

CSCI-544

APPLIED NATURAL LANGUAGE PROCESSING

Classification of Non-English Songs Based on Lyrics

Author 1:

Swati Mutyala

USC ID: 4306529720

e-mail: swatimut@usc.edu

Author 2:

Md Shadman Rafid

USC ID: 4106906001

e-mail: rafid@usc.edu

Author 3:

Pruthvi Gollahalli

Niranjana

USC ID: 3154106640

e-mail: gollahal@usc.edu

Instructor:

Prof. Ron Artstein

Author 4 :

Aarju Goyal

USC ID: 9288448242

e-mail: aarjugoy@usc.edu

Contents

Contents	1
1 Introduction	2
1.1 <i>Interesting / Challenging:</i>	2
1.2 <i>Existing Work:</i>	2
1.3 <i>Our Contributions:</i>	2
2 Methodology	3
2.1 <i>Materials:</i>	3
2.2 <i>Procedure:</i>	4
2.3 <i>Evaluation:</i>	6
3 Results	9
3.1 <i>Spanish Songs Classification:</i>	10
3.2 <i>French Songs Classification:</i>	11
3.3 <i>Italian Songs Classification:</i>	12
4 Discussion	14
4.1 <i>Conclusions:</i>	14
4.2 <i>Future Research:</i>	15
5 References	16
6 Division of Labor	17
A Appendix	18
List of Figures	18
List of Tables	18

1 Introduction

Companies nowadays use music classification, either to recommend new music to their customers (Spotify, Soundcloud) or simply as a product (Shazam). ML and NLP techniques has been quite successful in extracting trends and patterns from large pools of data. We were interested in exploring the relationship between a song's lyrics and its mood. ***We aimed to build classifiers that would learn what lyrics are prevalent in given themes to predict a song's mood solely from its lyrics.***

1.1 *Interesting / Challenging:*

There are not many recommendation systems for Non-English languages[1]. This project aims at addressing this by classifying songs of languages other than English, namely - **Spanish, French and Italian**. It was challenging because there is a scarcity of datasets of Non-English songs based on lyrics, therefore we had to crawl and collect data ourselves along with ***manually labeling each song's lyrics in training and validation data as happy or sad.***

1.2 *Existing Work:*

Music mood recognition and classification to build a genre classification system was done based on audio feature extraction (audio signals)[2].. Mood analysis and classification is used by many companies to improve their recommendation systems[1]. Many classifiers are based on Audio type and beats of songs, but in some songs ***the tempo of the song can contradict or mask the actual essence of the song.*** Hence, we use lyrics to classify the songs.

1.3 *Our Contributions:*

Our project is an extension of the previously mentioned work. Additionally, We ***explored the relationship between a song's lyrics and its mood.*** We built classifiers that would learn what lyrics are prevalent in given themes to predict a song's mood solely from its lyrics. This could also be used to compare how well the moods can be analyzed based on audio signals (done in previous works/projects) and based on lyrics.

2 Methodology

2.1 *Materials:*

There are very limited datasets for song lyrics in different languages other than English. We used [BeautifulSoup](#) in Python for web scraping. We used this to crawl the site [lyricwiki](#) to obtain the lyrics of the songs. **All the three languages' lyrics are in their respective native scripts.**

We have developed *DataFrames* with columns {Lyrics, Mood} of both training data (2000 songs) and validation data (400 songs) for Spanish, Italian and French Languages each. For mood classification, 'Happy' and 'Sad' **labelling is being done manually** for both training and validation datasets.

We used the following Python packages in our project -

- ***requests***: Used to get the urls for all the song links from lyricwiki page.
- ***pandas***: Used for maintaining both training and validation dataframes with 'Lyrics' and 'Mood' columns.
- ***sklearn***:
 - ***sklearn.preprocessing.LabelEncoder***: Used for labeling 'Sad' as 1 and 'Happy' as 0.
 - ***sklearn.feature_extraction.text.CountVectorizer***: Used for employing Count Vectorizing method.
 - ***sklearn.feature_extraction.text.TfidfVectorizer***: Used for employing TF-IDF Vectorizing method.
 - ***sklearn.naive_bayes.MultinomialNB***: Used for incorporating Multinomial Naïve Bayes.
 - ***sklearn.naive_bayes.BernoulliNB***: Used for incorporating Bernoulli Naïve Bayes.
 - ***sklearn.pipeline.Pipeline***: Used for employing Four pipelines in Grid Search.

