last.fm tags are text labels that users associate to songs, artists, and albums via the Last.fm social platform.

Both Spotify audio features and Last.fm tags have been used as input data mostly for classification tasks, such as music genre recognition, where, given a set of Spotify audio features and/or Last.fm tags, the model estimates the music genre(s) of a particular track. Previous studies, however, have not experimented with these values as target outputs, to the best of our knowledge. This unexplored aspect reveals what we believe is a potential research opportunity in music analysis and recommendation.

In particular, this article focuses on predicting Spotify audio features, given a set of Last.fm tags. By predicting Spotify audio features, we explore the relationship between the subjective perception captured by Last.fm tags, and the concrete musical features that Spotify computes. This approach might help to identify patterns and hidden correlations between how music is perceived, consumed, and discovered.

Additionally, the predicted Spotify audio features could be used in recommendation systems to provide users with explainable recommendations. Music recommendations are typically difficult to interpret from the perspective of the listener. Users often get recommendations without meaningful explanations or justifications. By predicting Spotify features as an intermediate step in the recommendation pipeline, we could use these features to explain users why the algorithm suggests a particular track. This process could be part of an explainable recommendation pipeline, where users enter a set of tags, and as a result they get the predicted audio features, the closest tracks to those

[1]https://librosa.org/
[2]https://essentia.upf.edu/index.html

features (as recommended tracks), and the distance values between each track and the predicted features.

In the remainder of the article, we explain the data gathering and preparation process, as well as the data input formats and varios models. We will explore various models for the same track and provide insights on how accurately the prediction can be, by using only Last.fm tags. UNDER DEVELOPMENT

2 Data Sources

2.1 Last.fm Tags

Last.fm is an online music community where users keep track of their music listening habits. Users apply tags to artists, tracks, and albums to categorize and describe music from their own perspective, which implies that the Last.fm tags space does not fit into any structured ontology or data model. A tag can refer to any aspect that users consider to be a valid descriptor, such as genre, emotion, or user listening context.

For nearly two decades, many music aficionados have collectively contributed their own unique, personal interpretation, opinions and feelings as tags in Last.fm. Although many of these tags are single-worded descriptors (e.g *rock*, *dance*, or *happy*), users also use short sentences to describe music, such as *I like this track*, or *on the beach*.

Tags are available via the Last.fm API.

2.2 Spotify Audio Features

Spotify, one of the leaders in the music streaming industry, provides information about the tracks in their catalog via the Spotify developers API. Among the different data entities exposed, the API provides access to the *Audio Features* for each track.

The Spotify audio features are numerical values that represent high-level audio information computed from a specific track. These values characterize a track, musically speaking, by measuring musical aspects that, in many of these features, are related to the user perspective or recommendation factors. For example, a danceability value of 0.95 means that a particular song is highly suitable for dancing.

The features provided by the Spotify API are listed in Table 1. While Spotify provides a description of the audio features. How they compute or estimate these values is not publicly available. The reader can find further details about each feature in the Spotify API documentation[5].

[5] https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

Table 1 Spotify audio features. These features provide high-level musical information about a track.

Feature name	**Description**
acousticness	The track is acoustic. From 0 to 1
danceability	The track encourages (or is adequate for) dancing. From 0 to 1
duration_ms	Duration in milliseconds
energy	The track is perceived as energetic. From 0 to 1
instrumentalness	The track is instrumental. From 0 to 1
key	Key categories encoded as integers. From C (0) to 11
liveness	The audience is audible. From 0 to 1
loudness	In decibels. From -60 to 0
mode	Major (1) or minor (0)
speechiness	Does the track contain speeches? From 0 to 1
tempo	In beats per minute (BPM)
valence	How happy is the track (BPM). From 0 to 1

3 Related Research

The objective of this study is to predict Spotify audio features from Last.fm tags, so the objective of this section is to explore existing literature that is related to the problem and review the facts that are important or relevant for our experimentation.

The use of Last.fm tags and Spotify audio features has been common and prolific in studies that have applied machine learning to resolve MIR challenges.

An important concept to consider is the arousal-valence cuadrants, present in music emotion recognition, related to some studies that will be reviewed in the following sections, and also related to the valence and energy Spotify audio features.

3.1 Last.fm Tags

In the last decade, researchers have studied the use of Last.fm tags in classification and regression tasks. Last.fm tags have been a popular source of metadata for MIR tasks, because they potentially contain subjective information related to the genre, mood, and style of music, and might be used to characterize certain features of a music piece. Additionally, Last.fm tags constitute a useful source of input knowledge when the audio signal is not available, for example, due to copyright limitations.

Several studies have used Last.fm to predict music sentiment, mood, and even audio features. For example, Laurier et al. analyzed how Last.fm tags categorize mood. In their study, they created a semantic mood space based on Last.fm tags [2].

Çano and Morisio performed an analysis of Last.fm tags to create a dataset of music lyrics annotated with Last.fm tags. In the creation process, they concluded that Last.fm tags are mostly related to music genre and positive moods [3].

In a similar direction, Bodó and Szilágyi generated a dataset for lyrics genre classification by combining the Last.fm with *MusicBrainz* data [4]. MusicBrainz is an online database of music editorial metadata[1].

