UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CIÊNCIA DA COMPUTAÇÃO

CLASSIFICAÇÃO MULTIRRÓTULO APLICADA A DADOS MUSICAIS

TRABALHO DE GRADUAÇÃO

Vitor da Silva

Santa Maria, RS, Brasil

2014

CLASSIFICAÇÃO MULTIRRÓTULO APLICADA A DADOS MUSICAIS

Vitor da Silva

Trabalho de Graduação apresentado ao curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientadora: Prof.ª Dr.ª Ana Trindade Winck

Trabalho de Graduação Nº 366
Santa Maria, RS, Brasil

2014

Universidade Federal de Santa Maria
Centro de Tecnologia
Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

CLASSIFICAÇÃO MULTIRRÓTULO APLICADA A DADOS MUSICAIS

elaborado por
Vitor da Silva

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Ana Trindade Winck, Dr.ª
(Presidente/Orientador)

Luís Álvaro de Lima Silva, Dr. (UFSM)

Juliana Kaizer Vizzotto, Drª. (UFSM)

Santa Maria, 20 de Janeiro de 2014.

”Si vis pacem, para bellum”

— PUBLIUS FLAVIUS VEGETIUS RENATUS

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

CLASSIFICAÇÃO MULTIRRÓTULO APLICADA A DADOS MUSICAIS

AUTOR: VITOR DA SILVA
ORIENTADORA: ANA TRINDADE WINCK
Local da Defesa e Data: Santa Maria, 20 de Janeiro de 2014.

Diversos estudos vem sendo realizados com a grande quantidade de dados musicais aos quais se tem acesso atualmente, desde a classificação de músicas em emoções até organização automática de bibliotecas de áudio. Para essas tarefas a classificação multirrótulo tem ganhado importância devido ao modo como representa os dados. Assim, esse trabalho tem o objetivo de apresentar esse tipo de classificação, suas métricas e realizar uma comparação entre diversos algoritmos de classificação multirrótulo existentes atualmente. Também é gerado um conjunto de dados de gêneros e subgêneros musicais visando a sua utilização nesse trabalho e a sua disponibilização para futuras pesquisas.

Palavras-chave: Classificação Multirrótulo. Aprendizado de Máquina. Mineração de Dados.

LISTAGENS DE CÓDIGO

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O rápido crescimento da internet e os avanços de suas tecnologias possibilitaram a geração e obtenção de grandes quantidades de dados musicais. Estes dados incluem sinais acústicos, letras de músicas e rótulos geográficos, rótulos de estilo e rótulos de humor, muitos destes criados por usuários de redes sociais voltadas à música. Este progresso melhorou a experiência de ouvir música, mas revelou o problema de como organizar estes dados e, de forma mais geral, como programas de computador podem auxiliar nesta prática (SHAO, 2011).

Nas ultimas décadas tem se tornando bastante clara a expansão rápida da quantidade de pesquisas sobre obtenção de informações musicais, visando sistemas automáticos para busca e organização de músicas e dados relacionados. Categorias comuns de busca e seleção, como artista e gênero único, tem resultados de fácil obtenção e já receberam bastante atenção em pesquisas, mas a classificação em categorias mais complexas, como detecção automática de humor em músicas, apesar de estar em estágios iniciais, tem interessado cada vez mais pesquisadores da área (KIM et al., 2010).

O aprendizado de máquina, campo de estudo da Inteligência Artificial, preocupa-se em criar modelos baseados em características previamente conhecidas capazes de realizar a predição de propriedades associadas a uma coleção de dados. Para isso são utilizadas técnicas como a implantação direta de conhecimento, aprendizado por instrução, por analogia, por exemplos ou por observação e descoberta (CARBONELL; MICHALSKI; MITCHELL, 1983). Tais técnicas podem ser usadas para a tarefa de classificação de dados, inclusive dados musicais, em categorias variadas, onde a classificação multirrótulo se destaca.