- ***sklearn.model_selection.GridSearchCV***: It is an exhaustive search over specified parameter values for an estimator. Grid Search implements a ‘fit’ and a ‘score’ method. It fits the data into the vector representation as per the requirements of the estimator. After estimation, it gives the score (F1-score) achieved by the technique.
- ***sklearn.metrics***: Used for both confusion matrix and scores.
- ***nlTK***:
 - ***nlTK.stem.SnowballStemmer***: Used for stemming Spanish, French and Italian songs.
 - ***nlTK.word_tokenize***: Used for tokenization and lemmatization.
 - ***nlTK.corpus.stopwords***: Used for removing stopwords.

We tried downloading user-provided tags from [Last.fm](https://last.fm), but soon found out that tags like happy and sad (and other related adjectives) were only available for a very small subset of songs and were also sometime mislabeled.

2.2 Procedure:

- **Preprocessing Data:**
 - **Standardize lyric formatting**: We changed all lyrics to lower-case and removed any special characters[3].
 - **Stop Word Removal**: We imported a list of stop words from `nlTK.corpus` and removed stop words from the lyrics[3].
 - **Tokenization and Lemmatization**: We tokenized characters and changed all variations of words to the same base words for better calibration of our data such that different tenses or plural forms are interpreted as the same word[3].
 - **Snowball Stemming**: We employed snowball stemming which is a small string processing language designed for creating stemming algorithms for use in information retrieval.
- **Mood Label Classification**: The following guidelines were used to assign *happy* and *sad* mood labels to the songs. If the song had a

somewhat dark theme, e.g. violence, war, killing, etc. we labeled it as sad. Also, if the artist seemed to be upset or complaining about something, or if the song was about a “lost love”, it was labeled as sad. Everything else was labeled as happy.

- **Model Creation and Classification:**

- **Vector Representations:** This feature extraction aims to convert raw text to numbers that represent information well, i.e. that quantify signature words in a text. In this project we employed:
 - * **Count Vectorizer:** It provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary[4][5][3].
 - * **TF-IDF Vectorizer:** It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus[4][5][3].
- **Naïve Bayes Classifications:** We decided to compare two Naive Bayes classifiers since they are known to perform well in document classification and text analytics:
 - * **Multinomial NB:** This model treats the data as though it is discrete and each conditional probability $P(X_i | y)$ is a multinomial distribution[6][7].
 - * **Bernoulli NB:** This model treats the data as though it is binary, or can only take on two values. In this case, it classifies a song by generating an indicator for each word, either 1, indicating the presence of the word in the text, or 0, indicating absence[6][8].
- **Grid Search:** GridSearch implementation in scikit-learn made the search for the “optimal” combination between pre-processing steps and estimator parameters very convenient.

- **Grid Search Combinations:**
 - * Count Vectorizer + Multinomial NB
 - * Count Vectorizer + Bernoulli NB
 - * TF-IDF + Multinomial NB
 - * TF-IDF + Bernoulli NB

2.3 *Evaluation:*

For evaluating the best combination of Vectorization and Naïve Bayes, we used Grid Search which optimizes cross-validated grid-search over a parameter grid[5]. There are four combinations and using Pipeline module, we created four Pipelines:

- **Pipeline_1 (Count Vectorizer + BernoulliNB):** We ran it for n_gram range from unigrams to trigrams.
 - ***Spanish:*** The best values are -
 - * best_score: 0.017
 - * best vect_ngram_range: (2, 2)
 - ***Italian:*** The best values are -
 - * best_score: 0.168
 - * best vect_ngram_range: (1, 1)
 - ***French:*** The best values are -
 - * best_score: 0.132
 - * best vect_ngram_range: (1, 1)

- **Pipeline_2 (TF-IDF Vectorizer + BernoulliNB):** We ran it for n_gram range from unigrams to trigrams.
 - ***Spanish:*** The best values are -
 - * best_score: 0.274
 - * best vect_ngram_range: (1, 1)
 - ***Italian:*** The best values are -
 - * best_score: 0.265
 - * best vect_ngram_range: (1, 1)
 - ***French:*** The best values are -
 - * best_score: 0.177
 - * best vect_ngram_range: (1, 1)
- **Pipeline_3 (Count Vectorizer + MultinomialNB):** We ran it for n_gram range from unigrams to trigrams.
 - ***Spanish:*** The best values are -
 - * best_score: 0.443
 - * best vect_ngram_range: (2, 2)
 - ***Italian:*** The best values are -
 - * best_score: 0.393
 - * best vect_ngram_range: (1, 1)
 - ***French:*** The best values are -
 - * best_score: 0.480
 - * best vect_ngram_range: (1, 1)