Although different datasets that contain Last.fm tags have been created, so far, the *Last.fm dataset*[2] has been the most widely used in research. This dataset is a complementary dataset of the Million Song Dataset (MSD) [5].

In general, these studies confirm the possibility of extracting knowledge from Last.fm tags.

We introduced this section talking about machine learning but only refeferred to studies that use Last.fm tags in exploration and data set creation

3.2 Spotify Audio Features

Historically, the Spotify audio features features were called the *Echo Nest audio features*. *The Echo Nest* was an online music intelligence platform that provided users and clients with music analysis services. Among these services, the Echo Nest offered a database and an API to retrieve audio features for each of the tracks in the database [3]. Spotify acquired the Echo Nest in 2014. As a result, the Echo Nest API was eventually deactivated and Spotify migrated these audio features to the Spotify API.

Nowadays, the two terms can be found in published research. Whereas the most recent studies refer to the Spotify audio features, earlier studies use the Echo Nest denomination.

Regardless of the term used, the features remain the same. These features are a set of high-level descriptors, such as energy and danceability, which are related to the audio but also to the listeners perception.

Similarly to Last.fm, Spotify (or Echo Nest) audio features are commonly present in MIR research. For example, Wang and Horvát used these audio features to study the differences between male and female artists [6]. And what did they achieve?

Jamdar et al. used Echo Nest audio features, combined with lyrics data to classify songs into emotion tags. These classes were first defined based on a Last.fm tags emotion mapping [7].

Similarly, Non-negative Matrix Factorization was applied in combination with EchoNest audio features for song recommendations [8].

Panda and Redinho explore the use of Spotify high-level features applied to Music Emotion Recognition (MER) [9]. In particular, they identify that the energy, valence, and acousticness values, provided by the Spotify API, are highly relevant for emotion classification. They also achieve better performance on MER models by using their own top-100 features, and they determine that, although these three Spotify features are relevant in terms of characterizing emotion, more features are needed to effectively recognize emotion. Another

[1]https://musicbrainz.org/

[2]Last.fm dataset, the official song tags and song similarity collection for the Million Song Dataset, available at: http://millionsongdataset.com/lastfm.

[3]https://en.wikipedia.org/wiki/The_Echo_Nest

interesting observation of this study is the the identification of energy as a surrogate of arousal valence

In general, Spotify audio features have been used as predictive input variables. We, to the best of our knowledge, are unaware of studies that use these features as target variables, or studies that have addressed the problem of audio features regression, based solely on Last.fm tags.

Similarly to Last.fm tags, Spotify audio features can be found in a number of datasets. Publicly available datasets of Spotify audio features can be found online, as a result of open-source and research communities collecting data from the Spotify API and publishing the results. It is unclear, however, whether these published datasets completely meet the Spotify API terms of service.

For example, *P4kxspotify* is a publicly available dataset that combines music review texts with Spotify audio features. The dataset creators argue that, although the terms of service prohibits scraping, their work is ethical [10].

Another publicly available dataset is the *Spotify Audio Features* Kaggle dataset[4]. This dataset contains more than 116,000 unique tracks, and includes audio features for each track.

There are various reasons why we chose to generate our own dataset. First, we wanted to generate a dataset that combined both Last.fm tags and Spotify audio features. Second, there is lack of clarity regarding the conditions under which Spotify allows the use of the audio features. Third, we wanted to explore a research line where machine learning models are trained by using the listening history of a single user.

4 Generating a Dataset

Before conducting experiments to predict audio features from tags, we constructed a dataset of Last.fm tags and Spotify audio features, indexed by track, by gathering the data from the Last.fm and Spotify APIs.

The tracks were selected from the listening history of a single user.

4.1 A Single-user Dataset

This work is scoped within our single-user research line [11]. In this area, we explore the development of music recommender systems that characterize the music preferences and listening context only for a single user. Therefore, we extracted the data from the listening history of the corresponding author in Last.fm[5].

By training our system in a single-user space, we also raise the following question: Is it possible to train recommender systems, and in particular, user-centric systems, by using a single-user dataset? Additionally, we wanted to explore the idea of mimicking the fact that each human perceives music individually. If we train a system on data from different users, then the system would share the view of multiple individuals.

[4]https://www.kaggle.com/datasets/tomigelo/spotify-audio-features
[5]https://www.last.fm/user/jimmydj2000

Using a single-user data set might sound counterproductive in a machine learning scenario, specially considering how machine learning breakthroughs have attempted, and succeeded in many cases, to generalize in a particular problem. This is not objective of this study, which explores how a machine learning model can represent the music consumption experience of a single human. The model must be able to generalize, but only within the context of the user's musical taste, which be believe can be possible, given a sufficiently large listening history.

4.2 Gathering Listening History and Tags from Last.fm

Last.fm uses the term *scrobble* to refer to the action of playing a track at a specific moment in time. Last.fm started monitoring user listening activity with a desktop application called *Scrobbler*. Users used to install this application on their computers to monitor their activity on players such as Winamp or iTunes. With the advent of music streaming services, the possibilities for users to scrobble their music habits expanded. Integrations where developed to integrate the scrobbler into popular platforms, such as Spotify, YouTube, or SoundCloud. Mobile versions of the Scrobbler were also developed for Android and iOS devices, while open source initiatives flourished too[6].

