EMOTION DETECTION IN SONG LYRICS STANZAS

TEXT ANALYTICS - A.Y. 2024/2025

Group 2 - Barbieri, Bosco, Ferrara, Rotellini, Zizza

Abstract

Lyrics serve as one of the main foundations of songs, playing a crucial role in expressing feelings in many different ways. The emotional tone of songs can also serve various purposes, such as automatized playlist creation or songs' organization, offering an alternative to the more traditional genre-based classification. This study focuses on developing four Machine Learning models to classify emotions conveyed in English song lyrics, at the stanza level. The models were chosen for their proven effectiveness across various domains and their diverse approaches, providing a thorough investigation of different techniques and depths for emotion classification in text. The study begins with the preprocessing of the *Genius Song Lyrics* dataset, which our analysis is based on. This was a deep and time-consuming process, which ultimately resulted in a complete reshape of the originale dataset. Then, the models were trained through transfer learning. The ground truth was created with an already existing model: Albert Base v2, which was chosen to avoid the computational costs of larger models like BERT. The resulting labelled dataframe was then splitted following the train/test split protocol. The study proceeded with the training and the performance analysis of the following models: Random Forest and Support Vector Machine, followed by a One-Dimensional Convolutional Neural Network and a Recurrent Neural Network. Lastly, the results are discussed to highlight the specific challenges encountered.

Introduction

This study focuses on capturing emotional dynamics in song lyrics by assigning emotion labels to individual stanzas rather than entire songs. This approach allows for a more granular analysis of emotional shifts within the text. The emotion labels correspond to Robert Plutchik's eight primary emotions [1] (shown in figure 1), providing a comprehensive range for representing various emotional states.

Figure 1: Plutchik's eight primary emotions

The goal of this report is to provide a comprehensive overview of the project, detailing its methodology, results, and key insights. The *Methods* chapter contains a detailed explanation of the data and procedures used in the project, providing descriptions of each part implemented in the project. The *Results* chapter presents an overview of the obtained outcomes.
These results are further explored in the final sections, *Discussion* and *Conclusions*, which interpret the general findings, present possible future directions and recap the primary objectives of the work.

Methods

The dataset used in this project is a sampled subset of English-language songs derived from the *Genius Song Lyrics Dataset*[2]. The original dataset contained numerous attributes; the ones considered relevant for model training are:

- **title:** the song's title;
- **lemmatized_stanzas:** lyrics of the single stanza;
- **stanza_number:** identifies the position of the stanza in the song;
- **is_chorus:** boolean variable that attests whether the stanza is a chorus or not;
- **is_country, is_pop, is_rap, is_rb, is_rock:** boolean variables, result of a one-hot encoding process, that represent songs genres;

- **label:** represents the emotional classification of the stanza, assigned by Albert Base v2[3].

All of these attributes, except for `title`, were the result of the preprocessing phase, as described in the preprocessing section. Due to limited computational power, the labeling process was time-intensive, ultimately resulting in a limited dataset, with a few more than 100.000 entries.

Preprocessing

The preprocessing phase began by sampling the dataset while maintaining genre distribution. Text cleaning focused on removing irrelevant noise, such as square-bracketed items, and splitting lyrics into individual stanzas using stanza-related keywords (e.g. ”chorus”, ”verse”, ”bridge” etc.). Resulting uninformative stanzas, such as empty or very short ones, were discarded. The output of this preliminary preprocessing phase was a dataset where the records were no longer whole songs but individual stanzas, each numbered according to its position within the song. A boolean feature, `is_chorus`, was then added to mark repeated or chorus-related stanzas, and duplicate stanzas were removed to eliminate redundancy. Further cleaning involved removing stanza headers and newline characters, producing cleaner stanzas for labeling.

Lemmatization was performed using the `spaCy` library, generating tokenized stanzas by filtering out punctuation. Lemmatization was chosen over stemming because it produces more accurate and meaningful results, particularly for tasks requiring semantic understanding, such as the one at hand.