- **Pipeline_4 (TF-IDF Vectorizer + MultinomialNB):** We ran it for n_gram range from unigrams to trigrams.
 - ***Spanish:*** The best values are -
 - * best_score: 0.344
 - * best vect_ngram_range: (1, 1)
 - ***Italian:*** The best values are -
 - * best_score: 0.183
 - * best vect_ngram_range: (2, 2)
 - ***French:*** The best values are -
 - * best_score: 0.035
 - * best vect_ngram_range: (3, 3)

From the above values, we learnt that Pipeline_3 which is the combination of **Count Vectorizer and Multinomial Naïve Bayes** stood out in the comparative study.

Manual checking is a tedious and time-consuming task requiring human resource. There is no gold standard dataset available for comparing performances of the developed systems. Thus, **manual evaluation** was performed on the validation data and the results.

3 Results

For each language model in Pipeline_3 mentioned above, we used **a training set of 2000 songs and a validation set of 400 songs** . We evaluated each model by using *accuracy_score*, *precision_score*, *recall_score* and *f1_score* from *sklearn.metrics*.

Accuracy computed is the subset accuracy, meaning that the predicted labels must exactly match the actual labels provided.

$$\frac{(TruePositives + TrueNegatives)}{(TruePositives + TrueNegatives + FalsePositives + FalseNegatives)} \quad (3.1)$$

Precision is the fraction of predicted positive events that are actually positive.

$$\frac{TruePositives}{(TruePositives + FalsePositives)} \quad (3.2)$$

Recall (otherwise known as sensitivity) is the fraction of positive events that predicted correctly.

$$\frac{TruePositives}{(TruePositives + FalseNegatives)} \quad (3.3)$$

F1-Score is the harmonic mean of recall and precision.

$$\frac{(2 * precision * recall)}{(precision + recall)} \quad (3.4)$$

In addition, we generated a **confusion matrix** for each model to understand how the data was classified at a micro-level.

3.1 *Spanish Songs Classification:*

- **Training set:** 2000 songs

Spanish Confusion Matrix - Training dataset

A heatmap representing the confusion matrix for the training dataset. The y-axis is labeled 'actual class' with categories 'sad' and 'happy'. The x-axis is labeled 'predicted class' with categories 'happy' and 'sad'. The cells contain the following counts: (actual=sad, predicted=happy) is 37, (actual=sad, predicted=sad) is 1207, (actual=happy, predicted=happy) is 524, and (actual=happy, predicted=sad) is 232. The colors range from light blue for high counts to dark blue for lower counts.

actual class	predicted class	
	happy	sad
happy	524	232
sad	37	1207

Figure 3.1.1: *Spanish Training*

- **Validation set:** 400 songs

Spanish Confusion Matrix - Validation dataset

A heatmap representing the confusion matrix for the validation dataset. The y-axis is labeled 'actual class' with categories 'sad' and 'happy'. The x-axis is labeled 'predicted class' with categories 'happy' and 'sad'. The cells contain the following counts: (actual=sad, predicted=happy) is 14, (actual=sad, predicted=sad) is 227, (actual=happy, predicted=happy) is 60, and (actual=happy, predicted=sad) is 99. The colors range from light blue for high counts to dark blue for lower counts.

actual class	predicted class	
	happy	sad
happy	60	99
sad	14	227

Figure 3.1.2: *Spanish Validation*

- **F1-Score:**

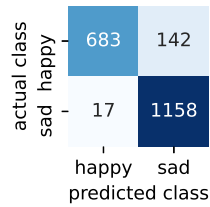
	ACC (%)	PRE (%)	REC (%)	F1 (%)
Training	86.55	93.40	69.31	79.57
Validation	71.75	81.08	37.74	51.50

