Semester Project Report

Department Of Data Science and Engineering

Emotion Patterns in Music Playlists

Authors:
Sara Giammusso
Mario Guerriero

Supervisors:
Raphael Troncy
Pasquale Lisena
Enrico Palumbo

June 17, 2018

EURECOM
Sophia Antipolis

Contents

Abstract

Music streaming services such as Spotify are revolutionizing the music world, enabling a transition from artist-created bundles of songs (CDs) to user-created playlists.

Different logics may be applied in the generation of a playlist: they can contain songs of a similar genre (e.g. "Rock playlist"), fit to a particular occasion (e.g. "New year's eve party"), to a particular context (e.g. "Gym"), to a particular mood (e.g."Happy") and so on.

The goal of this semester project is to unravel the emotion patterns underlying the sequences of songs in a playlist using automatic approaches of Emotion Detection on the lyrics.

Chapter 1

Introduction

1.1 Background

In the last few years the online music streaming services such as Spotify, Apple Music and Deezer introduced, among others, the possibility to create playlists thus opening new challenges on music recommendation.

One of the new possible tasks a modern Recommender System should perform is automatic playlist continuation. By suggesting appropriate songs to add to a playlist, a Recommender System can increase user engagement by making playlist creation easier, as well as extending listening beyond the end of existing playlists.

More precisely the task of automatic playlist continuation consists in: given a set of playlist features, the system shall generate a list of recommended tracks that can be added to that playlist, thereby "continuing" the playlist.

In fact, one of the final goals of this project was to produce numerical features to be used in a recommender system for music playlists. Specifically, this recommender system had to be built for the RecSys Challenge 2018[1], hosted for this year by Spotify, which provided a huge dataset of a milion playlist coming from their own platform. Since the goal of this challenge is to improve the playlist continuation feature of the Spotify's recommender system, we believed that putting emotions in the features used to group playlists could be useful for achieving better performances. For this reason, the output of this system is a vector where, each element, expresses the

[1] `https://recsys-challenge.spotify.com/`

likelihood score for the releated emotion. The details of how this vector is built will be made clearer throughout this report.

1.2 Project scope

One of the possible features to consider in developing a system for the automatic playlist continuation task is the emotion expressed in each song of the playlist and the more frequent transition patterns from one emotion to the other. Thus, not only the emotion of each song lyrics must be detected, but also the transitions between emotions must be analyzed.

Emotion Detection is a novel and promising field of study of Natural Language Understanding, which is able to automatically infer what are the emotions expressed in a text. It can be considered as a Sentiment Analysis task, which is the computational treatment of opinions, sentiments and subjectivity of a natural language text.

1.3 Results

Below we just reported the accuracy results we obtained throughout our project while classifying emotions both in song lyrics and in playlists, as shown in table 1.1 and 1.2 respectively.

10-fold Cross Validation Accuracy				
Dataset	ANN	LR	SVM	xgboost
MoodyLyrics	90.55%	-	90%	86%
MoodyLyrics4Q	55.97%	57.87%	58.04%	56.89%
Both together	68.41%	69.42%	69.32%	64.27%

Table 1.1: Emotion detection accuracies

Playlist Classification Accuracy		
Dataset	With outliers	Without outliers
MoodyLyrics	29%	29%
MoodyLyrics4Q	66%	66%
Both together	50%	47%

Table 1.2: Playlist classification accuracies

We will go into the details of how those results were obtained in the next chapters of this report.

1.4 Emotion Classification Utilities

The above results can be obtained using the code which can be found in the project's GitHub repository[2]. Specifically, using this code, we can perform classification at three different levels:

- **Lyric level**: using the function `classify(sid, artist, title)` of the `srcemoclassify.py` script
- **Playlist level**: using the function `robust_classify(playlist_vect)` from the `srcplaylist_classify.py` script
- **Spotify's dataset slice level**: using the function `classify_slice(slice_path)` from the `srcslice_classify.py` script

Along with those utilitiy programs we also provided some pre-trained models in the folder `srcmodel`, which can be used to reproduce the results we obtained. The details of how those models were built will be discussed later in this report.

1.5 Report outline

The report is structured as follow: *chapter 2* contains the analysis of the state of the art in emotion detection tasks. *Chapter 3* includes an exploration of

[2] `https://github.com/sgiammy/emotion-patterns-in-music-playlists`

the background knowledge for this project, *chapter 4* describes the lyrics emotion classification approaches tried, *chapter 5* the playlists classification methods and results while *chapter 6* contains conclusions and future works considerations.

Chapter 2

The State of the Art

In the following chapter we will briefly go through the knowledge we acquired while exploring already existing materials in the same context of our problem.

2.1 Sentiment Analysis

Sentiment Analysis (SA) is the computational study of people's opinions, attitudes and emotions toward an entity, the entity being an individual, event or topic. [27]

Sentiment Analysis can be considered a classification task as illustrated in Fig 2.1.

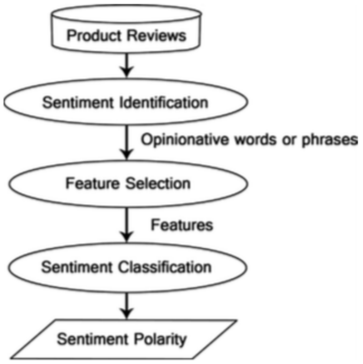


Figure 2.1: Sentiment analysis process on product reviews [27]

There are three main classification levels in SA:

- Document-level
- Sentence-level
- Aspect-level

Their difference is the granularity at which they operate. Indeed, while document-level SA aims to classify a document as expressing a positive or negative opinion or sentiment by considering the whole document as the basic information unit, sentence-level SA aims at classifying the sentiments/opinions expressed in each sentence. In both cases the first step is to identify whether the sentence/document is subjective or objective and if it is subjective determine whether the sentence expresses positive or negative opinions.

In certain applications, classifying text at the document level or at the sentence level may not provide the necessary details needed for detecting opinions on all aspects of the entity. Aspect-level SA, instead, aims at classifying the sentiment with respect to the specific aspects of entities. The first step is to identify the entities and their aspects. Then, all the different opinions on the same entity must be considered. Indeed, the opinion holders may also give different opinions for different aspects of the same entity, e.g. "This chair is ugly but it is comfortable".

Sentiment Analysis task is considered as a sentiment classification (SC) problem. The first step in the SC problem is to extract and select text features. Some of the most commonly used features are:

- **Terms presence and frequency**: These features are individual words or word n-grams and their frequency counts;
- **Parts of speech (POS)**: finding adjectives, pronouns, etc. as they are important indicators of opinions;
- **Opinion words and phrases**: these are words commonly used to express opinions including *good or bad, like or hate.* On the other hand, some phrases express opinions without using opinion words, e.g. *cost me and arm and a leg*;
- **Negations**: the appearance of negative words may change the opinion orientation like *not good* is equivalent to *bad.*

Sentiment Classification techniques can be roughly divided into machine learning approach, lexicon based approach and hybrid approach [27].

The *Machine Learning Approach (ML)* applies the famous ML algorithms and uses linguistic features.

The *Lexicon-based Approach* relies on a sentiment lexicon, a collection of known and precomiled sentiment terms. It is divided into dictionary-based approach and corpus-based approach which use statistical or semantic methods to find sentiment polarity.

The *Hybrid Approach* combines both approaches.

The various approaches and the most popular algorithms of SC are illustrated in Fig 2.2.

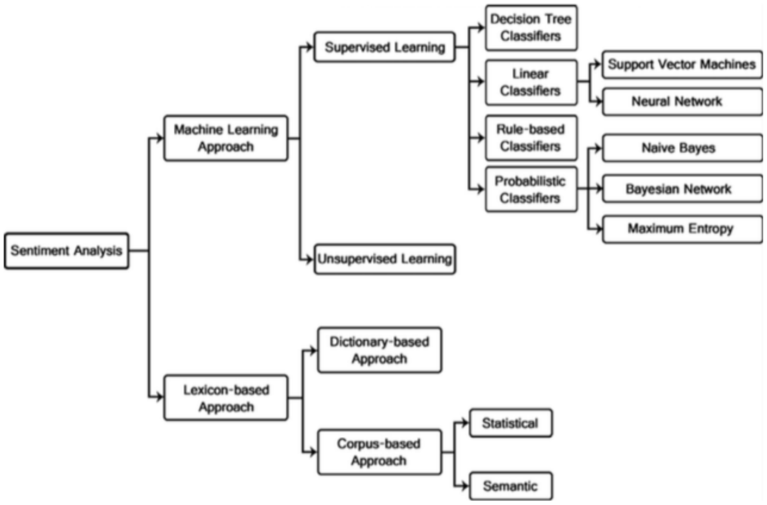


Figure 2.2: Sentiment classification techniques [27]

The text classification methods using ML approach can be roughly divided into supervised and unsupervised learning methods. The supervised methods make use of a large number of labeled training documents. The unsupervised methods are used when it is difficult to find these labeled training training documents.

The lexicon-based approach depends on finding the opinion lexicon which is used to analyze the text. There are two methods in this approach:

- Dictionary-based

- Corpus-based

In the Dictionary-based approach a small set of opinion words is collected manually with known orientations. Then, this set is grown by searching their synonyms and antonyms. The newly found words are added to the seed list then the next iteration starts. The iterative process stops when no new words are found. The dictionary based approach has a major disadvantage which is the inability to find opinion words with domain and context specific orientation.

The limitation of the dictionary-based approach is addressed by the corpus-based approach, which depends on syntactic patterns or patterns that occur together along with a seed list of opinion words to find other opinion words in a large corpus. One of these methods is called *sentiment consistency*: it starts with a list of seed opinion adjectives, and used them along with a set of linguistic constraints to identify additional adjective opinion words and their orientations. The constraints being for example *AND, OR, BUT, EITHER-OR,...*; the conjunction *AND* for example says that conjoined adjectives usually have the same orientation.