Since the dataset was not pre-labeled at the stanza level, ALBERT Base v2 was then fine-tuned to label approximately 100,000 stanzas with emotional categories. A preliminary class distribution analysis showed a slight imbalance, with *joy* being the most frequent (18%) and *disgust* the least (10%). Figure 2 illustrates the distribution across all classes.

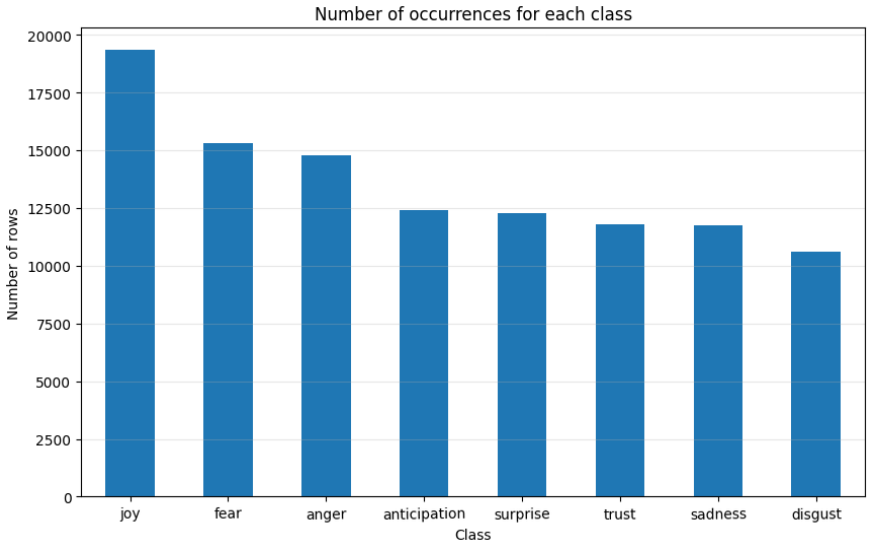


Figure 2: Number of stanzas for each label

Models

The selected model architectures are:

- **Random Forest**: A robust ensemble learning method known for its ability to handle complex, high-dimensional datasets effectively.
- **Support Vector Machine (SVM)**: A powerful classifier that excels in separating classes by finding the optimal hyperplane, particularly effective in text classification tasks.
- **One-Dimensional Convolutional Neural Network (1D-CNN)**: Designed to capture local patterns in sequential data, leveraging convolutional layers to learn hierarchical features.
- **Recurrent Neural Network (RNN)**: Utilized for its strength in processing sequential data, with the ability to capture contextual relationships between words across different stanzas.

Their different approaches and depths are an important point of the study, as they offer interesting insights into the possible different techniques and levels of complexity required for detecting emotional tones in complex pieces of text.

Static Models

The development of static models aimed at providing a benchmark for more complex neural networks. Both architectures, Random Forest and Support Vector Machine, consisted of a preprocessing layer, followed by a classifier. The preprocessing handled title and `lemmatized_stanzas` using TF-IDF for feature extraction.

To improve classification, feature extraction was conducted on the labeled dataset. Firstly, we deepened the preprocessing by creating a **custom stopword list** of frequent and generic words, punctuation, and common typographical errors. To this list were then added the NLTK stopword one and the NLTK numbers' one. Then, TF-IDF scores were computed for each emotion label, using the parameters `min_df` and `max_df` to minimize the influence of overly common or rare words. Despite these efforts, the analysis did not yield the expected results, with most labels sharing common features and low TF-IDF scores. This was likely due to the repetitive and generic nature of song lyrics.

With regard to hyperparameters tuning, Random Search and cross-validation were used to estimate model performance more accurately.

Neural Networks

The neural network architectures were developed and tuned through empirical testing. Both the Recurrent Neural Network and One-Dimensional Convolutional Neural Network share the same preprocessing steps: Non-Negative Matrix Factorization is applied to the title for extracting latent topics, following TF-IDF for richer text representation. The `lemmatized_stanzas` are processed through convolutional and recurrent pipelines, where elements are tokenized and padded to maintain consistent input shapes.