Figure 3.1.3: *Spanish F1-Score*

3.2 *French Songs Classification:*

- **Training set:** 2000 songs

French Confusion Matrix - Training dataset

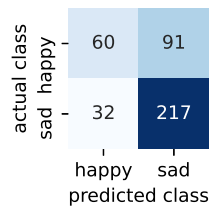


actual class	predicted class	
	happy	sad
happy	683	142
sad	17	1158

Figure 3.2.1: *French Training*

- **Validation set:** 400 songs

French Confusion Matrix - Validation dataset



actual class	predicted class	
	happy	sad
happy	60	91
sad	32	217

Figure 3.2.2: *French Validation*

- **F1-Score:**

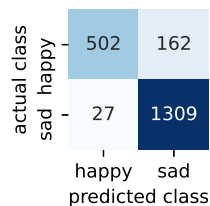
	ACC (%)	PRE (%)	REC (%)	F1 (%)
Training	92.05	97.57	82.79	89.57
Validation	69.25	65.22	39.74	49.38

Figure 3.2.3: *French F1-Score*

3.3 *Italian Songs Classification:*

- **Training set:** 2000 songs

Italian Confusion matrix - Training dataset

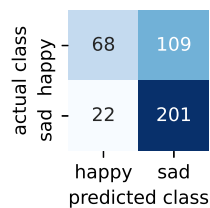


actual class	happy	502	162
	sad	27	1309
		happy	sad
		predicted class	

Figure 3.3.1: *Italian Training*

- **Validation set:** 400 songs

Italian Confusion matrix - Validation dataset



actual class	happy	68	109
	sad	22	201
		happy	sad
		predicted class	

Figure 3.3.2: *Italian Validation*

- **F1-Score:**

	ACC (%)	PRE (%)	REC (%)	F1 (%)
Training	90.55	94.90	75.60	84.16
Validation	67.25	75.56	38.42	50.94

Figure 3.3.3: *Italian F1-Score*

- **Comparison to existing work:** Since we found no existing song classifiers for these languages, we had to do manual evaluation to check our system's performance. Compared to our initial expectations, our system performed reasonably well.
- **What the system gets right:**
 - The classification accuracy is fairly good which shows that Naïve Bayes was a good document classifier to begin with. Moreover, Naïve Bayes works well on small datasets which agreed with our project.
 - The classification based on lyrics reflects the actual meaning of the song which might otherwise be lost if classified based just on audio.
 - Manual labeling helps get a human point of view.
- **What the system gets wrong:**
 - Differentiating songs into only two moods has restricted the actual classification where there are no proper boundaries defined for happy and sad. Having more moods would have helped classify songs precisely.
 - Our system uses only two types of Naïve Bayes because there was insufficient time to explore other methods.

4 Discussion

4.1 *Conclusions:*

- The datasets were heavily skewed towards ‘Sad’ but instead of discarding hundreds of thousands of eligible data points, we decided to keep our dataset as is and produce results despite this limitation.
- The combination of **1-gram tokenization**, stop word removal and feature vectors based on count vectorization in combination with a Multinomial Naïve Bayes model seemed to work best (in terms of the F1-score). **Final Classification : Count Vectorizer + Multinomial NB**
- The differences between various pre-processing steps and parameter choices were rather minor except for the choice of the n-gram sequence lengths.
- **Manual labeling** for training and validation sets gave better results.
- We tried using our Spanish Classifier to classify a bunch of French validation data. Using the French Classifier, we see that the F1-Score for that validation data is just 22.32%. From this we can conclude that different words might not be classified in the similar way among all languages.

	ACC (%)	PRE (%)	REC (%)	F1 (%)
Training	86.55	93.4	69.40	79.64
Validation	56.50	46.3	14.71	22.32

Figure 4.1.1: *French classification scores using Spanish Classifier*

Example: Consider the word ‘love’, being a neutral word it could be classified as ‘Sad’ in Spanish but most French songs might be happy ones with the word ‘love’ in the lyrics

- **Over-fitting:** It is an evaluation step to check whether the Grid Search is giving optimum results. The best n_gram range given by Grid Search for Italian is uni-grams. So, when using bi-grams in our final classification of Italian instead of uni-grams, we see that the classifier over-fits the data.