A classificação multirrótulo, foco principal desse trabalho, permite que os dados sejam diferenciados a partir de dois ou mais parâmetros distintos, técnica que também se mostra especialmente útil para a resolução de problemas presentes em diversas aplicações modernas, como classificação de funções de proteínas e classificação semântica de cenários (TSOUMAKAS; KATAKIS, 2007). Tomando como exemplo a classificação de músicas em gêneros e subgêneros, cada música pode se encaixar em mais de uma única classe, atribuindo-se à ela classificações como "blues", "country blues" e "blues rock" simultaneamente.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo gerar e disponibilizar ao público um conjunto de dados multirrótulo de gêneros e subgêneros musicais. Com este conjunto de dados é realizada uma comparação do comportamento de algoritmos de classificação multirrótulo quando aplicados à grandes quantidades de dados musicais. As informações obtidas através desse estudo podem vir a facilitar o uso desse tipo de classificação nas mais diversas aplicações, como categorização detalhada de bibliotecas musicais extensas e sistemas automáticos de recomendação.

1.1.2 Objetivos Específicos

Este trabalho visa especificamente:

1. Analisar os principais métodos de classificação multirrótulo existentes;
2. Construir um conjunto de dados musicais a partir de informações reais obtidas e armazenadas ao longo do tempo, contendo rótulos descritivos de gêneros e subgêneros para cada entrada;
3. Testar algoritmos de classificação multirrótulo utilizando esses dados;
4. Realizar uma análise dos resultados obtidos.

2 REVISÃO BIBLIOGRÁFICA

Música é definida como a ciência ou arte de ordenar tons ou sons em sucessão, em combinação e em relações temporais a fim de produzir uma composição dotada de unidade e continuidade (MERRIAM-WEBSTER, 2013). O entendimento do ser humano sobre música está conectado diretamente a aspectos emocionais, culturais e sociais. A combinação destes fatores produz uma organização pessoal das músicas, a qual é a base para a classificação humana de gêneros musicais. Esta pode ser dividida em dois grandes grupos, sendo eles as Taxonomias de Especialistas e *Folksonomias* de Música (SORDO et al., 2008).

Taxonomias de Especialistas são aquelas criadas principalmente com intuito comercial por gravadoras, lojas de discos e lojas virtuais. Seu objetivo é principalmente guiar o consumidor com facilidade até o produto desejado e são de natureza hierárquica (por exemplo, gênero, subgênero, artista e álbum). Já as *Folksonomias* de Música são decorrentes da popularização de conteúdo gerado por usuários na internet, onde as músicas deixam de ser classificadas hierarquicamente e passam a contar com sistemas de rótulos (*tags*) atribuídos a partir de critérios pessoais, os quais podem incluir inclusive rótulos relacionados à emoções e rótulos geográficos. Esta mudança para um sistema não hierárquico de classificação gera maior confiança por parte do usuário na hora de classificar suas músicas, mas dificulta a manutenção das Taxonomias de Especialistas já que de tempos em tempos podem surgir novos termos (SORDO et al., 2008).

O interesse por utilizar Aprendizado de Máquina na classificação de músicas despertou após o ano 2000 (LI; OGIHARA, 2003). A presença de Taxonomias Especialistas, como o MP3.com (CBS, 2013) e de *Folksonomias* de Música, como o Last.fm (LAST.FM, 2013), e o fácil acesso aos dados nelas contidos impulsionou os pesquisadores à esta área.

No domínio da Inteligência Artificial, aprendizado de máquina é o ramo que se preocupa em estudar e construir sistemas que tem a habilidade de aprender a partir de dados sem a necessidade de programá-los explicitamente. Segundo (MITCHELL, 1997), para uma definição mais formal do processo podemos dizer que um computador aprende a partir da experiência E em relação a uma classe de tarefas T e medida de performance P se a sua performance nas tarefas contidas em T, de acordo com a medida P, melhora com a experiência E.

