1 2 90

UNIVERSIDADE Ð
COIMBRA

Rafael Alexandre Portugal Matos

MERGE Lyrics: Music Emotion Recognition next Generation - Lyrics Classification with Deep Learning

MSc Thesis

Dissertation in the context of the Master in Data Science and Engineering, advised by Professor Rui Pedro Paiva and Professor Ricardo Malheiro and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the
University of Coimbra.

September of 2022

Merge Lyrics: Music Emotion Recognition Next Generation – Lyrics Classification With Deep Learning

Rafael Alexandre Portugal Matos

MSc Thesis

FACULTY OF SCIENCES AND TECHNOLOGY
DEPARTMENT OF INFORMATICS ENGINEERING

September 2022

Dissertation in the context of the Master in Data Science and Engineering advised by Professor Rui Pedro Paiva and Professor Ricardo Malheiro and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering of the University of Coimbra.

This work is funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of project MERGE (Music Emotion Recognition - Next Generation) - PTDC/CCI-COM/3171/2021 and CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

Acknowledgements

The development of this project is perhaps the most important stage of the beginning of my career it could not be done without the help of those around me. I would like to start by giving thanks to everyone who has been directly or indirectly supporting me throughout this journey.

I would like to thank the MERGE project (Music Emotion Recognition - Next Generation , PTDC/CCI-COM/3171/2021), funded by Fundação para a Ciência e Tecnologia (FCT), which offered the contex for this work, and everyone in the MERGE team, namely my closest supervisors Rui Pedro Paiva and Ricardo Malheiro for their continuous availability, support and guidance during this period. Working with such friendly yet professional and interested colleagues made the tasks at hand feel more enjoyable and compelling.

I would also like to say thanks to all of my friends, the ones who remain in my life and to the ones who, because of the unpredictable course of life, do not. I will never forget the memories we shared, the support that was given and most importantly the necessary reprimands for my stubborn obsessions, every single moment will remain with me until the end.

A special thanks to the University of Coimbra, its professors and the remaining employees who were always there when it was needed. I would also like to say thanks to the Centre for Informatics and Systems of the University of Coimbra (CISUC) for the funding of this work via a research scholarship.

I owe much of this work to my close family who have always taken a big role in softening the hardships of the graduation process. To my father José, my mother Anabela, my grandfather Mário, my grandmother Maria and my sister Daniela I express tremendous gratitude as these four human beings were the core pillars of my dreams, supporting me through each decision, motivating me to go further and making sacrifice after sacrifice so that this could all be possible. These five people believed in me and always stood by my side despite the situation, for this reason I would like to dedicate this work to them.

Abstract

Music has become a key aspect of the everyday life for most individuals, this industry is constantly growing and from there comes the need to develop systems that can automate the organisation and partitioning of music pieces in different categories. For years researchers have attempted to determine perceived emotion from music with the aid of Classical Machine Learning (ML) algorithms, providing us good yet not great state of the art results of 77.1% F1-Score in Lyrics-based Music Emotion Recognition (LMER) and 67.4% F1-Score in Lyrics Emotion Variation Detection (LEVD).

Deep Learning (DL) approaches have become popular in the LMER field as researchers attempt to surpass traditional methods. We proposed a DL approach to this problem, using the existing 951 lyrics LMER dataset, which reached an F1-Score of 88.9% and consisted on a pre-trained model (BERT). This work also assesses LEVD in which an F1-Score of 85.3% was achieved by applying concepts such as Natural Language Processing (NLP) data augmentation and Long Short Term Memory (LSTM) models.

This research contributed to the improvement of our knowledge in DL approaches which enhanced the previously achieved classical ML results and proved beneficial to the better understanding of the limitations the data available imposes.

Keywords

deep learning, data augmentation, natural language processing, music emotion recognition, music emotion variation detection

Table of Contents

This page is intentionally left blank.

Abbreviations

BERT Bidirectional Encoder Representations from Text.

BOW bag-of-words.

CNN Convolutional Neural Network.

DL Deep Learning.

DNN Deep Neural Network.

GI General Inquirer.

GPT Generative Pre-Training Transformer.

GPU Graphics Processing Units.

LIWC Linguistic Inquiry and Word Count.

LMER Lyrics-based Music Emotion Recognition.

LSTM Long short-term memory.

MEVD Music Emotion Variation Detection.

MIR Musical Information Retrieval.

ML Machine Learning.

MLP Multi-Layer Perceptrons.

NLP Natural Language Processing.

POS part-of-speech.

RNN Recurrent Neural Networks.

SemBF Semantic-Based Features.

SOA State-of-the-art.

StruBF song-Structure-Based Features.

StyBF Stylistic-Based Features.

TL Transfer Learning.

List of Figures

List of Tables

Chapter 1

Introduction

Art and music are basic human activities. Humankind and art cannot function without one another. The interaction with sound is unavoidable, either to make it or take pleasure in it. People have always found music significant in their lives, whether for enjoyment in listening, the emotional response, performing, or creating. Music is something special. Some call it a universal language, while others call it the window to the soul. In the earlier days, we would give cassette tapes to the ones we secretly liked, since words could not express our deepest feelings. We used the emotion in music. And not much has changed.

Nowadays, sharing music has become easier, and it is quite evident that music has taken up an extremely important role. And for many people, and even brands, the music they relate to is an extension of themselves. As we know, a musical piece sets a mood and a vibe as we hear it in lounges, bars, parties, or other social events. In this era where digital music libraries are constantly growing, to create the right environment one must be able to pick the most suitable playlist. This is where Musical Information Retrieval (MIR) is crucial as it provides individuals more advanced, malleable and user-friendly ways of finding their musical preferences.

Music categorisation is one of the most valuable tools we have at our disposal for understanding and discussing artists' creations. When used flexibly and descriptively, not as a means of rigid division, these classifications have the power to substantially improve our comprehension, recognition, and enjoyment of the music we hear.

Music emotion recognition (MER) is a subfield of MIR that aims to determine the affective content of music by applying machine learning techniques. These systems have numerous applications such as music recommendation systems like the widely known Spotify, automatic playlist generation, music therapy, and so forth. However, to determine the emotional category of music is a defying process and several issues need to be addressed such as excerpt annotation, feature extraction and algorithm design and experimentation.

1.1 Problem Statement, Motivation and Scope

In current times, music plays a big role in the social and psychological functions of individuals. The amount of new records keeps on growing, making it challenging and highly expensive to manually annotate every single freshly produced song. This is where music emotion recognition systems come to aid, helping to develop music recommendation, automatic playlist generation and other systems scattered by different fields.

Automating a task as subjective as music annotation is quite problematic. The existing MER methods face some serious challenges, it is difficult to accurately express the ups and downs of music emotion based on the analysis of the entire music. Analysing these emotions based on the pitch, length, and intensity of the notes, can hardly reflect the soul and connotation of music. Research on early MER systems was based on audio composition breakdown (Lu, Liu, and Zhang 2006). Subsequent investigation revealed that combining lyrics and audio analysis and creating the so-called bimodal MER systems would conceive more accurate results (Laurier, Grivolla, and Herrera 2008, Hu and Downie 2010).

Lyrics-based music emotion recognition (LMER) methods became a key aspect in music information retrieval. More recent studies relied on feature extraction and traditional ML to create models capable of detecting and classifying emotion based on lyrics (Malheiro 2017, Malheiro et al. 2018). Deep Learning is starting to grow in this research field as most of the classical ML methods trust mostly on feature extraction which which requires substantial domain knowledge and, for this reason, is a highly complex and expensive process.

Researchers developed projects solely consisting of coming up with new features that could in some way retrieve more significant information from music to improve their models. The implementation of DL methods can help researchers escape some of this exhausting process by automatically capturing features from lyrics.

Emotion recognition from common text (e.g. world wide web corpus or social media interactions) has been widely explored but when considering lyrics, research is still scarce and emotion recognition, in our opinion, poses as an even more difficult task. This is mainly due to the characteristics of the data itself. Lyrics are not as straightforward as most of the text found on the web. At the end of the day these documents always were a way for artists to express themselves and expressiveness is often associated with ambiguity or confusion in the meaning of a sentence. This is a problem that is hard to overcome as it makes it difficult for the annotators when assigning labels and for the models in their training process. Another major issue detected in the lyrics was the presence of a chorus or multiple identical verses. Redundant information is typically (depending on the task at hand) reduced or discarded as equal samples tend to add less additional information and limit the generalisation capability of machine learning models. As real-world applications for the problem at hand consider high quantities of musical data, is it crucial that our models can adapt reasonably to new information.

In this work, we aim to provide solutions for both static Lyrics-based Music Emotion Recognition, where the song is treated and classified as a whole, and the dynamic approach, Lyrics Emotion Variation Detection, where the goal is to perceive the changes of emotion throughout the lyrics. As this is the continuation of previous work (namely, Prof. Ricardo Malheiro's doctoral thesis Malheiro 2017) we will be leaning on past research made with classical ML models and explore DL model architectures that rise above traditional approaches.

1.2 Main Objectives and Approaches

This dissertation was conducted in the scope of the MERGE project [1] (Music Emotion Recognition - Next Generation , PTDC/CCI-COM/3171/2021), funded by Fundação para a Ciência e Tecnologia (FCT), which offered the contex for this work. The research conducted along the two semesters helped clarify a path to follow during the development of this work, thus some objectives have been proposed for guidance (see Table 1.1), ranked by priority - *High*, *Medium* or *Low*. The goal is to strive as much as possible in order to become close to fulfil these tasks.

Objective	**Priority**
Static LMER - Develop new DL approaches and compare them against existing traditional ML models in controlled datasets	High
Static LMER - Explore NLP related methods for textual data augmentation	Low
Static LMER / LEVD - Testing pre-trained models (BERT/ GPT-3) performance	High
Static LMER / LEVD - Private database annotation	Medium
LEVD - Explore NLP related methods for textual data augmentation	High
LEVD - Develop new DL approaches for lyrics emotion variation detection in private dataset	High

Table 1.1: Goals for the second semester

Overall the main purpose of this project was to assess if there are DL approaches capable of reaching or even surpassing State of the Art results provided by classical methods. When we talk about new approaches, we do not refer to the development of new algorithms or tools, but to the incremental improvements that new architecture designs bring to the table. The fact that Deep Learning models tend to be complex and computationally demanding needs to be taken into consideration when weighting the advantages and disadvantages of these methods over the classical designs.

The central piece of this work was the static portion of the dilemma (LMER). Different DL approaches were applied to the chosen datasets. Structures such as simple multilayer perceptrons, CNNs, LSTMs as well as powerful pre-trained models that perform well on many other NLP tasks were evaluated as possible solutions for the problem. To classify the songs we followed the 4 quadrants of the Russell emotion model. For the dynamic problem (LEVD) we have used a dataset of around 368 annotated sentences from multiple lyrics previously used. In this part of the work we also performed experiments using pre-trained models and applied data augmentation techniques commonly used in the NLP field in order to take the most out of the scarce amount of annotated data we had available. The literature on LEVD was still limited as this field is under development and most of the applications for it rely on pure curiosity, making it an even more challenging task.

[1] https://www.cisuc.uc.pt/en/projects/MERGE

There were changes made in our priorities throughout the work and some of the initial objectives were discarded. One example is the transfer learning for LEVD as most models were not available to the public or would not fit the needs of this project. Also, the experiments made using pre-trained models took a big toll in our time constrains as this was an area that required even more exploration and knowledge gathering than the remaining.

While building and testing the approaches present in this work, other tasks were defined such as the private database increase, which although was not used in this work, it is essential for future research. New lyrics were annotated along the time frame of this project in order to extend the existing datasets and prepare them for future work.

1.3 Results, Contributions and Limitations of the thesis

The main results achieved in this thesis:

- an F1-Score of **88.9%** for static LMER, which surpassed the current state-of-the-art for the datasets used in this work.
- an F1-Score of **85.3%** for LEVD, which was also an increase over the current state-of-the-art for the dataset used in this approach.
- improved F1-Score results in LMER by taking advantage of pre-trained models.
- improved F1-Score results in LEVD with NLP data augmentation approaches.

The contributions that impacted this work:

- a proposal of two DL architectures (one for static LMER and another for LEVD) which outperformed the state-of-the-art traditional ML approaches.
- the analysis of the impact of different inputs (by taking advantage of multiple word embedding models and data augmentation techniques) on DL models.

The limitations encountered in the whole process:

- the size of the datasets, which are still in constant expansion and still to be further on explored in future work.

1.4 Resources and Planning

1.4.1 Planning

In order to organise ourselves and have a better grasp of the tasks at hand we decided to create a Gantt diagram that would showcase the main assignments of this project and their respective duration. In Figs. 1.1 and 1.2 we display the work for the two semesters.

First Semester

Due to the initial lack of experience and information in the field, the opening weeks of the first semester were reserved to gather knowledge and come in touch with the subject. A lot of research was put into NLP and the multiple approaches on traditional machine learning and deep learning emotion recognition as well as text classification. The state-of-the-art (SOA) would rely on this investigation and therefore this starting point would be necessary for a good document.

This stage was purely focused on LMER as LEVD can be thought of a more in depth specialisation of this problem as it is explained later on this document. After the initial research, we needed to put our hands on the work already done by our predecessors to get the feel of how things operate in this area. Our first attempts to mimic traditional ML SOA results were not so successful but with time we would achieve our goals. Later on we started our first DL experiments. The objectives here were to get to know the existing datasets, use previously extracted features combined with simple DL architectures and also to process our lyrics with different word embedding techniques (these are necessary as a lot of the SOA work on deep learning approaches are based on them). The choice of these experiments relied on the need to assess the performance of simpler models first as our experience in DL was still raw and only later test with the more complex ones. Most of the time available in this semester was reserved for the experimental process. Mounting these architectures and applying them to the problem at hand took a significant toll on the initial arrangement.

Second Semester

At the start of the second semester the initial objectives were to build on the proposed architectures and lookout for improvement as simple changes (i.e. adding dropout layers), which often meant considerable performance boosts. By this time only a few experiments had been conducted when compared to the extensive variety of possible parameters passed to the models.

The deadline for this work was extended as multiple obstacles got in our path. The necessary time to invest in knowledge gathering on pre-trained models and their implementations was specifically underlooked, since these are complex architectures that not so often provide intuitive ways of using them. Another circumstance that impacted our time necessities was the shared server used for our tests (see Section 1.4.2). Although efforts were made to increase the amount of Graphics Processing Units (GPU) available this aid only came in the later stage of the work and much of the semester was a constant fight between the available processing units. As such, a big portion of the computation was performed in Google Colab in an attempt to overcome this scenario.

Research on possible approaches for LEVD transfer learning took longer than expected as well, with most of the methods not publicly available or providing inconsistent results.

Overall, this increase in time favored the work specially because at this time the servers availability had been greatly improved and it allowed for a larger and more rich experimentation process that led to our best results. Fig. 1.2 clearly demonstrates the discrepancy between the real and planned times for our tasks.

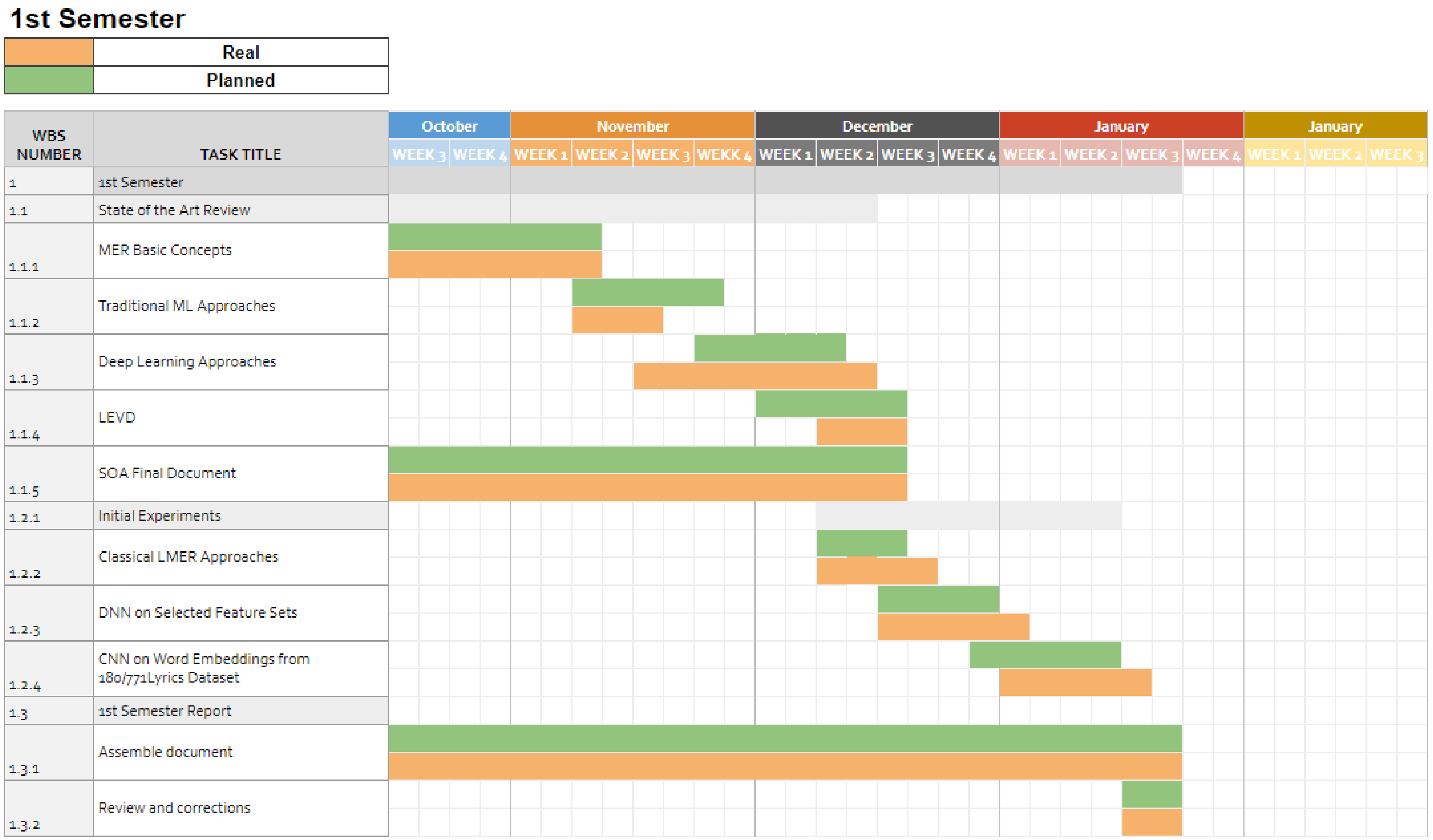


Figure 1.1: Gantt Diagram - 1st semester

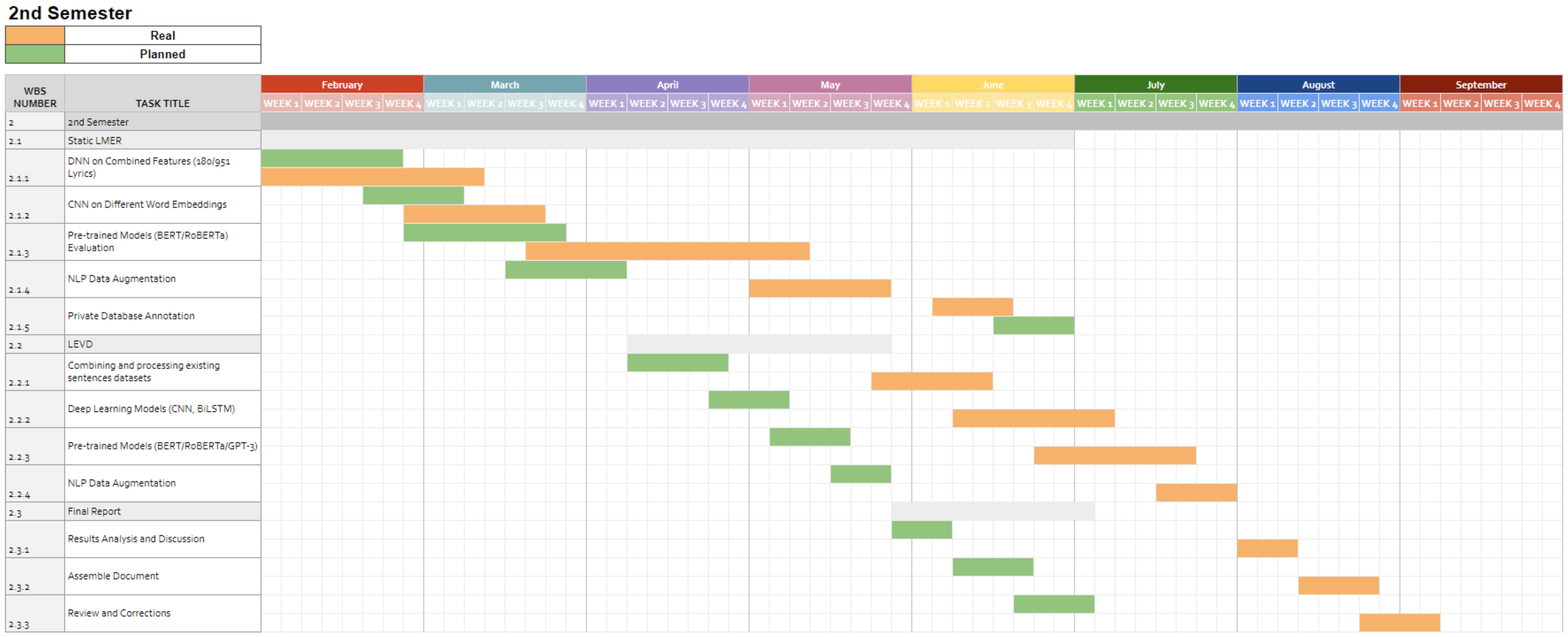


Figure 1.2: Gantt Diagram - 2nd semester.