2.2 Emotion Detection

Emotion detection (ED) is the process of identifying human emotions. It is a recent field of research that is closely related to Sentiment Analysis. Indeed, Sentiment Analysis aims to detect positive, neutral or negative feelings from text, whereas Emotion Analysis aims to detect and recognize feelings in natural language texts. Therefore we can look at ED as a finer grained task with respect to SA.

Emotion is expressed as joy, sadness, anger, surprise, hate, fear and so on. Since there is not any standard emotion word hierarchy, the focus is on the related research about emotion in cognitive psychology domain. In 2001, W. Gerrod Parrot, wrote a book named “Emotions In Social Psychology” [23], in which he explained the emotion system and formally classified the human emotions through an emotion hierarchy in six classes at primary level which are *Love, Joy, Anger, Sadness, Fear and Surprise* [26].

Emotion detection may have useful applications, such as [17]:

- Measure citizens happiness;

- Pervasive computing: this may include suggesting help when anxiety is detected through speech, or to check the tone of an email;
- Improving perception of a customer to increase brand reputation and sales.

Some of the biggest challenges in determining emotion are:

- *Context-dependence of emotions*: people use different regulation strategies in different social contexts. A phrase can have element of *anger* without using the word "anger" or any of its synonyms, e.g. *"Shut up!"*
- *Word-sense disambiguation*: identifying which sense a word (i.e. its meaning) is used in a sentence, when the word has multiple meanings;
- *Co-reference resolution*: pronouns and other referring expressions must be connected to the right individuals;
- Lack of labelled emotion databases.

The main methods used for text based emotion detection are:

- *Keyword Spotting*
- *Lexical Affinity*
- *Learning-based*
- *Hybrid*

Keyword Spotting The keyword pattern matching problem can be described as the problem of finding occurrences of keywords from a given set as substrings in a given string. These words are classified into categories such as disgusted, sad, happy, angry, fearful, surprised, etc. The process of Keyword spotting method is shown in Fig 2.3.

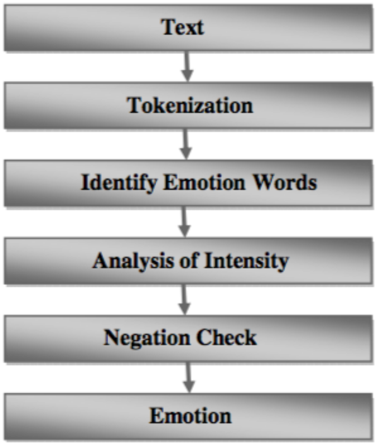


Figure 2.3: Keyword spotting technique

The first step is converting data into tokens, i.e. a sentence into words, then from these tokens emotion words are detected. The second step is analyzing the intensity of emotion words. Sentence, then, is checked whether negation is involved in it or not then finally an emotion class will be assigned.

Lexical Affinity method The Lexical Affinity approach is an extension of keyword spotting technique: apart from picking up emotional keywords it assigns *probabilistic affinity* for a particular emotion to arbitrary words. This technique has the main disadvantage of missing out emotion content that resides deeper than the word level.
For example the word 'accident', having been assigned a high probability of indicating a negative emotion, would not contribute correctly to the emotional assessment of phrases like *"I avoided an accident"* or *"I met my girlfriend by accident"*.

Learning-based methods Learning-based methods change the focus from "determining emotions" to "classify the input texts into different emotions". Indeed, learning-based methods try to detect emotions based on a previously trained classifier, which apply various theories of machine learning such as Support Vector Machines (SVMs).

Hybrid Methods Since keyword-based methods and naïve learning-based methods could not acquire satisfactory results, some systems use hybrid approach by combining both keyword spotting technique and learning based method, which help to improve accuracy.

However all these methods have some major limitations:

- *Ambiguity in Keyword Definitions*: words can have multiple and vague meanings that can change according to different usages and contexts. Moreover emotion labels could have different emotions in some extreme cases such as ironic or cynical sentences;

- *Lack of Linguistic Information*: these methods totally ignore syntax structures and semantics that also have influences on expressed emotions. For example the sentences *"He laughed at me"* or *"I laughed at him"* express two totally different meanings;

- *Incapability of Recognizing Sentences without Keywords*: sentences without any keyword would imply that they do not contain any emotion at all, which is obviously wrong.

Deciding a way to label emotions is another challenging aspect of ED. There are mainly two possible ways to label data [17]:

1. The label is one between the set of emotions, e.g. *anger, disgust, sad, happy, surpise, fear, neutral*;

2. *Slider approach*: the label is composed of percentages for each emotion, as described in Fig 2.4.

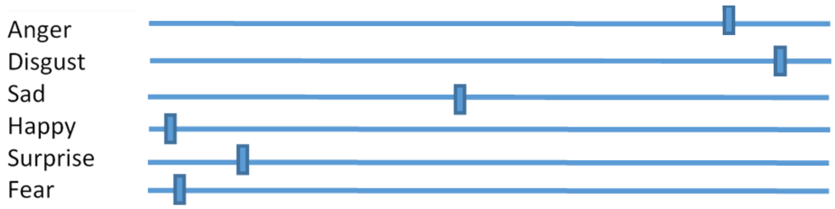


Figure 2.4: Slider approach [17]

The sliders approach certainly offers more information but it comes with additional computational complications. However, the probabilistic assignment produced in the sliders approach can be turned into distinct labels of single emotions, just like those produced by first labeling approach, but of course, we can not move from the first approach to the second one.

Chapter 3

Problem Background

In this chapter we will analyze what were the already existing systems and tools we found interesting when we started our project.

3.1 The problem

The goal of this semester project is to unravel the emotion pattern underlying the sequences of songs in a playlist using automatic approaches of Emotion Detection on the lyrics.

The problem can be divided into two main parts:

1. Classify emotions of songs based their lyrics
2. Analyze emotion patterns in the playlists

3.2 Related work

Emotion detection domain has already attracted many researchers from computer science, psychology, cognitive science and so on.

Before building our own emotion detection system we started by analyzing some of the already existent classifiers.

IBM Watson Natural Language Understanding [5] Watson is a question answering computer system capable of answering questions posed in natural language, developed by IBM.

Natural Language Understanding is a collection of APIs that allows to:

- Recognize the overall sentiment, in a scale from negative to positive [-1, 1];

- Detect the emotion percentage between: joy, anger, disgust, sadness, fear;

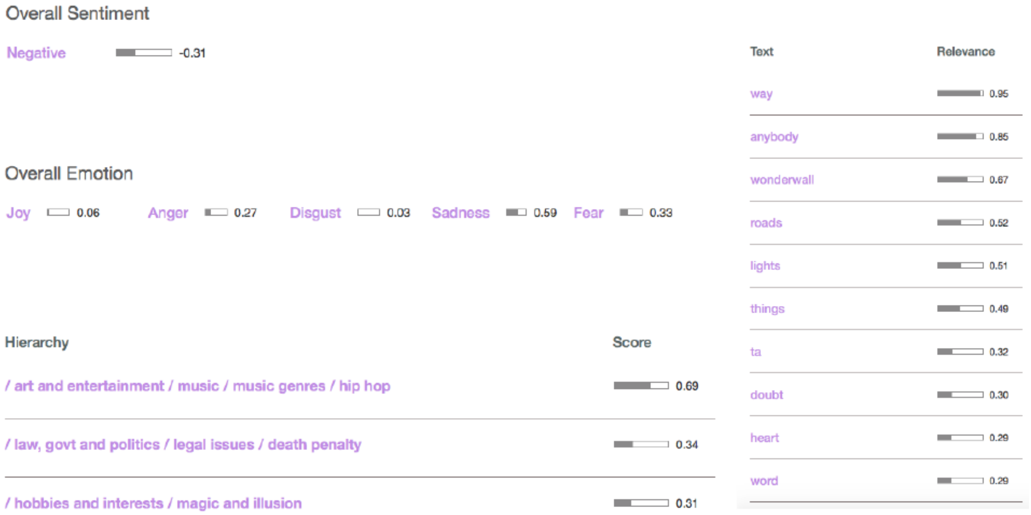
- Determine keywords ranked by relevance;

- Extract entities: people, companies, organizations, cities and other information;

- Classify content into hierarchical categories;

- Identify general concepts that may not be directly referenced in the text;

An example of IBM Watson NLU output is shown in figure 3.1.



Overall Sentiment

Negative -0.31

Overall Emotion

Joy 0.06 Anger 0.27 Disgust 0.03 Sadness 0.59 Fear 0.33

Hierarchy	Score
/ art and entertainment / music / music genres / hip hop	0.69
/ law, govt and politics / legal issues / death penalty	0.34
/ hobbies and interests / magic and illusion	0.31

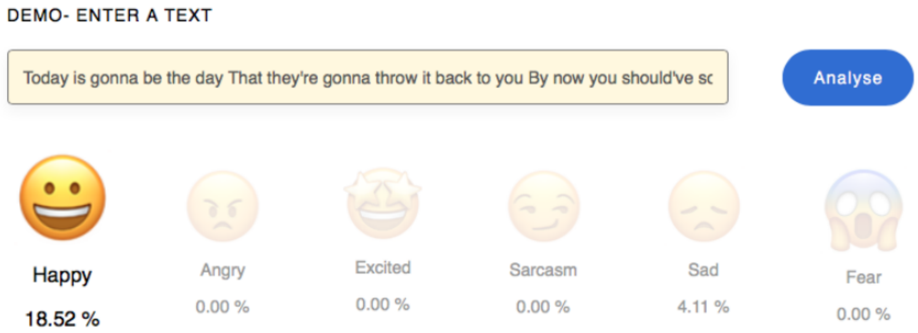
Text	Relevance
way	0.95
anybody	0.85
wonderwall	0.67
roads	0.52
lights	0.51
things	0.49
ta	0.32
doubt	0.30
heart	0.29
word	0.29

Figure 3.1: IBM Watson Natural Language Understanding output while analyzing the lyric of Wonderwall, by Oasis

IBM Watson Tone Analyzer [6] It uses linguistic analysis to detect joy, fear, sadness, anger, analytical, confident, and tentative tones found in text. It allows to select different sources: tweets, online reviews, email messages, or other text. It uses both:

- the document level: to get a sense of the overall tone;
- the sentence level: to identify specific areas where tones are the strongest.