For us, the first step to construct the dataset was to download the user listening activity. We queried the Last.fm API to download the user's scrobbling logs, reported from 2007 to 2022. For each scrobble, we have gathered the following information:

- Playback timestamp
- Track name
- Artist name
- Track tags

Last.fm maps each track (and artist) to a list of community-contributed tags. For each track-tag mapping, Last.fm includes a *count* value, which indicates the popularity of the given tag for the track. Last.fm normalizes this value in the 0-100 range, so the most popular tag for a track is tipically associated with a count value of 100. For example, if *jazz* is the most popular tag for a track, then the track might be probably associated to the following tuple `(jazz, 100)`.

Users typically listen to their favorite tracks several times, so the amount of unique tracks played is smaller than the number of track plays. In this case, the amount of individual tracks listened in a 15-year period is about 20 000 and the number of scrobblings is, approximately, 90 000. Therefore, the user has listened to each song, approximately, 4.5 times on average.

[6]https://github.com/elamperti/OpenWebScrobbler

4.3 Gathering Spotify Audio Features

After collecting the listening history and track tags from Last.fm, and identifying the unique tracks that represent the user music collection, we used the Spotify API to collect audio features for each one of the 20 000 individual tracks.

The mapping between Last.fm and Spotify tracks was performed on an artist-track basis. For each track, the artist and track name extracted from Last.fm were used as parameters of the Spotify Search API. The Last.fm API provides a unique identifier, the *MusicBrainz* ID. The Spotify API, however, does not provide this value.

4.4 Filtering Missing Values

After retrieving audio features, we identified that the Spotify API had failed to provide audio features for a portion of the tracks. Similarly, Last.fm also returned no tags for another subset of the tracks. To prevent problems with missing values, we decided to filter out these tracks from the dataset. After filtering tracks that were missing Last.fm tags or Spotify audio features, the dataset resulted in 14 009 samples. Compared to the original 20 000 unique tracks included in the listening history, approximately 6000 songs were missing either Spotify or Last.fm data. In other words, about 70% of the tracks in the user listening history included relevant information for the study.

4.5 Dataset Comparison

Considering that the data was gathered from a single user, we explored the data to verify that the distribution of the Spotify audio features was comparable to larger, and possibly more balanced, Spotify datasets. In particular, we verified that the distribution of the features, described in Table 2 and Figure 1, was comparable to the distribution of the Spotify Audio Features Kaggle dataset. Our dataset, which we call *Last.fm Single-user* dataset, presents mean (μ) and standard deviation(σ) values that are comparable to the same values of the Spotify Audio Features Kaggle dataset, as illustrated in Table 3 and Figure 2.

Table 2 Audio features description of the Last.fm Single-user dataset.

Feature	μ	σ
Danceability	0.60	0.19
Energy	0.63	0.23
Acousticness	0.22	0.30
Instrumentalness	0.51	0.38
Valence	0.44	0.28

Table 3 Audio features description of Spotify Audio Features Kaggle dataset.

Feature	μ	σ
Danceability	0.58	0.19
Energy	0.57	0.26
Acousticness	0.34	0.25
Instrumentalness	0.22	0.36
Valence	0.44	0.26

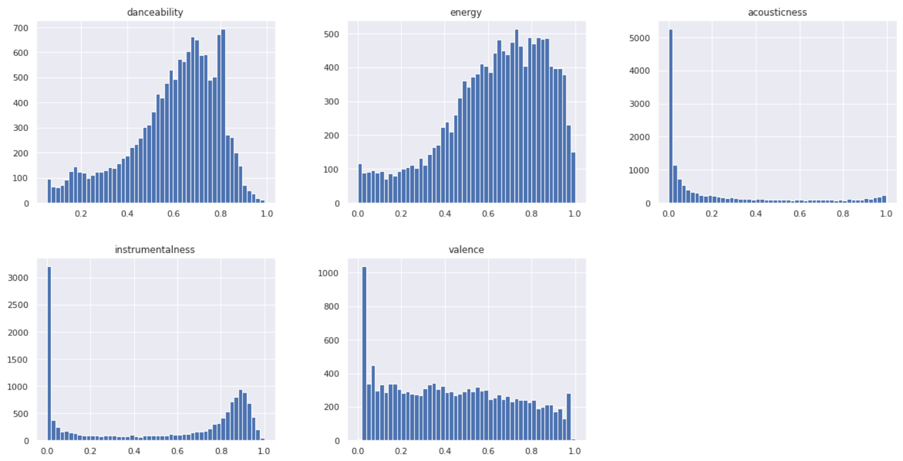

Fig. 1 Distribution of audio features in the Last.fm Single-user dataset.