1.4.2 Experimental Environment

The first experiments were performed in a shared server by the team that provided:

- Intel Xeon Silver 4214 CPU @ 2.20GHz × 48
- 256 GB System RAM
- 3 x NVIDIA Quadro P500 16GB

The models were running on the GPUs (Graphics Processing Unit) which would help save a considerable amount of time when in comparison to using the CPU for this task. Although this environment had an adequate overall performance, it was often being occupied by other users which limited the availability of the server. Later on we have decided to switch our experiments to the Google Research Colaboratory which allowed us to take advantage of the following hardware:

- Intel Xeon Silver 4214 CPU @ 2.20GHz × 48
- NVIDIA V100 PCIE 16 GB
- 24 GB System RAM
- Virtual machine lifetime of 24 hours

Adapting to this new tool took a fair amount of time which we were not counting on, specially when attempting to run previously made work, but these conditions allowed us to train and test our models when we decided to, with the only constrain being the virtual machine lifetime, limiting us to 24 hours of continuous usage.

Most of the available implementations were made with Python, thus this was the selected language for this work. We have taken advantage of libraries such as *tensorflow*, *keras*, *scikit-learn*, *numpy*, *pandas* and *pytorch* which offered us the optimised versions of functions/models necessary to fulfil our tasks.

1.5 Outline

The following Chapter 2 reviews the state-of-the-art approaches to LMER and LEVD. Section 2.1 gives insights on different emotion taxonomies and existing datasets are described in Section 2.2. Deep learning concepts are disclosed in Section 2.3 and information about current progress when it comes to static LMER and LEVD are showcased in Sections 2.4 and 2.5, respectively.

Chapter 3 aims to summarise the approaches and experiments made in order to come up with a DL solution for the static LMER problem. Section 3.1 contains information about the information available for the models to learn, followed by Section 3.2, which clarifies these methods and their results in more detail, and Section 3.3 which contains a critical analysis of these outputs.

Chapter 4 has a similar structure to the later, only this portion of the work is focused on LEVD.

Chapter 5 concludes the work and provides expectations for future work.

This page is intentionally left blank.

Chapter 2

Background Concepts and State of the Art

The following section covers the most significant principles necessary for the conclusion of this work. It summarizes the key background concepts related to this work (e.g., emotion and deep learning) and stands as a critical review of the research done in the field, particularly on static LMER and LEVD. The main datasets employed in these problems are also reviewed.

2.1 Emotion

The concept of emotion commonly comes associated with doubtfulness and uncertainty. In the following portion of this work we discuss the predominant aspects that are entangled with the concept of emotion: definition, types and models.

2.1.1 Defining Emotion

"Emotion" is a term infused in the English dialect in the seventeenth and eighteenth centuries as a translation of the French denomination "émotion". Most of the concepts we define as emotions today have been the object of theoretical analysis since Ancient Greece, under a variety of language-specific labels such as passion, sentiment, affection, affect, disturbance, movement, perturbation, upheaval, or appetite. The word "Emotion" comes with a tremendous amount of ambiguity to its side generated by a long and complicated history that created a wide variety of shared insights about the nature and function of emotions, but no consensual definition of what emotions truly are. As Fehr and Russell Fehr and Russel 1984 stated, "everybody knows what an emotion is, until you ask them a definition".

In modern psychology, emotion is often defined as a complex state of feeling that results in physical and psychological changes that influence thought and behaviour. Emotionality is associated with a range of psychological phenomena, including temperament, personality, mood, and motivation.

Generally we are used to hear that music expresses emotions or that we feel emotions in response to music. This relationship also leads one to intuitively assume that the emotion we feel when listening to a song is the emotion being expressed by that piece. To many, it makes sense that listening to an happy piece of music, for example, is likely to make us feel that way. As Gabrielsson Gabrielsson 2002 has suggested, however, this may not always be the case.

2.1.2 Expressed, Perceived and Felt Emotions

Typically emotions are branched into three categories: expressed, perceived and felt emotions. Gabrielsson 2002.

- Expressed emotion: designates the emotion the artist is trying to convey in its piece.
- Perceived emotion: refers to the emotion that the listener apprehends from the song (which, as stated before, can diverge from the emotion the author is trying to pass on).
- Felt/Induced emotion: defines ones emotional response to the song.

Expressed and perceived emotions commonly match but not in all cases, these two types of emotions are both related and dependent (just like felt emotions) on the person's emotional state, situational and musical factors, thus its natural that in some circumstances they might differ from each other. Although an author tries to convey happiness in his song (expressed emotion), a listener might perceive calmness and quietude (perceived emotion) which can then provoke a state of depression on the auditor (felt emotion).

2.1.3 Emotion Representational Models

As we have seen before, there is no universal agreement when it comes to a standard model to represent emotion, the currently existing models are divided in two branches: discrete/categorical and dimensional/continuous.

Discrete Models

Categorical models aim to categorise emotions, distinguishing them from each other. The famous Ekman's model Ekman 1982 classified emotions in six different categories: anger, disgust, fear, happiness, sadness and surprise. However, this model was ineffective in the music emotion recognition domain as it was originally established for encoding facial expressions.

Another well known model is Hevner's Hevner 1936, which divides emotions into eight clusters using a total of 67 adjectives (Fig. 2.1).

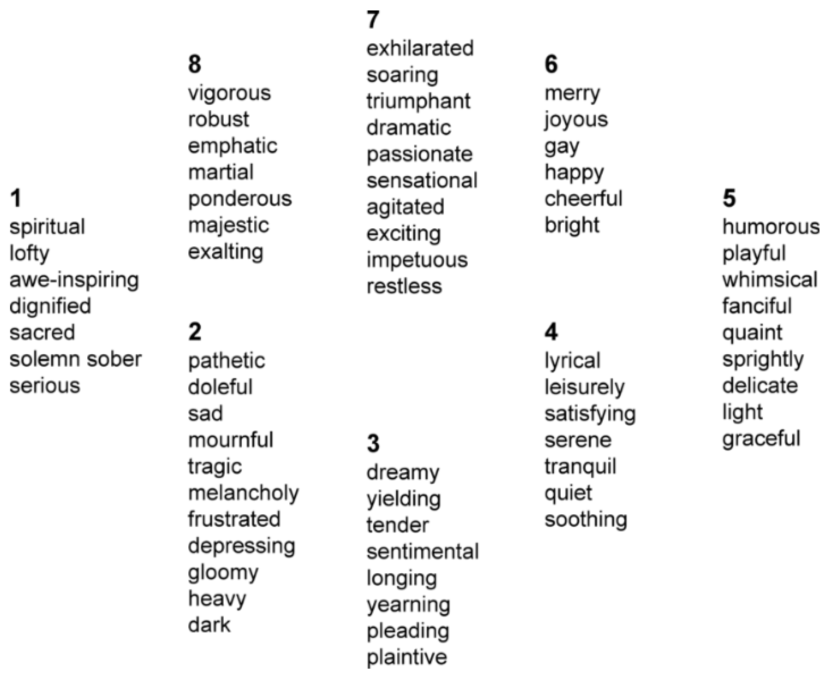


Figure 2.1: Hevner's emotion categories

The layout of this model is based on the emotions similarities, aggregates closer to each other have more affinity between them while those who are farther away are more distinct. For classification purposes, we consider that each emotion present in the same cluster has the same intent as the rest. This model reduces the plethora of emotions to these eight clusters making the understanding of the separation between them challenging and even overlapping some.

Dimensional Models

Discrete models revealed themselves ambiguous representations of emotions. By using a Cartesian space type of representation, Russell states that our emotion can be branched in two neurophysiological responses, arousal and valence, dividing this space in four quadrants (Fig. 2.2). The Y-axis represents arousal (energy and stimulation level) while the X-axis refers to valence (positive and negative affective states, a sense of pleasantness).

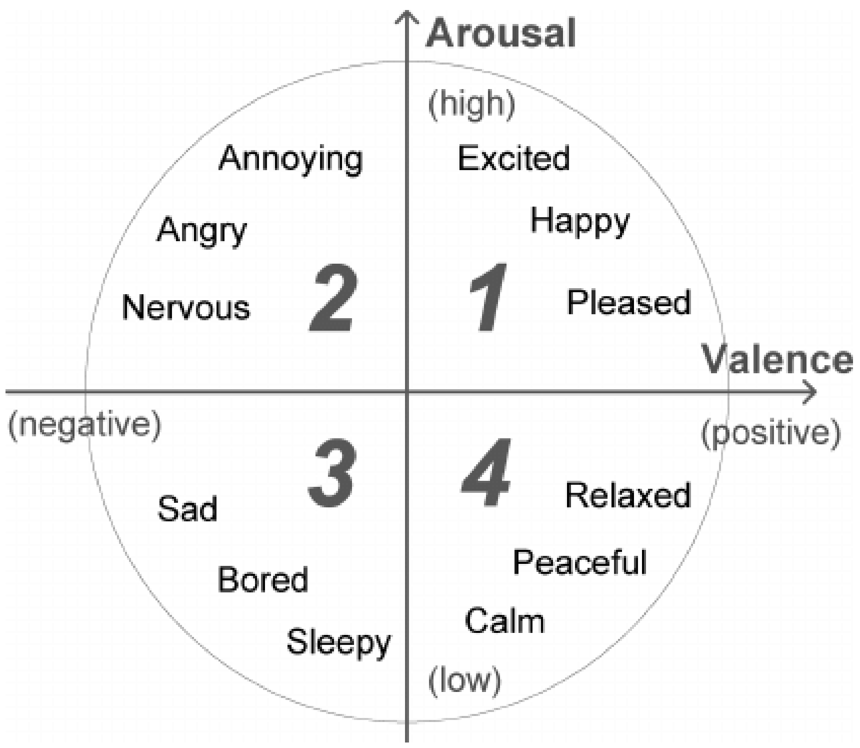


Figure 2.2: Russell's model

One can perceive this model as of having only four emotions, one for each quadrant (happy for quadrant 1, angry for 2, sad for 3 and relaxed for 4) but Russell came up with additional adjectives to classify emotions, distributing them in the plane (Fig. 2.2). This model allows for emotion representation based on discrete tags or continuous dots on the dimensional plane which remove the ambiguity as they're not subjective as linguistic terms. Although following the continuous annotation path provides more clear answers to our question, it's still a much more challenging task when it comes to giving labels for the lyrics than the discrete version of the model.

Russell's model has been reinvented in different ways, adding different dimensions to refer to dominance or potency (Tellegen, Watson, and Clark 1999, Schimmack and Reisenzein 2002) or separations with distinct tags Meyers 2007. Researchers find it challenging to establish pertinent comparison insights as different authors follow divergent emotion models (Fig. 2.3). Currently, the Russell AV (Arousal/Valence) model proves itself as the most consistent one and several MER projects are based on this representation of emotion (Juslin and Sloboda 2001, Laurier, Sordo, et al. 2009). This work is a continuation of previous research made in our laboratory, thus it will also be focused on the Russell AV model for classification purposes.

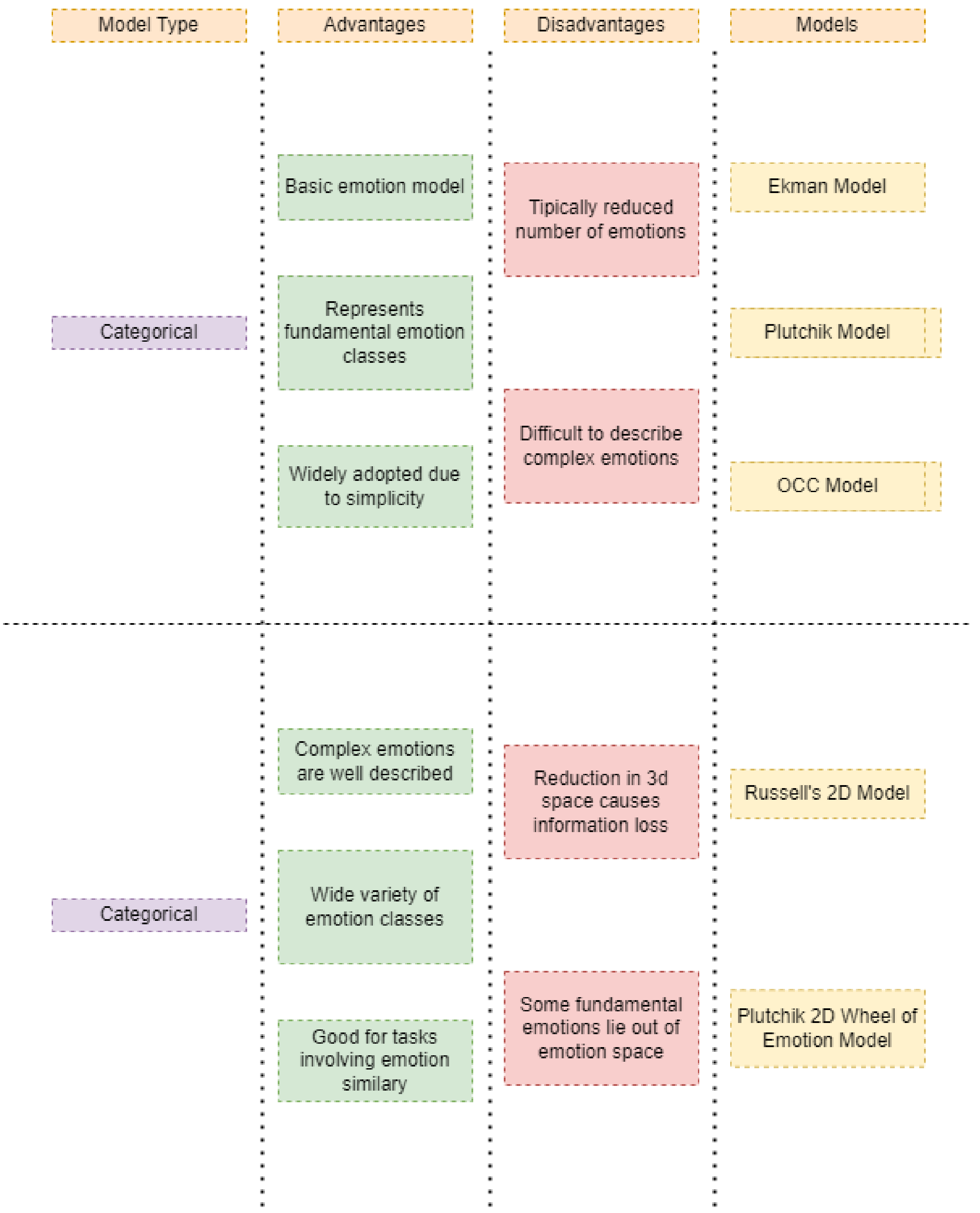

Model Type
Advantages
Disadvantages
Models
Basic emotion model
Tipically reduced number of emotions
Ekman Model
Categorical
Represents fundamental emotion classes
Plutchik Model
Difficult to describe complex emotions
Widely adopted due to simplicity
OCC Model
Complex emotions are well described
Reduction in 3d space causes information loss
Russell's 2D Model
Categorical
Wide variety of emotion classes
Some fundamental emotions lie out of emotion space
Good for tasks involving emotion similary
Plutchik 2D Wheel of Emotion Model

Figure 2.3: Summary diagram of the emotion representational models.

2.2 Music Emotion Datasets

To solve the problem of emotion recognition in music, researchers attempt to annotate lyric datasets in a subjective approach based on emotional perception. Manual and professional annotation of song emotions is labour intensive, restricting the amount of labelled samples each dataset offers. Also, the perception of emotions is influenced by various factors like age, gender, social context or professional background, and thus it is quiet challenging to come to an agreement for all of the instances in the annotation process.

Most of the available datasets do not even reach the amount of 1000 lyrics. Labelling the songs requires a heavy cognitive involvement of the subjects, therefore it is difficult to find individuals that are able to perform extensive work on the subject. Attempts to create bigger datasets often are created using crowdsourcing and develop poorly labelled information, compromising the quality of the whole dataset as machine learning algorithms, as we are going to showcase in the following sections, work better with quality over quantity.

In previous research made in our lab (Malheiro et al. 2018) smaller newly created datasets were employed and provided good results. The annotation process was prioritised which further on helped machine learning algorithms to find strong representations for each targeted emotion and extract features related to them. The following Table 2.1 showcases some of the datasets related to current research for LMER.

