Retrieving Similar Lyrics for Music Recommendation System

Braja Gopal Patra[1], Dipankar Das[2], and **Sivaji Bandyopadhyay[2]**
[1]School of Biomedical Informatics,
The University of Texas Health Science Center at Houston, Houston, Texas, USA
[2]Department of Computer Science & Engineering, Jadavpur University, Kolkata, India
brajagopal.cse@gmail.com, dipankar.dipnil2005@gmail.com,
sivaji_cse_ju@yahoo.com

Abstract

Presently, millions of music tracks are available on the web. Therefore, a music recommendation system can be helpful to filter and organize music tracks according to the need of users. To develop a recommendation system, we need an enormous amount of data along with the user preference information. However, there is a scarcity of such dataset for Western as well as Hindi songs. This paper presents a similar lyrics retrieval system for Hindi songs using features collected from lyrics. A romanized Hindi lyric dataset is collected from the web. The collected dataset is noisy, and several forms of a single word are present in it, thus an unsupervised stemming algorithm is proposed to reduce the size of N-grams. The Self-Organizing Feature Maps (SOFMs) based similar lyrics retrieval system achieves the maximum F-measure of 0.749.

1 Introduction

The improvement in digital technology has led the music digitally available to all Internet users. The development of nanotechnology made storage devices portable, and nowadays, any handheld devices can store thousands of tracks. Whenever a user has an enormous number of choices for listening to music (like browsing web or personal storage devices), the user is overwhelmed by options. The recommender system comes as a savior and filters the songs that are suitable for that user at that moment. It also maximizes the user's satisfaction by playing appropriate song at the right time, and, meanwhile, minimize the user's effort (Hu, 2014). The recommendation problem can be seen as a ranking problem, and it creates a list of suitable songs for users.

Many music streaming services for Western music emerged in recent years, such as *Google Play Music*[1], *Apple music*[2], *Last.fm*[3], *Pandora*[4], *Spotify*[5], and so forth; and some of them are not available in India. These music streaming applications store user preferences and recommend users what they want to listen. In India, several music streaming services were started recently and those are *Apple music*, *Gaana*[6], *Hungama*[7], *Saavn*[8], and *Wynk music*[9] etc. Most of them do not recommend songs and those are just a music library. Youtube is one of the video streaming services which provides recommendations based on the collaborative filtering (Davidson et al., 2010). It also provides facility to search using title of song and it can also search a video using any keywords within a lyric body only when full lyric is available in the description.

There is a keen interest in accessing music contents nowadays. Available search engines or information retrieval (IR) systems allow users to search a song by the metadata such as song title, artist, album name. Incorrect metadata can lead to wrongly searched data, and without any metadata, it is not possible to search a song. Again, the current IR systems give particular search results based on the query rather than similar lyrics to a query. It was observed that a lyrics provide different semantic information than audio for some of Hindi songs, i.e., the annotators perceived differ-

[1]https://play.google.com/music/listen
[2]https://www.apple.com/music
[3]https://www.last.fm
[4]https://www.pandora.com
[5]https://www.spotify.com
[6]https://gaana.com
[7]http://www.hungama.com
[8]https://www.saavn.com
[9]https://www.wynk.in/music

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 290–297

ent moods while reading lyrics and listening to the corresponding songs (Patra et al., 2016b). People are interested in listening to songs specific to situation and mood (Duncan and Fox, 2005). There is a need for recommendation system based on information within the music as well as the metadata of music such as mood, genre, artist name, and so on.

Music similarity measures can help to understand why two music pieces are perceived alike by the listener and to guide the user in efficiently retrieving desired piece of music (Schedl et al., 2011). Query by hamming helps to find an exact song with respect to a query humming. Again, a lyrics based retrieval system could be helpful for searching similar songs. Till date, there is no such lyrics retrieval system developed for Hindi songs.

In the present task, we collected a huge lyric dataset for Hindi songs written in Romanized English characters. Though, developing lyrics retrieval system for Hindi songs of *Romanized* characters is a difficult task. The main reason is that the processing of such text is difficult for an n-gram based system as a single word is written with different variations, for example, "*Ajnabi*" and "*Ajnabiii*". The existing stemming and parts-of-speech taggers are available for either *utf* or *WX* format of Hindi characters. All sentiment lexicons are also available in *utf* format and these can not be used for *Romanized* characters.

