Introduction

In our project we are studying the artist Eminem (Marshall Mathers).

Our main objective is to divide the artist songs in 2 different periods and that our model will be able to identify the correct period by the song lyrics.

We decided to divide the artist life in to 2 periods, before and after his rehabilitation from drugs.

About Eminem

Marshall Bruce Mathers III born October 17, 1972. known professionally as Eminem

He is credited with popularizing hip hop in Middle America and is regarded as among the greatest rappers of all time.

His success is considered to have broken racial barriers to the acceptance of white rappers in popular music. While much of his transgressive work during the late 1990s and early 2000s made him a controversial figure, he came to be a representation of popular angst of the American underclass and has been cited as influencing many musical artists

Eminem is known for being sober for 16 years (Since April 2008), in his early career he struggled with an addiction to many drugs such as: Ambien, Valium, Vicodin & Weed.

Eminem and his mother shuttled between states, rarely staying in one house for more than a year or two and mostly living with family members

He grew up without a father and a mother who was a drug addicted, in his early career he mentioned on one of his songs that his mom gave him drugs without him knowing when he was a kid.

Eminem developed Slim Shady, a sadistic, violent alter ego. The character allowed him to express his anger with lyrics about drugs, rape and murder

Data Base-

We created our own Data Set by scraping HTML code from the genius website, this site is famous for uploading and publishing the lyrics for most of the songs.

Most of the songs on our Data Base are songs from Eminem official Albums.

Due to a gap we had between the number of songs in each period we added songs to the first period which were singles he released, or songs he released that belongs to crews he was part of at that time such as: D12, Bad Meets Evil, Soul Intent. He was the writer of all the songs in our Data Base.

We found an existing code for scraping Lyrics from Genius and changed it to fit our purposes, a link to our project - link

Project

Our goal is to study Eminem's lyrics by using Natural Language processing (NLP) to find the best parameters to identify the differences between the periods.

In comparison, we used a training model – Random Forest Classifier with cross validation which on the best try managed to produce an accuracy result of 79.4%
We used the best number of features for the TF-IDF matrix which was - 3801.

In our project we will try to out preform the results from the Random Forest Classifier.

The code for this project is in CustomRandomForestClassifier.py class and to run it just run the class RandomForestMain.py

Parameters :

1. **word_frequency**: This parameter calculates the frequency of each word in the song lyrics. It tokenizes the lyrics, converts them to lowercase, removes non-alphanumeric characters, and then counts the occurrences of each word.
2. **named_entities**: This parameter extracts named entities (such as people, organizations, and locations) from the song lyrics using SpaCy's Named Entity Recognition (NER).
3. **dependency_parse**: This parameter analyzes the grammatical structure of sentences in the lyrics by counting the frequency of different dependency parse tags.
4. **topic_modeling**: This parameter performs topic modeling on the song lyrics using Latent Dirichlet Allocation (LDA) to identify common topics discussed in the song.
5. **non_real_words_freq**: This parameter calculates the frequency of non-real words (words not found in the English dictionary) in the lyrics.
6. **curse_words_freq**: This parameter counts the frequency of curse words in the lyrics, based on a predefined list of curse words.

7. **sentiment_analysis**: This parameter performs sentiment analysis on the song lyrics using TextBlob, calculating the overall sentiment polarity, which ranges from -1 (negative) to +1 (positive).

8. **found_albums_names_refs**: This parameter searches for album names mentioned in the lyrics.

9. **found_songs_names_refs**: This parameter searches for song titles mentioned in the lyrics.

   In order to resolve these parameters : predicted_curse_words, predicted_slang_words, predicted_names.
   We trained our own model inside the model which will try and predict curse_words, slang words and names from a list that we initialized at the beginning of the project, from these our goal is that the model will be able to identify similar words with the same context.

   Due to too many mistakes after testing the prediction model we decided to go with a different search techniques

10. **predicted_curse_words**: This parameter predicts the occurrence of curse words we downloaded a data set of 708 swear words from Kaggle(link) which we added to our swear words that we know that are being used by the artist.

11. **predicted_slang_words**: This parameter predicts the occurrence of slang words in the song ,we managed to find a consistency in the lyrics – when analyzing the songs. For each song that ends in "in'" we counted it as slang. When hearing the song you can hear that the word is not pronounced fully. To those words we added known slang words for example : "nigga" , "niggas" , "mcs" .