Dataset	Approach	Features	Size	Emotion Taxonomy	Observations
MoodyLyrics, Y. Agrawal, R. Agrawal, and Alluri 2021	Dynamic/Static	Word Embeddings and Preprocessed Lyrics	2595 Songs	Russells AV	Well balanced among the emotion classes and high number of unique artists/songs
YL AllMusic, D. Yang and W. Lee 2010	Static	Extraction based on the Harvard General Inquirer IV. 4 model	1032 Songs	Psychological model of 23 emotions	Explores an unusual model of emotion, no great diversity in feature extraction
MER 180 Lyrics, Malheiro 2017	Dynamic/Static	Features based on content, structure, style and semantics (e.g. unigrams, POS tags, GI and LIWC)	180 Songs	Russell AV	Diversity in feature extraction, good balance between emotion categories, strict annotation process but smaller sized dataset
MER AllMusic 771 Lyrics, Malheiro 2017	Static	Features based on content, structure, style and semantics (e.g. unigrams, POS tags, GI and LIWC)	771 Songs	Russell AV	Diversity in feature extraction, decent size for more complex classification models
Synchronised Lyrics Emotion, Parisi et al. 2019	Static	Word Embeddings	———-	Hevner Emotion Model	Good for bi-modal analysis as it is part of a bi-modal synchronised dataset
Baidu Web Chinese/English Lyrics, An, Sun, and Wang 2017	Static	Segmented Lyrics	3552 Songs	Thayer Emotion Model	Lack of annotation restrictions, most of the songs are in Chinese
Popular Chinese Lyrics, Xu et al. 2021	Static	Features extracted using SC-LIWC	3839 Songs	Russell AV	Low feature diversity and poor annotation criteria

Table 2.1: Datasets Summary

2.3 Deep Learning

In this section we point out and explain key concepts from the DL domain. These definitions will then be used frequently on the rest of this work, thus the importance of reviewing them properly.

2.3.1 Basic Concept

At its simplest, deep learning can be thought of as a way to automate predictive analytics. These algorithms are stacked in a hierarchy of increasing complexity and abstraction. A growing child can be a good analogy for this concept. At an young age we tend to point out objects and call them by the name we think classifies them best. For example, when a baby looks to a cat and says "Dog" it relies on its parent to confirm its classification and say "No, that is not a dog!" or "Yes, that is a dog!". As the toddler continues to point to objects, he becomes more aware of the features that all cats possess, clarifying the concept of cat by building an hierarchy in which each level of abstraction is created with knowledge that was gained from its preceding layer.

Neural networks (NN) are a necessary concept to explore when discussing deep learning. When they were first developed researchers were inspired by the human brain, their structure and functionality are similar to the way biological neurons operate.

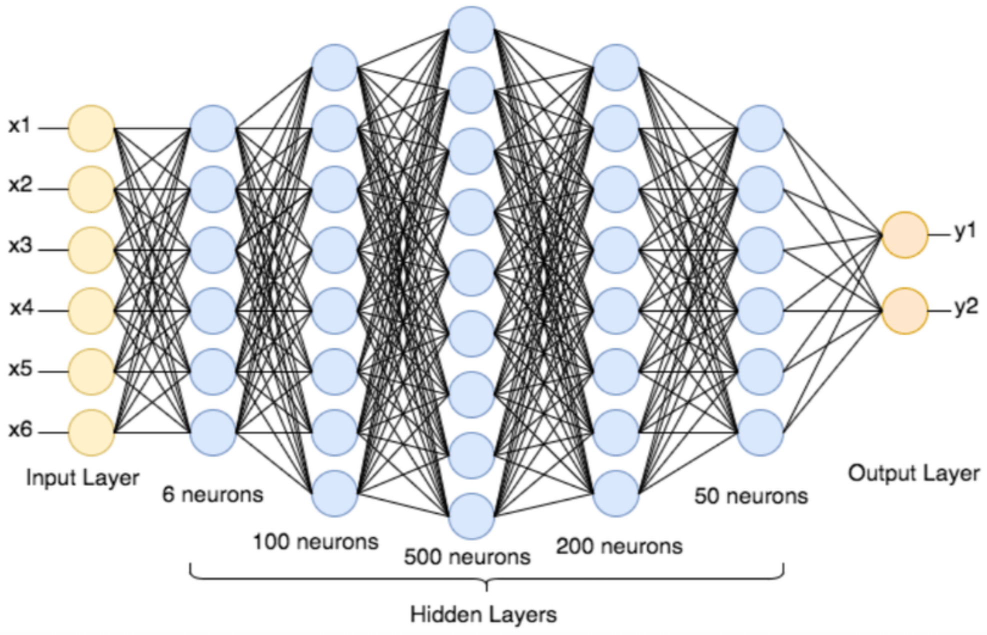


Figure 2.4: Fully Connected Neural Network Structure Example

A network can be split into three sections: the input, the hidden and the output layers.

Each one of these layers is composed of a set of neurons - mathematical functions similar to organic neurons. Once an input layer is determined, weights are assigned. These weights determine the relevance of a given variable, thus defining its importance for the output.

$$\sum_{i=1}^{m} w_\text{i} x_\text{i} + bias = w_1 x_2 + w_2 x_2 + \ldots + wmxm + bias \tag{2.1}$$

$$output = f(x) = \begin{cases} 1, & \text{if } \sum w_\text{x} + b \geq 1 \\ 0, & \sum w_\text{x} + b < 0 \end{cases} \tag{2.2}$$

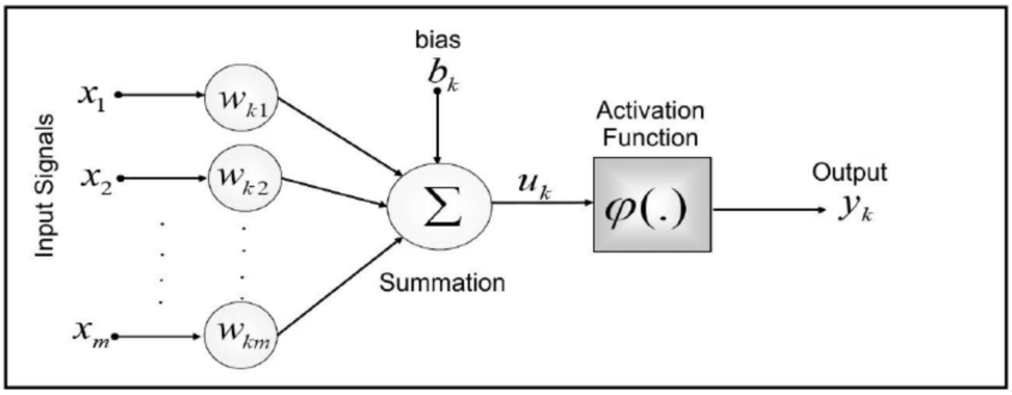


Figure 2.5: Neuron Structure

Further on, the inputs are multiplied by their respective weights and summed (Equation 2.1). The result of this process is then passed through an activation function (Equation 2.2) which determines the final output. If that output exceeds a given threshold, it activates the node, passing data to the next layer in the network (feed forward network). This is the case for a simple multi-layer perceptron, but other types of neural network structures are used in the various deep learning applications.

2.3.2 Convolutional Neural Networks

In this project we will be implementing a type of NN that is often used for computer vision purposes but can be applied with the right techniques to natural language processing, the Convolutional Neural Network or CNN. Fig. 2.6 showcases a diagram related to the employment of a CNN for a NLP task.

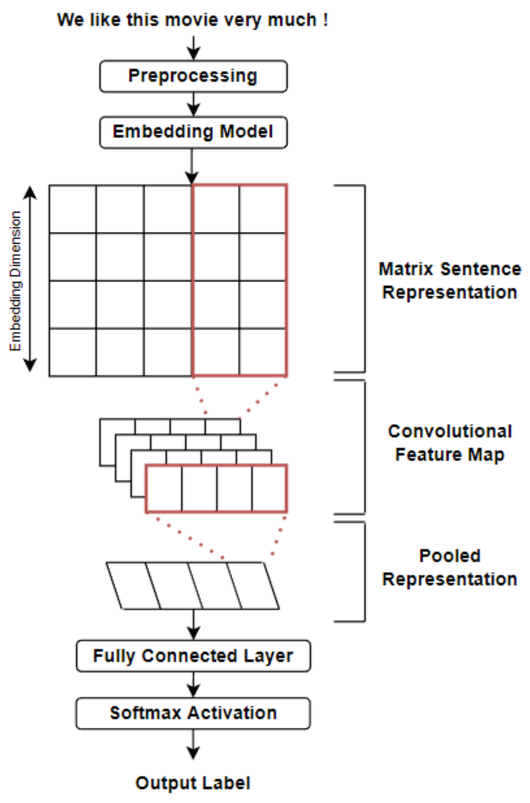


Figure 2.6: Usage of a CNN for NLP example diagram.

Since we are dealing with lyrics, before feeding information to the CNN we would have to express the words in the song with vectors representations. An embedding layer or a pre-trained model is used firstly to receive the sentences and transform them so they can further on serve as input for the CNN.

These networks are able to capture the spatial dependencies in an input through the application of relevant filters. The architecture performs a better fitting to the sequence vectors dataset due to the reduction in the number of parameters involved and re-usability of weights. After receiving the inputs, the filter runs through the feature vectors, from left to right it moves with a certain stride value until it parses the complete width of the data. Then, it goes down to the left of the vector with the same stride and continues the same analysis until the entire frame is traversed and new feature maps are created. In the end, the data goes through an activation function for classification.

2.3.3 Long Short-Term Memory Networks

Long Short Term Memory Network is a more advanced Recurrent Neural Network (RNN), capable of detecting dependencies in sequence prediction/classification tasks. LSTMs store previous information and use it to process the current input, just like a person reading a book remembers the last chapters. LSTMs are considered in a certain way an updated version of RNNs as they can handle long term dependencies by avoiding the vanishing gradient problem - when the gradient has such a small value that stops the weights of the network from changing.

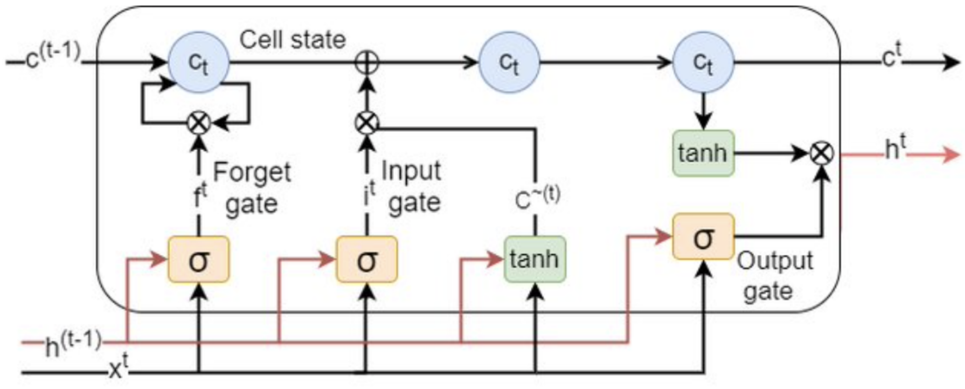


Figure 2.7: Structure of an LSTM unit.

LSTM units can be summarized by a cell state which basically contains the information gain in each unit and three main gates that interact with the later as shown in Fig. 2.7. The forget gate is decides which information is used for calculating the cell state and which data can be discarded. The input gate updates the cell state and helps to find out important information and store certain attributes in the memory that are deemed relevant. The output gate passes the hidden state, this is where the modified cell state is passed through a tanh function and is multiplied with the sigmoid output to calculate the hidden state.

These types of recurrent neural networks have been successfully applied to NLP tasks as they are effective in memorising important information. Fig. 2.8 showcases a simple architecture of a bidirectional LSTM model used for NLP. Words are transformed into their respective embedding representation with a fixed dimension and then given to two layers of lstm units. The hidden states that result from these units are concatenated and fed to a dense layer and processed by an activation function for classification.

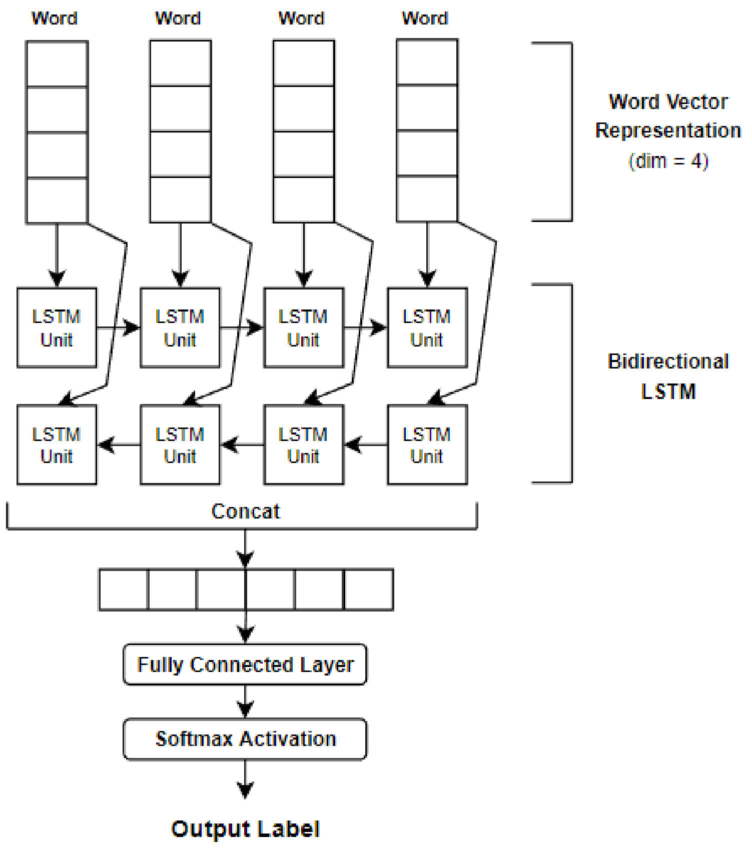


Figure 2.8: Usage of a LSTM for NLP example diagram.

Other methods mainly explore sentences as individual word inputs, thus not adapting to the whole meaning of the sentence and producing predictions that rely more on statistics. By using appropriate embedding and encoding techniques LSTMs can get more information from actual sentences. Another common application in the NLP field would be the Bidirectional LSTMs, which as the name expresses run through the input in both ways therefore using past and future information which can help improve performance in some cases.

2.3.4 Activation Functions

As we've seen, an activation function in a neural network defines how the input is transformed into an output from a node or nodes in a layer of the network. The choice of activation function has a large impact on the capability and performance of the neural network, and different activation functions may be used in different parts of the model.

Commonly researchers use the ReLU, the sigmoid or the tanh activation functions for the hidden layers, a neural network will almost always have the same activation function

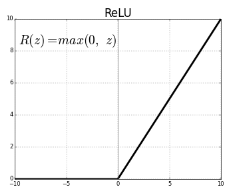


Figure 2.9: ReLU Activation Function

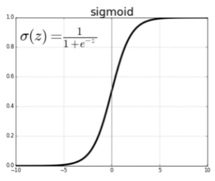


Figure 2.10: Sigmoid Activation Function

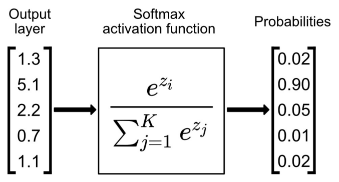

Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

Figure 2.11: Softmax Activation Function

in all hidden layers, it is unusual to vary the activation function through a network model. The type of function used in these layers is often related to the type of network we are dealing with.

- **Multilayer Perceptrons (MLP)** typically use the ReLU activation function.
- **Convolutional Neural Networks (CNN)** are also often constructed with the ReLU activation function.
- In **Recurrent Neural Networks (RNN)** the Tanh and/or Sigmoid activation functions are frequently employed.

For the output layers the linear, the logistic (sigmoid) or the softmax activation functions are the top choices. The linear function is common on regression problems, the sigmoid function is employed mostly in binary classification tasks and the softmax activation function is preferred when it comes to multi-class classification and is the one used in our models.

2.3.5 Loss Functions

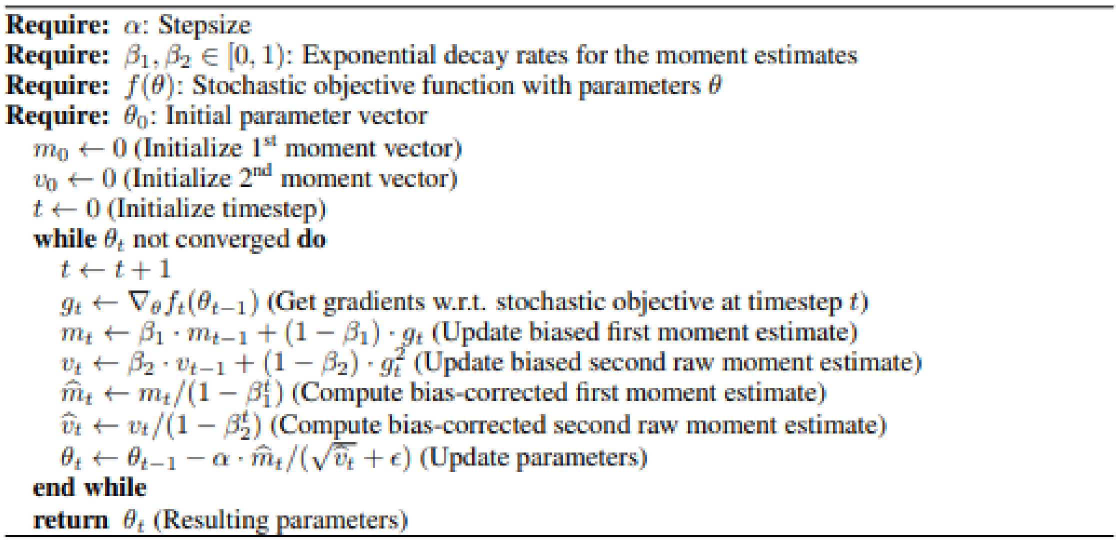
The tuning of the network weights can be seen as an optimisation algorithm, choosing the best set of weights that provides us the most accurate solution. The function we want to minimise or maximise is called the objective function or criterion. When we are minimising it, we may also call it the cost function, loss function, or error function. This function helps us measure the difference between the calculated solution (i.e Arousal and Valence) and the real value (i.e. annotated values). The network updates the weights by computing a gradient of the cost function.

$$Loss = -(\frac{1}{outputsize})\sum y \cdot \log \hat{y} + (1-y) \cdot \log(1-\hat{y}) \tag{2.3}$$

The most common are the Mean Squared Error (MSE, usually applied in regression tasks), the Binary Crossentropy (BCE, for binary classification) and the Categorical Crossentropy (CCE, for multi-class classification). This thesis focuses in classification problems with multiple labels, thus the CCE (Equation 2.3) is the appropriate loss function for our experiments.