O aprendizado de máquina pode ser dividido em duas categorias: Supervisionado e não supervisionado. Como (KHANAL et al., 2013) define, em problemas de aprendizado de máquina supervisionado, as possíveis classificações são predeterminadas. Os objetivos, chamados

de classes, podem ser visualizados como um conjunto finito previamente estabelecido por um humano. A tarefa do computador será a de procurar por padrões e construir um modelo matemático capaz de avaliar preditivamente variações nos dados, rotulando um segmento destes com as classificações disponíveis. Como, para esse trabalho, todas as classes (rótulos) dos dados são previamente conhecidas, pode-se dizer que a classificação multirrótulo é uma tarefa de aprendizado de máquina supervisionado. Problemas de aprendizado de máquina não supervisionado ocorrem quando as classes não são conhecidas, tendo como principal tarefa a descoberta de padrões que podem ser utilizados para a classificação dos dados (KHANAL et al., 2013).

2.1 Classificação Multirrótulo

Problemas de classificação tradicionais se preocupam em aprender a partir de um conjunto de instâncias que estão associadas a um único rótulo l de um conjunto de rótulos disjuntos L, onde a cardinalidade do conjunto é maior que 1.

$$l \in L, |L| > 1 \tag{2.1}$$

Se a cardinalidade do conjunto de rótulos for igual a 2 ($|L| = 2$), então o problema de aprendizado é chamado de classificação binária. Já se a cardinalidade for maior que dois ($|L| > 2$), então é um problema de classificação multiclasse (TSOUMAKAS; KATAKIS, 2007). Na classificação multirrótulo cada instância deixa de estar associada a um único rótulo $l \in L$, e passa a estar associada a um conjunto de rótulos $Y \subseteq L$.

Para uma descrição mais formal do processo de classificação multirrótulo, segundo (CARRILLO; LÓPEZ; MORENO, 2013), seja L um conjunto finito de rótulos e D o conjunto de instâncias.

$$L = \{\lambda_j : j = 1 \dots q\} \tag{2.2}$$

$$D = \{(x_i, Y_i) : i = 1 \dots m\} \tag{2.3}$$

onde x_i é o vetor de características de uma instância e $Y_i \subset L$ é o subconjunto de rótulos da instância i. O subconjunto Y_i é definido então como um vetor binário $Y_i = \{y_1, y_2, \dots, y_q\}$, onde cada $y_j = 1$ indica a presença de um rótulo λ_j no conjunto de rótulos relevantes para x_i. Usando esta convenção, o espaço dos resultados pode ser definido como $Y = \{0, 1\}^q$.

O objetivo da classificação multirrótulo é atribuir o conjunto de rótulos corretos para uma nova instância x, ou seja, predizer se um rótulo λ_j deve ou não ser atribuído à instância

x. Para isso, pode-se tanto transformar um problema de classificação multirrótulo em n problemas de classificação com um único rótulo, ou adaptar algoritmos existentes para trabalhar diretamente com múltiplos rótulos (TSOUMAKAS; KATAKIS; VLAHAVAS, 2010).

2.1.1 Métricas de Avaliação

A avaliação de métodos de aprendizado a partir de dados multirrótulo requerem métricas diferentes das utilizadas em dados de uma única classe (MAIMON; ROKACH, 2010). Para as definições das métricas a seguir é considerado um conjunto de dados com entradas $(x_i, Y, i), i = 1...m$, onde $Y_i \subset L$ é o conjunto de rótulos reais e $L = \lambda_j : j = 1..q$ é o conjunto de todos os rótulos. Dada uma entrada x_i, o conjunto de rótulos resultantes da predição de um classificador multirrótulo é denotado como Z_i.

- Perda de Hamming (*Hamming Loss*): A porcentagem de rótulos errados preditos em relação ao total de rótulos, definida como

$$\text{Hamming Loss} = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \triangle Z_i|}{M} \tag{2.4}$$

onde M é o número total de rótulos e $\triangle$ é a diferença simétrica de dois conjuntos, o que é, na teoria dos conjuntos, o equivalente ao operador XOR da lógica booleana(MAIMON; ROKACH, 2010).

