# Moodify: Analyzing Song Emotions with NLP

**Mattia Franzin (239930)**
mattia.franzin@studenti.unitn.it

## Abstract

This project describes an music recommendation system project designed to operate in song lyrics for mood prediction of the user, based on Natural Language Processing (NLP) techniques. The system seamlessly integrate Spotify and Genius APIs, which help give personalized recommendations in a user-friendly interface. Along with PyTorch Lightning for model training efficiency, HuggingFace Transformers library was used for obtaining pre-trained models, and Weights and Biases for real-time performance tracking. This system incorporates a (multilingual) transformer architecture for analyzing lyrics and correctly classifying emotion. It enhances music streaming experience by making song recommendations to listeners based on their current state of emotions and is an example of the practical applications of NLP. The results demonstrate promising performances in emotion detection despite the limitations in training only on English datasets.

## 1 Introduction

The approach to listening to music has been revolutionized in this digital age, thanks to music streaming services. Spotify, being one of the most-known platforms, offers its users very broad access to a library of songs from various genres and artists around the world.

Not only Spotify revolutionized our life, but the same is going on with Natural Language Processing (NLP) techniques, that raised new possibilities for an array of applications. They help machines understand and manipulate human language, and applied to songs and lyrics, it can provide much help. Integrating NLP in music listening experience would be interesting, because people use these services on an almost daily basis, and such models could benefit it, deepening into themes, emotions, and stories written through lyrics.

A multilingual transformer architecture-based network can provide text understanding and interpretation in various languages. A system which suggests songs to users based on the their mood has been created. This system collaborates with Spotify and Genius APIs to retrieve recent songs played by the user, extract the lyrics, and with the dominant emotion, obtain personalized recommendations for a specific user. A simple UI has also been developed, which allows users to interact with the system .

## 2 Tech Stack

The whole project is structured and completed with PyTorch Lightning, giving a more structured way than raw PyTorch by having an integrated entire backbone to the training, validation, and testing phases. This framework simplifies the workflow of developing deep learning models; it has built-in features to save/load checkpoints, search learning rates, determine batches size, establish actions at the start or end of epochs, and add loggers quickly.

The training of the model began with Google Colab. Nevertheless, many limitations appeared regarding the availability of time and resources, including those used during the debugging sessions.

To address these challenges, the training part has been transferred to a remote machine from **LightningAI** (Falcon and The PyTorch Lightning team, 2019). This platform provides is based on tokens, and 15 are gained each month ( 22 training hours). You can earn more tokens through their educational tutorials. The token-based system allows flexible access to multiple GPUs that can be scaled according to computational needs. In this particular case, a **Single L4 GPU** model was used, which has 50% more dedicated RAM than the **T4** model — with almost double the **TFLOPs** — 121 vs. 65.

*Weights and Biases* (Biewald, 2020) has been integrated into the setup for continuous monitoring and analysis of the progress made in training, validation and test. By tracking the model param-

```
class LyricsClassifier(LightningModule):
    def __init__(self, model, lr, labels, batch):
        ...
    def forward(self, x, labels=None):
        ...
    def on_train_epoch_start(self):
        ...
    def configure_optimizers(self):
        ...
    def training_step(self, batch, batch_idx):
        ...
    def validation_step(self, batch, batch_idx):
        ...
    def test_step(self, batch, batch_idx):
        ...
    def train_dataloader(self):
        ...
    def val_dataloader(self):
        ...
    def test_dataloader(self):
        ...
    @staticmethod
    def collate_fn(batch):
        ...
```

Figure 1: Example of a PyTorch Lightning Module

eters in real-time, this tool produces helpful and user-understandable visualizations. These directly facilitate further model adaptations.

Regarding the models themselves, they all come from the **HuggingFace** Transformers library. The library is well-known for providing an enormous collection of pre-trained models that can easily be plugged into the projects, and potentially published. Along with the models, the library also provides **tokenizers**, which are essential for the models to understand the text. Datasets also come from another **HuggingFace** library, *Datasets*, which provides fast and efficient access to a wide range of datasets, with easy manipulation and pre-processing.

## 3 Datasets

Unfortunately, no Italian text dataset is available with such emotion annotations. Therefore, the choice was to use English datasets along with multilingual models, hoping for generalization. The training was attempted on two different datasets; the second one gave the best result.