2.3.6 Optimisation Algorithm

Optimisation is defined as the search of a set of inputs to an objective function that results in a maximum or minimum evaluation for the given function. There are multiple algorithms used to perform optimisation. In our work, we used Adam (see Fig. 2.12), an algorithm based on stochastic optimisation.



```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Figure 2.12: Adam - algorithm for stochastic optimization, adapted from Kingma and Lei Ba 2017

This method was chosen due to its popularity in the area we are tackling and the fact that it only requires first-order derivatives (gradients) to choose the direction to move in the search space, thus being more memory efficient when compared to other methods.

2.3.7 Transfer Learning

Transfer learning (TL) is a machine learning method commonly applied to deep learning approaches that involves using a previously developed model as foundation for a new one that is created to solve similar problems. TL has its origins on the observation that humans can apply previous knowledge to solve new problems faster and intelligently, and it has been proposed to deal in convenient ways with the problematic situation of having training and test samples derived from different feature spaces and distributions.

This methodology is often used as the starting point on computer vision and natural language processing (NLP) tasks given the vast compute and time resources required to develop neural networks designed for these problems. TL is noticeably important in the beginning of a NN, helping in skipping usually expensive and irrelevant learning phases (Fig. 2.13).

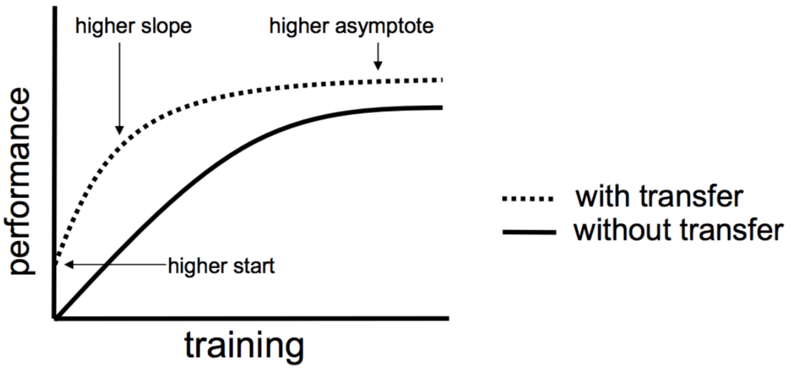


Figure 2.13: Learning curve with and without TL.

Previous studies in the speech emotion recognition field show that emotion recognition models previously trained on a specific speech corpus have the tendency to perform better to that same corpus than to new data. This is due to the characteristics of the speakers in each corpus, as well as the type of emotions being conveyed, the level of portrayal, among others. Because of this, TL has recently started to be applied to this area due to its multiple advantages in dealing with mismatches between training and test data sets. Pandeya and Lee (Pandeya and J. Lee 2020) proposed a supervised music video emotion dataset for a data-driven algorithm and used late feature fusion of audio and video representations after transfer learning.

Speech emotion detection is at the core of lyrics-based MER and previous research using denoising autoencoders (DAE), simple recurrent networks (SRN) and long-short term memory networks on speech and music emotion recognition have shown that using TL from this area of knowledge can be beneficial for the performance of DL models (Coutinho, Deng, and Schuller 2014).

BERT

Recent advances in Deep Learning approaches provided better results in the LMER field but this is still certainly a challenging task. Publicly available datasets are scarce and most of them are small in scale which makes it hard to train DL models in such low amount of information due to overfitting. Previous research (Devlin et al. 2018) highlighted the effectiveness of fine-tuning pre-trained models such as the Bidirectional Encoder Representations from Transformers (BERT), for specific tasks.

BERT is designed to pre-train deep bidirectional representations from unlabeled text. The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art classification models for a wide range of tasks. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (see Fig. 2.14). The input is embedded into vectors and processed in the neural network. Classification tasks related to sentiment analysis require an extra layer on top of the Transformer output to assign the labels.

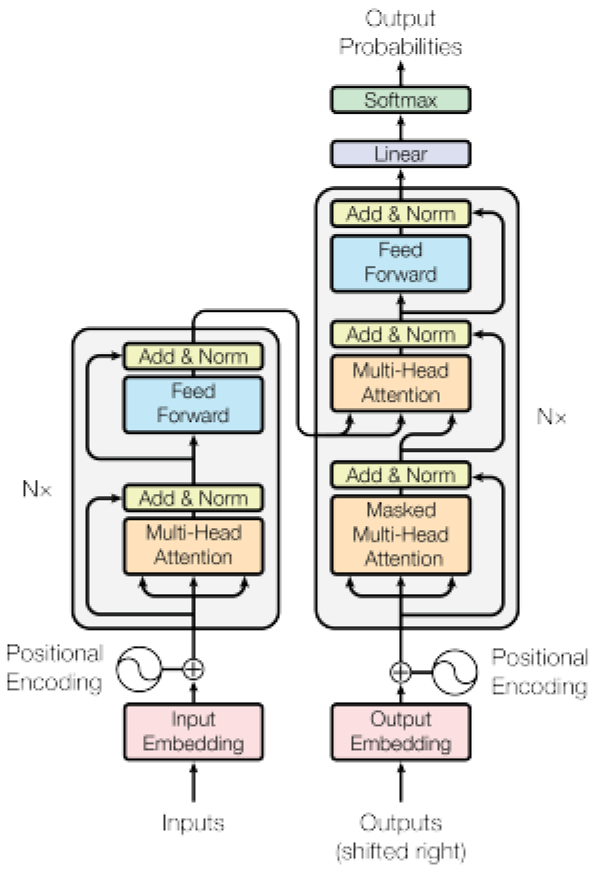


Figure 2.14: Transformer model architecture adapted from Vaswani, Shazeer, Parmar, Uszkoreit, Jones, A. Gomez, et al. 2017

Siriwardhana et al. 2020 used two pre-trained “BERT-like” architectures (Speech-BERT for the signal component and RoBERTa to process the textual information) to solve multi-modal emotion recognition. After tokenizing both the speech signals and text, the data was passed through the two pre-trained models and further on concatenated and sent through a fully connected and softmax layers for classification. Four emotion categories were defined: happy, sad, angry, and neutral. The experiments were conducted in different datasets, first four ablation studies were made for the IEMOCAP dataset Busso et al. 2008 obtaining an average of 84.68 F1-score for the four emotion labels by only using the RoBERTa model. Then, one of the largest multi-modal emotion recognition datasets available at the time (CMU-MOSEI, Zadeh et al. 2018) was used to train the models which obtained 88.08 F1-score. This work clearly shows that applying pre-trained models in the emotion recognition field is an advantageous process and can achieve very good results.

GPT-3

GPT-3, or "Generative Pre-trained Transformer 3" (Radford et al. 2018) was created by OpenAI and it consists on a large model that was pre-trained using one of the largest text corpuses ever (around 499 billion tokens or 2 trillion characters). The information fed into this model was mainly gathered from the textual data available on the world wide web.

GPT-3 has a straight forward interface, when fed a textual sequence it outputs the predictions. This model provides the means to achieve significantly good results on multiple tasks but one must learn how to specify the right inputs in order to extract the right knowledge for the problem at hand.

This tool has been used for text-classification in previous research, a study made with large pre-trained models for text classification was done and a benchmark was set - the RAFT benchmark (Real-world Annotated Few-shot Tasks, Alex et al. 2021). The analysis was done in multiple datasets for different purposes, the information included newspaper articles, law corpus, bibliographic databases and tweets among others. The results were impressive, GPT-3 scored first in the list of models of its kind with an average f1-score of 62.7, achieving better scores than its predecessors and only losing to human crowd-sourced predictions. At the time, the usage of this model for emotion recognition and sentiment analysis tasks is scarce but the capabilities of this large pre-trained algorithm can have a beneficial impact in the field.

2.3.8 Data Augmentation in NLP

DL models are only as good as the data they are fed, information augmentation techniques have been a huge step in increasing the amount of relevant data in smaller datasets. Data augmentation is defined as a collection of methods that artificially increase the amount of data by creating new data points from existing samples. The quantity of data available for the model has an heavy performance impact.

The goal here is to create additional synthetic data from the original samples. In NLP data augmentation should be done carefully due to the grammatical structure of the text. This thesis experiments only generate augmented data from the samples that are used to train the models as explained in Chapter 4.

There are multiple data augmentation methods available in NLP, in this thesis the focus revolves around synonym replacement (replacing a given number of words from a sentence with one of its synonyms chosen at random). The later is performed via word embeddings (numerical vector representations of words) as this is the most commonly used and effective path to find a synonym for a chosen word.

Non-contextual Word Embeddings

Word embeddings can be divided in contextual and non-contextual. Traditional word embeddings aim to learn a continuous representation for each single word in a document. These embedding models build a global vocabulary using unique words in the documents by ignoring the meaning of words in different context. Similar representations are learnt from the expressions that are most frequently close to each other in the documents. The typical models used for non-contextual embedding are Glove, word2vec and fastText.

Contextual Word Embeddings

Google BERT and related versions are the dominant choice for contextual embeddings. These consider the sequence of all of the words in the document to learn sequence level semantics. Contextual methods have the advantage of learning multiple meanings for the same word when in different situations (e.g. in "I left you at the left side of the bank" the word "left" has two distinct interpretations).

2.4 Static LMER

This section reviews and describes the current progress in LMER and provides information in previous work while also giving some insights on related approaches applied in different contexts that may prove useful for future work. During our investigation we have found other studies that could've been relevant for our project but due to the lack of information provided by the authors were discarded. We have also abdicated researches that didn't provide consistent results or were based on similar architectures of the ones we are about to showcase but didn't properly explore the possibilities and limitations of these models, thus providing misleading, redundant or unimportant information for our work.

Table 2.2 contains previous work related to LMER that is further on explained in detail.

Author	Approach	Database	Model Input	Models	Emotion Taxonomy	Results
Yang and Lee	Traditional ML	1032 lyrics from AllMusic	182 features from the Harvard General Inquirer IV.4 model	Decision Trees / Rule-based methods ensembles	Psychological model of 23 emotions	67.0% Accuracy
Malheiro et al.	Traditional ML	MER 180 lyrics	Novel Features based on content, structure, style and semantics (e.g. unigrams, POS tags, GI and LIWC)	SVM	Russell AV	77.1% F1-score
Malheiro et al.	Traditional ML	MER 180 lyrics for training/ AllMusic 771 lyrics for validation	Features based on content, structure, style and semantics (e.g. unigrams, POS tags, GI and LIWC)	SVM	Russell AV	73.6% F1-score
An et al.	Traditional ML	3552 Chinese/English lyrics extracted from Baidu web	Segmented lyrics	Naive Bayes Classifier	Thayer Emotion model (4 classes)	68.0% Accuracy
Xu et al.	Traditional ML	3839 popular chinese songs annotated by 87 uni. students.	98 Features extracted using SC-LIWC	RFR-based model	Russell AV	0.147 RMSE Arousal 0214 RMSE Valence
Abdillah et al.	Traditional ML	MoodyLyrics	Word embeddings extracted with the GloVe model	Naive Bayes Classifier	Russell AV	87% F1-Score
Parisi et al.	DL	Synchronized Lyrics Emotion Dataset	Word embeddings extracted with ELMo	LSTM	Hevner Emotion model (5 classes)	74.86% Accuracy 50.26% F1-score
Parisi et al.	DL	Synchronized Lyrics Emotion Dataset	Word embeddings extracted with fastText	Bi-LSTM	Hevner Emotion model (5 classes)	80.61% Accuracy 66.41% F1-score
Abdillah et al.	DL	MoodyLyrics	Word embeddings extracted with GloVe	Bi-LSTM	Russells AV	91% F1-score
Agrawal et al.	DL	MER 180 lyrics	Preprocessed Lyrics	Bidirectional transformer (XLNet)	Russells AV	88.60 % F1-score Quadrants 93.98% F1-score Valence 88.89% F1-score Arousal
Agrawal et al.	DL	MoodyLyrics	Preprocessed Lyrics	Bidirectional transformer (XLNet)	Russells AV	94.77% Quadrant

Table 2.2: LMER approaches

2.4.1 Classical Machine Learning Approaches

Traditional machine learning approaches (Fig. 2.15) are always deeply connected with a need to discover and create informational relevant features. In the MER field, it is no different, with a lot of research made on the audio features, researchers started to go deeper on LMER, creating novel features that could help in the development of models with performances close to the ones trained with audio signals. These studies created state of the art results for future research and even helped improve bi-modal analysis.

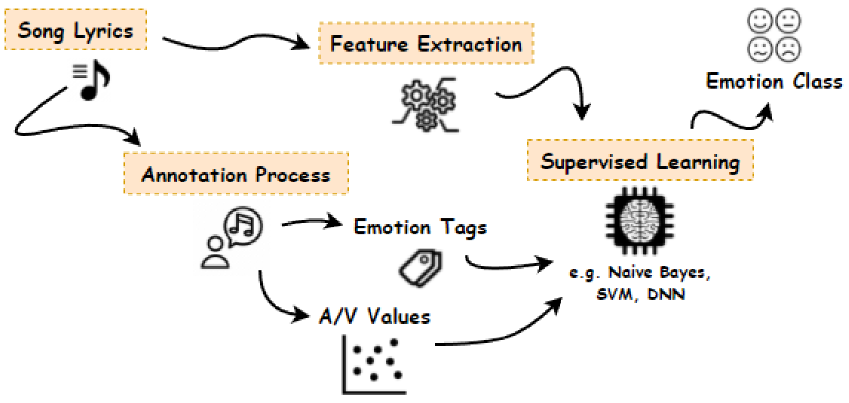


Figure 2.15: Traditional ML approach to MER diagram.

Yang and Lee (D. Yang and W. Lee 2010) used decision trees and rule-based methods ensembles to create a human-comprehensive model and identify emotion in lyrics. They gathered 1032 songs from the allmusic.com repository and extracted 182 features form the General Inquirer Harvard IV.4 model (Stone et al. 1966) to use as input for their algorithms as these features would help to reduce the dimensionality in contrast to the bag-of-words approach. For classification purposes the authors chose the psychological model of 23 emotions (Tellegen, Watson, and Clark 1999). The data was split into train and test and ran through a 10-fold cross-validation process with the chosen model. The results were impressive considering the 23-class classification problem at hand, the authors managed to achieve an accuracy of 67 percent with a fairly human comprehensive model.

Malheiro et al. (Malheiro et al. 2018) constructed a dataset by collecting 180 song lyrics annotated by 39 individuals coming from different background and education levels following the Russell's circumplex emotion model. Novel features concerning the content, structure, style and semantics of the lyrics were extracted and further ran through a selection process. The best features were used for classification, performed using Support Vector Machines (SVM) with grid search for parameter optimisation. Several experiments were made resulting on a mixed classifier (with the best models by feature type) that reached 77.1% F1-score for quadrant classification. This approach was then validated on a 771 lyrics dataset extracted from the AllMusic platform and still obtained 73.6% F1-score which showed how well the model could adapt to new samples.

An et al. (An, Sun, and Wang 2017) conducted a research using Naïve Bayes Classifier with four classes from the Thayer emotion model and got 68% accuracy. The study used a collection of 3552 song labels extracted using crawling in the web of Baidu music, the lyrics were from both English and Chinese music pieces. Abdillah et al. (Abdillah, Asror, and Wibowo 2020) used the same Naïve Bayes Classifier but with word embeddings extracted from the MoodyLyrics dataset using the GloVe method. The results of the model with word embeddings were surprisingly better, achieving an F1-score of 82% for Russell quadrant classification, making clear that feature extraction has an heavy impact on traditional approaches performance.

Xu et al. (Xu et al. 2021) attempted to predict values for valence and arousal combining both audio and lyrics from a dataset containing 3839 popular chinese songs manually annotated by 87 university students. In total, for each song 98 features were extracted using SC-LIWC (Gao et al. 2013). For the algorithms the authors decided to use multiple linear regression (MLR) and random forest regression (RFR) since the labels are continuous, making this a problem fit for regression (predicting values based on previous samples). Grid search was performed to achieve optimal parametization and cross-validation was used for model evaluation.

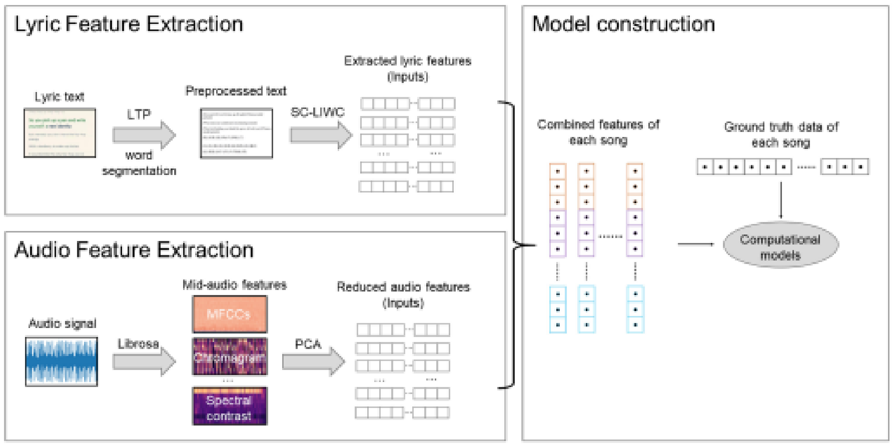


Figure 2.16: Xu et al. proposed architecture (adapted from Xu et al. 2021).

As we can see in Fig. 2.16 the research analyses training both combined data (audio and lyrics) and solo information (audio or lyrics only). Results showed that when it comes to arousal, lyrics are not nearly as relevant as audio but for the perceived valence of music, textual information becomes as relevant as audio features. This is a common result as even humans most often perceive music intensity better through sound but this weakness in lyric information can also be caused by poor feature extraction and selection, in my opinion the authors didn't explore much on this topic.

From this short review, we can see that traditional machine learning approaches rely heavily on feature extraction and selection processes. This can be a very time-consuming and expensive task and it is often challenging to find features that convey the relevant information the model needs in order to perform well on a specific assignment. We have found out a pattern in most of the classical approaches from the literature that disregard this crucial stage, these tend to fail in achieving decent results.

2.4.2 Deep Learning Approaches

Parisi et al. (Parisi et al. 2019) decided to use convolutional, long-short term neural networks for lyrics and audio emotion classification. Different text embeddings were tested and models like fastText, ELMo (Peters et al. 2018) and BERT were used for contextual feature extraction. A new emotion dataset containing synchronised lyrics and audio was created and the authors decided to follow the Hevner emotion representation, labelling their data with 5 different adjectives (Sad, Joy, Fear, Anger, Disgust). One of the main aspects of this work is that the text and audio data were synchronised, thus establishing a connection between the lyrics and its corresponding audio interval. Focusing on the lyrics component (Fig. 2.17), the authors used ELMo and BERT to extract the word embeddings and fed them into an LSTM layer followed by two dense layers for the first one or simply two dense layers for both (having the output layers softmax activation for prediction).

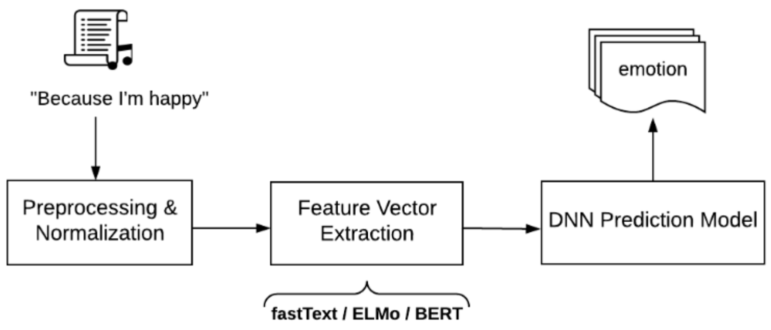


Figure 2.17: Parisi et al. proposed architecture (adapted from Parisi et al. 2019).