Several text normalization techniques and an unsupervised stemming algorithm have been implemented to handle unstructured data. Finally, we developed unsupervised IR systems to retrieve similar songs with respect to a query song using Self-Organizing Feature Maps (SOFMs) and Document level word embeddings followed by a baseline system using Fuzzy C-means (FCMs) clustering. The similar lyrics retrieval system can be combined with existing metadata based recommender to give a better performance. It is also useful for recommending a song where little or no metadata (genre, mood) is available.

This paper is organized as follows: Section 2 describes the related work on similar lyrics retrieval and works on Indian music. The dataset and preprocessing techniques are discussed in Section 3. Section 4 describes SOFMs and Doc2Vec for developing the retrieval system. The developed systems with comparisons are described in Section 5. Finally, Section 6 concludes and provides avenues for further work.

2 Related Work

Automatic playlist generation is one of the fundamental problem in music information retrieval (MIR) to overcome the manual song selection. Automatic playlist generation can be seen as recommendation problem. The biggest challenges faced while developing recommendation system are collecting a huge dataset and metadata, then getting user preferences or feedback. The recommender system can be developed based on both audio and lyrics to solve the problem of manual playlist selection or generation.

There have been multiple experiments which process lyrics. Mahedero et al. (2005) performed the language identification, structure extraction, theme extraction, and similarity searches mainly on Western lyrics. The mood (Hu et al., 2009; Zaanen and Kanters, 2010) and genre (Mayer et al., 2008) classification have also been performed using lyric features of Western music.

Another interesting task named as LYRICSRADAR was developed by Sasaki et al. (2014) and they visualized the topics of Japaneses lyrics by using a Latent Dirichlet Allocation (LDA). Several experiments were performed on retrieving similar lyrics for Western songs by (Mahedero et al., 2005; Knees et al., 2007; Schedl et al., 2011), Mandarin lyrics by (Wang et al., 2010), and Chinese lyrics by (Han et al., 2015).

2.1 Experiments on Indian Songs

MIR in Indian songs is at early stage. Recently, mood classification of Hindi songs have been performed using audio (Ujlambkar and Attar, 2012; Patra et al., 2013; Patra et al., 2016a), lyrics (Patra et al., 2015), and combination of both (Patra et al., 2016b; Patra et al., 2016c). The datasets used in above experiments are small and not adequate for development of recommendation system.

Some other tasks like raga identification of south Indian Carnatic music (Sridhar et al., 2011), multimodal sentiment analysis of Telugu songs (Abburi et al., 2016), melody identification of Carnatic music (Koduri et al., 2011), rhythm analysis of Indian art music (Srinivasamurthy et al., 2014) etc. have been performed till date. To the best of author's knowledge, almost no work exists for retrieving similar lyrics for any of the Indian songs.

3 Dataset and Preprocessing

3.1 Dataset

As no task on retrieving the similar lyrics has been performed till date, no dataset is also available. A total of 31,171 lyrics of Hindi songs have been collected from several websites[10][11][12] using a web crawler developed by us. Among them, 25,088 lyrics are in *Romanized* characters and 6,083 lyrics are in *utf-8* characters. It is good to develop similar lyrics retrieval system on the lyrics having *utf-8* characters, but the number of such lyrics is insufficient to develop an IR system. Thus, we discarded the latter set of lyrics for the current experiment and developed similar lyrics retrieval system only on the *Romanized* lyrics.

There were many HTML tags and other junk characters, thus, several preprocessing steps were performed on the collected dataset to ensure the quality. The variations in *Romanized* words motivated us to remove the duplicate characters and perform stemming.

3.2 Preprocessing

We removed HTML tags and junk characters from the lyrics. *Mukhda* (the starting stanzas of a song) is repeated in lyric and the importance of words in *mukhda* is higher than the words in *antara* (inside stanzas of a lyric) (Beaster-Jones and Sarrazin, 2016). Again, we observed that the systems are biased towards *mukhda* because of the higher word frequency. We performed our experiments both before and after removing repeated lines from the lyrics.