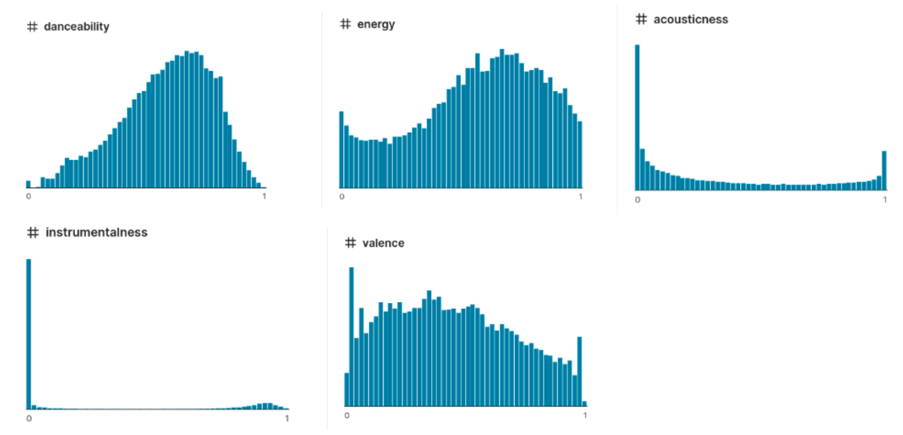

Fig. 2 Distribution of audio features in the Spotify Audio Features Kaggle dataset.

5 Experiments

We trained three commonly used machine learning models to predict Spotify audio features from Last.fm tags by using the Last.fm Single-user dataset. Considering that predicting the audio feature values is a regression problem, we tested the following models:

- Boosted tree regressor [12]
- Bayesian Ridge Regressor [13]
- GPT-2 model (fine-tuning) [14]

5.1 Models

The *Boosted tree regressor* is a specific implementation of the *XGBoost* algorithm for regression tasks. This algorithm is an ensemble learning technique that combines simple multiple decision trees to create a stronger predictive model. This model has proved to be a powerful solution for classification and regression problems on high-dimensional, structured data. REF

The *Bayesian Ridge regressor* is a regression model that combines the principles of Bayesian statistics with ridge regression. Compared to linear regression, which assumes that the model estimated parameters have a deterministic value, the Bayesian Ridge regressor treats the model coefficients as variables with a prior distribution. By incorporating prior distributions into the learning process, the model is able to capture uncertainty. The ridge regression technique enables the model to handle noisy, collinear, structured data.

The *GPT-2* model is a transformer model. Transformers are a type of neural network architecture that has been widely used in natural language processing (NLP) tasks, such as text generation and question answering. GPT-2 has been trained on a large corpus of text, and typically used as a language model to generate text that resembles human-written language. However, in this study we have experimented with GPT-2 as a regressor. Rather than retraining the model from scratch, we have fine tuned the model to behave as a regressor and predict Spotify audio features. The basic idea behind this approach is to feed a string of concatenated Last.fm tags as input to the GPT-2 model, and then use the model's output as the predicted value of a specific Spotify audio feature.

5.1.1 Models Parametrization

Due to the different models and input encodings that we employed in the models, we decided to limit the dimensionality of the experiments, by using mostly the default hyperparameters that the most common libraries set for these models.

For the Bayesian ridge regressor, we used the default hyperparameters set by the *scikit-learn* library. The training parameters for the Bayesian ridge are listed in table 4.

To fine tune the GPT-2 model, we used the *GPT2ForSequenceClassification* model of the *Transformers* Python library. The parameters used are listed in table 5.

Similarly, for the boosted tree regressor, we used the default parameters set in the *xgboost* Python library. We configured the boosted tree regressor model with the training parameters listed in table 6.

5.2 Last.fm Tags Input Format

TODO: Create diagram

The preceding models require specific input formats. In particular, the boosted tree and bayesian ridge regressors require structured data (e.g a set of

Table 4 Training parameters for Bayesian Ridge regressor

Parameter	Value
Maximum iterations	300
Tolerance[1]	1×10^{-3}
alpha 1	1×10^{-6}
alpha 2	1×10^{-6}
lambda 1	1×10^{-6}
lambda 2	1×10^{-6}

[1]Tolerance for the stopping criteria.

Table 5 Training parameters for GPT-2 regressor

Parameter	Value
Batch size	10
Problem type	regression
Epochs	10
Sequence length (tokens)	256
Tokenizer	GPT-2 tokenizer

Table 6 Training parameters for XGBoost regressor

Parameter	Value
objective	reg:squarederror
base score	0.5
booster	gbtree
colsample bylevel	1
colsample bynode	1
colsample bytree	1
gamma[1]	0
learning rate	0.3
max delta step	0
max depth	6
min child weight	1
estimators	200
n jobs	12
num parallel tree	1
predictor	auto
random state	0
reg alpha	0
reg lambda	1
scale pos weight	1
subsample	2
tree method	auto

[1]Tolerance for the stopping criteria.

predictor features and a set of target variables), whereas GPT-2 expects text strings as input.

Each individual sample in the Last.fm singe-user dataset corresponds to a unique track, and contains the list of Last.fm tag-count tuples (e.g. `[(electronic, 100), (dance, 45), ...]`) and the values of Spotify audio features. The Last.fm single-user dataset was converted to the formats described in the following sections, and then fed into the corresponding models.

5.2.1 Last.fm Tags as Table Columns

The tabular format represents the Last.fm tags as columns in a table. Each tag is defined by a column and each cell contains the count value of a tag for a track. If a *tag* is not present for a particular *track*, then cell $c_{track,tag}$ is 0.

Counting the total amount of Last.fm tags in the user collection resulted, initially, in more than five million tags. Under this high-dimensionality scenario, building a tabular data set, in which every row contains millions of columns (i.e. Last.fm tags) was theoretically possible, but presented scalability problems. In addition to scalability limitations, classic machine learning models might not take advantage of using the full collection of tags present in the user history. These models might even perform poorly if too many input features are provided. The reason for this is spurious relations or redundancy between input features. Models might find relations that are not real, and latent, redundant variables might be accountable multiple times, which can lead to a biased outputs. Feature subject selection aims at solving these problems.