The results for these models were not that impressive, ELMo embbedings combined with LSTM layers was the best overall scoring 74.86 percent accuracy but only 50.26 f1-score. The open-source text representation library fastText was also used to get a fixed portrayal of text and this is where the work got interesting, by combining the later representations with bidirectional LSTM and attention-based classification layers (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, A.N. Gomez, et al. 2017) an accuracy of 80.81 percent and an f1-score of 66.41 were achieved.

Abdillah et al. (Abdillah, Asror, and Wibowo 2020) also proposed a Bidirectional Long-short Term Memory (Bi-LSTM) deep learning approach with word embeddings extracted from the MoodyLyrics dataset using the GloVe method to classify music emotions. The study follows Russells emotion model and the Bi-LSTM model with dropout layer and activity regularisation produced an F1-score of 91%. The authors also performed tests with regular LSTMs and CNNs but the results for both of these models felt 1% short of the ones achieved with the Bi-LSTMs. In this research a lot of experiments were conducted to evaluate the best parameters for the Bi-LSTM, the authors have discovered that the dropout, the activity regularisation and the learning rate decay parameters can reduce the difference between training loss and validation loss by 0.15. This is a relevant aspect as a large difference between the training and validation losses is related to overfitting making the choice of the parameters crucial to avoid it.

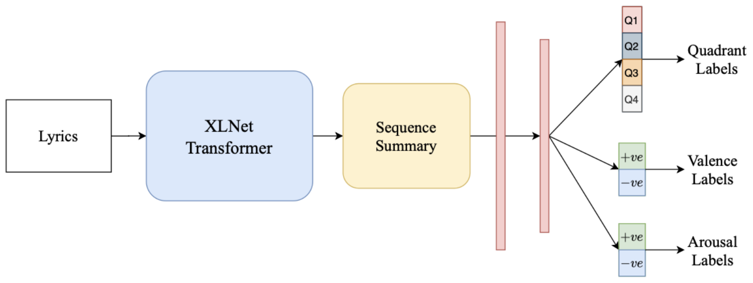


Figure 2.18: Agrawal et al. proposed architecture (adapted from Y. Agrawal, R. Agrawal, and Alluri 2021).

Agrawal et al. (Y. Agrawal, R. Agrawal, and Alluri 2021) developed a transformer-based approach for LMER that achieved the impressive F1-scores of 94.77% for quadrant classification on the MoodyLyrics dataset and 88.60%, 93.98%, 88.89% for quadrant, valence and arousal respectively on the 180 lyric dataset created in our lab (Malheiro 2017). Classification was performed using the four quadrants of Russells emotion model. The architecture was based on a large bidirectional transformer named XLNet (Z. Yang et al. 2019) that the authors describe as an improvement upon BERT in the retrieval of contextual information. The transformer outputs raw hidden states, which are processed and passed on to an hidden fully-connected layer which encodes the information and branches out into three complementary tasks for classification of quadrant, valence, and arousal separately. Such impressive results showcase the possibilities of DL approaches and set a new level for future research.

2.5 LEVD

When talking about lyrics-based emotion variation detection we're referring to text emotion recognition applied to the musical context. The main approaches related to this topic refer this as sentence/verse classification problem. Research is still scarce and in our investigation we have found a lot of consistent work applied to similar fields and it is unavoidable to mention these techniques as they are going to be laying the foundations for our future work. Table 2.3 showcases the most popular approaches selected as candidates to be applied in this type of task.

textbfAuthor	**Approach**	**Database**	**Model Input**	**Models**	**Emotion Taxonomy**	**Results**
Malheiro	Traditional ML	44 song lyrics / 951 song lyrics	Keyword Processed Sentences	Keyword-Based /SVM	Russells AV	67.35% F1-Score
Ge et al.	DL	SemEval-2019 Task 3	Word Embeddings	Bi-LSTM/CNN	3 Emotion Classes (Happy, Sad, Angry)	75.42% F1-Score
Ma et al.	DL	SemEval-2019 Task 3	Word Embeddings	Bi-LSTM/LSTM	3 Emotion Classes (Happy, Sad, Angry)	75.57% F1-Score
Ragheb et al.	DL	SemEval-2019 Task 3	Word Embeddings	Bi-LSTM	3 Emotion Classes (Happy, Sad, Angry)	75.82% F1-Score
Xiao	DL	SemEval-2019 Task 3	Ekphrasis tool raw pre-processed text	Pre-trained models (e.g. BERT, GPT-3, ULMit)	3 Emotion Classes (Happy, Sad, Angry)	76.86% F1-Score

Table 2.3: LEVD approaches

2.5.1 Classical Machine Learning Approaches

Malheiro (Malheiro 2017) had an important contribution for the groundwork in this field as he proposed a sentence-based approach for LEVD. In his work he states that sentences are the basic information units of any document, thus the document level emotion detection method depends on the emotion transmitted by the individual sentences of that specific document. Lyrics are created by grouping individual units of information such as verses (poetry) or sentences (prose) and by analysing each one of these units individually we can have a better grasp of the emotion conveyed by the whole song. For his work Malheiro collected 44 song lyrics and created a web application in order to facilitate the annotation process which was performed by labelling 330 sentences into 5 categories - 4 of them being each one of the quadrants from Russells model and 1 for a neutral emotion. Repeated sections and sentences with neutral emotion were discarded and the dataset ended up with 239 sentences - 86 for quadrant one, 67 for quadrant two, 47 for quadrant three and 39 for quadrant four.

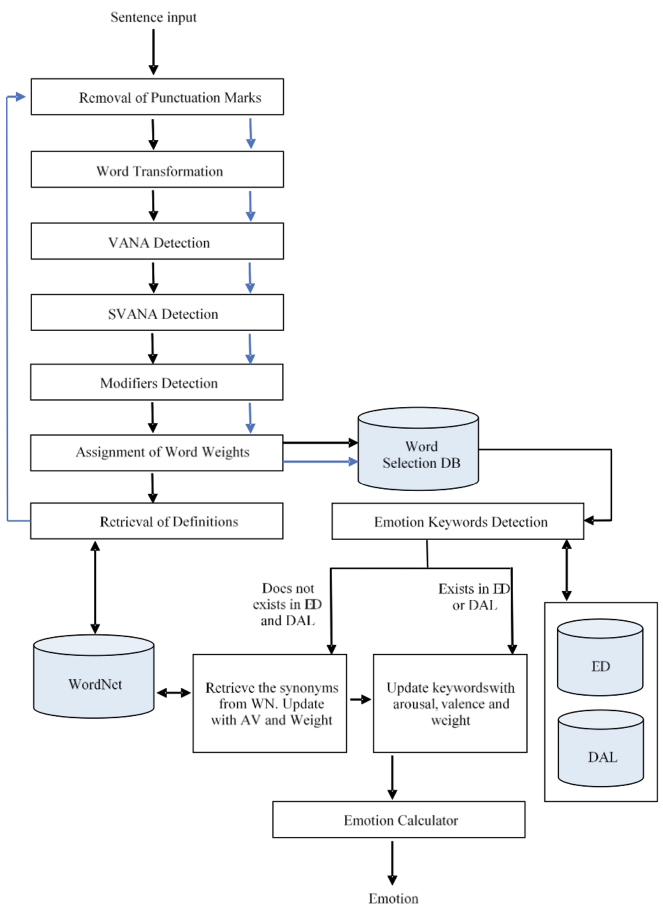

Sentence input
Removal of Punctuation Marks
Word Transformation
VANA Detection
SVANA Detection
Modifiers Detection
Assignment of Word Weights
Word Selection DB
Retrieval of Definitions
Emotion Keywords Detection
Does not exists in ED and DAL
Exists in ED or DAL
WordNet
Retrieve the synonyms from WN. Update with AV and Weight
Update keywordswith arousal, valence and weight
ED
DAL
Emotion Calculator
Emotion

Figure 2.19: Malheiro SERM proposed architecture (adapted from Malheiro 2017).

After the definition of the training and validation datasets Malheiro proposed a Keyword Based Approach (KBA - the Sentence Emotion Recognition Model (SERM) architecture that is shown in Fig. 2.19. Each sentence starts by running through a whole transformation process where words are treated one by one after which weights are assigned and values of valence and arousal are extracted using an Emotion Dictionary (ED) and the Dictionary of Affect in Language (DAL) (Whissell 1989). This is a complex process but it is well documented in the thesis Malheiro 2017. With this SERM approach, Malheiro was able to achieve a training F1-score of 70.82% and further on validated his model obtaining 67.35% which came relatively close to the training performance. The KBA was later compared with a classical ML model trained with a 951 lyrics dataset that resulted by combining the 180 lyrics and the 771 lyrics from AllMusic datasets. The best result that was achieved, testing on the 239-sentences dataset, was an F1-score of 52.7% which was far away from the 67.35% obtained by the KBA.

There are many approaches for text emotion recognition based on keywords and traditional learning methods applied to other tasks but as our work is specifically based on DL classification of music lyrics and the work we have reviewed in this section has similar goals and is heavily connected with our project we will consider this the baseline for our future endeavours in LEVD and start exploring available DL approaches.

2.5.2 Deep Learning Approaches

Ge et al. (Ge et al. 2019) proposed a deep learning model for emotion recognition in textual conversation which is showcased in Fig. 2.20. The approach was trained and tested using the SemEval-2019 Task 3 dataset which is made out of 38424 textual dialogues (Chatterjee et al. 2019). Three pre-trained word embedding models were used, which are word2vec-twitter (Godin et al. 2015), GloVe (Pennington, Socher, and Manning 2014), and ekphrasis (Baziotis, Pelekis, and Doulkeridis 2017), were used. The embedding layer is fed into a Bi-LSTM followed by an attention layer and a CNN. The outputs of the Bi-LSTM and the CNN are concatenated and global max-pooling is applied. From this point, a softmax activation computes the predictions. By combining the outputs of Bi-LSTM and CNN layers, the model was able to learn local features as well as long-term features, the model resulted in an average F1-score of 75.42% for classification in an emotion model composed of three classes (Happy, Sad and Angry).

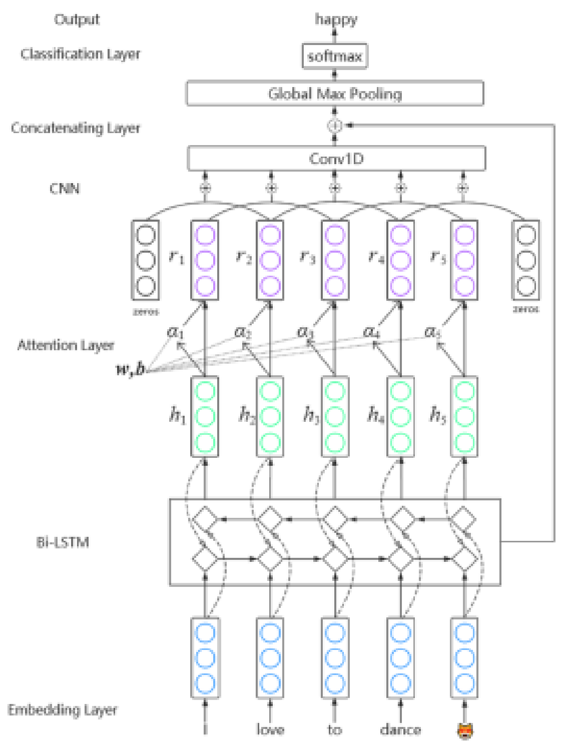


Figure 2.20: Ge et al. proposed architecture (adapted from Ge et al. 2019).

Ma et al. (Ma et al. 2019) developed a DL model with the same purpose as Ge et al. with the same dataset (SemEval-2019 Task 3). Emojis were replaced with words that would describe them the best in order to include this type of information in the word embeddings as these wouldn't accept this kind of special characters. The embeddings are fed into a Bi-LSTM layer, while an attention mechanism modifies the weights of the emotional relevant words. The inner product is taken from the output of the Bi-LSTM and the attention weights and fed into another Bi-LSTM and a pooling process is made in the outputs of this layer. The pooling scores are sent into a regular LSTM layer and classified using a softmax activation function. On average this approach produced an F1-score of 75.57% for classification in three emotions (Happy, Sad and Angry).

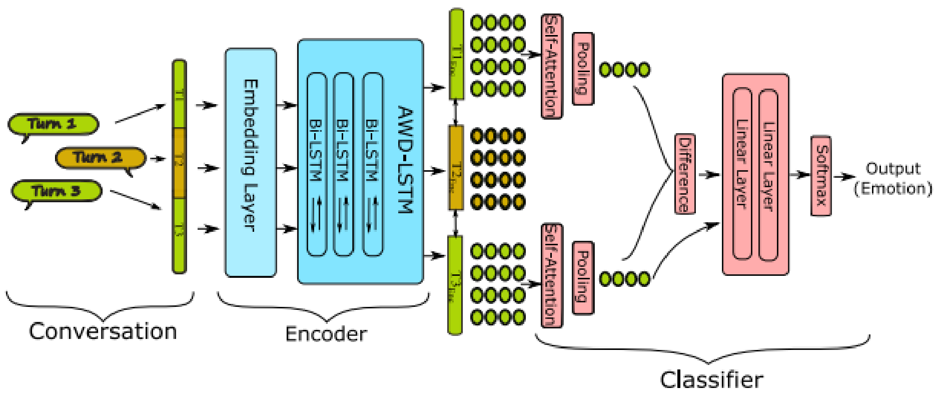


Figure 2.21: Ragheb et al. proposed architecture (adapted from Ragheb et al. 2019).

Ragheb et al. Ragheb et al. 2019 has implemented the structure presented in Fig. 2.21. The work was also based in the SemEval-2019 Task 3 dataset. The conversation data was concatenated and inputted into an embedding layer and fed into three consecutive Bi-LSTM layers trained by average stochastic gradient descent. After this process, an average pooling was applied after a self-attention mechanism and the difference between the pooled scores is taken as an input for softmax layers to obtain the emotion labels. The results exhibited low performance in the recognition of happiness which was the label out of the three possibilities with most erroneous predictions. Overall the model scored 75.85% F1-score.

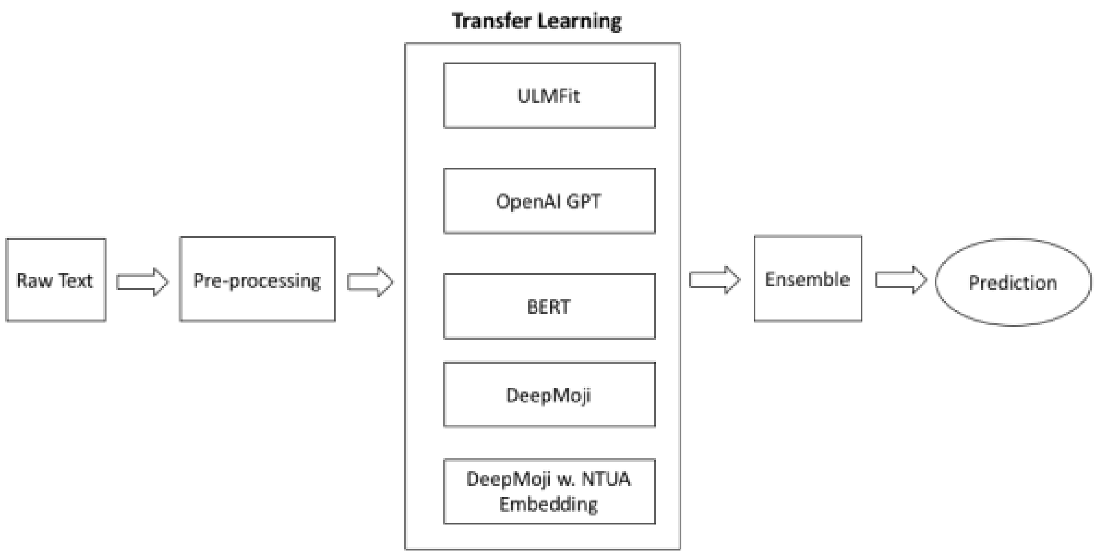


Figure 2.22: Xiao proposed architecture (adapted from Xiao 2019).

Xiao Xiao 2019 created an interesting study based on pre-trained models. This work is key for our future research as we are planning to implement this kind of design to our approaches. The models acted in the same dataset as the approaches stated before (Ge et al. 2019, Ma et al. 2019, Ragheb et al. 2019) which is highly important to establish a good comparison between standard deep learning approaches and pre-trained models. Fig. 2.22 demonstrates how the author designed the structure for his work. The ekphrasis tool (Baziotis, Pelekis, and Doulkeridis 2017) was once again used for pre-processing the text. Which makes this work even more important is that the researcher fine-tuned all the models used and also tested the combination of them at once. The best results of 76.86% F1-score were obtained by producing an ensemble and combining the strengths of the ULMFiT (Howard and Ruder 2018), the BERT (Devlin et al. 2018) and the Open-AI's GPT (Radford et al. 2018) models. This work showed that large pre-trained models can indeed be fine-tuned for text emotion recognition and are able to create state-of-the-art results.

LEVD is still an unpopular research topic in recent times but from our investigation, we believe that structures based on LSTM units that can exploit the dependencies between the elements of a verse and large pre-trained models that provide infinite possibilities, can create good DL foundations and provide new state-of-the-art results. The limitations on this field come from the lack of quality datasets, therefore our future work also involves a raise in the amount of samples in our current data, data augmentation techniques can be a necessary component when dealing with this kind of task.

Chapter 3

Static LMER

This section aims to give insights about the experiments made for LMER. Different approaches to the problem are explored with multiple classification model architectures and inputs.

3.1 Data

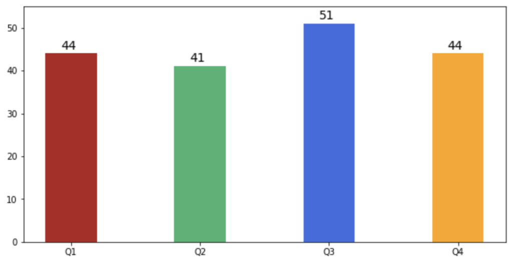
This section presents the datasets used and the way they were handled in order to be used for the static LMER models.

3.1.1 Database - MIR Lyrics Emotion (2016)

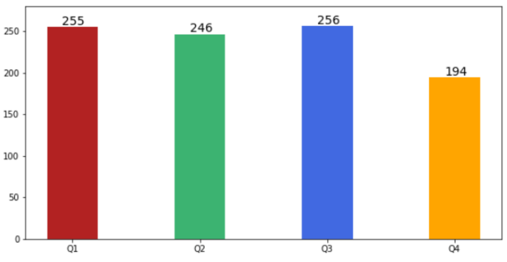
As this work is a continuation of previous research made by Malheiro (Malheiro 2017), we had at our disposal the data employed on his project. Specific information can be found in the thesis but to briefly summarise it: this dataset contains two parts, the first being a 180 lyrics dataset manually annotated with arousal and valence values based on Russell's emotion taxonomy and the second a 771 lyrics dataset annotated in the 4 quadrants of the same model, based on AllMusic tags. As both parts were carefully analysed in previous research, no critical flaws were noticed and no changes were performed when it comes to the amount of lyrics available. For this work we considered the 180 lyrics set as the most relevant, this is due to the fact that most experiments on previous work were specifically made on this set and also the way that it was built. Experiments for static LMER within the scope of this thesis were performed in the 180 lyrics set as well as in a 951 lyrics set that originated in the aggregation of both parts of the full dataset available (771 + 180 lyrics).

Quadrant Distribution

In the study we have performed classification ruled by the four quadrants of the Russell taxonomy. The annotation process for the 180 lyrics dataset was specifically performed by 39 people with diverging backgrounds and educational levels. Their job was to read the



(a) 180 Lyrics Dataset



(b) 951 Lyrics Dataset

Figure 3.1: MIR Lyrics Emotion (2016) Quadrant Distribution

lyrics, assign values for arousal/valence and identify the predominant emotion expressed without any previous research about the song. Further validation was performed and described on the thesis.

Figure 3.1 showcases the distribution of the songs from both datasets among the four quadrants. The information was reasonably balanced except for the number of songs from the fourth quadrant in the 951 lyrics dataset which deviated from the average.

3.1.2 Traditional Feature Sets

This thesis explored the feature sets suggested by Malheiro (Malheiro 2017), four different feature groups divided by the type of features were used in his traditional approach to the problem: CBF for content-based features, StyBF for stylistic-based features, StruBF for song-structure-based features and SemBF for semantic-based features.

CBF

This group contains features extracted with the bag-of-words (BOW) and part-of-speech (POS) tags methods. 10 feature sets are of this group: 6 for BOW (from unigrams to trigrams) after tokenization with and without stemming (st) and stopwords removal (sw); 4 are BOW (bigrams up to fivegrams) after the application of a POS tagger without st and sw.

StyBF

These features are related to stylistic aspects of the language. Grammatical classes occurrences are collected as well as the number of slang words, the total of words started with capital letters (First Capital Letter - FCL) and the amount of terms with all the letters in uppercase (All Capital Letters - ACL). For this category there are 2 feature sets: the number of occurrences of POS tags in the lyrics and the number of slang and capitalised words (ACL/FCL).

StruBF

Newly created features related to the structure of the lyrics, a few examples are the the number of times the chorus is repeated in the lyric, the amount of occurrences of the title in the lyric, the number of verses or the total of sections (verses and chorus) in the lyrics. Only one feature set was deducted out of this section covering all the structural features.

SemBF

These features describe the semantic aspects of the lyrics. Ricardo Malheiro used features based on previously developed frameworks as stated on the thesis. 4 feature sets were developed: the first with the features from Synesketch (Synesketch n.d.) and ConceptNet (ConceptNet n.d.); the second with the features from LIWC (LWIC n.d.); the third with the features from GI (GI n.d.); and the last with the features from gazetteers, DAL (Whissell 1989) and ANEW (Bradley and Lang 1999).

Feature Sets

Malheiro achieved his best results for quadrant classification with the feature sets present in Table 3.1. Each feature set went through feature selection and ranking with the ReliefF algorithm (Robnik-Šikonja and Kononenko 2003) before being passed to the models. An additional set (FS51) that initially contained 1083 features was built by combining all of the selected features from each one of the feature sets present in the table.

FS (Feature Set)	Feature Description	Group	#Features-#Selected Features
FS11	BOW (unigrams)	CBF	3567-200
FS12	POS+BOW (trigrams)	CBF	4687-700
FS21	#POS_tags	StyBF	34-20
FS22	#Slang+ACL+FCL	StyBF	3-3
FS31	Structural Lyric Features	StruBF	12-11
FS41	LIWC	SemBF	82-39
FS42	Gazetters	SemBF	20-20
FS43	FI	SemBF	182-90
FS51	All	All	1083-609

Table 3.1: Feature set organization

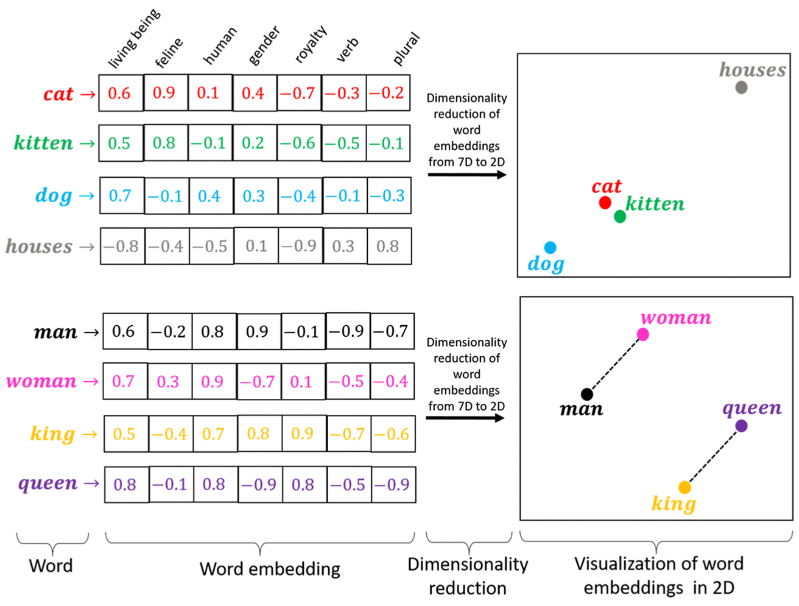
FS (Feature Set)	Feature Description	Group	#Features-#Selected Features
FS11	BOW (unigrams)	CBF	3567-385
FS22	#Slang+ACL+FCL	StyBF	3-3
FS42	Gazetters	SemBF	20-20
FS43	FI	SemBF	189-90
FS51	All	All	597-498