As the number of n-grams is quite high, and a huge computational power is required to perform the search. Thus, preprocessing is an important step reduce the n-gram size and the steps for preprocessing are sequentially discussed as follows.

3.2.1 Removing Duplicate Characters

There were many words having multiple repeated characters. To reduce the n-gram size, the frequency of any repetitive character were reduced to two. For example, the word '*Ajnabiiiiiiiii*' contains multiple '*i*' and those multiple occurrences of '*i*' was replaced by '*ii*'. At the end, word '*Ajnabiiiiiiiii*' became '*Ajnabii*'. Later on, the proposed stemming algorithm is used on above word to reduce *ii* to *i*.

3.2.2 Stemming Algorithm

It was observed that stemming algorithm improves the performance of any information retrieval system (Moral et al., 2014). Many words are written in different forms, for example, the word '*marega*' (to beat) is written as '*maregaa*' in another lyric. There are several tool for stemming or lemmatization, but all of them are for either *WX* or *utf-8* characters and these tools are not useful for current scenario. Thus, an unsupervised stemming algorithm was developed to reduce the number of n-grams present in corpus and to improve the system performance. We hope that the proposed unsupervised stemming algorithm is useful for handling noisy data from different languages.

The algorithm contains two main steps namely collecting suffix and stemming. The first step describes how suffixes are collected from words by comparing the similar words. Details of the first step is given below.

First, all unique words are stored in a dictionary after sorting them alphabetically and length wise. This step is performed to reduce the number of matching during stemming. For each word, a suffix is searched by comparing with another word starting with same character. If the difference between two words are less than equals to three then rest of the words (after removing the common characters) is considered as a suffix and the word matching is done from the left to right. For example, the words *Ajnabi* and *Ajnabii* have only single character difference, i.e. *i*. Thus, *i* is considered as suffix and inserted in the suffix list. If difference between the words is more than 3, then rest of the words after removing common characters may be a probable suffix. Such suffixes are collected and checked manually before implementing. The second step describes how stemming is performed using the collected suffixes and all inflected words are removed from the final dictionary. The pseudo code for normalizing the words is given in algorithm 1.

After using the stemming algorithm, the word '*maregaa*' is normalized to '*marega*'. Several words having different suffixes at the end were observed in lyrics, for example, '*dost*' (friend), '*dosti*' (friendship), '*doston*' (friends) and the words '*dosti*' and '*doston*' are normalized to '*dost*'.

We removed stopwords while constructing the

[10] http://www.lyricsmint.com
[11] http://smriti.com
[12] http://www.indicine.com

Algorithm 1 Pseudo code for unsupervised stemming

procedure COLLECTING SUFFIX
    Store all unique words in $dict$ after sorting them alphabetically and length wise
    for each $word_i$ in $dict$ **do**
        for each $word_j$ in $dict$ **do**
            if $word_j$.startswith($word_i$) and len($word_j$)-len($word_i$) ≤ 3 **then**
                $word_j$ is inflected form of $word_i$
                $suffix\_list$.append($word_j$.replace($word_i$))
            else
                if $word_j$.startswith($word_i$) and len($word_j$)-len($word_i$) ≥ 4 **then**
                    $diff\_suffix\_list$.append($word_j$.replace($word_i$))
                else
                    continue
    Each $suffix$ in $diff\_suffix\_list$ is manually checked
procedure STEMMING
    for each $word_i$ in $dict$ **do**
        for each $word_j$ in $dict$ **do**
            if $word_j$.startswith($word_i$) **then**
                x $\leftarrow$ $word_j$.replace($word_i$)
                if x in $diff\_suffix\_list$ **then**
                    $word_j$ is removed from $dict$
            else
                continue

n-grams. Initially, we had a list of stopwords for Hindi, but it was in *utf-8* format. Thus, another stopword list was prepared manually in *Romanized* format. This list contains all possible form of a single word, for e.g. '*yun*' and '*yuun*'. The stopwords list contains 307 words in *Romanized* format.