- Acurácia (*Accuracy*): A proximidade dos rótulos preditos em relação aos rótulos reais, definida segundo (MAIMON; ROKACH, 2010) como

$$\text{Accuracy} = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{2.5}$$

- Erro-Um (*One-Error*), que avalia quantas vezes o rótulo mais bem colocado, no *ranking* dos rótulos, não está no conjunto de de rótulos relevantes da instância.

Seja $r_i(\lambda)$ o *ranking* predito para um rótulo λ pelo método de classificação, onde o rótulo mais relevante recebe o *ranking* 1 e o menos relevante recebe o *ranking* q. O Erro-Um é então definido como

$$\text{1-Error} = \frac{1}{m}\sum_{i=1}^{m}\delta(\arg\min r_i(\lambda)), \lambda \in L \tag{2.6}$$

onde

$$\delta(\lambda) = \begin{cases} 1 & \text{se } \lambda \notin Y_i \\ 0 & \text{caso contrário} \end{cases} \tag{2.7}$$

- Precisão (*Precision*): A quantidade de rótulos preditos que são reais, definida segundo (MAIMON; ROKACH, 2010) como

$$\text{Precision} = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Z_i|} \tag{2.8}$$

- Exaustividade ou Recordação (*Recall*): A quantidade de rótulos reais que são preditos, definida por (MAIMON; ROKACH, 2010) como

$$\text{Recall} = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i|} \tag{2.9}$$

- Perda Zero-Um (*Zero-One Loss*): Perda na exatidão das predições. Sua definição, segundo (MAIMON; ROKACH, 2010), é dada por:

Seja S é o conjunto de treino com atributos de entrada $A = a_1, a_2, ..., a_n$ e um atributo-alvo nominal y de uma distribuição fixa D sobre o espaço das instâncias rotuladas, o objetivo é induzir um classificador ótimo com o menor erro de generalização .

O erro de generalização é definido como a taxa de classificação incorreta sobre a distribuição D.

$$\varepsilon(I(S), D) = \sum_{\langle x,y\rangle \in U} D(x, y) \cdot L(y, I(S)(x)) \tag{2.10}$$

onde $L(y, I(S)(x))$ é a função de Perda Zero-Um definida por

$$L(y, I(S)(x)) = \begin{cases} 0 & \text{se } y = I(S)(x) \\ 1 & \text{se } y \neq I(S)(x) \end{cases} \tag{2.11}$$

No caso de atributos numéricos o operador de soma é substituído pelo operador de integração.

2.2 Trabalhos Relacionados

Baseando-se nos artigos publicados, como (LI; OGIHARA, 2003), (WIECZORKOWSKA; SYNAK; RAŚ, 2006), (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008), (PACHET; ROY, 2009), (KATAKIS; TSOUMAKAS; VLAHAVAS, 2008), (WANG et al., 2009) e (YING; DORAISAMY; ABDULLAH, 2012), pode-se afirmar que os pesquisadores demonstram uma curiosidade particular em detecção de emoções, frequentemente intercambiando os termos emoção

e humor, sendo uma das referências mais antigas apresentada em 2003 (LI; OGIHARA, 2003), a qual propõe a classificação multiclasse para abordar tal problema, onde foram utilizadas técnicas de processamento de sinais acústicos para analisar 499 trechos de músicas. Os dados obtidos são decompostos em múltiplos problemas de classificação binária e posteriormente resolvidos com o uso de *Support Vector Machines* para descrever cada trecho a partir de um grupo de dez adjetivos. Os resultados desse método foram apenas parcialmente satisfatórios, atingindo uma média aproximada de 44% de precisão. Considerando o gênero musical das amostras, a precisão subia, em média, 4,2% (LI; OGIHARA, 2003).