The full range of feature sets previously created by Malheiro was not explored as this was not the main goal of this thesis. In this work we aimed to develop approaches that can automatically recognise relevant information from the provided data, thus reducing the work done in the feature extraction stage. These feature sets were relevant to observe how DL models perform when ran on traditional features.

3.1.3 Word Embeddings

Word embeddings became a popular way of representing text and are considered one of the main breakthroughs for the performance of DL models on diverse NLP tasks. Some of the models used in this project relied on word embeddings, thus it is necessary to explore this concept. This word portrayal technique allows words with similar meaning to have a similar representation and are considered a major breakthrough in increasing the performance of DL models built for NLP tasks.

Fig. 3.2 illustrates one simple way to explain word embedding methods: given word sequences (in our case pre-cleaned/processed lyric corpus) these algorithm output embedding vectors for each unique word. The words with similar vectors (that are closest in space) are most likely to have the same meaning or are used to convey the same sentiment.



living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Figure 3.2: Word vector representation, adapted from Borah 2021.

This simple description does not account for the training process of the models that provide these vector representations. There are many ways to learn word embeddings (Mikolov, Chen, et al. 2013), mostly based on the occurrence of words in the same context. For example, the algorithms can learn similar embeddings for words that appear many times in identical situations by guessing missing words in a huge corpus of text sentences.

There are different algorithms to perform word embedding. In this work we have mainly used the following: Word2Vec (Mikolov, Yih, and Zweig 2013), GloVe (Global Vectors for Word Representation) (Pennington, Socher, and Manning 2014) and fastText (Joulin et al. 2016).

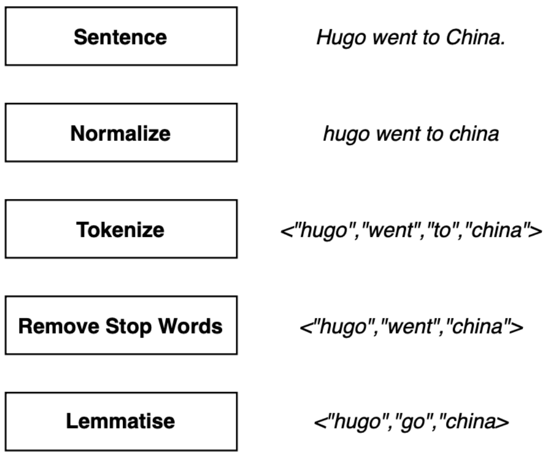


Figure 3.3: Sentence preprocessing.

From the experience obtained in the experiments, it was found that the models performed better when the information was processed before being passed to the embedding steps. For this reason, before creating these word representations the lyrics were modified as shown in Fig. 3.3. Before anything else, each individual corpus was tokenized and the word was considered the standard unit.

The system consisted in removing the "dirt" off each lyric. We started with deleting the punctuation and any number or unusual symbols that appeared. Then the lyrics had to be tokenized to words and in order to reduce redundant information, a set of expressions that occur commonly in the corpus (stop words) are cut out. Finally the data goes through a lemmatization process that aims to stem the words to their root form while trying to conserve their meaning, this is done by using a dictionary that stores the context of each word (e.g. the word "geese" turns to "gees" after stemming and "goose" after lemmatization).

3.2 Methods and Results

This section intends to explain the different static LMER approaches and showcase their results.

3.2.1 Evaluation Methods

To evaluate the performance of our models we chose to use the k-fold cross-validation technique with 10 folds (k=10) and 10 repetitions (one for each test fold), thus assessing the efficiency of the algorithms with ten different information splits.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3.1}$$

When it comes to evaluation metrics we decided to use the F1-score (see Equation 3.1) as it is considered one of the most important metrics for evaluating the capabilities of a given predictor by combining both the Precision and Recall metrics. The F1-Score has been the main metric used by researchers in the last years (as shown previously in Chapter 2) and in order to facilitate the results comparison we chose this as our leading metric. In the aggregation of F1 scores from different classes, weighted F1-score aggregation was employed.

3.2.2 Classical ML

The first step for this problem was to replicate previously referenced results. To achieve this goal we followed the steps shown in the work done by Malheiro (Malheiro 2017) and performed grid search to optimise the parameters of an SVM with a polynomial kernel. The models performance was evaluated with a 10-fold cross validation repeated 10 times. The inputs for this model were based on the feature sets created by Malheiro, these were adapted and shown before (see Table 3.1). We also had available different versions for each feature set, depending on the feature selection made in past research. The best result found was 77.03% F1-Score with FS51 (the combination of 609 features). This procedure was necessary in order to further on compare this result with the DL approaches performances.

3.2.3 Deep Learning

When reviewing different practical approaches to similar problems it became clear that complexity not often means performance in accordance with Occam's razor. With this in mind and considering the computational resources available, the experimental process started with the most basic DL architectures and further on climbed the complexity ladder.

Features

Our first model is one of the simplest yet still effective deep learning algorithms available. It consists in a dense neural network fed with the lyrical features extracted before: CBF, StyBF, StruBF and SemBF.

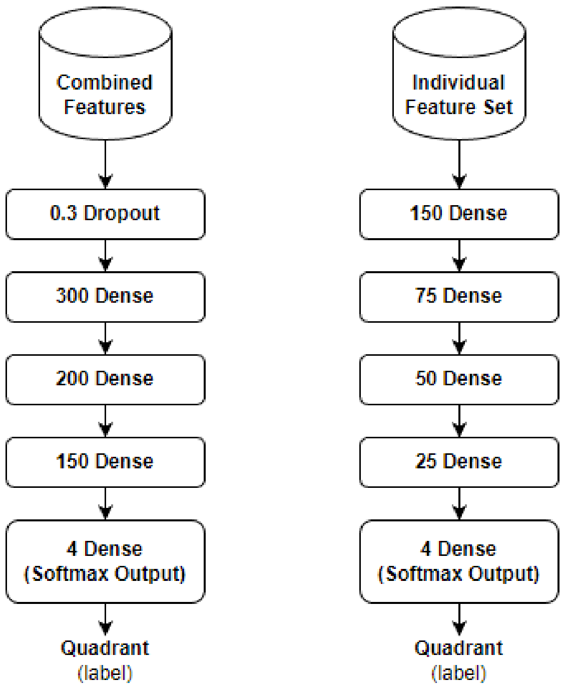


Figure 3.4: DNN models structure diagram.

Fig. 3.4 represents the structure for our best performing models with the combined and individual feature sets. The image itself does not account for the multiple designs tested throughout the experimental process. For the individual feature sets, by varying the amount of hidden layers and respective units we could better adapt the model for each unique set. Classification was performed by taking advantage of a softmax activation layer in the end of the network.

To analyse the possibility of overfitting and have a better understanding of which would be the optimal model we also chose to vary some parameter values, namely the number of training epochs, learning rates and batch sizes. Finding the best parameters for any DL architecture often becomes a challenging task. Combining the number of training epochs with the structure of the model and the learning rate associated with the optimizer was challenging and on top of this we also had to find the most adequate batch size (i.e. the number of samples that the model has seen before updating the network weights).

The main goal was to achieve the best possible performance in the training set while preventing the model from overfitting on test data. This was specially demanding when experimenting in the smaller dataset of 180 lyrics. Lower amounts of training data implies that the model has less patterns available to learn from, making it difficult to classify new samples. Different approaches were tested and it was observed that often feature sets that had less information available would require less complexity and reducing the amount of dense layers would help increase performance.

At first the only features extracted belonged to the 180 lyrics emotion dataset, but as in general DL models tend to need higher volumes of information, we have also tried extracting the same type of features from the 951 lyric dataset. In this version only some of the traditional feature sets could be extracted as some of the original extraction methods were not at our disposal, thus the experiments were made in a mixed dataset of only 498 features. Although less features were available in this set, there was still a considerable amount of additional information to be treated as there were more lyrics at our disposal and models would tend to overfitt more often. In an attempt to treat this issue, a 0.3 dropout layer was added as shown in Fig. 3.4. The purpose of this layer was to randomly ignore 30% of the information from the neurons (by changing them to 0), making them insignificant for the next layer.

Table 3.2 summarises the hyperparameter selection for the experiments. For the 180 lyrics dataset, the training epochs ranged from 10 to 250 in increments of 10 epochs. The batch size was related with the total amount of samples in the training set, in this case, it started in 1 and ended in 130 in intervals of 10 samples. The learning rate with Adam went from 1×10^{-6} up to 1×10^{-1} by shifting through the decimal places (every one fourth of the current decimal place is also tested e.g. 1×10^{-3} shifts to 1×10^{-2} after testing the model with the learning rates equal to 0.25×10^{-3}, 0.50×10^{-3} and 0.75×10^{-3}). On the 951 lyrics dataset, the number of training epochs was the same as well as the learning rate, only the batch size changed, ranging from 1 to 500 with a step size of 25 samples.

Parameter	**180 lyrics (min/step/max)**	**951 lyrics (min/step/max)**
Learning rate (Adam)	$0.1/0.25 \times 10^{-n}/1 \times 10^{-6}$	$0.1/0.5 \times 10^{-n}/1 \times 10^{-6}$
Epochs	10/10/250	10/10/250
Batch size	1/10/130	1/25/500

Table 3.2: Hyperparameters intervals for each dataset

When it comes to individual feature sets, the best results obtained were **77.8%** F1-Score with the Gazetteers features (FS42) in the 180 lyrics dataset and **76.1%** F1-Score with unigrams (FS11) in the 951 lyrics dataset. These were impressive results, considering the small number of features fed into the model in some of the feature sets.

Although with different amount of features available, the mixed feature sets (FS51) were also taken in consideration for both datasets. The model fed with the 498 features extracted from the 951 lyrics dataset reached a top performance of **72.1%** F1-Score. Better results were achieved when testing with the 609 features extracted form the 180 lyrics as this model achieved an F1-Score of **78.3%** which is barely statistically significant result when compared with the 77.03% of the SVM model fed with the same dataset, with a p-value of 0.0381 < 0.05. Table 3.3 showcases the confusion matrix and the F1-Score (both in percentage) per quadrant for this model.

	Q1	**Q2**	**Q3**	**Q4**	**F1-Score**
Q1	20.3	4.3	1.8	0.6	78.2
Q2	3.4	23.2	0.4	1.6	82.5
Q3	0.7	0.1	19.4	3.9	77.9
Q4	0.5	0	4.1	15.7	74.5

Table 3.3: Confusion matrix (percentage values) and F1-Score per quadrant for DNN with 951 lyrics and 498 features dataset

From this experiments solely we started to identify cues of a larger pattern that would be present in most of the tests made in this work: the third and fourth quadrants were the ones the models would struggle to classify the most and the second quadrant rarely had a lower accuracy than the others.

CNN

The second approach is based on a CNN fed with word embeddings. This model was chosen given most of the SOA research on text classification and sentiment analysis mentioned this concept and demonstrated good results for multiple tasks.

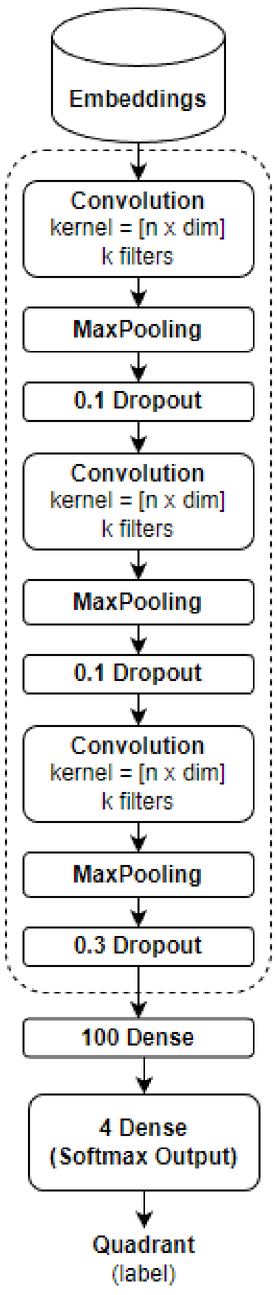


Figure 3.5: CNN structure diagram.

After much experimentation that the process of building an architecture for a deep learning model requires, the CNN structure presented in Fig. 3.5 was the top performer. The word vectors were achieved with three different models: Word2vec, GloVe and fastText. These two dimensional embeddings, which had a height of the size of the embedding dimension and a width equal to the length of the largest lyric in the dataset, pass through three sets of convolutional layers where feature maps are extracted by taking advantage of *n x dim* sized filters (where *n* is an integer between 2 and 5 and *dim* is the dimension of the embeddings). Each one of these layers was directly followed by a pooling process that helps reduce the dimension of the feature maps and feed them to dropout layers. In the final portion of the structure, information is passed through a dense layer with 100 neurons and classification was done by adding a softmax activation layer after the later.

Changing embedding models provided interesting results. This modification allowed us to see the impact of multiple word representations in the final performance of our model. Overall, the CNNs would perform better when fed with fastText embeddings, achieving more than a 2% increase in F1-Score most of the times. Word2vec and GloVe embeddings were toe to toe with each other, one scoring better than the other and interspersing in different scenarios.

The lookout for the best hyperparameters was done by grid search similar to our previous model (see Table 3.2), the learning rates and batch sizes were the same, however the maximum number of epochs tested was exaggerated and could have been reduced as the model started to overfitt after 50 epochs or less most of the times (with some particular exceptions as can be observed further on). Embedding dimensions ranged from 200 to 400 with increments of 50 and **k** (the number of filters in each convolution) was either 16, 32, 64 or 128. The kernel size for the convolutional layers and dropout percentage final values decision process required a considerable amount of time as when these were combined with the change of inputs and hyperparameterization, the performance would be deeply influenced.

	Q1	**Q2**	**Q3**	**Q4**	**F1-Score**
Q1	19.6	4.6	1.9	0.8	73.4
Q2	4.2	20.3	0.3	1.9	76.6
Q3	0.8	0.9	17.3	4.9	71.6
Q4	1.9	0.5	4.9	15.2	67.1

Table 3.4: Confusion matrix (percentage values) and F1-Score per quadrant for fastText+CNN and 951 lyrics dataset

An F1-Score of **68.6%** was the best performance achieved with the 180 lyrics dataset, using Word2vec embeddings. When fed with the 180 instances processed with the GloVe and fastText algorithms, the same model achieved F1-Scores of **61.3%** and **59.1%**, respectively. For the 951 song lyrics the initial idea was to increase the number of filters per layer as in general the more information available in an input the higher the number of filters required in a CNN, but in this case we ended up with **72.2%** F1-Score as our best result (see Table 3.4) with only 32 filters, the main change being the inputs that were produced with the fastText model. This is a statistically significant result when compared to the one produced by the same model with the 180 lyrics, having a p-value of 2.44×10^{-7}. In this case, using Word2vec as the embedding model resulted in **69.6%** F1-Score and when fed with GloVe embeddings the CNN reached only **67.6%**.

Augmented Data

Considering the 180 lyrics as our relevant source of information, it was clear that the scarcity of samples would be an issue. Data augmentation in NLP posed as a necessary resource to experiment on our work. In this field, augmentation is most commonly done by taking advantage of synonym replacement and word embeddings. Six approaches were tested using three non-contextual embedding models (word2vec, GloVe and fastText), a contextual embedding model (BERT), a lexical database where each word is grouped into a set of cognitive synonyms (WordNet) and a sequential algorithm in which lyrics would be augmented by the different algorithms in order.

Before passed to the models, lyrics were preprocessed in order to remove irrelevant characters/words and solely increase the amount of relevant information. The amount of lyrics available was scarce but cross-validation was intended so 6 different splits were made, each one with 120 and 60 lyrics for training and testing, respectively. Augmentation was performed solely on the training sets by increasing the number of training samples by three times the original size (360 lyrics) and the test sets were left untouched (60 lyrics). Each training set would be augmented with one of the six approaches referenced previously as shown in Fig. 3.6.

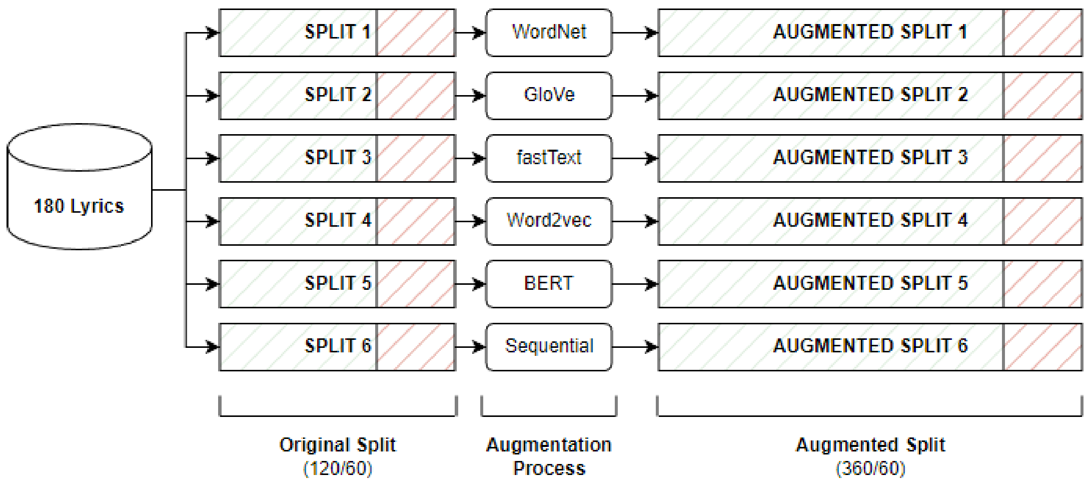


Figure 3.6: Training/Test splits with augmented data

The lenght of each lyric posed a problem, some of the models were computationally expensive and our system would run out of resources and stop the execution of the process many times specially if the amount of new samples requested was too large. By visually observing the new corpus we could identify mistakes in the freshly created lyrics. Often the words replaced by these models wouldn't fit the original sentiment of the lyric and also the amount of redundant information would increase severely in many of the songs. In more rare scenarios some of the words that replaced the original ones came with unusual characters like dots, commas, hashtags, etc., in the middle of them or were straight antonyms of the original words, thus conveying a different emotion in the lyric.

	Q1	Q2	Q3	Q4	F1-Score
Q1	12.2	9.1	3.2	1.4	50.9
Q2	5.3	14.8	0.5	3.8	56.9
Q3	2.4	1.4	13.9	7.3	49.1
Q4	2.1	2.3	8.2	12.1	52.9

Table 3.5: Confusion matrix (percentage values) and F1-Score per quadrant for fastText+CNN and 540 lyrics BERT augmented dataset

The augmented data splits were processed by the three usual embedding models and fed into the CNN model previously shown in Fig. 3.5. When processing the augmented samples with the Glove algorithm the CNN achieved **49.3%** F1-Score and **49.7%** when using Word2vec. The model averaged the best F1-score of **52.5%** when processed with fastText embeddings (see Table 3.5) and the top performing split was achieved when using BERT to create new samples with an F1-Score of 58.6%. GloVe and fastText originated folds both reached 52.4% F1-Score, followed by the WordNet and Word2vec splits with 51.9% and 50.9% F1-Score, respectively. The split built with instances created using the sequential algorithm had the lowest score of 49.4% F1-Score.