A total of 95,415 unigrams were obtained from the whole corpus after removing the HTML tags and junk characters. The number reduced to 94,960 after removing the stopwords from lyrics. Further, the duplicate characters were removed and this process obtained a total of 75,620 unigrams. Finally, the unsupervised stemming technique was used and it reduced the unigram size to 37,693, though some errors were observed during the stemming process. For example, stemming algorithm trimmed the word '*waaris*' (heir) to '*waar*' (attack) after comparing with the later and after removing the suffix *is*.

4 Methods

4.1 Doc2Vec

Bag-of-words gained immense popularity in the field of text processing, though they have two weaknesses: they lose the ordering of the words and also ignore semantics of the words (Le and Mikolov, 2014). The document level word embeddings have been quite successful in several classification tasks, and it has the advantage over the word embeddings that it is trained to reconstruct linguistic contexts of words. Similarly, Doc2Vec is an extension of word embeddings that learns to correlate labels and words, rather than words with other words.[13] Doc2Vec has been successfully used for several NLP related tasks such as summarization (Pontes et al., 2016), sentiment analysis (Le and Mikolov, 2014) etc.

Initially, we trained all the lyrics using Gensim[14] library. Then, top 10 retrieved vectors for each of the query vectors have been collected for manual checking. We also trained Doc2Vec model on lyrics after stemming all words using the proposed stemming algorithm.

4.2 Self-Organizing Feature Maps

SOFMs are useful for clustering several tasks (Vesanto and Alhoniemi, 2000) and it has been successfully used for information retrieval (Ahuja and Goyal, 2012). SOFMs

[13] https://deeplearning4j.org/doc2vec
[14] https://radimrehurek.com/gensim/models/word2vec.html

are a class of artificial neural networks, which employ competitive learning (Kohonen, 1982). SOFMs cluster similar data without the help of training instances, and hence are said to perform unsupervised learning. The algorithm is started by initializing a set of randomly weighted neurons in the input feature space, and care is taken not to initialize two neurons with identical weights. SOFMs work in two phases, namely self-organizing phase and recall phase. In the self-organizing phase, each neuron's weight vector is matched with an input vector, and the best matching neuron and its neighborhood's weights are adapted to match the selected input. As this kind of learning progresses, input-vectors located far away from each other are mapped to distant neurons. Thus, a grouping of close-by input neurons is formed. In the recall phase, an input vector which is unknown to the SOFMs are matched with all the neurons, and the neighborhood which forms its closest match is associated with that new input vector (Kar et al., 2015).

In self-organizing phase, we considered feature vectors (N-grams) of songs and these were mapped to the neurons to form an SOFMs cluster. In recall phase, a query lyric feature vector was matched with the cluster neighborhoods created during the self-organizing phase. The nearest matching SOFM neighborhood was selected as the set of song ids corresponding to the query song. The detailed steps of SOFMs are given in algorithm 2.

N-gram feature: The n-gram feature plays an important role in information retrieval. The *Term Frequency-Inverse Document Frequency* (*tf-idf*) scores of up to trigrams were considered as feature because the sparsity of the feature vector increases significantly and the results get worse after including higher order n-grams.

4.3 Evaluation

Manual checking is a tedious and time-consuming task requiring human resource. There was no gold standard dataset available for comparing performances of the developed systems. Thus, manual evaluation was performed for calculating similarity between a query and retrieved lyrics. To keep the annotation process simple and reduce the manual checking load, we selected only top ten retrieved lyrics for each query lyric.

We asked the annotator, whether the song is similar to query lyrics or not. Second, whether they would like to listen to retrieved song after listening to the query-song. The annotators were asked to provide a score on a scale from 0 to 1 to each of the ten retrieved lyrics for a single query lyric based on above mentioned points. The retrieved lyric is considered to be matched with the query lyric if the similarity score provided by the annotator is more than 0.7; the threshold was selected experimentally. We wanted a trade off between the system performance and quality of the annotation. This value is selected to reduce the annotation disagreement as well as the subjectivity of the annotators. Two annotators checked each of the results. We also calculated the inter-annotator agreement and it was 87%. Finally, precision (P), recall (R) and F-measure (FM) are calculated based on the manual checking.