Posteriormente, em 2006, é apresentada a possibilidade de classificação multirrótulo para o mesmo problema de detecção de emoções em músicas (WIECZORKOWSKA; SYNAK; RAŚ, 2006). Desta vez, rótulos são manualmente inseridos para cada instância presente do conjunto de dados, a partir de um conjunto de 13 rótulos, sem limite na quantidade de rótulos para cada instância. A classificação multirrótulo se dá posteriormente, a partir de um grafo personalizado construído a partir de informações fornecidas pelo usuário. Assim, o sistema é capaz de lidar com a subjetividade de cada usuário, retornando para cada um resultados diferentes e aumentando a taxa de sucesso da classificação musical.

Um estudo realizado em 2008 compara quatro algoritmos de classificação multirrótulo aplicados ao problema de detecção de emoções, sendo eles *Binary Relevance* (BR), *Label Powerset* (LP), *Random k-Labelsets* (RAkEL) e *Multilabel k-Nearest Neighbor*(MLkNN), onde foi observado que o algoritmo RAkEL obteve o melhor resultado, classificando entre 75,8% e 90,3% dos rótulos corretamente (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008).

Estudos vem sendo conduzidos desde então considerando diversos aspectos, como a classificação multirrótulo em larga escala de títulos de músicas, incluindo centenas de milhares de entradas e mais de 600 rótulos por entrada (PACHET; ROY, 2009), sugestão automática de rótulos (KATAKIS; TSOUMAKAS; VLAHAVAS, 2008), classificação de estilos musicais utilizando hipergrafos (WANG et al., 2009) até a classificação multirrótulo de gêneros baseado em mineração de textos em letras de músicas (YING; DORAISAMY; ABDULLAH, 2012), todos com níveis de sucesso promissores.

3 METODOLOGIA

A sequência de atividades realizadas neste trabalho está listada a seguir.

1. Geração de um conjunto de dados musicais;
2. Estudo dos métodos e algoritmos de classificação multirrótulo;
3. Elaboração de uma estratégia de classificação multirrótulo aplicável a dados musicais;
4. Testes dos algoritmos com o conjunto de dados gerado;
5. Análise dos resultados.

3.1 Construção do Conjunto de Dados

No domínio musical, o conjunto de dados multirrótulo mais recorrente, disponível publicamente, é o *emotions* (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008), voltado para a classificação de músicas em relação à emoções. Segundo (SANDEN; ZHANG, 2011), não existe ainda um conjunto de dados voltado especificamente para a tarefa de classificação multirrótulo de gêneros musicais. Deste modo, criar um conjunto de dados multirrótulo para a classificação de músicas em gêneros se torna um passo essencial para o desenvolvimento deste trabalho. As etapas envolvidas na criação deste são descritas em detalhe abaixo.

1. Seleção dos arquivos de áudio;
2. Extração de características dos sinais acústicos;
3. Obtenção e seleção dos rótulos relacionados a gêneros musicais;
4. União dos dados e pré-processamento inicial.

Para a etapa 1, 1360 músicas em formato MP3 (MPEG-1/2 Audio Layer III) de um total de 104 artistas distintos foram selecionadas. Estas músicas foram escolhidas por terem em comum o rótulo *rock*, possibilitando assim a existência de uma maior quantidade de gêneros relacionados, como por exemplo, *metal*, *melodic metal* e *melodic power metal*, segundo os rótulos de usuários. A obtenção destes rótulos é explicada mais adiante.

De modo a utilizar os arquivos de áudio em uma tarefa de classificação qualquer, seja de rótulo único ou multirrótulo, faz-se necessária a extração de características descritivas do áudio. Para a geração do conjunto de dados, são utilizados os valores da Centroide Espectral, do Grau de Inclinação da Função de Transferência Espectral, do Fluxo Espectral, dos Cruzamentos de Zero no Domínio do Tempo e Coeficientes Cepstrais das Frequências de Mel. Estas características, descritas logo abaixo, são posteriormente associadas aos rótulos, permitindo que as diferenças e semelhanças pertinentes a cada gênero possam ser identificadas pelos algoritmos de classificação.