ParallelDots API [10] It uses an Emotion Analysis classifier trained on a proprietary dataset and tells whether the underlying emotion behind a message is: Happy, Sad, Angry, Fearful, Excited, Funny or Sarcastic. An example of its output is shown in figure 3.2.



DEMO- ENTER A TEXT

Today is gonna be the day That they're gonna throw it back to you By now you should've sc

Happy	Angry	Excited	Sarcasm	Sad	Fear
18.52 %	0.00 %	0.00 %	0.00 %	4.11 %	0.00 %

Figure 3.2: ParallelDots API applied on the lyrics of Wonderwall, from Oasis

The results obtained with ParallelDots API and IBM Watson are quite different. Moreover we should consider that we don't know how the two tools internally parse the lyrics, i.e. how they do consider line breaks, line breaks without commas, etc.

QEmotion [11] Qemotion detects the main emotion of the speech and define the corresponding emotion in terms of temperature:

- From 31° to 40° → Happiness
- From 21° to 30° → Surprise
- From 11° to 20° → Calm
- From 6° to 10° → Fear
- From −5° to 5° → Sadness
- From −14° to −6° → Anger
- From −20° to −15° → Disgust

3.3 NLP libraries

In order to select the best Natural Language Processing library for our purpose we also analyzed pros and cons of the main Natural Language Processing libraries available, i.e. NLTK, TextBlob, Standord's CoreNLP and SpaCy.

NLTK: Natural Language Toolkit NLTK is a leading platform for building programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries [8]. It is recommended only as an education and research tool.
Pros:

- its modularized structure makes it excellent for learning and exploring NLP concepts;
- over 50 corpora and lexicons, 9 stemmers, and a dozens of algorithms to choose from (this can also be considered as a con).

Cons:

- Heavy library with a steep learning curve;
- Slow and not production-ready.

TextBlob Built on top on NLTK [15].
Pros:

- More intuitive;
- Gentle learning curve.

Stanford's CoreNLP Stanford CoreNLP provides a set of human language technology tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get the quotes people said, etc. [14] It is Java

library with Python wrappers.
Pros:

- fast;
- support for several major languages.

SpaCy It is a new NLP library designed to be fast, streamlined and production-ready [13].
Pros:

- minimal number of options;
- its philosophy is to only present the best algorithm for each purpose.

Cons:

- it is new, so its support community is not as large as other libraries, but it is growing very fast.

3.4 Word embedding techniques

Word embeddings are a set of feature learning techniques mapping words or phrases from the vocabulary to vectors of real numbers.

These techniques map sparse word vectors into continuous space based on the surrounding context. For example if *"salt"* and *"seasoning"* appear within the same context, the model will indicate that *"salt"* is conceptually closer to *"seasoning"*, than another word, say *"chair"*.

There are three main embedding techniques: Word2vec [16], GloVe [24] and FastText [4].

The first two approaches treat each word in corpus like an atomic entity generating a vector for each of them. Specifically, word2vec tries to learn a vector representation of word occurring in similar contexts by using feedforward neural networks. On the other hand, GloVe, is a count-based model, which means that it firstly constructs a large co-occurrence matrix for the terms it finds, and then it factorizes (reduces the dimensionality) of this matrix to obtain fixed size vectors.

FastText, instead, treats each word as composed of character ngrams, so the vector for a word is made of the sum of this character n-grams [4].

For example, for FastText, the word vector for *"apple"* is a sum of the vectors of the n-grams "ap", "app", "appl", "apple", "ppl', "pple", "ple", "le" assuming 3 and 6 as minimum and maximum ngrams size.
The difference between Word2vec/GloVe and FastText manifests as follows:

1. *Rare words*: even if words are rare, their character n-grams are still shared with other words - hence the embeddings with FastText can still be good;

2. *Out of vocabulary words*: FastText can construct the vector for a word from its character n-grams even if word does not appear in training corpus;

3. *Hyperparameters choice*: FastText requires to choose the minimum and maximum n-grams sizes, and this directly impacts the computation time and the memory requirements.

3.5 Tools Choice

Given the considerations expressed above, for our project we decided to use SpaCy and an already trained language model provided by the same developers, built using GloVe.

SpaCy was choosen among the other analyzed libraries for its efficiency and for its simplicity.

GloVe, instead, was choosen because we did not believe that we needed all the complications introduced in FastText. Specifically, the language model we used provides 685k unique 300-dimensional word embedding vectors, which were trained on natural language text coming from OntoNotes 5 [9] and Common Crawl [2].

Chapter 4

Emotion Classification Approaches

In this chapter we will go through the experiments we made for this project. Specifically, we will show how we tried to build some first models, solely based on the MoodyLyrics dataset, with no particular preprocessing nor feature engineering techniques, just by using the tools we already mentioned. Then we will move to more complex feature engineering techniques and classifiers.

Along with that, we will also provide a formal validation of the POS tagger tool we used.

4.1 Public datasets

A big challenge in emotion detection is the lack of a labelled emotion database to enable active innovation. Currently, few publicly accessible databases are available.

MoodyLyrics [19] contains around 2500 songs manually annotated through Amazon Mechanical Turk with 4 different emotion labels, i.e., happy, sad, angry and relaxed.

EmoInt [22] contains manually annotated tweets classified according to the intensities of anger, fear, joy and sadness. *EmoBank* [3] instead contains 10.000 sentences, each of which has been annotated according to both the emotion expressed by the writer and the emotion perceived by the reader.

Certainly, the most appropriate dataset for our problem among those we found is MoodyLyrics.

4.1.1 MoodyLyrics

As already mentioned, MoodyLyrics is a manually annotated dataset of 2595 songs labeled according to 4 different emotion labels: happy, sad, angry and relaxed.

MoodyLyrics creators used content words of lyrics and their valence and arousal norms to assign songs with those labels. In particular, they used Russell's Valence-Arousal model with the 4 mood categories described for the annotation process [25]. Valence, which describes the pleasant-unpleasant degree, and Arousal, which describes aroused-sleepy degree, values of songs are computed adding the corresponding values of each word of lyrics that is found in a lexicon which was build by combining ANEW (Affect Norm of English Words), WordNet and WordNet-Affect.

In the context of MoodyLyrics, Valence represents the positive or negative intensity of an emotion whereas Arousal indicates how strongly or rhythmically the emotion is felt [19].

Based on this assumptions, MoodyLyrics creators classified lyrics according to the planar model shown in figure 4.1, which relates arousal and valence.

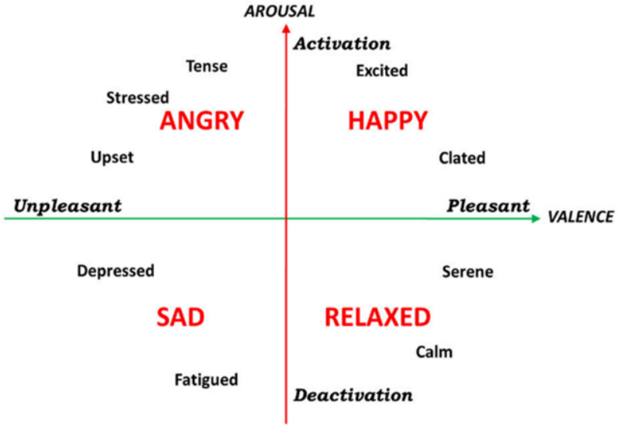


Figure 4.1: Valence and Arousal planar model used by MoodyLyrics creators to annotate lyrics [19]

MoodyLyrics duplicates bug

During the analysis of MoodyLyrics described in the previous section we detected the presence of duplicated songs inside the dataset. Moreover, sometimes different emotions were associated with the duplicated songs. Thus, to continue our analysis we eliminated duplicated rows and we chose as emotion label the most frequent emotion between all the duplicates.

This bug in the dataset was reported to MoodyLyrics creators. As a response to our bug report email, MoodyLyrics creators suggested to use MoodyLyrics4Q [18], that, according to them is a more accuratelly labeled version of MoodyLyrics.

This advice opened us three possibilities: continue using MoodyLyrics, start using MoodyLyrics4Q or create a new dataset as the concatenation of the previous two. We decided to start using all these three models, in order to understand which one, at the end, will give us a better playlists classification. The complete MoodyLyrics emotion classification analysis can be found at Notebook 1, while the MoodyLyrics4Q and the emotion detection analysis in the merged datasets can be found at Notebook 2.

4.1.2 MoodyLyrics4Q

MoodyLyrics4Q contains 2000 songs and has the same annotation schema as MoodyLyrics. Figure 4.2 shows the emotions distribution comparison between the two MoodyLyrics versions.

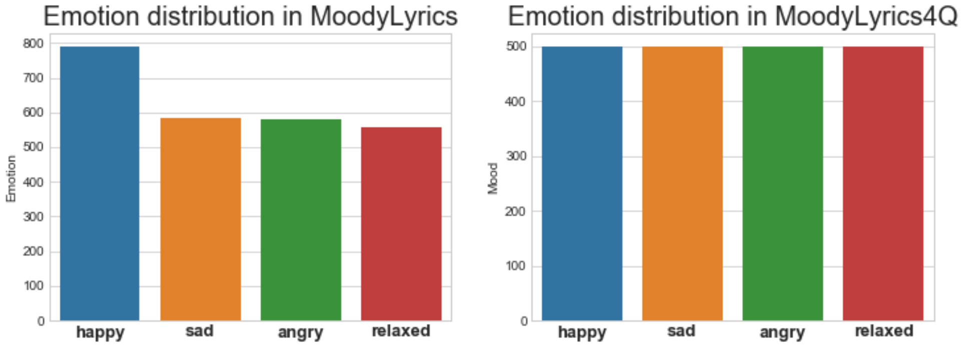


Figure 4.2: Emotions distribution comparison between MoodyLyrics and MoodyLyrics4Q

MoodyLyrics4Q classes are much more balanced, however MoodyLyrics4Q contains less songs with respect to MoodyLyrics.