One-Dimensional Convolutional Neural Network

The Convolutional part of the architecture is specifically designed to extract and learn local patterns in `embedding_lyrics`. Its structure consists of three convolutional layers, each applying filters of

varying sizes. This allows to detect patterns at different granularities. These layers are followed by Global Max Pooling, to reduce the previous output's dimension to a fixed-length vector, as well as retaining focus on the most informative patterns. A dropout layer is then applied, to introduce regularization and prevent overfitting.

Recurrent Neural Network

The Recurrent part of the architecture is specifically designed to extract and learn local patterns in `embedding_lyrics`. Its structure consists of three Gated Recurrent Units (`GRU` layers) to model temporal relationships. These are characterized by progressively smaller numbers of units; this allows pattern capture at different abstraction levels. All three layers in the architecture use the `tanh` activation function to compute the hidden state and the `sigmoid` activation function for the recurrent gate. The first and second layers return the full sequence of hidden states for each time step in the input sequence, enabling richer learning of patterns over time. Dropout is applied on every layer, to prevent overfitting and add regularization.

Shared Components

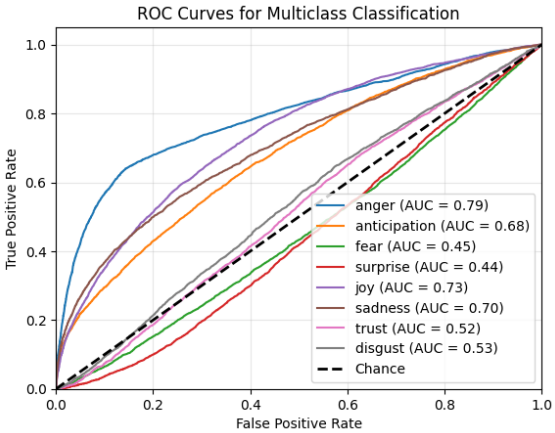
Other features are processed through a simple pipeline, which concatenates inputs, passes them through a dense layer, and combines them with the lyrics-processing output. The final output layer uses 8 units with softmax activation for emotion classification.
The models are trained using categorical cross-entropy as the loss function, and categorical accuracy as the evaluation metric. Other metrics, such as top-k categorical accuracy with $k = 2$, were tested but discarded due to limited performance improvements and lower precision.

Results

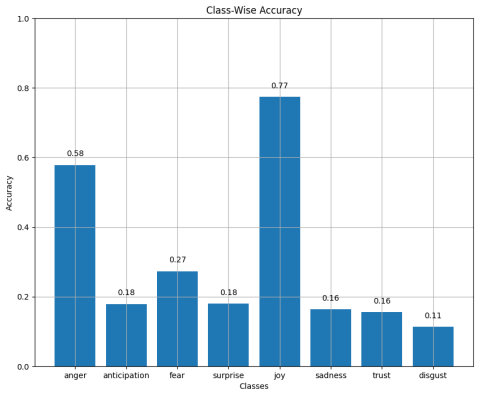
The performance metrics considered for evaluation in these analysis are the following: accuracy, precision, recall, and F1-score. Considering all of these metrics is crucial to accurately evaluate how well each model performs. Regarding the static models implemented in this project, the classification report revealed an accuracy of 34% for the Random Forest algorithm and 24% for SVM. These results can be considered reasonable, given that the task at hand is a multi-class classification

problem with 8 classes.

For the Random Forest model, the class with the highest F1-score is `anger`, which is the third class as for support, sitting at 4436. `Joy` was the most supported class, with 5854 instances, but it came second as for the F1-score, which was of 0.40. The second class based off support, which was `fear` with 4652 instances, had a lower F1-score of 0.31. The remaining classes showed comparable support and F1-scores, averaging around 3500 instances and 0.25 respectively, only `disgust` had a considerably lower F1-score, at 18%. In image 3a, the ratio of true positives versus false positives is displayed, with the Random Forest classifier scoring lower than a random guesser in two out of eight classes. Image 3b shows class-wise accuracy.



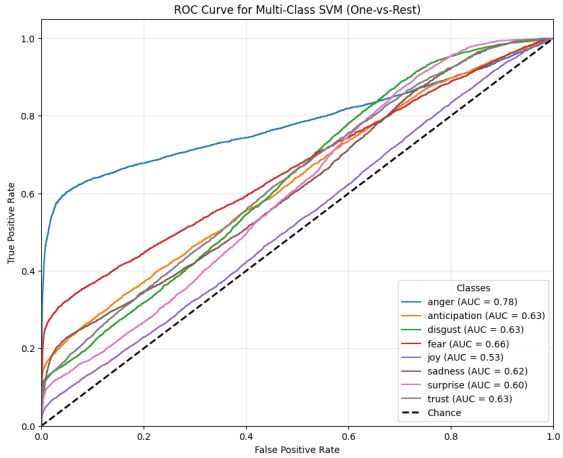
(a) ROC Curve for the Random Forest Classifier



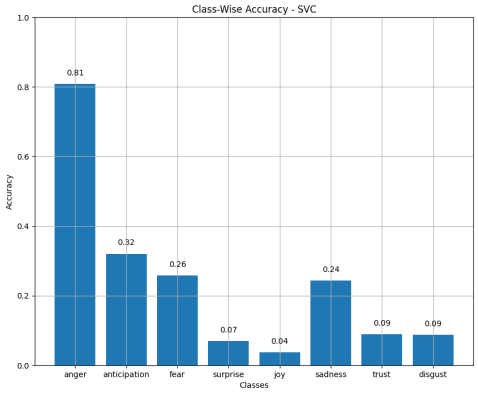
(b) Class-wise Accuracy - Random Forest

Figure 3: Random Forest - plots

For the Support Vector Machine model, the class with the highest F1-score is *fear*, followed by *anger* at 0.31 and *anticipation* at 0.25. As for the other classes, the scores were not as comparable as for Random Forest, oscillating between the 22% of *sadness* and the 7% of *joy*, with *disgust* and *trust* sitting at 16%, and *surprise* at 12%. Surprisingly, *joy* is the least performing class for SVC, dispite having the highest support. In image 4a, the ratio of true positives versus false positives is displayed. The SVM model performs better than a random guesser in all of the classes, therefore surpassing Random Forest despite the lower scores of accuracy. Lastly, image 4b shows the class-wise accuracy by SVM, highlighting a clear and strong disparity in the classification performances among the classes.



(a) ROC Curve - SVM



(b) Class-wise Accuracy - SVM

Figure 4: SVM - plots

As mentioned in the previous chapter, the development of neural networks iterated testing phases and adjustments over different aspects of training. Semi-supervised learning through generation of pseudo labels via the partially trained models generally yielded poor results; on the other hand, downsampling into evenly represented labels gave some minor improvements, for both architectures. The neural networks generally underperformed compared to the static models: the convolutional neural network ultimately reached a test categorical accuracy of 0.2128, while the recurrent neural network had a test accuracy of **MANCA QUALCOSA**.

The graphs below show various performance metrics of the network with the better performances for the convolutional architecture.

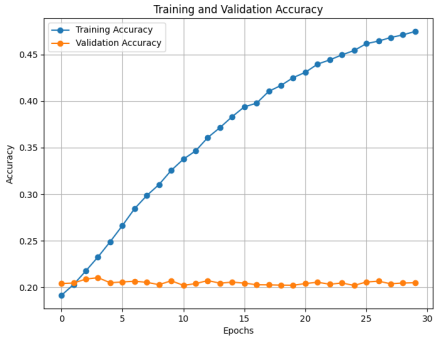


Figure 5: Convolutional Neural Network's training and validation accuracy

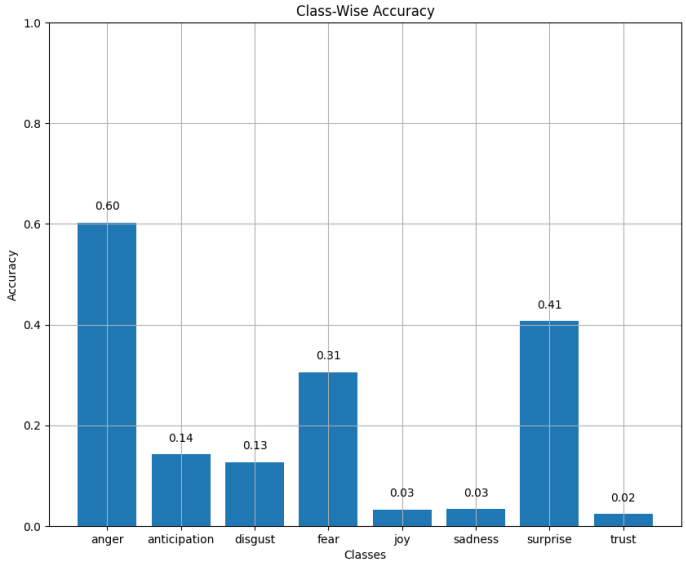


Figure 6: Convolutional Neural Network's class-wide accuracy

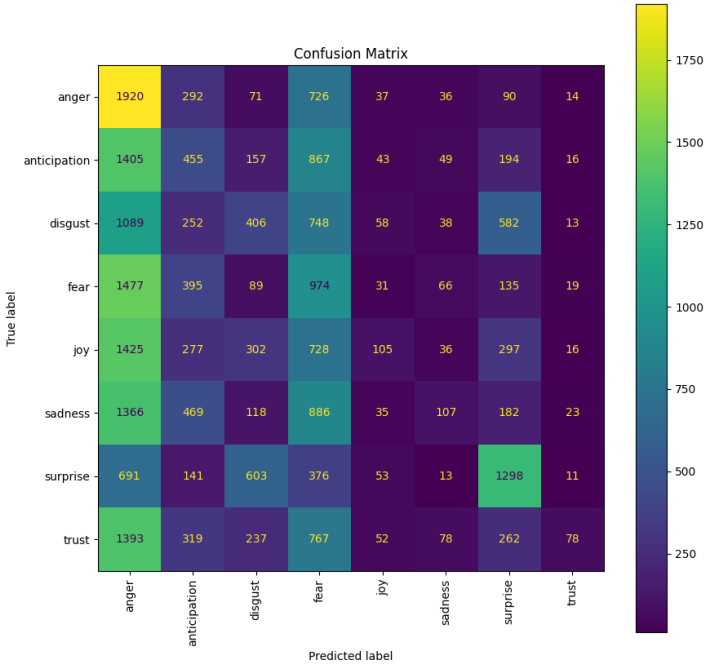


Figure 7: Convolutional Neural Network's confusion matrix

The graphs below show various performance metrics of the network with the better performances for the recurrent architecture **GRAFICI RECURRENT**.

Discussion

The results presented in the previous chapter show general uncertainty in the models. This has a few possible explanations: one of the most likely is the significant overlap of features.

There are clear indications of this issue in the performances of neural networks, particularly the recurrent architecture. This is evident during training, where significant improvements in training accuracy occur only in later epochs, as shown in the training and validation accuracy plot (figure[...]**AGGIUNGERE**): once training accuracy begins to rise meaningfully, the model quickly overfits. Additional insights are provided by the confusion matrix and class-wise accuracy plot (figures [...]**AGGIUNGERE**), which consistently show a dominant class with a high number of predictions and a secondary class with relatively high accuracy, while the remaining classes receive significantly fewer predictions overall.

The convolutional architecture shows similar behavior, though less prominently (figures 5, 6, 7). In this case, training accuracy improves earlier in the epochs, but with minimal corresponding gains in validation accuracy. The confusion matrix and class-wise accuracy plot reveal analogous patterns, though to a lesser extent.

To address this issue, one solution could be applying a filter to exclude the most common words, allowing the remaining words to better convey clear emotional meanings. Another potential unexplored approach is the use of custom metrics during training that focus on class-wise accuracy instead of overall model accuracy.

Another probable issue is the flaw in the generated ground truth, which may also contribute to the poor performance and could be linked to the vague distinction between classes. This possibility can be investigated through explainability analysis, with an example provided below (figure 8).

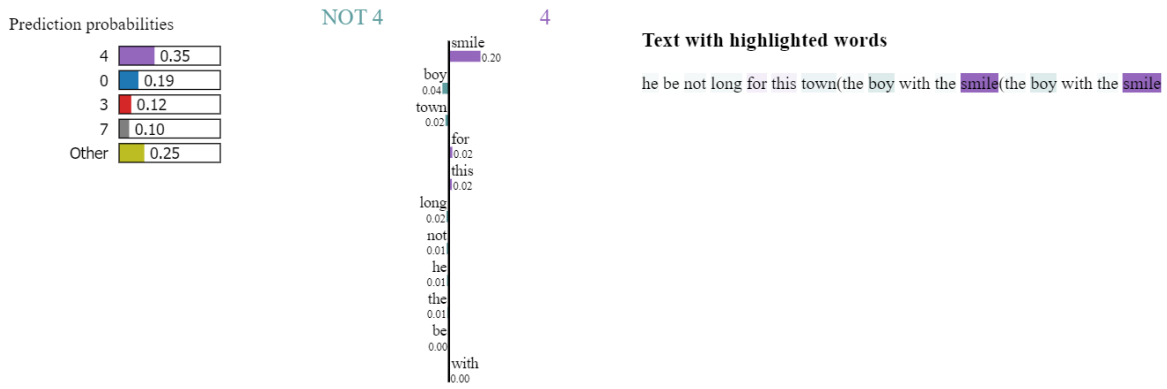


Figure 8: Explainability - visualization

The left section of the graph displays the predicted probabilities for each class. In the center section feature importances are ranked from most to least relevant and divided into two groups: on the right features with a positive influence on the predicted label; on the left, those with a negative influence that suggest the model should consider other classes. The right section of the graph highlights the values of the most important features, using bright colors to indicate features with a positive influence on the prediction.

The example in question illustrates a prediction where the model assigned the label *joy* to the stanza under analysis, but the expected label, assigned by the ALBERT model, was *sadness*. However, the word *smile*, which is brighty highlighted, intuitively suggests that *joy* might be a more plausible class for this stanza, even one that ALBERT could reasonably assign. This showcases the aforementioned issue: the transfer learning approach used to create the ground truth appears to have some limitations. In some instances, labels generated by ALBERT do not seem to be appropriate.

One possible solution is to switch to a different pseudo-labeling model. Alternatively, using probability vectors rather than direct classification for labeling could guide the models toward a regression-based approach. This strategy may provide more informative inputs for the models and facilitate a deeper understanding of their performance issues, helping to identify where and why

they struggle.

Conclusions

This study aimed at exploring various Machine Learning techniques perform an emotion detection task on songs, which are irregular, complex texts. The particular field has many practical applications, such as improving recommendation systems.
The results of the project point towards better performances obtained by more straightforward, simpler models; neural networks generally struggled to find general, meaningful patterns and correlations, leading into suboptimal training and testing performances. These results might have been caused by a series of factors, such as the likely feature overlap between different classes and poor labeling from the model used to generate the ground truth. On the other hand, static models obtained acceptable results.

As mentioned in the previous chapter, there are things that can be done to solve both these issues, such as using alternative models, techniques or sources for generating the ground truth. An other possible approach is to use a probabilistic approach for the labeling process, which can indeed guide models into more informed decisions.

In conclusion, since the static models performed better in this particular study, it may be possible to achieve better performance on the emotion detection task, ultimately advancing its applicability in real-world scenarios.

Bibliography

[1] Robert Plutchik. “A general psychoevolutionary theory of emotion”. In: *Emotion: Theory, research, and experience* 1 (1980).

[2] *Genius Song Lyrics*. URL: https://www.kaggle.com/datasets/carlosgdcj/genius-song-lyrics-with-language-information?select=song_lyrics.csv.

[3] *Albert Base v2*. URL: https://huggingface.co/albert/albert-base-v2.

List of figures