Pre-trained Models

Pre-trained models have taken over current research as these algorithms revealed great capability of handling different text classification tasks with great performance. The main issues with most of these architectures are their complexity, computational costs and the fact that many of them do not make it out to the public for testing. Most of the models that are made available often require monetary funds to operate or have limited usage. However, Google has developed an algorithm that has been in the core of their search engine and is available worldwide. Since Google BERT was made accessible, many versions of this algorithm flooded the world wide web in which many were designed for specific tasks. In our experiments for the static classification problem, only the base version and the Robustly Optimized BERT Pre-training Approach (RoBERTa) were tested.

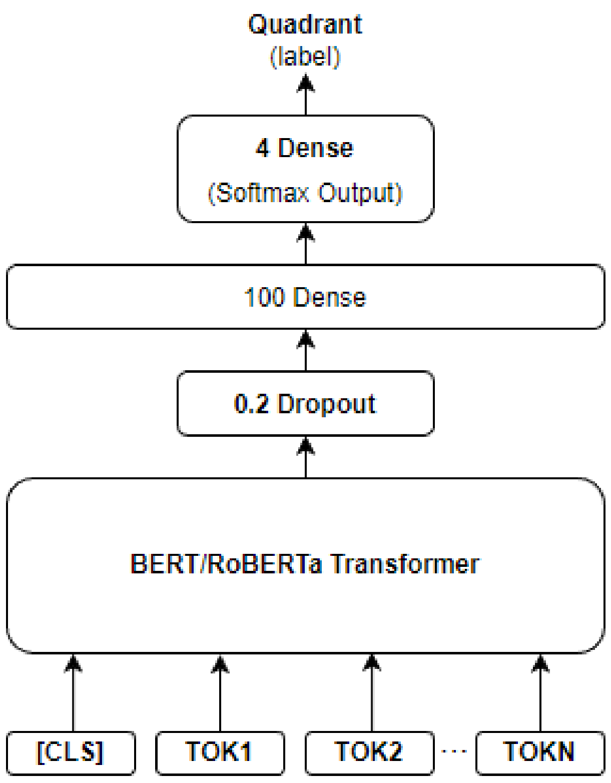


Figure 3.7: BERT model structure diagram.

Both these models have demands regarding the format of the its inputs. The song lyrics had to be tokenized and a CSL token had to be in the beginning of each sample (see Fig. 3.7). After loading the pre-trained weights and passing the information throughout the algorithm, the outputs of the model were processed by a dropout and dense layers before reaching the softmax classification and final layer.

These large algorithms were constantly overfitting and it was useless to train them with a large amount of epochs. Despite the fact that this situation reduced the room for broad testing, it was also an advantageous factor as BERT and its versions, as stated previously, require large amounts of computational power even with smaller datasets. Experiments were performed with no more than 10 training epochs since after the fourth or fifth epoch performance would start to deteriorate.

RoBERTa is often assumed to be an improvement upon BERT because of its dynamic masking, however, our experiments did not properly prove this statement was truthful for all cases. For the 180 lyrics dataset, RoBERTa achieved an F1-Score of **85.6%** while BERT only got up to **83.4%**. It was a different scenario when testing with the larger 951 lyrics dataset where the base version finished off with **88.9%** F1-Score (see Table 3.6), which is statistically significant with a p-value of 1.03×10^{-5} when compared with the RoBERTa model trained with the 180 lyrics. The modified version stuck with **84.2%** for the 951 lyrics dataset.

	Q1	**Q2**	**Q3**	**Q4**	**F1-Score**
Q1	22.6	2.8	0.2	0.2	87.1
Q2	2.7	23.4	0	1.1	87.0
Q3	0.4	0.3	22.1	2.2	91.9
Q4	0.4	0.1	0.8	20.7	89.6

Table 3.6: Confusion matrix (percentage values) and F1-Score per quadrant for BERT and 951 lyrics dataset

The BERT model had an interesting behaviour, other models tested in this work would find it difficult to classify samples from the third and fourth quadrants whereas BERT, as can be seen in Table 3.6, would assign these labels correctly in many more experiments than the other algorithms. The data this model has learned from and used to set its pre-trained weights could be the major influence for this drastic improvement.

3.3 Analysis of the Results

The hyperparameters, results and computation time for each model are illustrated in Table 3.8.

Concerning the DNN models, initial thoughts were to create and test a large diversity of structures, ones more complex than others so that we could take advantage of the multiple feature sets available. Having this amount of feature groups definitely forced us to adapt each DNN to each group. Reducing the amount of layers and their respective neurons would work best in simpler features sets with less features or number of samples. Larger amount of information required more computation and specific techniques to prevent overfitting (i.e. adding dropout layers in the combined feature set, FS51 with 609 and 498 features depending on the dataset, improved the model generalisation capabilities in the test sets).

The CNN model was perhaps one of the most tested structures in this work, having explored multiple hyperparameter combinations and architectures. We could argue that the lack of sufficient quality data for this type of model had an impact on the results.

Different embedding models were tested and their impact was noticeable. One type of embeddings and its dimension would often perform better with specific CNN structures. This made the whole experimental process longer. Just like the majority of the models tested in this thesis, CNNs struggled when classifying the unseen third and specially the fourth quadrant instances which diminished the overall scores. By taking a deeper look into our outputs and analysing the confusion matrices that resulted from our tests, a pattern could be identified: often lyrics from the third quadrant would be classified as belonging to the fourth and vice versa. This revealed that the model would have difficulties distinguishing the sentiment related to each one of these quadrants.

Augmentation was also performed and did not end up improving our CNN results as the lyrics created by these models would deviate from the original sentiment of the song or generate redundant/misleading data, as previously explained. More work should be put in improving this process, splitting the lyrics in sentences, either manually or with an automatic process, and augmenting them individually could improve the quality of the augmented data since some of the models like contextual embeddings take in account more than just the word or part of the lyrics that is being replaced.

Regarding pre-trained models, although these approaches required a large amount of implementation and knowledge gathering time due to their complexity, they also proved to be one of the best possible solutions for the problem at hand. Taking advantage of these architectures led to a substantial increase in the third and fourth quadrant classification accuracy (see Table 3.7) which, because of the way these models are trained, takes us back to the necessity of improving the data we feed our models.

	Q1	**Q2**	**Q3**	**Q4**	**F1-Score**
Q1	22.6	2.8	0.2	0.2	87.1
Q2	2.7	23.4	0	1.1	87.0
Q3	0.4	0.3	22.1	2.2	91.9
Q4	0.4	0.1	0.8	20.7	89.6

Table 3.7: Confusion matrix (percentage values) and F1-Score per quadrant for BERT model

Complex architectures like these have high demands in computational resources and the training time is usually two or even three times higher than with other approaches but no conclusive observations can be taken out of the time stamps recorded in our work as the models were trained in two setups with different specifications due to the lack of GPU availability in the main server.

Model	Input	Kernel Size, #Filters	Emb. Dimension	Learning Rate	Batch Size	#Epochs	F1-Score (%)	Time (min)
CNN	Word2vec Emb. (951x640x400)	5x400, 32	400	0.0005	100	40	69.6	342.75
CNN	GloVe Emb. (951x640x300)	5x300, 64	300	0.00075	25	40	67.6	322.32
CNN	fastText Emb. (951x640x300)	5x300, 32	300	0.00125	75	30	72.2	285.47
CNN	Word2vec Emb. (180x498x250)	3x250, 32	250	0.00125	50	50	68.6	221.56
CNN	GloVe Emb. (180x498x250)	3x250, 32	250	0.005	30	30	61.3	202.49
CNN	fastText Emb. (180x498x200)	3x200, 16	200	0.001	10	40	59.1	218.27
Top ind. features for 180 lyrics DNN	Feature Set 42 (180 x 20)	N/A	N/A	0.01	10	20	77.8	40.30
Top ind. features for 951 lyrics DNN	Feature Set 11 (951 x 200)	N/A	N/A	0.005	25	50	76.1	120.30
Combined 180 lyrics features DNN	Feature Set 51 (180 x 609)	N/A	N/A	0.00075	50	200	78.3	310.65
Combined 951 lyrics features DNN	Feature Set 51 (951 x 498)	N/A	N/A	0.001	150	100	72.1	259.95
Augmented Split CNN	Word2vec Emb. (420 x 501 x 200)	5 x 200, 64	200	0.0015	50	25	49.7	182.13
Augmented Split CNN	GloVe Emb. (420 x 501 x 300)	5 x 300, 64	300	0.00075	40	30	49.3	203.79
Augmented Split CNN	fastText Emb. (420 x 501 x 250)	5 x 250, 64	250	0.001	50	25	52.5	192.48
BERT	180 lyrics	N/A	200	0.0001	20	2	83.4	638.2
BERT	951 lyrics	N/A	400	0.00005	15	3	88.9	801.36
RoBERTa	180 lyrics	N/A	200	0.000125	25	3	85.6	740.36
RoBERTa	951 lyrics	N/A	400	0.00001	20	3	84.2	878.42

Table 3.8: Best results for DL static LMER approaches

Chapter 4

LEVD

The purpose of this section is to showcase the data used for the LEVD problem and detail the experimental process.

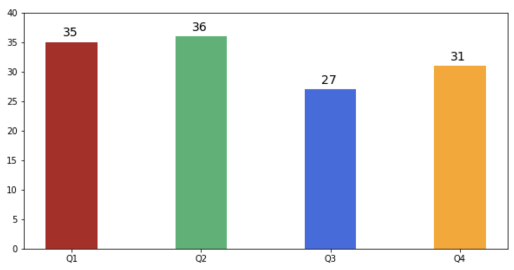
4.1 Data

This section presents the datasets used and the way they were handled in order to be used for the LEVD models.

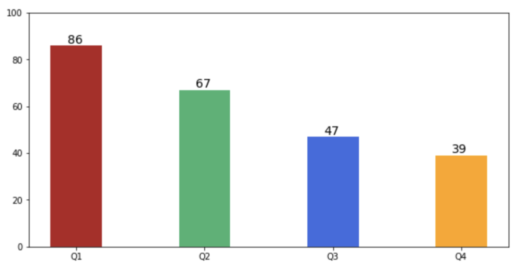
4.1.1 Database - MIR Lyrics Emotion Sentences (2016)

When it comes to the LEVD problem, we also had a previously built dataset (which has also been carefully explored by Malheiro in Malheiro 2017). In his thesis, Malheiro considered a sentence (verse) as the basic unit for the lyric, thus this dataset consisted in a total of 368 sentences manually annotated into the 4 quadrants of Russell's model. The sentences came from 44 lyrics that belonged to several musical genres and were annotated through a web application that allowed users to read the specific verse and chose one of five options: four for each one of the quadrants of Russell's emotion model and an extra one that would be used if the annotator felt like the sentence would not convey any emotion (Neutral). For each verse, the final label was the option with the most votes (when in a draw, the verse was ignored) and the sentences classified as "Neutral" were left behind. After the annotation process we were left with the quadrant distribution shown in Fig. 4.1.

There was a significant difference between the number of samples for each label, specifically when comparing the first with the third and fourth quadrants. This could have been one of the factors that negatively impacted the results of the models tested with the initial data. Originally the dataset was split into 129 and 239 sentences for model training and testing, respectively, but our methods revolve around a different approach to data splitting as detailed further on.



(a) 129 Sentences Dataset



(b) 239 Sentences Dataset

Figure 4.1: MIR Lyrics Emotion Sentences (2016) Quadrant Distribution

4.2 Methods and Results

This section intends to explain the different LEVD approaches and showcase their results.

4.2.1 Classical ML

The goal for this portion of work was to develop DL approaches for LEVD that would reach or surpass previously obtained results as well. Malheiro built a keyword-based approach (KBA) which achieved an F1-Score of 67.4% Malheiro 2017 when applied to the same dataset we have at our disposal.

4.2.2 Deep Learning

At this stage, and considering the data available for this problem, LEVD consists in a diminished version of LMER: instead of a full lyric classification task we now face a sentence/verse labelling problem. Although it looked like we had a simpler scenario at hand, this was not the case. Most of the models adjusted too well to the training data and were unable to generalise when shown new samples. The evaluation of our models follows the same rules as in Chapter 3.

CNN

Initial experiments in LEVD started with the same concept applied in the CNN from Section 3.2.3. Quickly we found out that adjustments to the model itself had to be made as there was a big discrepancy between the high performance on the training sets and the much lower one in the test sets: the model was prone to overfitting. We started by modifying the actual structure of the model and reducing the convolution and pooling layers to only one of each as shown in Fig. 4.2, which by itself produced far better results.

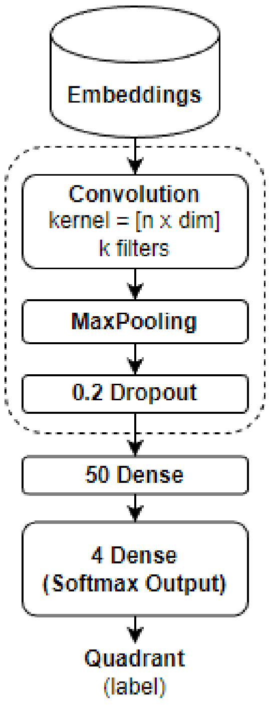


Figure 4.2: CNN structure diagram.

At this stage the complexity of the model was severely reduced but there was still room for improvement as the intricacy of our inputs was significantly reduced. Kernels with smaller dimensions typically obtained the best performances. The number of filters in each convolution was reduced as well and only layers with 16 and 32 filters were tested as higher values would make the model overfitt quickly. After the convolution and pooling processes information was passed through a dropout layer, whose values varied between the interval from 0.2 to 0.4 (achieving the best results with 0.2), and a dense layer with 50 neurons before the softmax activation and final layer. Again, grid search was performed to tune hyperparametes, the learning rates and embedding dimensions tested in Chapter 3 were kept, however the batch sizes went from 5 to 100 in steps of 5 and the epochs ranged from 5 to 75 also in increments of 5.

This structure achieved a top result of **59.8%** F1-Score (see Table 4.1) when fed with the sentences processed with the GloVe model, followed by fastText and Word2vec with **57.1%** and **53.4%**, respectively. The best results for the CNN model are still statistically significant, with a p-value of 3.18×10^{-14}, when compared to the improved keyword-based approach result of 67.4% F1-Score.

	Q1	Q2	Q3	Q4	F1-Score
Q1	22.4	2.9	2.2	2.0	76.0
Q2	3.7	24.4	2.9	1.7	74.6
Q3	5.1	4.9	9.8	1.0	47.1
Q4	7.1	1.7	1.1	7.1	41.4

Table 4.1: Confusion matrix (percentage values) and F1-Score per quadrant for GloVe+CNN and 368 sentences dataset

LSTM

LSTMS are widely used for NLP as these architectures maintain a memory based on history information, which allows the model to predict an output conditioned on distanced features. In this work experiments were made using both unidirectional and bidirectional long short-term memory cells. Once again, the architecture built for the problem at hand could not surpass a certain level of complexity, otherwise the model would overfitt.

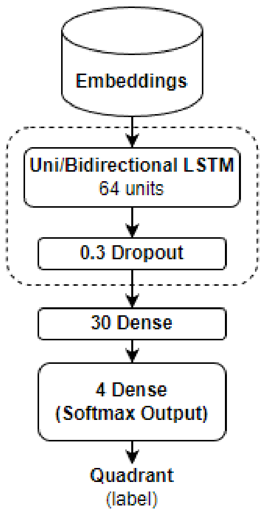


Figure 4.3: Unidirectional/Bidirectional LSTM structure diagram.

The first structure tested is showcased in Fig. 4.3, this architecture was one of the simpler versions of an LSTM-based model, composed of an LSTM layer with 64 units, a 0.3 dropout layer (which was inserted in the model not long after the first experiments), a fully connected layer with 30 neurons and, as usual, a softmax activation layer. Grid search was similar to the one made in the previously discussed CNN model. With an unidirectional version of the model it achieved an F1-score of **60.7%** combined with fastText, **58.9%** with Glove and **57.2%** with Word2vec embeddings. When using a bidirectional layer the F1-Score increased to **64.0%** (see Table 4.2) by using the fastText model for embeddings and **61.3%** for both Glove and Word2vec embeddings. The BiL-

STM still was not an improvement on past results, being statistically significant compared to the KBA with a p-value of 1.17×10^{-5}.

	Q1	**Q2**	**Q3**	**Q4**	**F1-Score**
Q1	22.8	3.8	2.2	4.5	67.2
Q2	3.3	18.8	4.3	1.6	69.5
Q3	4.3	2.4	12.0	2.0	58.2
Q4	4.3	1.1	2.2	10.4	57.1

Table 4.2: Confusion matrix (percentage values) and F1-Score per quadrant for fastText+BiLSTM and 368 sentences dataset

Augmented Data

The information available to train our models was scarce, in hope for an increase in performance, new datasets were built by taking advantage of data augmentation methods designed for NLP related tasks. On previous research Malheiro 2017 the dataset was divided on two splits, one with 239 sentences for training the models and another with 129 sentences in order to test them on unseen data. Again, to perform data augmentation and maintain cross-validation we decided to unite these initial splits and, from the total 368 lyrics, create 6 different folds each made of 268 and 100 lyrics for training and testing, respectively.

Just like in Section 3.2.3, in each split solely the training set was used to perform augmentation and the 6 folds had a different augmentation method for every single one of them, thus we were able to create 6 new different training sets with three times the number of original training verses in which one of them relied on synonym replacement with the WordNet lexical database, three of them were based on non contextual embeddings (GloVe, fastText and Word2vec), one of them coming from contextual embeddings (BERT) and the remaining consisted in a the same sequential algorithm that created new samples with each one of the previous methods. Each one of these new folds would have a total of 904 instances in which 804 would be used to train our models and 100 of them to further on test them.

The process of developing these new datasets had to take into consideration the class imbalance of the original samples as the new information would derive from them. If new sentences were created from each one of the initial training verses then the more the training portion increased, the bigger the imbalance would be. In order to correct this, classes with less samples would have priority in the augmentation process. This method allowed us to build balanced training sets (with 201 instances per quadrant) where the classes that originally had less samples now contain more augmented information than the rest.

	Q1	Q2	Q3	Q4	F1-Score
Q1	33.4	2.2	1.1	1.5	87.5
Q2	0.6	26.7	0.9	0.4	93.4
Q3	2.0	1.2	14.1	0.5	79.3
Q4	1.3	0.6	1.0	12.5	80.8