We studied the qualitative difference between the two version comparing the classification given to the songs contained in both datasets to establish what version, according to us, is more correct. The intersection between the two versions contains 47 songs, and 21 over 47 have been classified differently. We noticed that in 15 over this 21 songs the two datasets confuses *happy* with *relaxed* and *angry* with *sad*. Indeed, only 6 of 21 songs are classified totally differently if we assume that we can merge happy and relaxed and angry and sad emotion. However, reading the lyrics of each of this 6 songs, we could not establish which version is the best one. A more detailed analysis can be fount at Notebook 5.

4.2 POS Tagger Validity Check

Before digging into more complex types of analysis, we took the time to check if the tools we were using could be really good for our purpose.

Specifically, we had some doubts on the Part-Of-Speech (POS) tagger. Indeed, those kind of systems are generarly trained on texts coming from sources whose type of language is very different from those we would expect to find in lyrics. In fact, the SpaCy's POS tagger implemented in the language model we are using is trained on OntoNotes 5 [9] and on Common Crawl [2], which are both made of pieces of text taken from news, conversational telephone speech, weblogs, usenet newsgroups, broadcast and talk shows. Obviously, this type of natural language texts are much different from a lyric and we just wanted to make sure that we were using an appropriate tool.

Before going into the details of what we did, we must state that SpaCy's POS tagger provides two tags per words, which will both be considered in our analysis. Those two types of tag are:

- **POS**: coarse-grained part-of-speech e.g. VERB
- **TAG**: fine-grained part-of-speech e.g. PAST_TENSE

In order to obtain reliable insights of the functionalities of our POS tagger we considered three songs: one with a common language and very few slang words, one filled with slangs and another one with some vulgar words.

The first song we considered was "Polly" from Nirvana.

As a first approach we tried to tag the words considering one line of the lyric at the time. Here is an example of what we obtained:

```
Polly wants a cracker
```

```
PROPN VERB DET NOUN

Polly = PROPN NNP -> noun, proper singular
wants = VERB VBZ -> verb, 3rd person singular present
a = DET DT -> determiner
cracker = NOUN NN -> noun, singular or mass
```

As a first attempt, our POS tagger completelly succedeed in recognizing the phrase in the exact way we were expecting. In fact, the absence of weird words, e.g. slangs, made the task much easier.

Because of the almost complete absence of punctuation marks in our lyrics, we expected the POS tagger to fail while analyzing an entire song as whole. Instead, we were quite surprised to see that SpaCy's POS tagger does one desirable thing for our goal: it treats each line as a standalone sentence, even though they are not specifically separed by a stopping mark. Therefore, the same positive behaviour we observed on the first line was totally replicated on the other lines, giving us the exact tagging we were expecting by visually inspecting our song's lyrics.

We omitted the entire tagging process output for brevity reasons.

Our first experiment served to the purpose of arriving to a conclusion: SpaCy's POS tagger is a good tool for lyrics. However it did not solve our doubts about its ability of recognizing "weirder" words such as slang words or abbreviations. For this reason, we moved on analysing the lyrics of "Kiss You Back", from Underground Kiss.

The first interesting thing we noticed was that the POS tagger properly recognized abbreviations such as "'ll". Moreover, another important feature we noticed was clearly visible while tagging this line: "Yeah, we chocolate cross-over". Indeed, here the word "chocolate" is used as a verb (even though chocolate is clearly not defined as a verb in the dictionary) and the POS tagger was able to recognize this exception. This is quite important because, in songs, those situations happen very often.

Other additional things we noticed while analyzing this song came our of the tagging output of the following line:

```
Jus't havin' fun with it, man, know what I'm sayin'?
NOUN VERB NOUN ADP PRON PUNCT INTJ PUNCT VERB NOUN PRON VERB VERB PUNCT

Jus't = NOUN NNS -> noun, plural
havin' = VERB VBG -> verb, gerund or present participle
```

```
fun = NOUN NN -> noun, singular or mass
with = ADP IN -> conjunction, subordinating or preposition
it = PRON PRP -> pronoun, personal
, = PUNCT , -> punctuation mark, comma
man = INTJ UH -> interjection
, = PUNCT , -> punctuation mark, comma
know = VERB VB -> verb, base form
what = NOUN WP -> wh-pronoun, personal
I = PRON PRP -> pronoun, personal
'm = VERB VBP -> verb, non-3rd person singular present
sayin = VERB VBG -> verb, gerund or present participle
' = PUNCT '' -> closing quotation mark
? = PUNCT . -> punctuation mark, sentence closer
```

One very interesting result we can notice comes from the following two lines:

```
havin' = VERB VBG -> verb, gerund or present participle and
ayin = VERB VBG -> verb, gerund or present participle
```

In fact it looks like our POS tagger is able to recognize verbs in their correct tense even if they are abbreviated in an unconventional way.

One thing which really impressed us, was the word "man" being recognized to be an interjection from time to time. "An interjection is a part of speech that shows the emotion or feeling of the author. These words or phrases can stand alone or be placed before or after a sentence. Many times an interjection is followed by a punctuation mark, often an exclamation point"[1]. This description perfectly fits with the usage of the word "man" in their contextes when it was recognized to be an interjection.

Those kind of things are not trivial to detect and this ability of our POS tagger convinced us even more of its impressive skills.

The last thing we were left to analyze at this point was a vulgar song. For this purpose we considered "The Ballad Of Chasey Lain", from Bloodhound Gang.

In this case we have no special remarks to report. We can just say that everything was tagged and recognized in the exact way we expected.

We did not report entire lyrics nor the full POS tagger output in the interest of brevity. However, those analysis are available on the public GitHub

[1] http://examples.yourdictionary.com/examples-of-interjections.html

repository for this project, at Notebook 3.

4.3 Initial Classification

The first approaches to classify emotions we tried, were entirely based on converting MoodyLyrics entries into features. Specifically, since MoodyLyrics simply provides the artist and the title of its songs, we firstly downloaded the lyrics and then we built word embedding vectors for them by using SpaCy's built-in methods, based on GloVe. Those vectors we worked on are made of 300 dimensions.

MoodyLyrics songs were annotated according to the arousal and valence dimension of their lyric, as explained above. Therefore we thought that, if we could be able to project the resulting word embedding vector of each lyric in a bi-dimensional space, we could have been able to clearly see how they are separated.

In order to visualize the mentioned idea, we applied Principal Component Analysis (PCA) on MoodyLyrics songs and plotted the resulting points in 2-dimensional space, as can be seen in figure 4.3. However the resulting figure is not exactly what we were expecting.

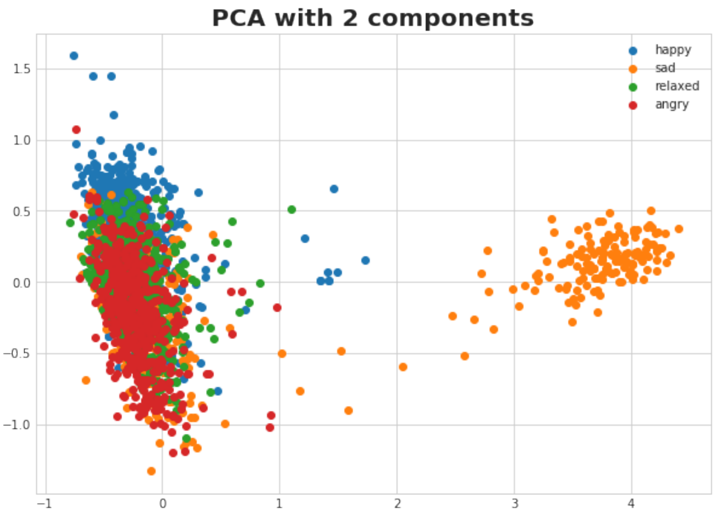


Figure 4.3: Principal Component Analys on MoodyLyrics data with 2 dimensions as output

Indeed, because of the explained MoodyLyrics annotation schema, we were expecting to see 4 clearly separated group of points. Instead, what we can see is that sad songs are clearly separated from the others while the songs belonging to the classes happy, angry and relaxedare quite overlapped and can be easily confused between them.

Despite the fact that this result seemed weird, after listening to several songs and trying to manually classify them according to their lyrics, we came to the conclusion that it is very easy for a human to confuse happy, angry and relaxed song. Therefore, if we keep in mind that MoodyLyrics was manually annotated, we can explain the anomaly in our plot as a consequence of human error during the annotation process.

After this preliminary data analysis, we moved on producing models to actually classify our dataset. The result of these process are visible in table 4.1 where the reported accuracy values were obtain after a 10-fold cross validation.

Classifier	**Accuracy**)
k-Nearest Neighbour	82%
Support Vector Machine	90%
Gradient Boost	86%
Neural Network	90.55%

Table 4.1: Accuracy results for different classifiers on MoodyLyrics

As we can see, the best results were certainly achieved by a Neural Network whose architecture was as follows: one input layer with 175 neurons and sigmoid activation function, one hidden layer with 175 neurons and sigmoid activation function, one output layer with 4 neurons and softmax activation function.

Anyway for certain songs we noticed that the expressed emotion could be easily guessed from its title. Therefore we tried to build also songs titles word embedding vectors and classified them. The results of this operation are visible in table 4.2. Again, those results were obtained after a 10-fold cross-validation.

Classifier	**Accuracy**)
Support Vector Machine	67%
Gradient Boost	65%
Neural Network	70%

Table 4.2: Accuracy results for different classifiers on MoodyLyrics, using just the title's word embedding vector

Moreover, performances obtained while building classifiers based on the songs title's word embedding vector only did not improve with respect to the previous case. Therefore we decided that, probably, we could have given a try at using both the title's and the lyric's word embedding information for each song. What we did was to use both lyrics vectors and title vectors norm. The results of this process are shown in table 4.3.