12. **predicted_names**: This parameter predicts the occurrence of names, eminem is referring many artists in his songs so we used another data set of 2079 artists(link) we cleaned the data from the data set and used only the artists names, to that we added known names of artists, famous people and Eminem family members.

Aggregated Metrics:

- **total_curse_words**, **total_slangs**, **total_names**: These parameters sum the occurrences of predicted curse words, slang words, and names respectively.
- **total_words**: This parameter sums the occurrences of all words in the song.

- **total_unique_words**: This parameter counts the number of unique words in the song.

Structure Analysis Integration:

- **structure_analysis**: This analysis includes metrics such as the number of verses, lines, characters, and average lengths of verses, lines, and words.

Analyzing:

process_files():

Processes each file in 'file_paths' and aggregates results by period. After processing albums individually, it combines these results into a period-level summary and saves it to a CSV file.

analyze_period(df):

Purpose: Analyzes the combined album results to generate period-level statistics.

Calculations:

Average, Median, Standard Deviation, and Coefficient of Variation (CV) for various metrics like avg_total_unique_words, avg_total_words, avg_total_names, etc.

Summation and Counting for aggregated metrics like summed_predicted_curse_words, summed_predicted_slang_words, summed_predicted_names, summed_word_frequency.

Reference Combination: Combines all album and song name references within the period.

Sentiment Aggregation: Aggregates sentiment analysis across all albums in the period.

Named Entity Aggregation: Aggregates named entity counts across all albums in the period.

combine_references(df, column_name):

Combines references (either album or song names) from all entries in a period to create a unified list of references.

aggregate_sentiment(df):

Aggregates sentiment values from all albums in a period to calculate the overall sentiment for the period.

aggregate_named_entities(df):

Aggregates named entities from all albums in a period, calculating the total number of named entities and the number of unique entities.

These methods work together to generate a comprehensive analysis of each period by processing the results from individual albums and combining them into period-level metrics.

<u>Algorithm:</u>

The ourAlgorithm class is designed to analyze a song by comparing it against statistical summaries from two different periods. The class utilizes a weighted vector to score different aspects of the song, and the final decision on which period the song belongs to is based on these scores.

Weighted Vector

The class uses a dictionary called weights, where each key represents a specific parameter used in the song analysis (e.g., word frequency, named entities, etc.). The corresponding value is a weight that signifies the importance of that parameter in the overall analysis. For example:

'word_frequency': 10

'named_entities': 2

'curse_words_freq': 15

In this context, word_frequency has a weight of 10, meaning it is moderately important in the overall analysis. On the other hand, curse_words_freq has a higher weight of 15, indicating that it is more significant when determining the song's period.

These weights are applied during the scoring process. After the song is analyzed, the difference between the song's parameter value and the average value from the two periods is calculated. This difference is then multiplied by the corresponding weight to contribute to the final score of each period.

Coefficient of Variation (CV)

In the context of this algorithm, the Coefficient of Variation (CV) is used to normalize the scores based on the variability of the data for each parameter. CV is calculated as the ratio of the standard deviation to the mean, and it helps to account for the relative variability of each parameter.

For instance, in the case of total_unique_words, the algorithm computes the CV for both periods and uses it to adjust the scores. This adjustment ensures that a parameter with high variability (i.e., a high CV) has a more moderated impact on the final score, reflecting the uncertainty or inconsistency in that parameter.

By incorporating CV, the algorithm can make more balanced and fair comparisons between periods, especially when the data has significant variations. This approach adds robustness to the decision-making process, reducing the likelihood of biased results due to outliers or inconsistent data.

Conclusion

The combination of the weighted vector and the use of CV allows the ourAlgorithm class to perform a nuanced analysis of songs, taking into account both the importance of different parameters and the variability within the data. This ensures that the final decision on which period the song belongs to is both weighted by significance and normalized by variability, leading to more accurate and reliable results.

How to run

1.Run main.py to get song_analysis results for each period.
2.Run CSVAnalyzer.py to get results summary for each period.

3.Run ourAlgorithm.py to get our model results.