Algorithm 2 Pseudo code for SOFMs

1: **procedure** SELF-ORGANIZING PHASE
2: Initialize a neuron field of $k \times k$ dimension, each having $1 \times d$ dimensional weight vector (no two weight vectors would be the same).
3: Select winning weight vector having the least Euclidean distance ($d_{i,j} = \sqrt{\sum_{i=1}^{d} (x_{i,j} - w_{i,j})^2}$) to input vector
4: The winning neuron adapted using $w_{k,j} = w_{k,j} + \eta(x_{k,j} - w_{k,j})$
5: **for** each iteration **do** decrease learning rate η and neighborhood size till convergence
6: **procedure** RECALL PHASE
7: Map input data to nearest clusters centers (weight vectors).

5 Results and Discussion

We have selected a total of 100 query lyrics for testing system performances. For the test, we chose lyrics by searching the top lyrics on the web which are present in the collected dataset. The rest 31,070 lyrics from entire dataset were used for training the system. The systems and their performances with detailed analysis are described below.

5.1 Baseline System

A baseline system was developed for identifying similar lyrics using Fuzzy C-means (FCMs)

clustering algorithm on 34,571 unigrams, and it achieved F-measure of 0.42.

5.2 Doc2Vec based System

We trained Doc2Vec model using all data (i.e. 31,171 lyrics). There were two models, with and without the unsupervised stemming algorithm. We evaluated the Doc2Vec based system using same 100 query lyrics, and the systems achieved F-measures of 0.670 without using the stemming algorithm. Whereas F-measure increased to 0.692 after implementing the stemming algorithm, i.e. an improvement of 0.022 was observed after performing the stemming.

5.3 SOFMs based System

For the SOFMs based system, we changed the distance function from Euclidean to cosine similarity as : $Cosine\ \ Similarity\ (CS) = 1 - \frac{u \cdot v}{||u||^2||v||^2}$

The n-grams were collected after removing stopwords and duplicate characters as well as implementing the unsupervised stemming algorithm. The words having frequency one were also removed from the total 37,693 unigrams and the final unigrams dimension was 34,571. The main reason was that we observed an improvement in the F-measure of 0.008 after removing the unigrams with one frequency. The unigram based system yields F-measure of 0.671 using the Euclidean distance, and there was an improvement of 0.004 when cosine similarity was used for calculating the distance.

After adding bigrams to the above system, the feature dimension increased to 50,321. The bigrams having frequency one is also removed from the lists. The SOFMs based system obtained F-measure of 0.711 using Euclidean distance, and cosine similarity improved the F-measure by 0.007. We developed another system using n-grams up to three, and the feature dimension was 57,321. The similar lyrics retrieval system achieved F-measure of 0.737 using SOFMs with Euclidean distance. An improvement of 0.012 in F-measure was observed using cosine similarity. The detailed results were shown in Table 1.

The higher order n-grams were not included in the study due to computational complexity. The proposed stemming algorithm provides significant computational cost cutting. In fact, we believe that implementing SOFMs for this problem would not be useful if the stemming algorithm was not used. Finally, we removed the repeated lines of *mukhda* from lyrics and developed another system using only unigrams. We observed that the F-measure fell by 0.12 in comparison to system developed using all the words of *mukhda*. Thus, we have not performed any experiments further using this setting.

Algorithms	P	R	FM
FCMs$_U$(Baseline)	0.431	0.410	0.420
Doc2Vec$_{WTS}$	0.670	0.670	0.670
Doc2Vec$_{WS}$	0.691	0.693	0.692
SOFMs$_U$ (EU)	0.721	0.663	0.671
SOFMs$_U$ (CS)	0.721	0.667	0.674
SOFMs$_{UB}$ (EU)	0.727	0.696	0.711
SOFMs$_{UB}$ (CS)	0.732	0.704	0.718
SOFMs$_{UBT}$ (EU)	0.764	0.710	0.737
SOFMs$_{UBT}$ (CS)	0.779	0.718	**0.749**

Table 1: Performances of SOFMs and FCMs based systems.
EU: Euclidean Distance, **CS**: Cosine Similarity, **WTS**: Without Stemming, **WS**: With Stemming, **U**: Unigram, **UB**: Unigram + Bigram, **UBT**: Unigram + Bigram + Trigram

5.4 Discussion

The similar lyrics retrieval systems based on SOFMs and Doc2Vec performed well as compared to baseline system using FCMs. The Doc2Vec based system failed to perform as good as the system developed using SOFMs with n-grams up to three. The Doc2Vec requires a huge amount of training data to train itself and this may be one of the reasons for low performance of Doc2Vec system. Calculating the similarity in the general text is much easier than doing it in the lyrics due to the free word order nature and this may be another reason for poor performance of Doc2Vec based system. It can be stated that the SOFMs work well for clustering similar lyrics. By improving the accuracy of unsupervised stemming algorithm, performances of SOFMs based similar lyrics retrieval system can be increased. Again, adding more number of lyrics can significantly improve the accuracies of such systems.

We investigated the errors in SOFMs based system. We found that there were some mistakes due to spelling variations. For example, the song "*ab to hai tumase har kushii apanii*" does not match with a single song. There are many spelling mistakes in this lyrics such as "*kushii*" (it should

be "*khusii*"), "*apanii*", "*tumase*", "*mashahuur*", "*budanaam*" etc. After removing stopwords, only these words left for the test; thus no match is found with the training dataset. The use of similar words in a different sense makes it harder to identify the similar lyrics in the case of SOFMs based systems. In the case of Doc2Vec, we have not removed stopwords while training the Doc2Vec model; again this model observes the context information rather than only syntactic information (matching the exact word); thus the above lyric has fetched the results.

Searching the similar lyrics was also performed by Mahedero et al. (2005) for Western songs. They used cosine similarity to identify similar lyrics. Another task, identifying similar lyrics based on topics in Japanese songs was performed by Sasaki et al. (2014). They identified topics of lyrics using LDA, and the evaluation was performed using the perplexity. To the best of author's knowledge, no other comparable task has been performed in either in Hindi or Western music.

6 Conclusions and Future Work

We developed a Hindi lyric dataset and implemented several techniques to clean the unstructured data. An unsupervised stemming algorithm was proposed to reduce the number of n-grams. We hope these methods can be used in IR systems for cleaning several unstructured data. The SOFMs based similar lyrics retrieval system achieved the maximum F-measure of 0.749 calculated on 100 query lyrics. We believe that this research will facilitate the development of recommendation in Indian music specifically for Hindi songs.

There are several directions for future work. One of the most immediate tasks is to evaluate performance of the proposed unsupervised stemming algorithm. We used document level word embeddings though, word level embeddings and latent dirichlet allocation (LDA) can be used in future for developing lyrics retrieval systems.

In future, the inter-cluster cosine similarity can be used for automatic evaluation. A weighted score can be assigned to each portions of lyrics (starting, middle, and end) for evaluation.

The mood words from lyrics can be collected using unsupervised approach such as word embeddings and the derived mood information can be used for ranking the results of similar lyrics retrieval system. Ranking the retrieved lyrics is another important factor for recommendation system and is not considered during the evaluation of current system. It is one of the limitations of current developed system and ranking based evaluation can be implemented in future.

Acknowledgments

The work reported in this paper is supported by a grant from the "*Visvesvaraya Ph.D. Scheme for Electronics and IT*" funded by *Media Lab Asia* of *Ministry of Electronics and Information Technology* (*MeitY*), *Government of India*. The authors are also thankful to the anonymous reviewers for their helpful comments.

References

Harika Abburi, Eswar S. A. Akkireddy, Suryakanth Gangashetti, and Radhika Mamidi. 2016. Multimodal sentiment analysis of telugu songs. In *Proceedings of the 4th Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2016)*, pages 48–52.

Sudhir Ahuja and Rinkaj Goyal. 2012. Information retrieval in intelligent systems: Current scenario & issues. *arXiv preprint arXiv:1206.3667*.

Jayson Beaster-Jones and Natalie Sarrazin. 2016. *Music in Contemporary Indian Film: Memory, Voice, Identity*. Taylor & Francis.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, et al. 2010. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM.

Nancy Duncan and Mark Fox. 2005. Computer–aided music distribution: The future of selection, retrieval and transmission. *First Monday*, 10(4).

Yong Han, Li Min, Yu Zou, Zhongyuan Han, Song Li, Leilei Kong, Haoliang Qi, Wenhao Qiao, Shuo Cui, and Hong Deng. 2015. Lrc sousou: A lyrics retrieval system. In *Proceedings of the International Conference of Young Computer Scientists, Engineers and Educators*, pages 464–467. Springer.

Xiao Hu, J Stephen Downie, and Andreas F Ehmann. 2009. Lyric text mining in music mood classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 411–416.

Yajie Hu. 2014. *A Model-Based Music Recommendation System for Individual Users and Implicit User Groups*. Ph.D. thesis, University of Miami.

Reshma Kar, Amit Konar, Aruna Chakraborty, Basabdatta Sen Bhattacharya, and Atulya Nagar. 2015. Eeg source localization by memory network analysis of subjects engaged in perceiving emotions from facial expressions. In *International Joint Conference in Neural Networks (IJCNN-2015)*. IEEE.

Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. 2007. A music search engine built upon audio-based and web-based similarity measures. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 447–454. ACM.

Gopala K. Koduri, Marius Miron, Joan Serrà Julià, and Xavier Serra. 2011. Computational approaches for the understanding of melody in carnatic music. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 263–268.

Teuvo Kohonen. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Jose P.G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 475–478. ACM.

Rudolf Mayer, Robert Neumayer, and Andreas Rauber. 2008. Rhyme and style features for musical genre classification by song lyrics. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 337–342.

Cristian Moral, Angélica de Antonio, Ricardo Imbert, and Jaime Ramírez. 2014. A survey of stemming algorithms in information retrieval. *Information Research: An International Electronic Journal*, 19(1).

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2013. Unsupervised approach to hindi music mood classification. In *Proceedings of the Mining Intelligence and Knowledge Exploration*, pages 62–69. Springer International Publishing.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2015. Mood classification of hindi songs based on lyrics. In *Proceedings of the 12th International Conference on Natural Language Processing (ICON- 2015)*, pages 261–267.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016a. Labeling data and developing supervised framework for hindi music mood analysis. *Journal of Intelligent Information Systems*, 48(3):633–651.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016b. Multimodal mood classification - a case study of differences in hindi and western songs. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1980–1989.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016c. Multimodal mood classification framework for hindi songs. *Computación y Sistemas*, 20(3):515–526.

Elvys L. Pontes, Juan-Manuel Torres-Moreno, Stéphane Huet, and Andréa C. Linhares. 2016. Tweet contextualization using continuous space vectors: Automatic summarization of cultural documents. In *Proceedings of the CLEF (Working Notes)*.

Shoto Sasaki, Kazuyoshi Yoshii, Tomoyasu Nakano, Masataka Goto, and Shigeo Morishima. 2014. Lyricsradar: A lyrics retrieval system based on latent topics of lyrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 585–590.

Markus Schedl, Time Pohle, Peter Knees, and Gerhard Widmer. 2011. Exploring the music similarity space on the web. *ACM Transactions on Information Systems (TOIS)*, 29(3):14:1–14:24.

Rajeswari Sridhar, Manasa Subramanian, B. M. Lavanya, B. Malinidevi, and T. V. Geetha. 2011. Latent dirichlet allocation model for raga identification of carnatic music. *Journal of Computer Science*, 7(11):1711–1716.

Ajay Srinivasamurthy, André Holzapfel, and Xavier Serra. 2014. In search of automatic rhythm analysis methods for turkish and indian art music. *Journal of New Music Research*, 43(1):94–114.

Aniruddha M. Ujlambkar and Vahida Z. Attar. 2012. Mood classification of indian popular music. In *Proceedings of the CUBE International Information Technology Conference*, pages 278–283. ACM.

Juha Vesanto and Esa Alhoniemi. 2000. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600.

Chung-Che Wang, Jyh-Shing Roger Jang, and Wennen Wang. 2010. An improved query by singing/humming system using melody and lyrics information. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 45–50.

Menno Van Zaanen and Pieter Kanters. 2010. Automatic mood classification using tf* idf based on lyrics. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 75–80.