- Multimodal EmotionLines Dataset (MELD)
- GoEmotions

### 3.1 Multimodal EmotionLines Dataset (MELD)

(source)

MELD (Poria et al., 2019) is a multimodal dataset of conversations from the TV series Friends. It is transcribed and annotated with emotions and polarity. A total of 13,000 dialogues constitute it, divided in a ratio of 72/8/20 into training, validation, and test sets. Only texts and emotions (Anger, Disgust, Sadness, Joy, Neutral, Surprise, Fear) were used.

### 3.2 GoEmotions

GoEmotions (Demszky et al., 2020) is a text dataset with Reddit comments annotated with emotions. The simplified version of the annotations was taken, which included 54.300 comments, cleaned from unclear examples, in an 80/10/10 ratio for the train/valid/test sets out of 211,000 overall comments. As well, the annotated emotions are 27 plus neutral, which were reduced to 6 plus neutral, as the focus is general classification rather than specificity of emotions, especially since many are very similar to each other.

The emotions were grouped as follows, using Paul Ekman's basic emotions (Ekman, 1992):

- **anger**: anger, annoyance, disapproval
- **disgust**: disgust
- **fear**: fear, nervousness
- **joy**: joy, amusement, approval, excitement, gratitude, love, optimism, relief, pride, admiration, desire, caring
- **sadness**: sadness, disappointment, embarrassment, grief, remorse
- **surprise**: surprise, realization, confusion, curiosity
- **neutral**: neutral

## 4 Models

All the models used are based on the transformer architecture, relying on the transformer's self-attention mechanism, handling long-range dependencies more efficiently than RNNs. Using this architecture, Google introduced BERT (Bidirectional Encoder Representations from Transformers), followed by many others.

As opposed to traditional models, instead of reading text from left to right or right to left, BERT reads the entire text at once, allowing it to understand the context of a word based on all of its

surroundings (https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270). To learn better representations, BERT was trained on two tasks:

- **Masked Language Model (MLM)**: randomly masking 15% of the words in each sentence and predicting them.
- **Next Sentence Prediction (NSP)**: predicting whether two sentences are consecutive or not

All tested models makes use of the *base* version, instead of the *large* one.

All models has been instantiated *ForSequenceClassification*, a class that wraps a transformer model for sequence classification, with a classifier on top of the transformer output, to be able, in this case, to classify the emotions of the lyrics.

### 4.1 RoBERTa

One year after BERT, RoBERTa (Liu et al., 2019), *A Robustly Optimized BERT Approach*, was introduced by Facebook AI, aiming to improve BERT's training process and performance. To do that, the NSP task was removed, the embedding changed from *WordPiece* to *Byte-Pair Encoding* (BPE), and network training was optimized, with a larger batch size, more data, and more training steps. Moreover, dynamic masking was introduced.

The variant used in this project is the base one, with 125M parameters, trained on 160GB of data.

### 4.2 BigBird

BigBird (Zaheer et al., 2021), on top of the RoBERTa model, introduces a sparse attention mechanism, which allows the model to attend to only a few tokens, rather than all of them, reducing the computational cost. This is particularly useful when dealing with long sequences, as the model can focus on the most important tokens, ignoring the others.

It can be useful when dealing with lyrics, since it supports up to 4096 tokens, which means an entire song can be processed at once, capturing the context of the entire song.

### 4.3 XLM-RoBERTa

XLM-RoBERTa (Conneau et al., 2020) is a multilingual model based on RoBERTa, trained on 2.5TB of data in 100 languages. To accommodate multiple languages, the network size was increased to 270M parameters. It is expected to generalize better on different languages, including Italian.

This is the model used in the project, as the lyrics can be in different languages, not just English.

## 5 Results and Benchmarks

Since a model is being fine-tuned, the learning rate is a crucial hyper-parameter. The learning rate is set very low, around *5e-5* or *5e-6*. This was done to avoid **catastrophic forgetting**, a phenomenon where the model forgets what it has learned during pre-training, and overfits the new data, as shown on Figure 2. But even with this setup, this phenomenon was observed, so a scheduler was added to decrease the learning rate every two epochs with a 0.1 factor. Everything is then trained for at least 12 epochs with **AdamW** optimizer with weight decay of 0.01.