Table 4.3: Confusion matrix (percentage values) and F1-Score per quadrant for fastText+BiLSTM and augmented sentences dataset

Augmented data was processed by the three embedding models (fastText, Word2vec and Glove) and passed through the CNN, LSTM and BiLSTM architectures described previously. The CNN model achieved **76.5%** of F1-Score with the Word2vec model as its best performance which is statistically significant to when compared with the KBA approach with a p-value of 2.04×10^{-16}. The LSTM increased this score up to **81.3%** with the same embeddings and the BiLSTM had the best performance of all with **85.3%** F1-Score using fastText embeddings (see Table 4.3) which is a statistically significant result compared with both the KBA and the unidirectional version of the model, with the p-values of 1.25×10^{-26} and 1.53×10^{-6}, respectively. Although these were very basic models results were impressively high, proving performance was not always dependant on high levels of complexity and is greatly impacted by the inputs the model receives.

Pre-trained Models

Curiosity about the capability of pre-trained models grew with the impressive results achieved in Section 3.2.3. From the previous experiments on the LMER dataset and based on the constant overfitting problem shown by the developed architectures for LEVD, there was a major concern that these models would adapt well in training and, just like the later, substantially decay in performance when given test set instances.

The process was similar to the one presented in Section 3.2.3. The sentences were formated accordingly to the BERT/RoBERTa input requirements and fed into the models. The major differences were the amount of neurons the fully connected layer was given (reduced to 75 for these experiments) and the dropout value of its preceding layer which was now set to 0.4 instead of the 0.2 used for LMER. BERT reached an F1-Score of **82.9%** while the modified version, RoBERTa, achieved **84.6%** F1-Score (see Table 4.4).

The final result obtained by RoBERTa is statistically significant when compared with the base version, having a p-value of 0.0021, but this is not true when compared with the BiLSTM trained on augmented data since the p-value in this case is 0.5308 > 0.05.

	Q1	Q2	Q3	Q4	F1-Score
Q1	32.0	2.3	1.1	1.6	86.5
Q2	0.6	27.5	0.9	0.3	93.3
Q3	2.1	1.2	12.9	0.6	77.0
Q4	1.3	0.8	1.0	13.8	81.7

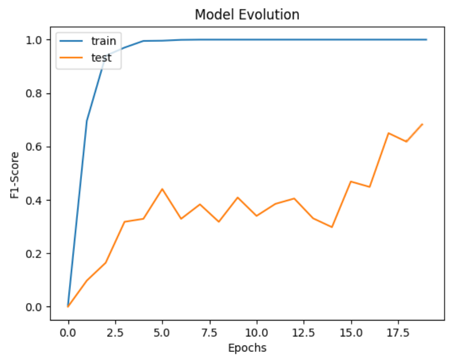
Table 4.4: Confusion matrix (percentage values) and F1-Score per quadrant for RoBERTa and 368 sentences dataset

Another pre-trained model was tested with the 368 sentence dataset, the GPT-3 developed by OpenAI. It is necessary to mention that the experiments conducted with this model were restricted. By the time the tests were made OpenAI only offered a trial version which limited the amount of computation available by giving the user a certain amount of credit and taking advantage of a credit per usage policy. The best result achieved, given the limitations imposed by OpenAI, was an F1-Score of **71.4%** which was not by any means impressive when compared to the remaining pre-trained competitors.

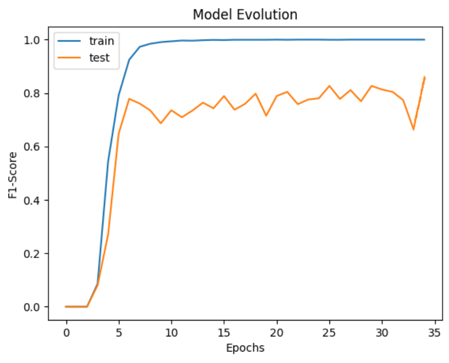
4.3 Analysis of the Results

The hyperparameters, results and computation time for each model are illustrated in Table 4.5.

Many times when investigating a problem we tend to overcomplicate the approaches developed in order to solve it. Our best results came from one of the simplest BiLSTM architectures designed for NLP tasks. Soon enough we have realised that, in this particular case, high performances would require some work on the model inputs as the issues analogous to the task at hand were highly related with the lack of information available in our dataset. NLP data augmentation techniques were key to achieving a top result of **85.3%** F1-Score. It is well worth to note that when fed with these smaller inputs (verses instead of full lyrics) the augmentation algorithms often kept the core emotion of the lyric, making much less mistakes when replacing parts of the input and producing more meaning full information than when applied to the static dataset.



(a) BiLSTM (trained on the 368 sentences dataset) F1-Score evolution per epoch



(b) BiLSTM (trained on the augmented split dataset) F1-Score evolution per epoch

Figure 4.4: BiLSTM Model Evolution

Fig. 4.4 demonstrates the impact these methods had in our models performance. The BiLSTM trained on the original dataset would perform well when training but lacked generalisation capabilities as it struggled to classify new samples. Although it took more training instances, with more patterns to learn from, the model fed with augmented data quickly adapted to the test set, reaching an initial peak around the first 5 epochs and stabilising more after this large momentum.

Similar results were obtained when using the transformer pre-trained model RoBERTa. This shows how larger models do not necessarily mean better outcomes. By being pre-trained on a large corpus RoBERTa is also exposed to information that is irrelevant for the problem at hand which can be a cause of a diminished performance. This model was not only akin to the BiLSTM on end results, but the intrinsic quadrant classification mistakes were alike, having similar confusion matrices.

CNN architectures were also tested as a possible solution to this problem but the LSTM models outperformed them in most cases. This can be related to the way that the later are capable of capturing dependencies in the sentences, specially when a bidirectional version is applied. Assuming that words are not individually independent from each other and that emotion is more often related to an aggregate of expressions, is an important step to understand the capabilities of these methods.

Model	Input	Kernel Size, #Filters	Emb. Dimension	Learning Rate	Batch Size	#Epochs	F1-Score (%)	Time (min)
CNN	Word2vec Emb. (368x156x200)	3x200, 16	200	0.00125	5	20	53.4	83.85
CNN	GloVe Emb. (368x156x250)	3x250, 16	250	0.001	5	20	59.8	87.96
CNN	fastText Emb. (368x156x200)	3x200, 16	200	0.0005	10	25	57.1	96.37
LSTM	Word2vec Emb. (368x156x200)	N/A	200	0.001	50	20	57.2	130.85
LSTM	GloVe Emb. (368x156x250)	N/A	200	0.001	45	15	58.9	131.96
LSTM	fastText Emb. (368x156x200)	N/A	200	0.00125	45	20	60.7	128.37
BiLSTM	Word2vec Emb. (368x156x200)	N/A	350	0.00175	40	20	61.3	249.85
BiLSTM	GloVe Emb. (368x156x250)	N/A	300	0.001	30	15	61.3	265.96
BiLSTM	fastText Emb. (368x156x200)	N/A	300	0.0015	30	20	64.0	253.37
Augmented Split CNN	Word2vec Emb. (904x154x200)	3x200	200	0.00075	20	25	76.5	301.77
Augmented Split LSTM	Word2vec Emb. (904x154x200)	N/A	300	0.00125	50	30	81.3	410.26
Augmented Split BiLSTM	fastText Emb. (904x154x200)	N/A	400	0.0015	100	35	85.3	675.19
BERT	368 Sentences	N/A	300	0.00015	60	3	82.9	244.87
RoBERTa	368 Sentences	N/A	250	0.000125	40	2	84.6	272.95

Table 4.5: Best results for DL LEVD approaches

This page is intentionally left blank.

Chapter 5

Conclusion and Future Work

This chapter addresses, in a summarised way, the main contributions and discoveries from this work. A rundown of the best results and approaches is done in Section 5.1 and the elements that deserve further exploration are detailed in Section 5.2.

5.1 Conclusion

Overall this work came as a beneficial experience for personal growth and development. There were portions of the project where the end result did not come out as initially expected, specifically when talking about the expected time frames produced in the most early stages. This can be justified on our own inexperience, being the first time we confront DL tasks related to a real-world scenario. Nonetheless, efforts were made to diversify the approaches taken and not just for the classification models, but for their inputs as well (i.e. Word2vec, GloVe, fastText embeddings). This work covers the simplest architectures available in DL (i.e. the DNN and the CNN), and follows on to more complex ones later on (i.e. pre-trained models).

The experimental process revealed itself as a highly time consuming task, testing a multitude of hyperparameters and structure designs for each model most often took longer than initially expected. As we became more familiar with the concepts, we started to get better results and our top performer (BERT) achieved **88.9%** F1-Score for static LMER.

Regarding LEVD, data augmentation posed as a beneficial process for the performance of our models. By combining this process and the use of LSTM layers, we have achieved the best result, to our knowledge, in the literature for the dataset in question - **85.3%** F1-Score.

5.2 Future Work

We propose that the following guidelines should be considered for future work:

- expanding the currently available datasets for both LMER and LEVD, with special regard to the class imbalance present in the LEVD dataset.

- considering implementing the LEVD sentence classification approach for static LMER (i.e. splitting the full lyrics in sentences, classifying these verses one by one and combining the results in order to obtain a single label for the whole lyric).
- increasing the efforts made to explore NLP augmentation techniques for static LMER.
- tuning and experimenting with different architectures and hyperparameters for the DL approaches.
- exploring more versions of pre-trained models for both problems, as the results obtained with these approaches were impressive.
- investing in understanding why misclassification happens in specific lyrics/sentences.

We optimistically believe that if tackled with the right amount of effort, the proposed future research lines can result in better outcomes for these problems.

References

Hevner, K. (1936). “Experimental studies of the elements of expression in music”. In: *American Journal of Psychology*.

Stone, P.J. et al. (1966). “The General Inquirer: A computer approach to content analysis.” In: *MIT Press, Cambridge, MA*.

Ekman, P. (1982). “Emotion in the Human Face”. In: *Cambridge University Press*.

Fehr, B. and J. Russel (1984). “Concept of Emotion viewed from a prototype perspective”. In: *Journal of Experimental Psychology*.

Whissell, C. (1989). “Dictionary of Affect in Language”. In: *Plutchik and Kellerman (Eds.) Emotion: Theory, Research and Experience, Vol. 4, pp. 113–131, Academic Press, NY*.

Bradley, M. and P. Lang (1999). “Affective Norms for English Words (ANEW): Stimuli, Instruction Manual and Affective Ratings”. In: *Technical report C-1, The Center for Research in Psychophysiology, University of Florida*.

Tellegen, A., D. Watson, and L. Clark (1999). “On the Dimensional and Hierarchical Structure of Affect”. In: *Psychological Science, Vol.10, SAGE Publications*.

Juslin, P. and J. Sloboda (2001). “Music and Emotion: theory and research”. In: *Oxford University Press*.

Gabrielsson, A. (2002). “Emotion Perceived and Emotion Felt: Same or Different?” In: *Musicae Scientiae (special issue)*.

Schimmack, U. and R. Reisenzein (2002). “Experiencing activation: energetic arousal and tense arousal are not mixtures of valence and activation”. In: *Emotion (Washington, D.C.)*

Robnik-Šikonja, M. and I. Kononenko (2003). “Theoretical and Empirical Analysis of ReliefF and RreliefF”. In: *Machine Learning, Vol. 53*.

Lu, L., D. Liu, and H. Zhang (2006). “Automatic mood detection and tracking of music audio signals”. In: *IEEE Transactions on Audio, Speech, and Language Processing*.

Meyers, O. (2007). “A mood-based music classification and exploration system”. In:

Busso, C. et al. (2008). “Iemocap: Interactive emotional dyadic motion capture database”. In: *Language resources and evaluation, vol. 42,*

Laurier, C., J. Grivolla, and P. Herrera (2008). “Multimodal music mood classification using audio and lyrics”. In: *International Conference on Machine Learning and Applications*.

Laurier, C., M. Sordo, et al. (2009). “Music mood representation from social tags”. In: *International Society for Music Information Conference*.

Hu, X. and J. Downie (2010). “Improving mood classification in music digital libraries by combining lyrics and audio”. In: *10th annual joint conference on Digital libraries*.

Yang, D. and W. Lee (2010). “Music Emotion Identification from Lyrics”. In:

Gao, R. et al. (2013). “Developing simplified chinese psychological linguistic analysis dictionary for microblog”. In: *Imamura K, Usui S, Shirao T, Kasamatsu T, Schwabe L, Zhong N, eds. Brain and health informatics. Lecture notes in computer science.*

Mikolov, T., K. Chen, et al. (2013). “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv preprint arXiv:1301.3781*.

Mikolov, T., W. Yih, and G. Zweig (2013). “Linguistic Regularities in Continuous Space Word Representations”. In: *Microsoft Research, Redmond, WA 98052*.

Coutinho, E., J. Deng, and B. Schuller (2014). “Transfer Learning Emotion Manifestation Across Music and Speech”. In: *International Joint Conference on Neural Networks (IJCNN)*.

Pennington, J., R. Socher, and C. D. Manning (2014). “GloVe: Global Vectors for Word Representation”. In: *Computer Science Department, Stanford University, Stanford, CA 94305*.

Godin, F. et al. (2015). “Named entity recognition for twitter microposts using distributed word representations”. In: *Proceedings of the workshop on noisy user-generated text. Association for Computational Linguistics,Beijing, pp 146–153*.

Joulin, A. et al. (2016). “Bag of Tricks for Efficient Text Classification”. In: *arXiv preprint arXiv:1607.01759*.

An, Y., S. Sun, and S. Wang (2017). “Naive Bayes Emotion Classifiers for Music Emotion Classification Based on Lyrics”. In: *Proceedings - 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS*.

Baziotis, C., N. Pelekis, and C. Doulkeridis (2017). “Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis”. In: *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*.

Kingma, D. and J. Lei Ba (2017). “Adam: a method for stochastic optimisation”. In: *ICLR 2015*.

Malheiro, R. (2017). “Emotion-based Analysis and Classification of Music Lyrics”. In: *Doctoral Program in Information Science and Technology. University of Coimbra*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, et al. (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, et al. (2017). "Attention is all you need". In: *NIPS*.

Devlin, J. et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805*.

Howard, J. and S. Ruder (2018). "Universal Language Model Fine-Tuning for Text Classification". In: *Proceedings of the 56th annual meeting of the association for computational linguistics, vol 1. Association for Computational Linguistics, Melbourne, pp 328–339*.

Malheiro, R. et al. (2018). "Emotionally-Relevant Features for Classification and Regression of Music Lyrics". In: *IEEE Transactions on Affective Computing*.

Peters, M.E. et al. (2018). "Deep con- textualized word representations". In: *NAACL, pages 2227– 2237*.

Radford, A. et al. (2018). "Improving Language Understanding by Generative Pre-Training". In:

Zadeh, A.B. et al. (2018). "Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Lin- guistics (Volume 1: Long Papers)*.

Chatterjee, A. et al. (2019). "SemEval-2019 Task 3: EmoContext, Contextual Emotion Detection in Text". In: *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Ge, S. et al. (2019). "Dialog Emotion Classification using Attentional LSTM-CNN". In: *Proceedings of the 13th international workshop on semantic evaluation. Association for Computational Linguistics, Minneapolis, pp 340–344*.

Ma, L. et al. (2019). "Emotion Detection with Emotion Oriented Neural Attention Network". In: *May J, Shutova E, Herbelot A, Zhu XZ, Apidianaki M, MohammadSM (eds) Proceedings of the 13th international workshop on semantic evaluation*.

Parisi, L. et al. (2019). "Exploiting Synchronized Lyrics and Vocal Features for Music Emotion Detection". In:

Ragheb, W. et al. (2019). "Attentive Conversation Modeling for Emotion Detection and Classification". In: *May J, Shutova E, Herbelot A, Zhu XZ, Apidianaki M, MohammadSM (eds) Proceedings of the 13th international workshop on semantic evaluation*.

Xiao, J. (2019). “Ensemble of Transfer Learning Methods for Contextual Emotion Detection”. In: *May J, Shutova E, Herbelot A, Zhu XZ, Apidianaki M, MohammadSM (eds) Proceedings of the 13th international workshop on semantic evaluation*.

Yang, Z. et al. (2019). “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems. pp. 5753–5763*.

Abdillah, J., I. Asror, and Y. Wibowo (2020). “Emotion Classification of Song Lyrics using Bidirectional LSTM Method with GloVe Word Representation Weighting”. In: *RESTI journal, System Engineering and Information Technology, Vol. 4 No. 4 (2020) 723 - 729*.

Pandeya, Y.R. and J. Lee (2020). “Deep Learning-based Late Fusion of Multimodal Information for Emotion Classification of Music Video”. In:

Siriwardhana, S. et al. (2020). “Jointly Fine-Tuning “BERT-like” Self Supervised Models to Improve Multimodal Speech Emotion Recognition”. In: *Augmented Human Lab, Auckland Bioengineering Institute, The University of Auckland*.

Agrawal, Y., R. Agrawal, and V. Alluri (2021). “Transformer-based approach towards music emotion recognition from lyrics”. In:

Alex, N. et al. (2021). “RAFT: A Real-World Few-Shot Text Classification Benchmark”. In:

Borah, A. (2021). “Word Embeddings: How do organizations use them for building recommendation systems?” In:

Xu, L. et al. (2021). “Using machine learning analysis to interpret the relationship between music emotion and lyric features”. In: