Q-learning algorithm application in successful song lyrics creation process

Group 5
Valentina Brivio, Enrico Cipolla Cipolla, Filip Juren, Beatrice Laini, Arianna Zottoli

Abstract

This paper presents a method to improve the success rate of song lyrics using a reinforcement learning inspired approach. Focusing on the decade 1990-2000, we developed a classifier to predict hit songs from "Billboard top 100" using a Random Forest with TF-IDF vectorization, genre and sentiment scores. We integrated the classifier into a Q-Learning algorithm to modify lyrics, with the aid of Google LLM *Gemma* aiming to maximize hit potential, guided by a reward function composed of a linear combination of the probability given by our classifier and a coherence score. Despite a limited action space, this methodology presents a promising approach to support writers in their song creation process.

1 Introduction

The idea behind our study is to provide a creative data-based support for emerging artists. The digital age has significantly lowered barriers to entry, causing market saturation. Figures from *Alpha Data* show that less than 10% of artists account for over 90% of all streams across major platforms. Being discovered has become significantly more difficult, but the effective use of social media and marketing can be a great launch pad. Virality has in fact become the goal of many artists on social media, and we provide a formula for it.
In this study, we analysed a large dataset of songs from 1960 to 2022, focusing specifically on the 1990-2000 decade. We decided to focus on this decade given the trends of revival in the latest years, led by artists like *TheWeeknd*. Our initial goal was to develop a model to distinguish hit songs from non-hits. We developed a Random Forest classifier that uses TF-IDF vectorization of lyrics, sentiment scores, and genre information. We also appropriately weighted each class considering the imbalance between positive and negative samples.
We integrated this classifier into a Q-learning-inspired algorithm, a class of Reinforcement Learning Algorithm, where an agent (in this case, the song) learns the best actions to take to maximize a reward function (in this case, based on our classifier). We used Google's LLM *Gemma 2B*, known for its efficiency and speed, suitable for environments with limited resources.
To start, we outline the steps involved in collecting data and constructing our models. Next, we present the outcomes of the two parts and discuss limitations and potential improvements of our approach.

2 Method

2.1 Data Collection and Cleaning

The data collection process began by identifying successful songs in the United States. The success was determined by *Billboard* based on radio airplay, online streaming, record sales, and YouTube views. *Billboard* publishes annually the top 100 songs of the year. We grouped them into decades, spanning from 1960 to 2022, obtaining a dataset of nearly 1000 successful songs per decade. We collected the unsuccessful songs from a Kaggle dataset called "English Songs Lyrics"[1], which contains 5 million songs from the same period. We then merged the datasets, eliminating duplicates. Lastly, we obtained lyrics and genre of the songs. Using Selenium and Beautiful Soup, we scraped the lyrics from the Genius website, whereas we obtained the genres by scraping Last.fm and Wikipedia.

[1] https://www.kaggle.com/datasets/razauhaq/english-songs-lyrics

2.2 Classifier Construction

In the classifier construction phase, we decided to focus on songs from 1990 to 2000. The dataset contained 256,805 songs, with title, genre, artist, year, and lyrics for each one. We assigned the label 1 to the 1,083 Billboard Top songs, while labeling the rest as 0. To manage the significant class imbalance, we performed stratified sampling by label and genre. We reduced the number of non-top songs to about 10,000 and kept all the top songs. The resulting subset has 11,080 songs and for each of them we calculated positive, negative, and neutral sentiment scores using *RoBERTa for Sentiment Analysis*. See more in the Appendix A.
Firstly, we conducted an **exploratory analysis** (see more in the Appendix B), then we built a classifier to predict the likelihood of a song being a "Billboard top song". We used the labels indicating whether a song is top (1) or non-top (0) as the response variable. The song lyrics, genres, and the sentiment scores were the predictor variables. For the variable *genre*, we applied one-hot encoding, converting the categorical labels into binary vectors. To represent text data, we transformed the lyrics in vector representations testing 2 different methods: **TF-IDF** with 1-2 grams and **Sentence Transformers**. We combined these techniques with the following classification methods: **Logistic Regression, Random Forest** and **XGBoost**. To handle the class imbalance, we added higher weights to top songs, to improve performance on them. We trained the models on the training set and assessed the final models' performance on the test set. We selected the best hyperparameters using the 5-Fold Cross Validation.

Model selection: As shown in *Table 1*, we compared the models using various metrics on the test set: the average precision, recall, F1-score, and accuracy. We focused on the F1-score because it balances precision and recall, providing a better measure of the classifier's performance in the presence of class imbalances. The best classification model is Random Forest, which uses the genre, sentiment scores, and TF-IDF vector representation of the lyrics as predictors. We observed that including genre and sentiment scores as predictors improves performance compared to considering only the lyrics. Dimensionality reduction, instead, does not improve the model's performance. Based on the results of all the different models, we decided to implement in the q-learning algorithm the TF-IDF Random Forest classifier.

Model	Precision	Recall	F1-score	Accuracy
Tf-idf and Logistic	0.67	0.79	0.70	0.85
Sentence Transformer and Logistic	0.63	0.77	0.65	0.81
Tf-idf and Random Forest	**0.72**	**0.77**	**0.74**	**0.90**
SentenceTransformer and Random Forest	0.68	0.70	0.69	0.88
Tf-idf and XBGoost	0.73	0.73	0.73	0.90
SentenceTransformer and XGBoost	0.77	0.68	0.72	0.92

Table 1: Model Performance Comparison

2.3 Q-learning Algorithm

To modify the song lyrics according to the Billboard 100 criteria, we implemented a simplified version of a Q-learning algorithm (*Figure 1*). A Q-learning algorithm is a type of **reinforcement learning strategy**. The technique allows an agent to know how to optimally act in an environment, maximizing a given reward function.

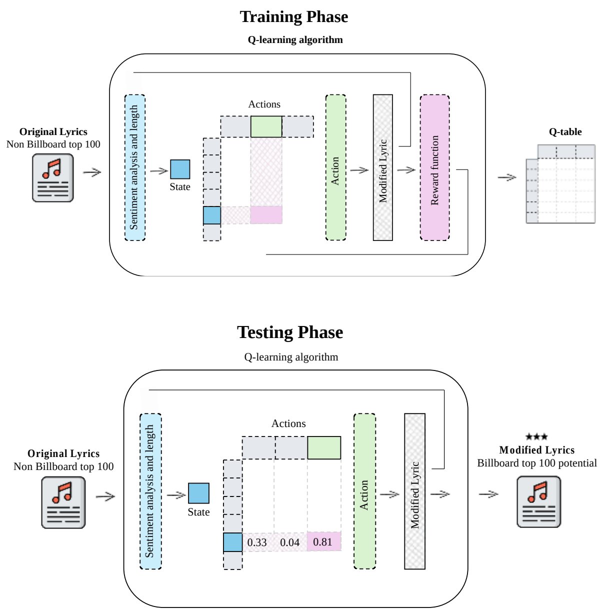

Figure 1: Q-learning algorithm

In our application, the algorithm has the goal of learning how to modify the lyrics to maximize the probability of being a *Billboard Top 100* song. The algorithm learns by updating the Q-table of Q-values. The Q-table has 3 components: the states (the rows), the actions (the columns), and the Q-values or rewards.

- **Q-table States:** The features of a lyric define its state. The states are all the possible combinations of positive, negative, neutral sentiment scores, and number of lines. We compute the sentiment scores of the lyrics using *RoBERTa for Sentiment analysis*. See more in the Appendix A.
- **Q-table actions:** The algorithm can either modify, delete, or add a new line to the lyrics. To execute the generative actions of the algorithm (add and modify), we adopted *Gemma*, a Large Language model developed by Google. For prompt engineering, see more in the Appendix C.
- **Q-values:** For each state, the Q-values are the outputs of the reward function in response to an action of the algorithm.
- **Reward function:** The reward function is a weighted average of the prediction of the Classifier (0 or 1) and the Coherence Score of the lyrics (0 to 1). The Coherence Score avoids that overly modified lyrics receive a high reward, as it measures the similarity of the modified lyrics to the original lyrics. See more in the Appendix D.

In classical Reinforcement Learning settings, the initial state remains fixed among different runs of the training process. In our setting, though, following this approach would overfit on that specific song, and our model would not generalize. Thus, the training dataset comprises a portion of songs from the 1990s, not included in *Billboard Top 100*. The lyrics were then split into lines through a pre-trained sentence tokenizer. In the training phase, for every lyrics, the algorithm analyzes the features: sentiment and length. Based on the features, it analyzes the possible actions, and if the state is new to the algorithm, it chooses a random action. Once the algorithm modifies the song according to the action, it computes the output of the reward function and updates the Q-table.

3 Results

3.1 False Positives Analysis

Firstly, we decided to analyze the False Positives predicted by our Classifier of choice. The tag distribution indicates that R&B (53 instances), pop (44 instances), and rock (31 instances) are the most common genres among the false positives. This suggests that the model struggles more with these genres compared to others. The lyrical and thematic content in R&B, pop, and rock songs might be more diverse or complex, leading to misclassifications. The model might not be as finely tuned to the nuances of these genres as it is for others like rap or country.

The most common words in the false positive lyrics include "love," "baby," "know," "yeah," "want," and "girl." These words are typically associated with popular themes in music, particularly in genres like pop, R&B, and rock. The high frequency of these common words might indicate that the model is overfitting on these terms, which are prevalent in both successful and unsuccessful songs.

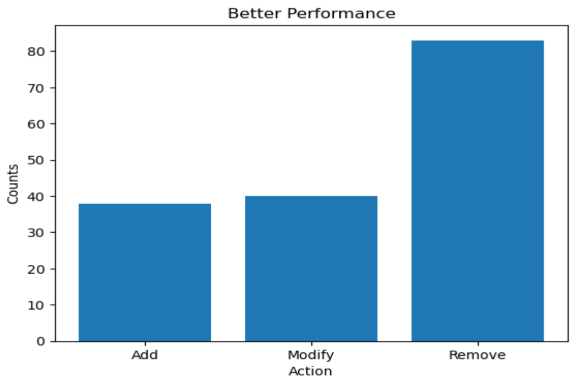

Figure 2: Results from the Q-matrix

3.2 Q-learning algorithm Deployment (Enri)

We can use the final Q-table to decide changes to an existing lyric, non *Billboard Top 100*. Similarly to the training phase, we pass a song to the model, which segments the lyric into lines and computes the state features of it (length and sentiment). Then, the algorithm identifies the state corresponding to these characteristics and selects the action that yielded the highest expected return among the 3. On the notebook (q_learning_algorithn.ipynb) you can find an example of the use of the model.

Results from the Q-table (*Figure 2)* tells us that, on average, removing a line contributes in a better way compared to adding or modifying a line, which shows comparable performances.

4 Related work

The literature related to the use of Large Language Models (LLM) aimed at modifying song lyric is quite extensive. The research by Xichu Ma et Al. (2021) conducted at the National University of Singapore introduced AI-Lyricist, a model that generates lyrics for a given piece of music. Specifically, they design AI-Lyricist to produce lyrics that match appropriately melody, style, and that are semantically meaningful. More similarly to our implementation, the paper by Luketina et al. (2019) discusses the different applications of NLP techniques in Reinforcement Learning (RL). The study highlights how NLP methods can be used to shape RL reward functions, creating context-specific reward systems. We decided to implement the Coherence Score in the reward function for this very same reason. The survey also outlines the challenges involved in integrating natural language with RL, such as the need for better models that can understand and generate natural language. We too experienced the challenges highlighted by the survey. In fact, we encountered multiple issues with the consistency of LLM models, crucial for the RL success.

The literature that attempts at classifying and predicting if a song is a hit is also vast. The majority typically focuses on sound and rhythmic elements, however, the study by Dahnaraj & Logan (2005) proves the predictive power of textual components. The paper shows that only focusing on acoustic information could lead to slightly less accurate results in classification. Abhishek Singhi and Daniel G. Brown (2015) identify the main reasons why a song could enter in the top rankings, focusing on textual elements. Similarly to our study, their analysis relied on a binary classification into “hit” (Billboard *Year-End Hot 100* singles) and “flop”. The study utilizes the Rhyme Analyzer’s set of 24 rhyme and syllable features. This tool analyses structural elements like rhyme density, syllable variation, and meter. They found a positive correlation between the complexity of rhymes and meter and the likelihood of a hit. Despite different papers sharing similar construction methods, we were not able to find a study with the same exact implementation.

5 Conclusions

We built a classification system for Billboard top 100 Song lyrics in the decade 1990-2000, using a Random Forest Classifier with a TF-IDF Vectorizer. We incorporated this classifier as a reward function in a Q-learning inspired algorithm, that leverages Google’s LLM *Gemma* to modify song lyrics.

Although being a very simple setup because of computational and time constraints, we believe that this approach has significant implications for the music industry. It represents a systematic and data-based approach to enhance the likelihood of hit potential songs. We believe that early-career song lyricists could benefit from this approach, having a Grammarly-like aid dedicated to the song lyrics writing process.

5.1 Limitations

We tested the system on a small sample of lyrics due to computational and time constraints. This implied that the matrix ended up being very sparse, with many states that were not visited.

Additionally, parsing the output of the LLM was really challenging. The output is often inconsistent, even with very precise prompts. Around 30% of the training steps involving modifying or adding a line failed because of this problem.

Lastly, the action space remains small, restricted to 3 actions leveraging the LLM. Other actions or suggestions could be included in the action space.

5.2 Future Improvements

Given the worse performances of adding or modifying the lyric compared to removing a line, the first possible improvement involves linking the findings from our classifier directly in the modification process, for example fine-tuning the LLM on the dataset of Billboard top 100 songs. The fine-tuning process would enable also more consistency in the outputs of the LLM.

Regarding the small action space we could introduce actions, such as removing the line based on the sentiment, or some other textual characteristics.

Finally, validation and consultation with industry professionals are important to understand hidden drivers of song success that we did not consider, such as the rhyme scheme.

References

Rolli, M. (2020, September 3). *The top 1% of musicians collect 90% of royalties from streaming. Rolling Stone.* https://www.rollingstone.com/pro/news/top-1-percent-streaming-1055005/

Van Hasselt, H., Guez, A., & Silver, D. (2015). *Deep reinforcement learning with double Q-learning.* arXiv preprint arXiv:1509.06461.

Barbieri, F., Camacho-Collados, J., Neves, L., & Espinosa-Anke, L. (2020). *TweetEval: Unified benchmark and comparative evaluation for tweet classification.* arXiv preprint arXiv:2010.12421

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., & Rocktäschel, T. 2019. *A survey of reinforcement learning informed by natural language*. arXiv preprint arXiv:1906.03926.

Raza, A. H., & Nanath, K. (2020, July). *Predicting a Hit Song with Machine Learning: Is there an apriori secret formula?.* In 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA) (pp. 111-116). IEEE.

Xichu Ma, Ye Wang, Min-Yen Kan, Wee Sun Lee (2021). *AI-Lyricist: Generating Music and Vocabulary Constrained Lyrics.* School of Computing, National University of Singapore, Singapore.

Abhishek Singhi and Daniel G. Brown (2015). *Can Song Lyrics Predict Hits?* University of Waterloo, Cheriton School of Computer Science.

Ruth Dhanaraj, Beth Logan (2005). *Automatic Prediction of Hit Songs.* HP Laboratories Cambridge.

Appendices

A

The methodology employed for the sentiment analysis is a pretrained BERT model, namely Twitter-RoBERTa-base for Sentiment analysis. This is a RoBERTa base model trained on about 58 million tweets and finetuned for sentiment analysis with the TweetEval benchmark proposed in "*TWEETEVAL: Unified Benchmark and Comparative Evaluation for Tweet Classification" (Barbieri et Al. 2020)*. In this paper, it is proved that although Roberta base performs well on the majority of the tasks, Roberta RT, that is Roberta Base trained on the twitter corpus with the TweetEval benchmark, proves more effective.

The model evaluates the song lyrics according to three labels: "negative", "neutral" and "positive". The model then assigns for each document and for each label a score from 0 to 1. These scores are the model's confidence level: the higher the score, the more likely the model believes the text is expressing a particular label. Being standardized, they sum up to one and can be generally interpreted as probabilities.

B

In the exploratory analysis, we examined the distribution of genres, artists, and average sentiment scores between top and non-top songs. The main results are as follows:

- **Genres**: as expected given the period, the three most popular genres in our dataset are pop, rock, and rap, followed by R&B and country. Regarding the top songs, the most popular genres are pop, R&B and rock, while for the non-top songs, they are pop, rock, and rap. The results are shown in *Figure 3*.
- **Artists**: Consistent with what we found before, the top 10 Billboard artists from the 1990s to the 2000s are mainly from rock, pop and R&B genres. The top 3 artists are: Mariah Carey, Madonna and Janet Jackson.
- **Average Sentiment Scores**: Comparing the average sentiment scores of top and non-top songs, the top songs show higher average positive and neutral scores compared to non-top songs. The average negative score is higher for non-top songs compared to top songs. The results are shown in *Figure 4*.

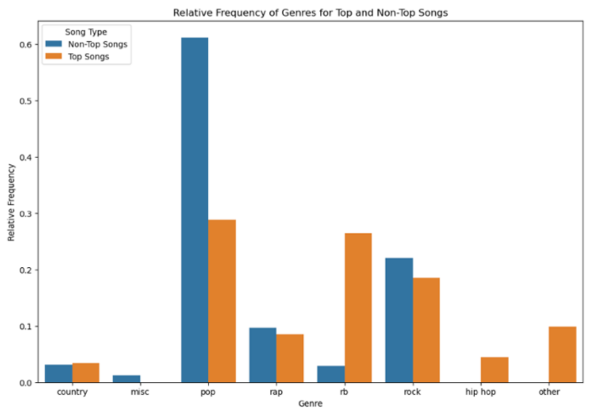

Figure 3: Relative Frequency of Genres for Top and Non-Top Songs.

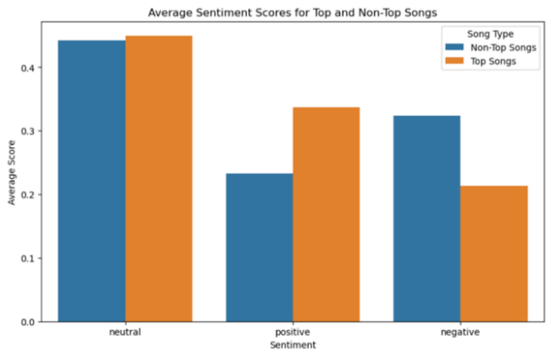

Figure 4: Average Sentiment Scores for Top and Non-Top Songs

C

Gemma Large Language Model is utilized for the task of lyric enhancement and generation. We chose this model This involves iteratively refining existing lines or generating new lines that are semantically coherent with the original text. At the same time however, the modifications need to be coherent with popular Billboard top 100 songs. Therefore, prompt engineering becomes fundamental. The prompts developed were two: one for the function *to_add* and one for the function *to_modify*.

```
input_text_modify = f'This is a lyric:
"{input_lyric}".Write another lyric in place
of the one in square brackets making it
closer to lyrics of top 100 billboard songs
of the years 1990-2000. The line must have
max lenght of {input_length} words and rhyme
with the word {rhyming_word}.'

input_text_add = f'Write one line that
continues this lyric "{input_lyrics}" and
that gets closer to lyrics of top 100
billboard songs of the years 1990-2000. The
line must have lenght of {input_length//2}
words and rhyme with the word {rhyming_word}'
```

The function *to_add* takes an index and adds a new line at said location, based on the previous lyric. The function *to_modify* instead takes an index and modifies the indexed line. Input lyric varies in the two functions, in the modify function the input text is the lyric starting from the line immediately before the target. In the case of add, we decided to provide more context by including in the input text 2 lines before the target. The *rhyming_word* is also established, based on the last word of the line preceding the target line.

Overall, the prompt works effectively, as the LLM provides the lyrics requested. The format of the output is however highly variable. Providing even more indications led to confused results and worsened the performance. We noticed that the best approach would be to let the LLM output the entire *input_lyric* modified and select with regex the section of interest.

```
pattern = r'\n\n\"(.*?)\.\"<eos>'
```

However, due to the high variability of output format, the regex is not always capable of capturing the output. We believe that extensions of the paper might involve improvements in this section.

D

The Coherence Score measures the semantic similarity between two sets of lyrics, namely, the original lyrics and newly generated lyrics. This score evaluates how well the newly generated lyrics maintain thematic and contextual alignment with the original. We utilize the *Universal Sentence Encoder (USE)* from *TensorFlow Hub*, specifically version 4 of the pre-trained model. The encoder creates high-quality sentence embeddings, which can capture the underlying semantics of the text. The model was chosen because it is quite efficient and highly suitable for tasks requiring understanding of sentence relationships.