Classifier	**Accuracy**)
Support Vector Machine	91%
Neural Network	54.14%

Table 4.3: Accuracy results for different classifiers on MoodyLyrics, using both the lyric's and the title's word embedding vectors

For this case we did not try to learn the same number of classifiers as above because, based on the previous experiments, we noticed that SVM and ANN should be the best algorithms for our problem.

The accuracy values obtained in this experiment seem quite abnormal. Indeed, while the SVM provides an accuracy score which is comparable to those obtained while using only the lyrics content, the ANN performances drastically decreased. However, we were expecting to obtain performances very similar to those we got while using just the lyrics content as, in this case, we are basically adding one features (which is the norm of the title's word embedding vector).

In conclusion, the best and more stable performances were those obtained by learning our model solely on the lyrics word embedding vectors. Those results are also quite encouraging and made us think that, with some more advanced feature engineering techniques, we may be able to achieve even better results.

4.4 Feature Engineering

In order to improve our model performances, we focused our attention on feature engineering. Specifically we tried to extract stylometric, structural, orientation and vocabulary based features [20]. Apart from this we also generated a word embedding vector of the words contained in each song's lyric by using SpaCy's [13] pre-trained language model based on GloVe [24].

Here is a comprehensive list of the features we extracted from our dataset, followed by a brief description:

Title_vector: word embedding vector of the song's title

Lyric_vector: word embedding vector of the lyric content

%Rhymes: defined as the percentage of the number of rhymes over the number of total lines. A rhyme is defined as a rhyme between two following lines

Line_count: number of lines in the lyric

Word_count: number of words in the lyric

%Past_tense_verbs: defined as the the percentage of the number of past tense verbs over the total number of verbs

%Present_tense_verbs: defined as the the percentage of the number of present tense verbs over the total number of verbs

%Future_tense_verbs: defined as the the percentage of the number of future tense verbs over the total number of verbs, where future is just will + base form

%ADJ: percentage of adjectives over the total number of words

%ADP: percentage of adpositions (e.g. in, to, during) over the total number of words

%ADV: percentage of adverbs (e.g. very, tomorrow, down, where, there) over the total number of words

%AUX: percentage of auxiliaries (e.g. is, has (done), will (do), should (do)) over the total number of words

%INTJ: percentage of interjections (e.g. psst, ouch, bravo, hello) over the total number of words

%NOUN: percentage of nouns over the total number of words

%NUM: percentage of numerals over the total number of words

%PRON: percentage of pronouns (e.g. I, you, he, she, myself, themselves, somebody,...) over the total number of words

%PROPN: percentage of proper nouns (e.g. Mary, John) over the total number of words

%PUNCT: percentage of puntuctuation (e.g. ., (,), ?) over the total number of words

%VERB: percentage of verbs over the total number of words

Selfish_degree: percentage of 'I' pronouns over the total number of pronouns

%Echoism: percentage of echoism over the total number of words, where an echoism is either a sequence of two subsequent repeated words or the repetition of a vowel in a word

%Duplicate_Lines: number of lines duplicated across the lyric text

isTitleInLyric: boolean, true if the title string is also a substring of the lyric

Sentiment: sentiment between -1 and 1

Subjectivity_degree: degree of subjectivity of the text

Since the word embedding vectors we generated had length 300, at the end we were able to obtain 623 distinct numerical features for each of the songs in our dataset.

4.4.1 Feature Selection

Having to deal with 623 different features for discriminating songs among 4 classes is probably exaggerated and many features may be redundant or may not bring any useful information to our goal. Indeed, after running many experiments, we tried to keep our models as simple as possible by trying to select the fewer number of features possible.

In the end, we obtained the best results just by using the following features: *Lyric_vector*, *%Echoisms*, *%Duplicate_Lines*, *isTitleInLyric*, *%Past_tense_verbs*, *%Present_tense_verbs*, *%Future_tense_verbs*, *%ADJ*, *%PUNCT*, *Sentiment* and *Subjectivity_degree*. This process of feature selection left us with just 310 distinct features per song.

4.5 Beyond the lyrics dataset: EmoInt

One major limitation we had to face during our work on this project was the shortage in terms of labeled data. Indeed, most of the already labeled datasets which can be found online have been created for type of texts which were much different from lyrics i.e. news items, blog posts, Facebook posts, tweets, etc.

In order to overcome this limitation we thought that, if we found a dataset whose items language is close enough to the common lyrics language, we would have had some performance improvements in our classifiers. This is the reason why we tried to combine our dataset together with EmoInt [22]. A more detailed analysis of this work can be found at Notebook 4.

EmoInt provides several tweets annotated according to an emotion (anger, fear, joy, sadness) and to the degree at which the emotion is expressed in text. As EmoInt provide intensity levels together with emotion labels, we decided to take into account only those tweets for which the intensity was greater that 0.50 (50%).

Our original dataset, MoodyLyrics, contains "happy", "sad", "angry" and "relaxed" as labels. Therefore, in order to perform a sort of interjection with EmoInt, we used only the tweets corresponding to the anger, joy and sadness emotions, discarding those belonging to the fear emotion as we would not have been able to map them into our original work.

Moreover, it is important to mention that EmoInt was manually annotated using Best-Worst Scaling (BWS) [21], an annotation scheme proved to obtain very reliable scores. Therefore, we choose EmoInt because it looked like a realiable choice.

As a single preprocessing technique, we dropped hashtags and removed the tag characters (e.g. "Hey @MrTwitter how are you? #cool" became "Hey MrTwitter how are you?") because we had to compare tweets with songs and songs do not have those kind of things. Also, this sort of preprocessing should maximize the chances that everything is properly recognized by our POS tagger.

After having preprocessed EmoInt, we combined it to our lyrics based dataset (MoodyLyrics) and tried some different modeling approaches to see if we could obtain any performance improvements.

After having performed several different trials, we came to the conclusion that the best subset of features to use for our new dataset (MoodyLyrics + EmoInt) was the following: *Lyric_Vector*, *Word_Count*, *%Echoisms*, *Selfish_degree*, *%Duplicate_Lines*, *isTitleInLyric*, *%Present_tense_verbs*, *%Past_tense_verbs*, *%Future_tense_verbs*, *%ADJ*, *%PUNCT*, *Sentiment* and *Subjectivity_degree*.

Classifier	**Accuracy**)
k-Nearest Neighbour	46%
Support Vector Machine	48%
Gradient Boost	46%
Neural Network	82.38%
Multinomial Naïve Bayes Classifier	49%

Table 4.4: Accuracy results for different classifiers on MoodyLyrics and EmoInt combined

Using the data obtained as explained, we built several classifiers with which we obtained the results shown in table 4.4. We split our dataset in a training and a test part and the shown results are those obtained while classifying the test set.

As it was when we used MoodyLyrics alone, the best results were obtained by using a Neural Network. This network had a very simple architecture: one input layer with sigmoid activation function and 120 neurons, one hidden layer with softmax activation function and 60 neurons and an output layer with 4 neurons and softmax activation function.

Anyway, it is quite clear that this approach did not lead us to real improvements over our previous cases. This experiment served to the purpose of understanding that EmoInt is probably too much different from what we have classify and it may not improve our predictive abilities at all. Therefore we believed that the best choice was to keep using a lyrics based dataset as it is MoodyLyrics.

4.6 Results

In this section we present a summary of the result obtained while predicting one of the four emotion labels *relaxed*, *happy*, *sad*, *angry*, using an artificial neural network, a support vector machine, a logistic regression and xgboost. The accuracies have been computed with a 10-fold cross validation. The presented results were computed using both MoodyLyrics4Q and the union of MoodyLyrics and MoodyLyrics4Q. Of course, all of the feature engineering approaches described above were applied.

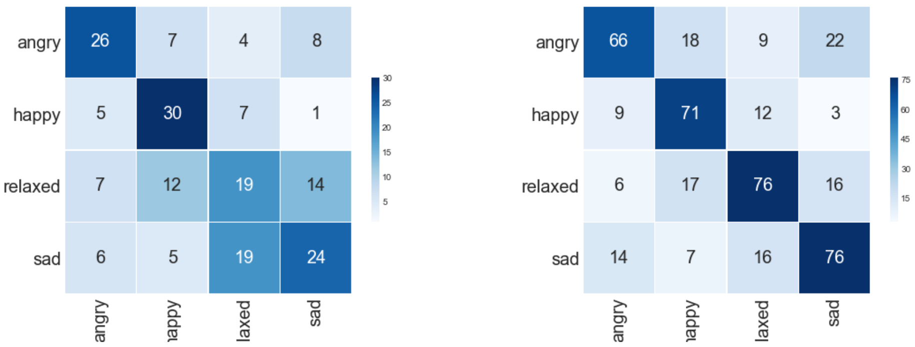
Here we omitted MoodyLyrics alone results because, after several experiments, we noticed that MoodyLyrics creators were right when saying that their first version of the dataset should be used as a ground truth dataset because it contains unappropriate labelings which could deteriorate performances in real world cases.

All the implementation details can be found at Notebook 2.

10-fold Cross Validation Accuracy				
Dataset	ANN	LR	SVM	xgboost
MoodyLyrics4Q	55.97%	57.87%	58.04%	56.89%
Both together	68.41%	69.42%	69.32%	64.27%

Table 4.5: Emotion classification accuracies

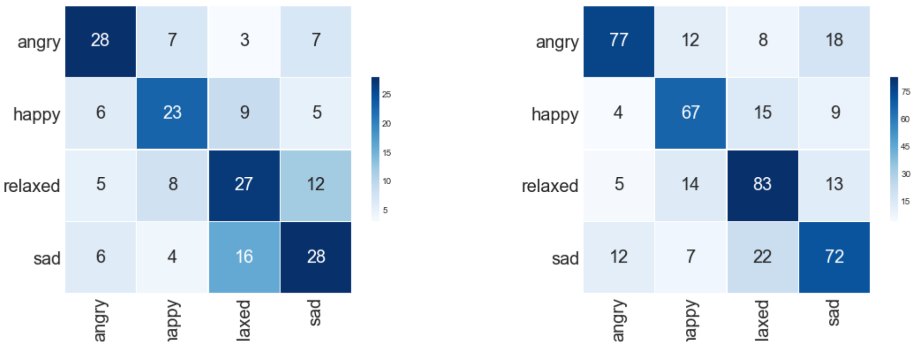
Figure 4.4, 4.5 and 4.6 represents the confusion matrix obtained while training each model on the 90% of the toal dataset and testing it on the randomly sampled remaining 10%.



(a) ML4Q

(b) ML + ML4Q

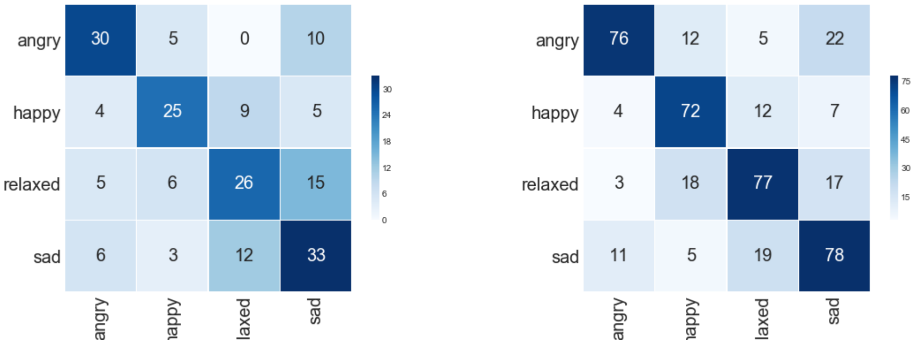
Figure 4.4: Artificial Neural Network - Confusion Matrix



(a) ML4Q

(b) ML + ML4Q

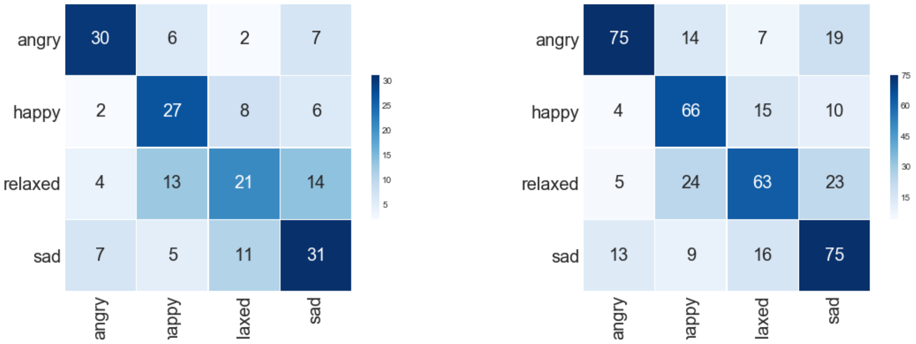
Figure 4.5: Logistic Regression - Confusion Matrix



(a) ML4Q

(b) ML + ML4Q

Figure 4.6: Support Vector Machine - Confusion Matrix



(a) ML4Q

(b) ML + ML4Q

Figure 4.7: Xgboost - Confusion Matrix

Chapter 5

Playlists analysis

Now we will move our focus from lyrics to playlists. Indeed, we will now demonstrate how we extended our work on single songs to whole playlists.

5.1 Playlist classification

Once we have a model to classify song emotions, our next step it trying to classify playlists emotion which could be a really useful information for a Recommender System which want to perform playlist continuation.

For the emotion classification task we decided to use three artificial neural networks, with the same architecture, but trained with three different datasets: MoodyLyrics, MoodyLyrics4Q and the merged dataset. Figure 5.1 shows the different accuracies reached with each dataset.

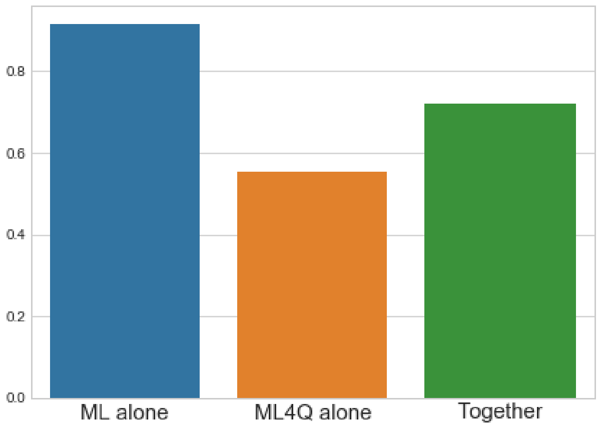


Figure 5.1: 5-fold ANN accuracies for each model

The method we chose to classify a playlist is the following: given a `song emotion array` = [sad%, angry%, happy%, relaxed%] where the sum of the emotion percentage for each song is equal to one, the playlist emotion is the emotion with the maximum percentage column sum over the rows. We found this approach very flexible, indeed, normalizing by the number of songs inside a playlist we obtain a probability of the emotion for that playlist. This would add 4 additional features which could be used by a Recommendation System.

However our main issue has been the absence of a labeled dataset through which compute an accuracy score of our classification method. In order to overcome this problem a perfectly balanced silver standard dataset has been generated considering 40 playlist inside Spotify RecSys [12] dataset with four really expressive titles: *rage* for angry playlists, *crying* for sad, *party!* for happy and *sleeping* for relaxed playlists.

The first step to classify the emotion for each song has been downloading the lyrics, but, unfortunately, we noticed that we could not download the entire number of song lyrics for every playlist. Figure 5.2 shows, for each playlist, the number of songs for which we could download the lyrics, while figure 5.3 shows the percentage of available songs for each playlist.

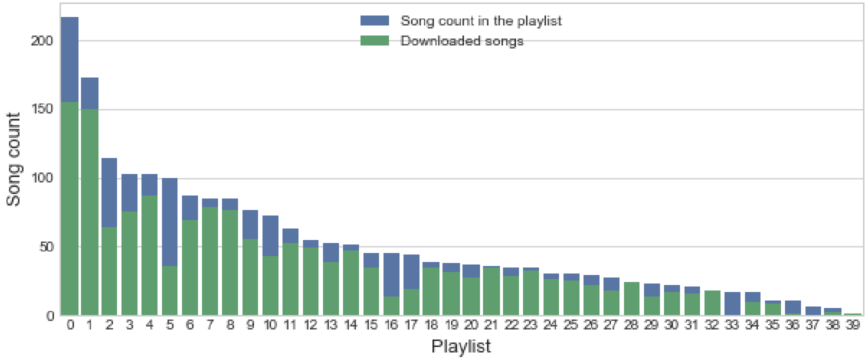


Figure 5.2: Real number of available songs for playlist

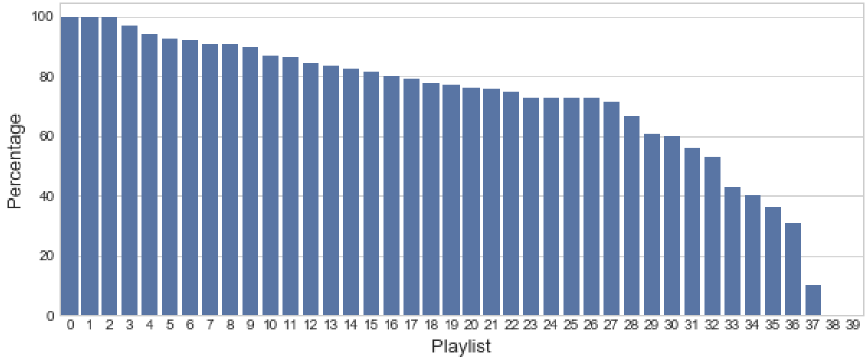


Figure 5.3: Percentage of available songs for playlist

Unfortunately two *relaxed* playlist totally disappeared, unbalancing our silver standard.

We then featurized each song using the features described earlier in this report and classified them using the three artificial neural networks (one trained with MoodyLyrics, one with MoodyLyrics4Q and one with the merged datasets) we previously built.

5.1.1 Robust Playlist Classification

As we already said, playlist classification was done by averaging the emotion vectors which are given as output of the lyrics classification process. However, in order to make the classifications more robust, we tried to remove outliers from our predictions.

Basically, for each playlist, we built a matrix of dimension Nx4, where N is the number of songs in the playlist and 4 is the number of target emotions (happy, sad, angry and relaxed). In this matrix, each row corresponds to the probabilistic assignment of emotions to each of song of the playlist. In order to remove outliers we considered a column (an emotion) at the time and we followed those steps:

1. compute the third (q3) and the first quartile (q1)
2. compute the interquartile range (IQR) defined as the difference between the third and the first quartile
3. define an upper bound value as $upper = q3 + (IQR * 1.5)$
4. define a lower bound value as $lower = q1 - (IQR * 1.5)$
5. eliminate from the column all those values outside the range $[lower, upper]$
6. average the remaining values to compute the label score for the emotion corresponding to the current column of the matrix

After having repeated the above process for all the 4 columns of our Nx4 matrix, we obtain a vector of 4 real values, respectively representing the playlist score for each of our target emotion.

5.1.2 Playlist Classification with a subset of the songs

In order to improve our computational speed we came up with the idea that probably a playlist's emotion is clearly expressed in just a subset of its songs, making it useless to compute emotion classification of each of its song. For this purpose, we tried to consider just a small sample of the total songs at the time, incrementally increasing our sample size.

We did this operation both using our simple, average based, classification scheme and the other, more robust, and outlier sensitive approach.

Figures 5.4 and 5.5 shows the accuracy results obtained while using the simple average classification approach, while figures 5.6 and 5.7 shows the results obtained while using the robust classification approach.

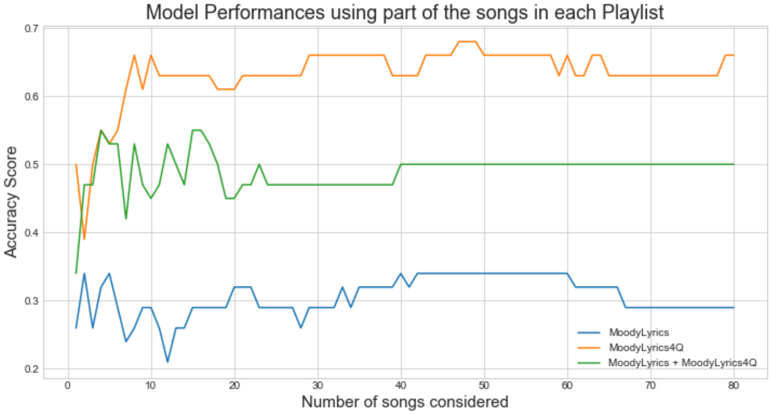


Figure 5.4: Classification accuracy on playlists obtained while considering just a certain amount of songs

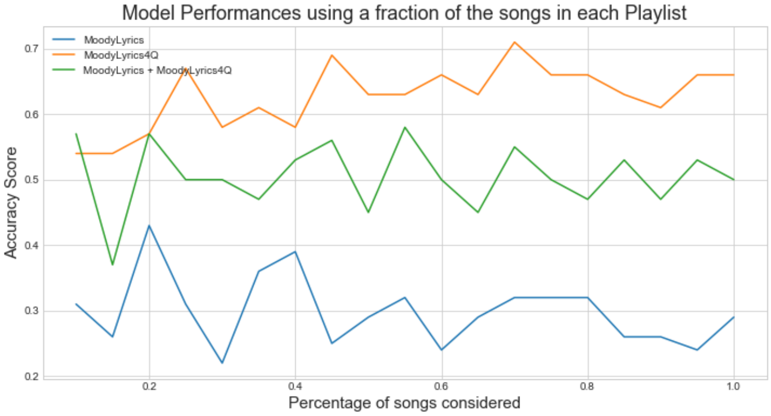


Figure 5.5: Classification accuracy on playlists obtained while considering just a percentage sample of the total number of songs

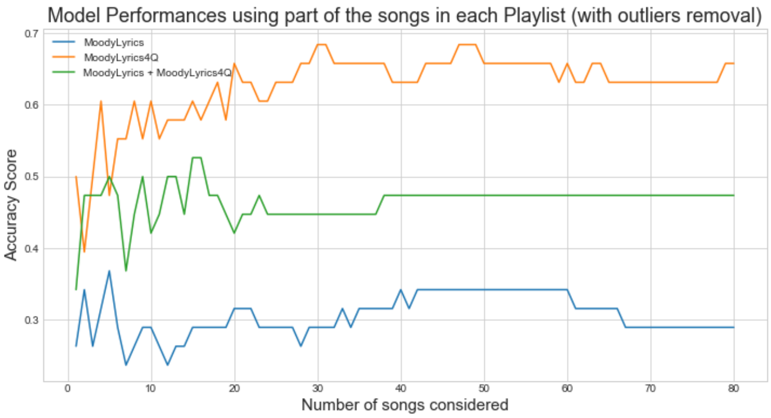


Figure 5.6: Classification accuracy on playlists obtained using robust classification while considering just a certain amount of songs

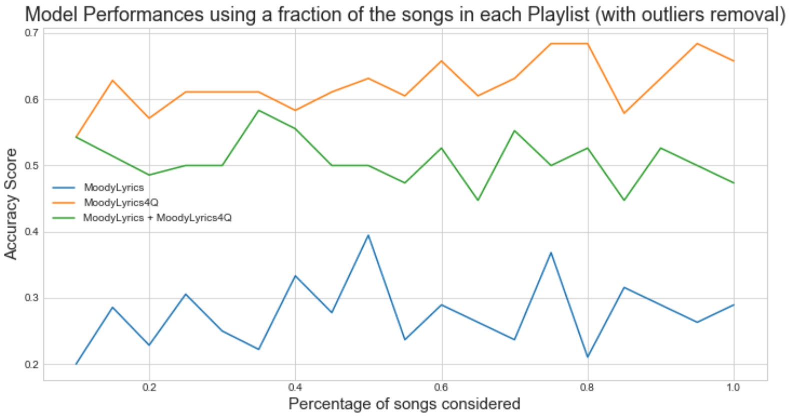


Figure 5.7: Classification accuracy on playlists obtained while using robust classification considering just a percentage sample of the total number of songs

As we can see from those figures, in order to converge to a stable classification accuracy value, the algorithm always needs a huge number of songs. Therefore we believed that this experiment told us that, the best way to classify our playlists is still to consider all of the songs contained in them.

5.1.3 Results

Table 5.1 shows the different accuracies reached classifying the playlist emotions using the method previously explained.

Playlist Classification Accuracy		
Dataset	With outliers	Without outliers
MoodyLyrics	29%	29%
MoodyLyrics4Q	66%	66%
Both together	50%	47%

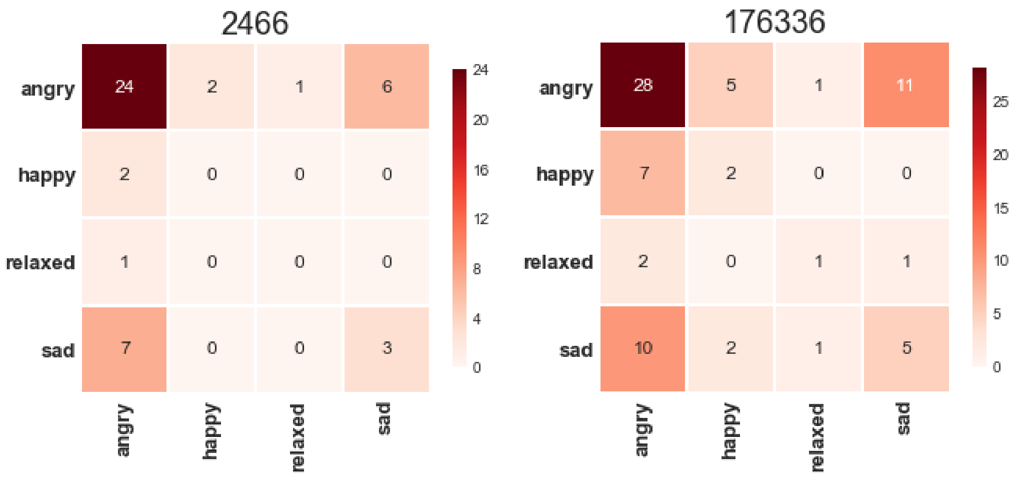
Table 5.1: Playlist classification accuracies comparison

5.2 Emotion patterns in playlists

In order to visualize emotion patterns inside a playlist we will use the Silver standard previously created, in which playlists are already classified. We will instead classify the emotion of each song inside a playlist and count the transitions between the emotions.

We will use a (4x4) matrix, where $matrix[i][j]$ is the total count of transition from emotion i to emotion j in that playlist. We count a transition from emotion i to emotion j only if the two songs are really consecutive, i.e. if the sequence IDs are consecutive inside the playlist. Indeed since sometimes we cannot download lyrics for a song inside a playlist it is possible that two songs are not really consecutive.

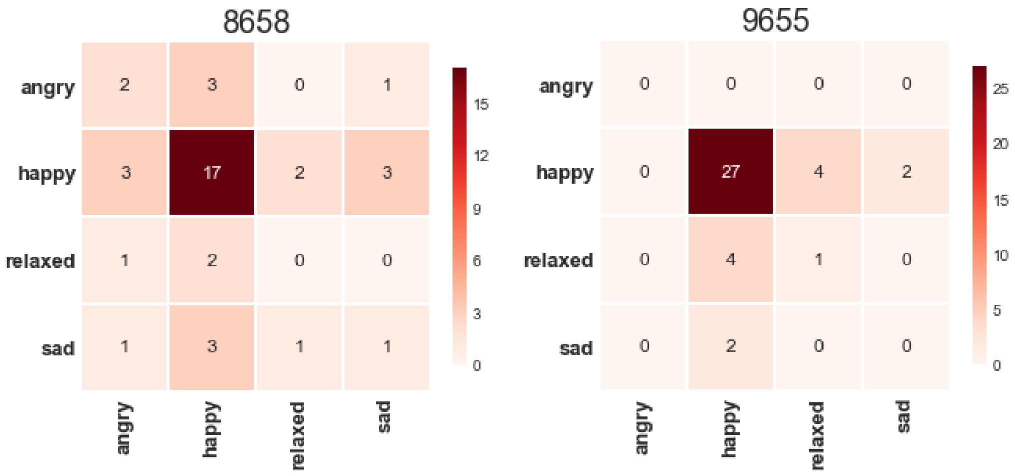
In the following figures we show emotion patterns obtained for 2 angry, 2 happy, 2 relaxed and 2 sad playlists.



(a) Playlist PID 2466

(b) Playlist PID 176336

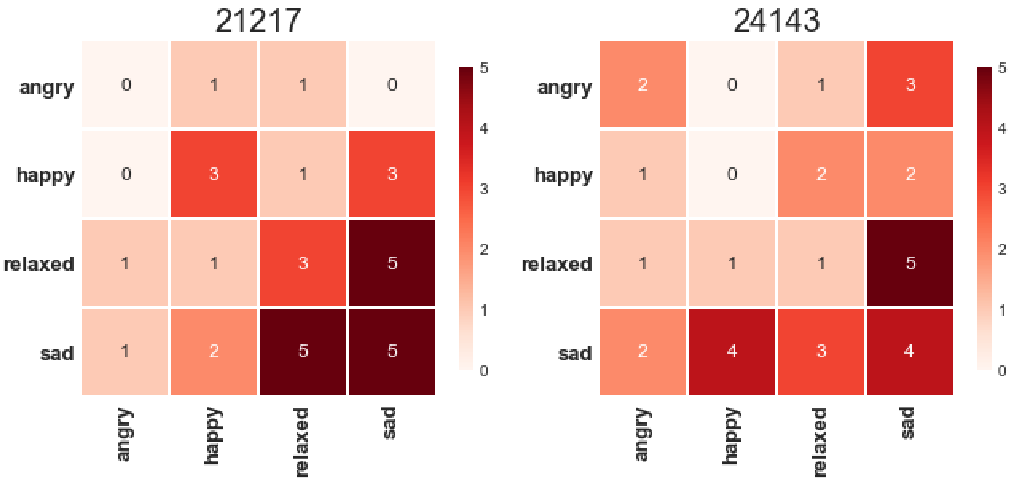
Figure 5.8: Emotion patterns in angry playlists



(a) Playlist PID 8658

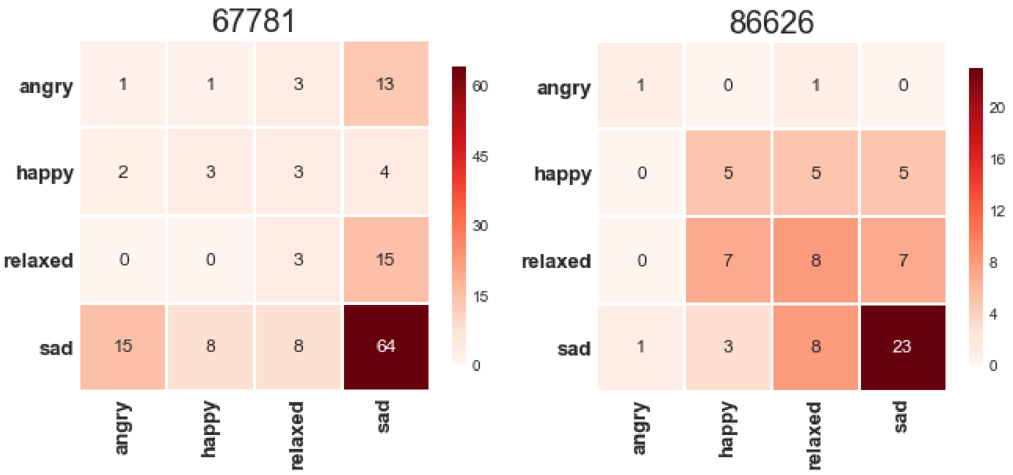
(b) Playlist PID 9655

Figure 5.9: Emotion patterns in happy playlists



(a) Playlist PID 21217 (b) Playlist PID 24143

Figure 5.10: Emotion patterns in relaxed playlists



(a) Playlist PID 67781 (b) Playlist PID 86626

Figure 5.11: Emotion patterns in sad playlists

We can notice that in most of the cases, the greatest number of transitions represents the transition from and to the playlist emotion itself. For example in *angry* playlists not only $matrix[0][0]$ has the biggest count but also the entire first row and column have the most populated cells, meaning that the majority of times pass from *angry* to *angry* or from *happy*, *relaxed*, *sad* to *angry*.

However the *relaxed* playlists represent a more random emotion pattern, meaning that or emotion songs (especially relaxed songs) are badly classified, or simply relaxed playlist includes a more uniform variety of songs.

Chapter 6

Conclusions and Future Works

In this report we detailed all the experiments and trials we did in the context of our semester project. Our main goal was to firstly build a system capable of classifying songs lyrics based on their emotion, and then to use the acquired knowledge to understand emotional patterns in music playlists.

The first thing we did, was focusing on the word embedding vectors generated for the lyrics of the already labeled songs we found in the MoodyLyrics dataset, which classifies lyrics according to 4 possible emotions: angry, sad, relaxed and happy. As we understood that this approach was very limiting, we started working on some smarter feature engineering approaches, in which we extracted our own features from the analyzed lyrics.

Under the suggestion of the dataset's creator, we then moved from MoodyLyrics to its updated and more reliable version: MoodyLyrics4Q. This new dataset became then the gold standard for our experiments.

We tried to combine our lyrics dataset also with data coming from different data sources, e.g. tweets. However we understood that the best way to proceed in our project was to keep working with just the lyrics based dataset, as it is obviously the most suitable one for unraveling emotional patters in music playlists.

At the end of the lyrics classification process, our output was, for each song, an emotion vector, composed of the intensities at which each sentiment was expressed.

Once we understood which was the best subset of features and the best classifiers, we moved our focus on classifying playlists. For evaluating our playlist classification performances, we compared our outputs to those of the silver standard playlist dataset we built.

We used two different approaches for playlist classification: in the first one we just computed the emotion classification vector for each song in the playlist and then we averaged them; in the second approach, before doing the average, we exclude from the emotion vectors those values which were considered to be too different from the others (outliers).

Before having a look at the emotional patterns in the playlists, we tried to ease out our computational load by trying to see if it was possible to properly classify a playlist based on just a subset of its songs. However, we found out that we need all the songs in each playlist if we want to achieve the best results possible.

To conclude, we explored the emotional patterns inside each playlist, meaning that we saw how the emotion classification vector evolved inside each playlist, transitioning from one emotion to another.

In conclusion, the results we obtained are shown in Table 6.1 and Table 6.2.

10-fold Cross Validation Accuracy				
Dataset	ANN	LR	SVM	xgboost
MoodyLyrics	90.55%	-	90%	86%
MoodyLyrics4Q	55.97%	57.87%	58.04%	56.89%
Both together	68.41%	69.42%	69.32%	64.27%

Table 6.1: Emotion detection accuracies

Playlist Classification Accuracy		
Dataset	With outliers	Without outliers
MoodyLyrics	29%	29%
MoodyLyrics4Q	66%	66%
Both together	50%	47%

Table 6.2: Playlist classification accuracies

Despite the interesting results we obtained, there is still much room for improvements.

Indeed, one thing we did not explore in much depth is the usage of Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), employed by many modern systems built for emotion classification tasks.

We made several small experiments using some complex CNN and RNN based networks we found in the literature [7]. However, those models produced very poor performances, which is the reason why we did not even mention them in the current report.

Anyway, those small experiments served to the purpose of letting us understand which direction to take in the future. Indeed, neural networks components, such as Long Short Term Memories (LSTMs), seems to be perfectly suited for those kind of problems and, if properly tuned, may help us in reaching better performances.

One current limitation we saw when running the mentioned experiments with deep neural networks is the shortage of training data. Indeed, effective training of neural networks requires a huge amount of data. In the low-data regime, parameters are underdetermined, and learnt networks generalise poorly.

The simplest way to solve this problem, would be to manually label more and more song lyrics, eventually using automated mechanisms e.g. Amazon Mechanical Turk [1]. However this solution is not practical and expensive (especially if we use Amazon Mechanical Turk).

Certainly, a better approach would be to use some automated data generation techniques. Indeed, we may think of using some of the Data Augmentation mechanisms currently employed in deep learning. We found several interesting techniques [28] which could be employed in our domain and could help alleviate issue by using Generative Adversarial Neural Networks (GANN), which are able to generate new data from the already existing one.

Bibliography

[1] Amazon Mechanical Turk. `https://www.mturk.com/`.

[2] Common Crawl, an open repository of web crawl data that can be accessed and analyzed by anyone. `http://commoncrawl.org/`.

[3] EmoBank. `https://github.com/JULIELab/EmoBank`.

[4] fastText - Library for efficient text classification and representation learning. `https://fasttext.cc/`.

[5] IBM Watson: Natural Language Understanding APIs. `https://www.ibm.com/watson/services/natural-language-understanding/`.

[6] IBM Watson: Tone Analyzer. `https://www.ibm.com/watson/services/tone-analyzer/`.

[7] Multi-class Emotion Classification for Short Texts. `https://github.com/tlkh/text-emotion-classification`.

[8] Natural Language Toolkit. `https://www.nltk.org/`.

[9] OntoNotes Release 5.0. `https://catalog.ldc.upenn.edu/ldc2013t19`.

[10] ParallelDots API. `https://tone-analyzer-demo.ng.bluemix.net/`.

[11] QEmotion. `http://www.qemotion.com/demo.php`.

[12] RecSys Challenge 2018. `https://recsys-challenge.spotify.com/`.

[13] SpaCy - Industrial-Strength Natural Language Processing in Python. `https://spacy.io/`.

[14] Stanford Core NLP. `https://stanfordnlp.github.io/CoreNLP/`.

[15] TextBlob: Simplified Text Processing. `http://textblob.readthedocs.io/en/dev/index.html/`.

[16] word2vec - Tool for computing continuous distributed representations of words. `https://code.google.com/archive/p/word2vec/`.

[17] Emotion detection and recognition from text using Deep Learning. `https://www.microsoft.com/developerblog/2015/11/29/emotion-detection-and-recognition-from-text-using-deep-learning/`, 2015.

[18] Erion Çano and Maurizio Morisio. Music mood dataset creation based on last.fm tags. pages 15–26, 2017.

[19] Erion Çano and Maurizio Morisio. Moodylyrics: A sentiment annotated lyrics dataset. In *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, ISMSI '17, pages 118–124, New York, NY, USA, 2017. ACM.

[20] Michael Fell and Caroline Sporleder. Lyrics-based analysis and classification of music. In *COLING*, 2014.

[21] Svetlana Kiritchenko and Saif M. Mohammad. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. *CoRR*, abs/1712.01765, 2017.

[22] Saif M. Mohammad and Felipe Bravo-Marquez. Emotion intensities in tweets. In *Proceedings of the sixth joint conference on lexical and computational semantics (*Sem)*, Vancouver, Canada, 2017.

[23] W. Gerrod Parrott. *Psychological Perspectives on Emotion in Groups.* Palgrave Macmillan UK, London, 2016.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[25] J.A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161–1178, 1980.

[26] Shiv Naresh Shivhare1 and Prof. Saritha Khethawat. Emotion detection from text. 2012.

[27] Hassan Walaa, Medhat Ahmed and Korashy Hoda. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 1994.

[28] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks. *CoRR*, abs/1711.00648, 2017.