	ACC (%)	PRE (%)	REC (%)	F1 (%)
Training	99.85	99.70	99.85	99.77
Validation	62.25	77.08	20.90	32.89

Figure 4.1.2: *Over-fitting (Italian Song Classifier)*

- **Overall contribution to the general NLP community:** Providing a basic Music Classifier for languages - Spanish, Italian and French. This could be extended to different languages or it can be improved by adding other uncommon NLP models/techniques. The classification based on lyrics is fairly new and is still being explored.

4.2 *Future Research:*

- Sentimental analysis can be done on the lyrics: Classifying lyrics as negative, positive or neutral might be interesting to investigate the common sentiments of a given genre.
- Classify song lyrics based on other features like “**year of release**”: Some moods/genres might be more popular in some years than others, so adding year released as a feature might be informative.
- Utilizing **audio file features** in addition to lyrics to classify songs to their respective moods. Our current analysis doesn’t focus on the rich auditory features of a song, which would be valuable to classify genres.
- Look into more models that might produce higher accuracy: Try the less common methods (like Word2Vec Feature Extraction, Neural Networks, Decision Trees, Logistic Regression) in our project, which might produce models with better accuracy.

5 References

- [1] Patra, B. Gopal, D. Das, and S. Bandyopadhyay, “Retrieving similar lyrics for music recommendation system,” *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, 2017.
- [2] “Prediction of song genres.” <https://medium.com/better-programming/predicting-a-songs-genre-using-natural-language-processing-7b354ed5bd80>.
- [3] “Text analytics for beginners using nltk (n.d.).” <https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk>. Retrieved: March 19, 2019.
- [4] “Generating wordclouds in python. (n.d.).” <https://www.datacamp.com/community/tutorials/wordcloud-python>. Retrieved: March 19, 2019.
- [5] A. Long, “Understanding data science classification metrics in scikit-learn in python.” <https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>. Retrieved: March 19, 2019.
- [6] P. Kaviani and S. Dhotre, “Short survey on naive bayes algorithm,” *International Journal of Advance Engineering and Research Development*, vol. 4, 2017.
- [7] S. Xu, Y. Li, and W. Zheng, “Bayesian multinomial naïve bayes classifier to text classification,” *Advanced Multimedia and Ubiquitous Engineering: MUE/FutureTech*, pp. 347–352, 2017.
- [8] “The bernoulli model (n.d.).” <https://nlp.stanford.edu/IR-book/html/htmledition/the-bernoulli-model-1.html>. Retrieved: March 19, 2019.

6 Division of Labor

Member Name	Contribution
Swati Mutyala	Proposal + Web Crawling + Manual Classification (1800 songs) + Naive Bayes Classification + Grid Search (CV and BNB, CV and MNB, TF-IDF and BNB) + Report
Pruthvi Gollahalli Niranjana	Proposal + Web Crawling + Manual Classification (1800 songs) + Naive Bayes Classification + Grid Search (TF-IDF and MNB) + Report
Md Shadman Rafid	Web Crawling + Manual Classification (1800 songs) + Vectorization(CV) + Evaluation + Online Repository + Report
Aarju Goyal	Web Crawling + Manual Classification (1800 songs) + Preprocessing + Vectorization(TF-IDF) + Evaluation + Poster

Table 6.1: *Division of Labor Between Teammates*

1. CV : Count Vectorizer
2. TF-IDF : Term Frequency - Inverse Document Frequency Vectorizer
3. BNB : Bernoulli Naïve Bayes
4. MNB : Multinomial Naïve Bayes

Word count for the document: 1942

A Appendix

List of Figures

3.1.1	<i>Spanish Training</i>	10
3.1.2	<i>Spanish Validation</i>	10
3.1.3	<i>Spanish F1-Score</i>	10
3.2.1	<i>French Training</i>	11
3.2.2	<i>French Validation</i>	11
3.2.3	<i>French F1-Score</i>	11
3.3.1	<i>Italian Training</i>	12
3.3.2	<i>Italian Validation</i>	12
3.3.3	<i>Italian F1-Score</i>	12
4.1.1	<i>French classification scores using Spanish Classifier</i>	14
4.1.2	<i>Over-fitting (Italian Song Classifier)</i>	15

List of Tables

6.1	<i>Division of Labor Between Teammates</i>	17
-----	--	----

List of Formulae

3.1	<i>Accuracy</i>	9
3.2	<i>Precision</i>	9
3.3	<i>Recall</i>	9
3.4	<i>F1-Score</i>	9