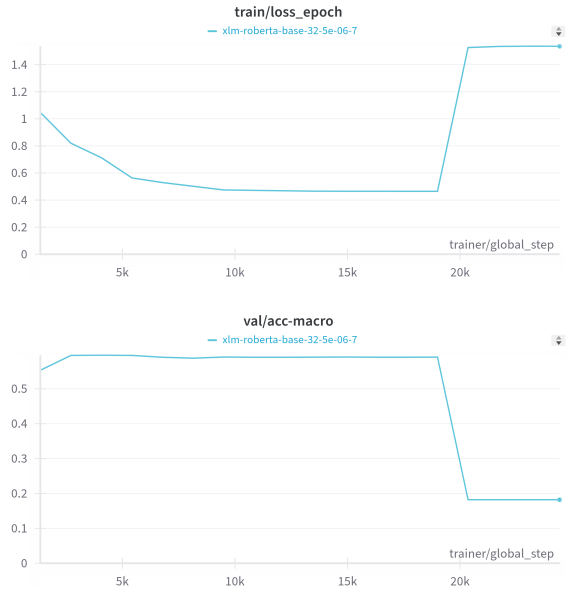


Figure 2: Catastrophic forgetting effect on train loss and validation accuracy

Both the multiclass and multilabel approaches were tested, but the best result was obtained by using the latter. The metrics used were accuracy and f1, both macro-average, to avoid that the largest classes influence the result too much.

Moreover, multilabel approach has been chosen because sentences may carry more than one emotion, and this allows the model to predict multiple emotions for a single sentence.

On the other hand, when counting class predictions with the multiclass approach, the model tends to predict *neutral* or *surprise* more often than the

other classes, even when the text is clearly *anger* or *sad*. So, to reduce this bias, when training with the multiclass approach, if the sample has more than one emotion and *neutral* is one of them, it is ignored; if there are more than one emotion and *surprise* is one of them, this is ignored too. If more than one emotion is still present, a random one is chosen. Not the best approach, but it improves results.

As already mentioned, the performance on Italian lyrics has been tested only in part, during the inference phase, with a quite satisfying result, considering that the model has never seen Italian text during the fine-tuning phase.

The test plots of **GoEmotions** dataset can be found in Figure 3 and Figure 4. **Meld** dataset results are slightly worse than to **GoEmotions** multiclass plots, and thus are omitted.

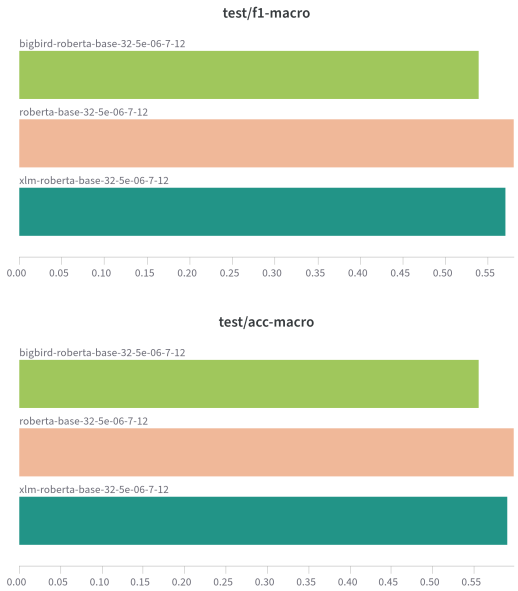


Figure 3: Test metrics on multiclass models

As shown in Figure 3 and in Figure 4, the multiclass approach present a slightly better **f1-score**, whereas the multilabel has an enormous advantage in terms of **accuracy**. Since we are more interested in predicting *true positives* and *true negatives*, a slight **decrease** in *f1-score* is still acceptable.


Figure 4: Test metrics on multilabel models

## 6 Conclusion

In conclusion, the model created, once integrated with Spotify and Genius, is able to detect the emotions present in the lyrics of the songs. To avoid that too short stanzas are mistakenly classified as neutral, it was decided to group four consecutive lines into a single line, scrolling one line at a time. This allows for a more precise classification, and limits the effect of incomplete stanzas being misclassified.

Clearly, the model can only rely on the text, and not on the figurative meaning that some words or songs may be carrying, but it is still a good starting point. To avoid the model predicting emotions at random in the absence of other information, the 'neutral' emotion was used as a bucket for emotions not present, ignoring it if other emotions are present.

When computing the most dominant emotions, the logits predicted by the model are sum up for each group of lines and songs, considering only the ones with a score higher than 0.50. The top-2 moods are then extracted and if the *neutral* emotions is the dominant, the second one is returned. If *neutral* is the only one, it is obviously returned.

Overall, the entire pipeline works as follow:

- The user logs in with Spotify credentials
- The last week's listening history is retrieved from Spotify

- The lyrics of the songs are extracted from Genius
- The lyrics are processed by the model group of four lines at a time
- The emotions are predicted and the most common one is chosen
- The songs are recommended based on the emotion computed
- The playlist is optionally created and user is redirected to Spotify

All the models are published on HuggingFace, and the code is available on GitHub, upon invitation, since it contains API keys and tokens. The same applies to Spotify APIs.

## References


Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *Preprint*, arXiv:1911.02116.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions. *Preprint*, arXiv:2005.00547.

Paul Ekman. 1992. Are there basic emotions? *Psychological Review*, 99(3):550–553.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *Preprint*, arXiv:1810.02508.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences. *Preprint*, arXiv:2007.14062.


## A Mood mapping to Spotify features

*Valence*, *Energy*, *Danceability* values ranges from 0.0 to 1.0. Values have been discovered manually through Spotify APIs

| Emotion | Valence | Energy | Danceability |
|---|---|---|---|
| Anger | 0.0 - 0.3 | 0.7 - 1.0 | 0.3 - 0.6 |
| Fear | 0.0 - 0.3 | 0.6 - 1.0 | 0.0 - 0.4 |
| Sadness | 0.0 - 0.3 | 0.0 - 0.4 | 0.0 - 0.4 |
| Happiness | 0.7 - 1.0 | 0.7 - 1.0 | 0.6 - 1.0 |
| Surprise | 0.5 - 1.0 | 0.7 - 1.0 | 0.4 - 0.7 |
| Disgust | 0.0 - 0.3 | 0.3 - 0.6 | 0.0 - 0.3 |
| Neutral | 0.4 - 0.6 | 0.4 - 0.6 | 0.4 - 0.6 |

Table 1: Mapping Ekman's Emotions to Spotify's Audio Features

## B Plots

### B.1 Multiclass

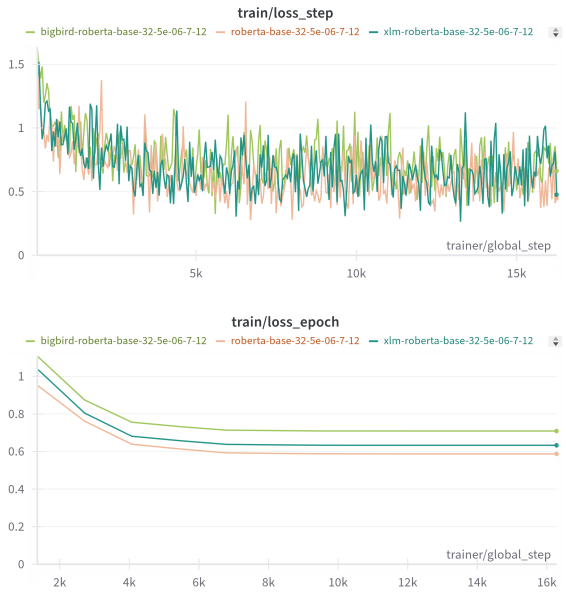


Figure 5: Train metrics on multiclass models

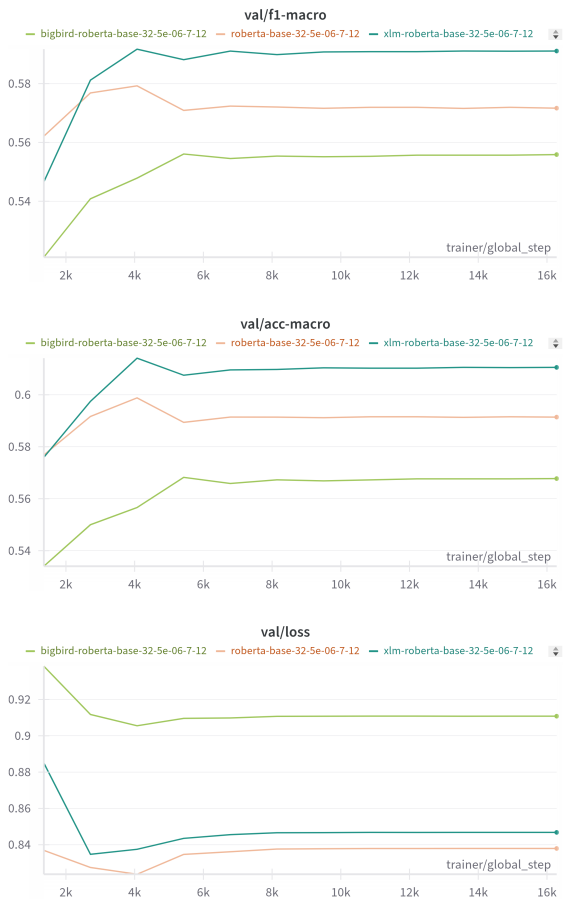


Figure 6: Validation metrics on multiclass models

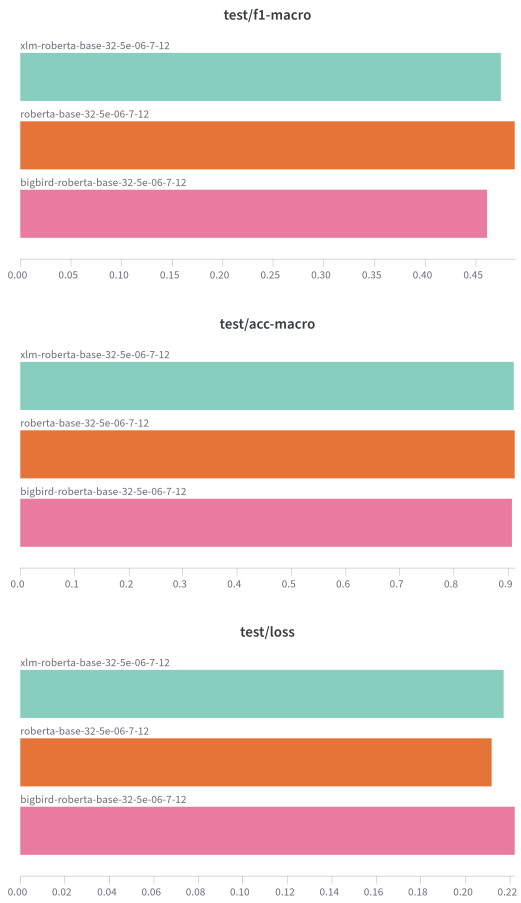


Figure 7: Test metrics on multiclass models

## B.2 Multilabel

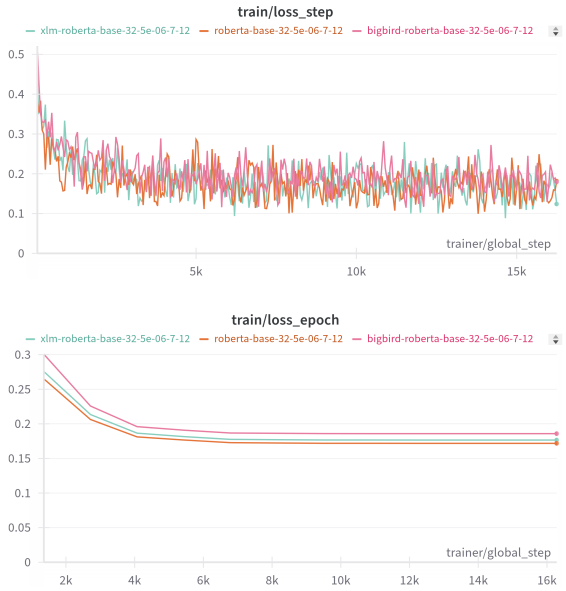


Figure 8: Train metrics on multilabel models


Figure 9: Validation metrics on multilabel models


Figure 10: Test metrics on multilabel models