Therefore, we reduced the number of tags by picking a subset of the most relevant tags. We used a the feature selection algorithm by using a basic data aggregation algorithm: grouping the data by tag, aggregating by summing the *count* values for each tag, ordering by the aggregated count, and finally selecting the top-K tags of the ranking. Three values of K, were used, thus generating three subsets: 100, 1000, and 10 000. We consider this reduction approach an initial approach approach in our experiments. For this particular aspect, dimensionality reduction algorithms, such as PCA, are good candidates for future work.

After selecting the top-K tags, the Last.fm single-user dataset was formatted as follows:

- Given that $Tags_K$ is the set of most K frequent Last.fm tags in the user listening history , where each $tag \in Tags_K$.
- Given that $Audio$ is the set of Spotify audio features, where each $feat \in Audio$.
- For each $track$:
  - $X_{tag,track}$ is the *count* value of tag for $track$. This value is in the $0 - 100$ range.
  - $y_{track},_{feature}$ is the value of the audio feature y for $track$.

An example of this data format is provided in table 7.

Table 7 Tabular data format for Last.fm tags in XGBoost and Bayesian regressors

Track	$X_{electronic}$	$X_{ambient}$	$X_{...}$	y_{energy}	$y_{valence}$	$y_{...}$
Massive Attack - Blue Lines	62	6	...	0.496	0.947	...
The Beta Band - Squares	40	3	...	0.446	0.507	...
...	...	...	...	...	...	...

The main drawback of this representation is data sparsity. For most of the tracks, many columns are 0.

This format was tested with the Bayesian Ridge and the Boosted Tree regressors.

5.2.2 Tokens as Table Columns

This format is also structured, but the input data is a set of tokens instead of tag count values. These *tokens* are the result of passing the tags, concatenated as a string, through a tokenizer. Tokenizers are crucial elements in the preprocessing of text data. A tokenizer dissects a piece of text into smaller units, called tokens. These tokens can be words, subwords, or even characters.

When breaking down the text into tokens, the tokenizer assigns a unique numerical identifier to each token. These identifiers are based on the vocabulary that the tokenizer has been trained on. For example, when the tokenizer processes the `"pop rock"` string, the `pop` token might be assigned to ID `123` and the `rock` token might be assigned to ID `34534`. Consequently, the result of tokenizing `pop rock` would be `[123, 34534]`.

Note that, although tokenizers are most commonly used in combination with transformer models, in this paper we test the possibility of using a tokenizer to preproccess the data passed to the models that require structured data.

Because the tokenizer requires a string as the input, we converted the set of tags for each track into a string, in a process that we called *stringification*. To *stringify* the tags, we concatenated Last.fm tags by following these three strategies:

- Order by count: `"rock, pop"`.
- Include tag count: `"rock 2, pop 1"`.
- Duplicate tags *count* times: `"rock rock, pop"`.

In this particular case, the X values of the tabular input data are the token IDs. These tokens are obtained from passing the string of concatenated Last.fm tags through the GPT-2 tokenizer, as the following procedure explains:

- Given that S is the stringification strategy.
- Given that X_L is the token vocabulary, where L is the maximum vocabulary length.
- Given that $Audio$ is the set of Spotify audio features, where each $feat \in Audio$.
- For each $track$ and S:
  - $tags_{track,str}$ is the string of concatenated tags produced by strategy S.
  - $tokens_{track}$ is the list of token IDs produced after tokenizing $tags_{str}$.
  - $X_{n,track}$ is token ID found at position n of $tokens_{track}$.
  - $y_{feature,track}$ is the value of the audio feature y for $track$.

An example of this data format is provided in table 8.

Table 8 Tabular data format for tokens in XGBoost and Bayesian regressors

Track	X_0	X_1	X_2	$X_{...}$	y_{energy}	$y_{valence}$	$y_{...}$
Massive Attack - Blue Lines	101	5099	6154	...	0.496	0.947	...
The Beta Band - Squares	101	4522	2600	...	0.446	0.507	...
...	...	...	...	...	...	...	...

Similarly to the tags tabular format, we also defined fixed values for the number of columns: 10, 1,000, and 10,000. Describe the name of this particular limit (maxsequence length?)

Similar to the previous structured format, this format was also used in the Bayesian and Boosted Tree regressors. In contrast to the previous format, this format does not present sparsity. It incorporates however, other problems, such as the lack of normalization of the values, which do not represent scalars, but IDs. It is also harder to inspect, because the meaning of each token is unknown.

5.2.3 Text Strings

When using transformer models, the input data is a string. To use GPT-2, we had to represent the Last.fm tags, which are defined as $(tag, count)$ tuples, as strings. To this end, we applied the same three transformations used in the tabular tokens formats, concatenate the tags by ordering by count, by including the tag count, and by duplicating the tags.

After converting to a string, the formal definition of the input data is as follows:

- Given that X is tags represented as text.
- Given that $Audio$ is the set of Spotify audio features, where each $feat \in Audio$.
- For each $track$:
  - $X_{track,n}$ is set of tags for $track$, encoded as a single string.
  - $y_{track,feature}$ is the value of the audio feature y for $track$.

An example of this data format is provided in table 9.

Table 9 Text data format for GPT-transformer. In this particular case, the tags have been concatenated by ordering by tag count

Track	X	y_{energy}	$y_{valence}$	$y_{...}$
Massive Attack - Blue Lines	"hip hop, chill, bristol, ..."	0.496	0.947	...
The Beta Band - Squares	"alternative rock, folk, ..."	0.446	0.507	...
...	...	...	...	...

what is the limit on the sequence length here?

5.3 Experiments Execution and Results

The experiments tested the three models, and the three values of K: 100, 1000, 10 000. In the Bayesian Ridge and Boosted tree regressors, we tested the two structured data formats: tags and tokens as columns. In the GPT-2 model, we tested the text data format.

For the formats that required the concatenation of tags into text strings, we tested the three different stringification strategies.

The data was split into a training set (8654 samples), a validation set(2164 samples), and a test set (2705 samples). The quality of the model was evaluated by using the root mean squared error (RMSE). This metric was computed on the test set.

Table 10 summaries the experiment results. The table provides RMSE values for each experiment.

Table 10 Experiment results. Cells values correspond to the RMSE value.

M	Input format	Danceab...	Acoustic...	Energy	Valence	Instrumen...
Base		0.276	0.438	0.329	0.395	0.541
Bayes	100 tags[1]	0.159	0.261	0.197	0.243	0.307
Bayes	1000 tags	0.153	0.253	0.190	0.237	0.299
Bayes	10000 tags	**0.152**	**0.251**	**0.189**	**0.236**	**0.297**
Bayes	100 tokens D[2]	0.307	0.307	0.238	0.281	0.383
Bayes	1000 tokens D	0.201	0.315	0.249	0.297	0.399
Bayes	10000 tokens D	0.359	0.507	0.394	0.479	0.613
Bayes	100 tokens O[3]	0.191	0.305	0.237	0.282	0.376
Bayes	1000 tokens O	0.237	0.343	0.276	0.339	0.428
Bayes	10000 tokens O	0.237	0.343	0.276	0.339	0.428
Bayes	100 tokens TC[4]	0.191	0.304	0.236	0.281	0.380
Bayes	1000 tokens TC	0.202	0.320	0.247	0.248	0.404
Bayes	10000 tokens TC	0.234	0.341	0.274	0.321	0.430
Tree	100 tags	0.154	0.257	0.188	0.240	0.302
Tree	1000 tags	0.149	**0.249**	0.184	0.236	0.292
Tree	10000 tags	**0.148**	0.250	**0.181**	**0.235**	**0.291**
Tree	100 tokens D	0.274	0.274	0.212	0.256	0.330
Tree	1000 tokens D	0.173	0.278	0.215	0.268	0.339
Tree	10000 tokens D	0.172	0.276	0.217	0.266	0.342
Tree	100 tokens O	0.179	0.294	0.225	0.271	0.350
Tree	1000 tokens O	0.182	0.294	0.224	0.270	0.353
Tree	10000 tokens O	0.182	0.294	0.225	0.267	0.353
Tree	100 tokens TC	0.172	0.280	0.211	0.262	0.342
Tree	1000 tokens TC	0.174	0.282	0.215	0.268	0.344
Tree	10000 tokens TC	0.175	0.281	0.214	0.270	0.345
GPT	Duplicated[5]	0.157	0.244	0.193	0.245	0.322
GPT	Ordered[6]	0.149	**0.237**	0.188	0.235	**0.297**
GPT	Tags,Counts[7]	**0.145**	**0.237**	**0.187**	**0.233**	0.301

[1]Tags in tabular format. Given $(rock, 3)$, the cell in the *rock* column contains `3`.

[2]Duplicated tokens in tabular format. Tags $(rock, 3), (pop, 2)$, are converted to the `"rock, rock, rock, pop, pop"` string, which a tokenizer converts to the list of input tokens (e.g. `[101,1005,16588,1005,2531, ...]`). These tokens are passed to the model in tabular format. Columns are $token_1$, $token_2$, ..., $token_N$.

[3]Ordered tokens in tabular format. Tags $(rock, 3), (pop, 2)$, are converted to the `"rock, pop"` string, which a tokenizer converts to the list of input tokens (e.g. `[101,1005,16588,1005,2531, ...]`). These tokens are passed to the model in tabular format. Columns are $token_1$, $token_2$, ..., $token_N$.

[4]Tokens in tabular format from tags and counts. Tags $(rock, 3), (pop, 2)$, are converted to the `"'rock' 3, 'pop' 2"` string, which a tokenizer converts to the list of input tokens (e.g. `[101,1005,16588,1005,2531, ...]`). These tokens are passed to the model in tabular format. Columns are $token_1$, $token_2$, ..., $token_N$.

[5]String. Given tags $(rock, 3), (pop, 2)$, input is formatted as `"rock, rock, rock, pop, pop"`.

[6]String. Given tags $(rock, 3), (pop, 2)$, input is formatted as `"rock, pop"`.

[7]String. Given tags $(rock, 3), (pop, 2)$, input is formatted as `"'rock' 3, 'pop' 2"`.

TODO: as you have std deviations for your RMSE, can you say that .291 is really better than .297 ? (idem for other results)

The Spotify feature that the models had less trouble estimating was the danceability, and the energy. The instrumentalness, however, was the feature that presented the highest deviations. Acousticness and valence also presented high RMSE values.

The GPT-2 model was the best model to predict the danceability, with a 0.145 RMSE. The Boosted tree was the best model to predict the instrumentalness, with a RMSE value of 0.29.

In general, the three models performed similarly, with GPT-2 achieving slightly better results in danceability, acousticness, and valence. The Boosted Tree regressor was the best model to predict energy, and instrumentalness.

The two regressors achieved the best results when using the *Tags as columns* format, mostly with 10 000 tags. Interestingly, the Boosted Tree regressor achieved better results with tokens when estimating valence. Even better results than GPT-2. In the rest of the features, the regressors worked better with tags as columns.

Using a higher number of predictor variables was not a synonym of lower error values when using tokens a columns.

The Bayesian regressor was a bit more sensitive to changes in tag formatting. This model tended to work better with data formatted by using the Tag Count stringification method. The Boosted Tree regressor presented less sensitivity to format changes, although the model generally performed better with the Tags as columns format too.

The Duplicated stringification method consistently performed worse with the GPT-2 transformer model.

5.3.1 Results for Tabular Data Models

6 Conclusions

In general, we believe that this novel approach that has the potential to benefit both listeners and researchers. By combining subjective user-generated tags with objective audio features, we can gain new insights into the complex relationship between perception and audio signal in music.

Our approach also presents limitations. One limitation is the assumption of a strong relationship between Last.fm tags and Spotify features, which may not be true in all cases. Future work could explore other sources of input values, possibly related to the user context, to improve the accuracy of the predictions.

Track mapping between Last.fm and Spotify is another opportunity for research and improvement. Although Last.fm provides the MusicBrainz unique identifier, but Spotify does not provide this value. The mapping was performed by using the track artist and name, but this approach resulted on about 30% of the tracks not found on Spotify.

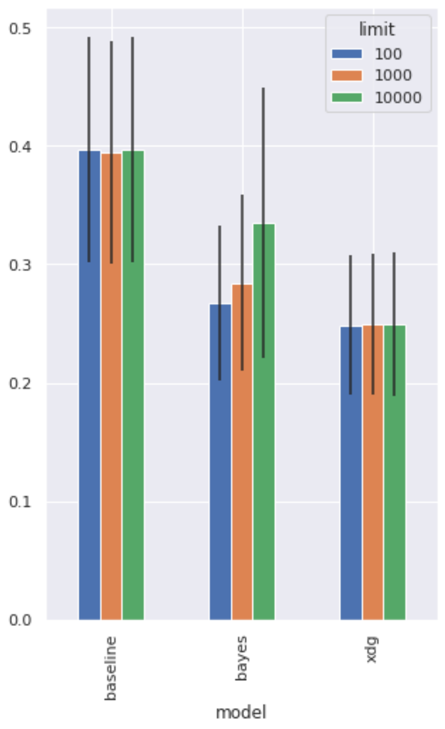

Fig. 3 RMSE mean and standard deviation by model and tags/tokens limit.

We consider this reduction approach an initial approach approach in our experiments. For this particular aspect, dimensionality reduction algorithms, such as PCA, are good candidates for future work.

TODO

7 Acknowledgments

TODO

References

[1] Ramirez, J., Flores, M.J.: Machine learning for music genre: multi-faceted review and experimentation with audioset. Journal of Intelligent Information Systems **55**(3), 469–499 (2020) https://doi.org/10.1007/s10844-019-00582-9

[2] Laurier, C., Sordo, M., Serra, J., Herrera, P.: Music mood representations from social tags. In: ISMIR, pp. 381–386 (2009)

[3] Çano, E., Morisio, M., *et al.*: Music mood dataset creation based on last. fm tags. In: International Conference on Artificial Intelligence and

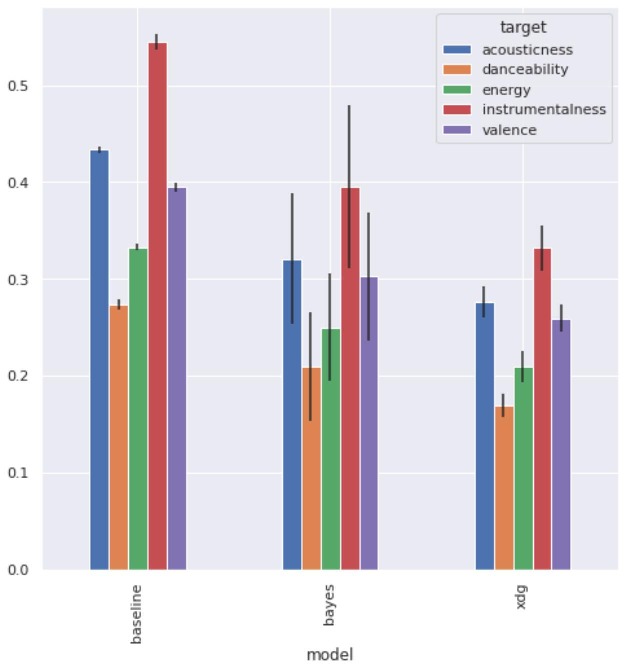

Fig. 4 RMSE mean and standard deviation by model and audio feature.

Applications, Vienna, Austria, pp. 15–26 (2017)

[4] Bodó, Z., Szilágyi, E.: Connecting the last. fm dataset to lyricwiki and musicbrainz. lyrics-based experiments in genre classification. Acta Universitatis Sapientiae, Informatica **10**(2), 158–182 (2018)

[5] Bertin-Mahieux, T., Ellis, D.P.W., Whitman, B., Lamere, P.: The million song dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR) (2011)

[6] Wang, Y., Horvát, E.-Á.: Gender differences in the global music industry: Evidence from musicbrainz and the echo nest. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 13, pp. 517–526 (2019)

[7] Jamdar, A., Abraham, J., Khanna, K., Dubey, R.: Emotion analysis of songs based on lyrical and audio features. arXiv preprint arXiv:1506.05012 (2015)

[8] Benzi, K., Kalofolias, V., Bresson, X., Vandergheynst, P.: Song recommendation with non-negative matrix factorization and graph total variation. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2439–2443 (2016). Ieee

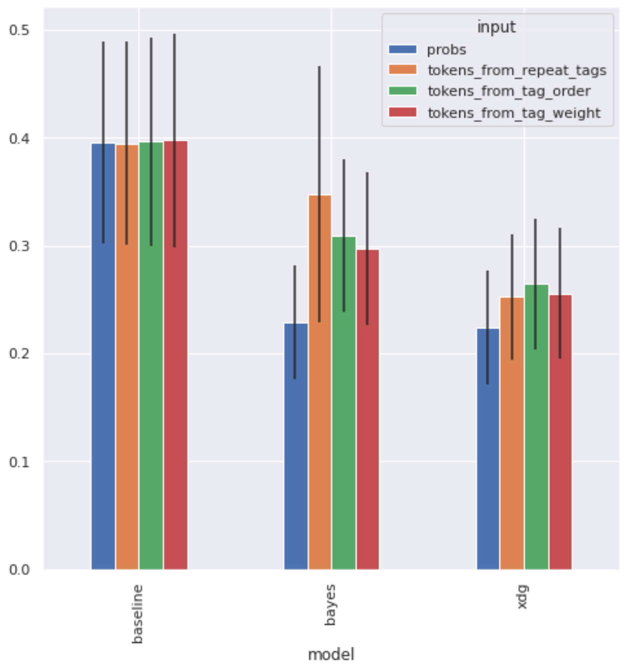

Fig. 5 RMSE mean and standard deviation by model and input type (tag probablities or tokens).

[9] Panda, R., Redinho, H., Gonçalves, C., Malheiro, R., Paiva, R.P.: How does the spotify api compare to the music emotion recognition state-of-the-art? In: 18th Sound and Music Computing Conference (SMC 2021), pp. 238–245 (2021)

[10] Pinter, A.T., Paul, J.M., Smith, J., Brubaker, J.R.: P4kxspotify: A dataset of pitchfork music reviews and spotify musical features. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 14, pp. 895–902 (2020)

[11] Ramirez-Castillo, J., Flores, M.J., Nicholson, A.E.: User-centric music recommendations. In: 16th Bayesian Modelling Applications Workshop, Conference on Uncertainty in Artificial Intelligence, Eindhoven, The Netherlands) (2022)

[12] xgboost: Xgboost. https://xgboost.readthedocs.io/en/stable/tutorials/model.html

[13] Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1**, 211–244 (2001)

[14] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.:

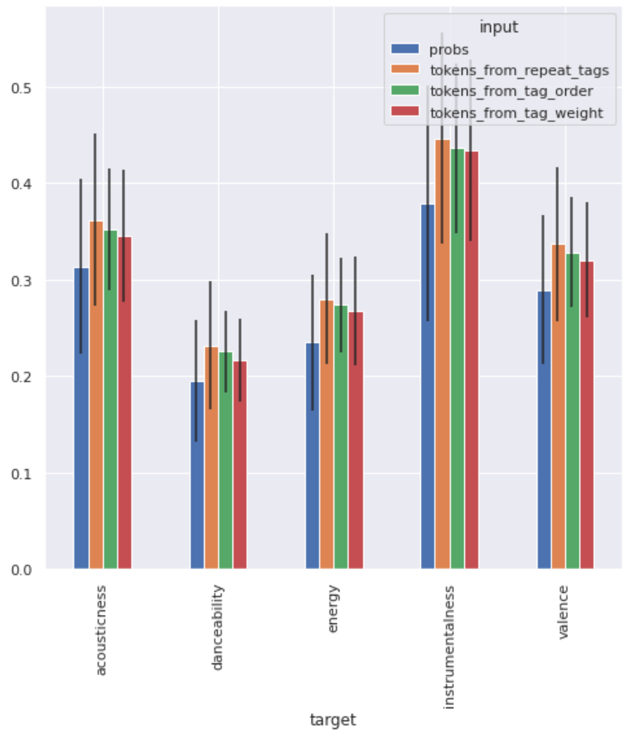

Fig. 6 RMSE mean and standard deviation by audio feature and input type (tag probablities or tokens).

Language models are unsupervised multitask learners (2019)