- Centroide Espectral (*Spectral Centroid*):

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] * n}{\sum_{n=1}^{N} M_t[n]} \tag{3.1}$$

A centroide espectral é definida como o centro de gravidade da magnitude do espectro da Transformada de Tempo Curto de Fourier (*Short-time Fourier Transform*), onde $M_t[n]$ é a magnitude da transformada de Fourier no período de tempo t e intervalo de frequência n. A centroide espectral é uma medida da forma do espectro do som, na qual valores altos correspondem a texturas com mais clareza e maior quantidade de altas frequências (TZANETAKIS; COOK, 2002).

- Grau de Inclinação da Função de Transferência Espectral (*Spectral Roll-Off*):

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 * \sum_{n=1}^{N} M_t[n] \tag{3.2}$$

O Grau de Inclinação da Função de Transferência Espectral é definido como a frequência sob a qual está concentrada 85% da distribuição de magnitude do sinal. Esta é outra medida da forma do espectro do som (TZANETAKIS; COOK, 2002).

- Fluxo Espectral (*Spectral Flux*):

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2 \tag{3.3}$$

O Fluxo Espectral é definido como o quadrado da diferença entre as magnitudes normalizadas de distribuições espectrais sucessivas onde $N_t[n]$ e $N_{t-1}[n]$ são as magnitudes normalizadas da transformada de Fourier nos intervalos de tempo t e $t-1$ respectivamente. O Fluxo Espectral é uma medida da variação local do espectro do som (TZANETAKIS; COOK, 2002).

- Cruzamentos de Zero no Domínio do Tempo (*Time Domain Zero Crossings*):

$$Z_t = \frac{1}{2}\sum_{n=1}^{N} |sinal(x[n]) - sinal(x[n-1])| \tag{3.4}$$

Um Cruzamento de Zero ocorre quando o sinal de áudio muda de positivo para negativo ou vice-versa em um determinado período de tempo. Na equação, tem-se que a função $sinal$ é 1 para argumentos positivos e 0 para argumentos negativos, $x[n]$ e $x[n-1]$ são os sinais de áudio de um intervalo de tempo t. Cruzamentos de Zero no Domínio do Tempo são uma medida da quantidade de ruído presente em um sinal (TZANETAKIS; COOK, 2002).

- Coeficientes Cepstrais das Frequências de Mel (*Mel-Frequency Cepstral Coefficients*):

$$C_n = \sqrt{\frac{2}{k}}\sum_{k=1}^{K}(\log S_k)\cos[\frac{n(k-0.5)\pi}{k}],\ n = 1, 2, \ldots, N \tag{3.5}$$

onde $S_k(k = 1, 2, \ldots, K)$ é o resultado de um banco de K filtros passa-faixa triangulares e N é o número total de amostras em uma unidade de áudio de t milissegundos (XU et al., 2005).

Na elaboração do conjunto de dados deste trabalho são extraídas todas as características citadas acima, notando-se que são utilizados 13 Coeficientes Cepstrais das Frequências de Mel distintos, bem como são calculadas as proporções de sons pertencentes a cada nota da escala cromática (Dó, Dó#, Ré, Ré#, Mi, Fá, Fá#, Sol, Sol#, Lá, Lá# e Si) para cada uma das amostras de áudio, somando um total total 29 características extraídas.

3.2 Obtenção dos Dados

A obtenção dos sinais de áudio pode ser executada com facilidade através do *framework* Marsyas (TZANETAKIS, 2013). O Marsyas disponibiliza uma ferramenta chamada "*bextract*", de simples configuração. Em Python, é necessário apenas garantir que o *bextract* possua as linhas de código mostradas na Listagem 3.1. O restante do código pode ser encontrado no repositório oficial do Marsyas (TZANETAKIS et al., 2013).

Listagem de Código 3.1: Extração de características com o *bextract*

```
...

net.updControl("TimbreFeatures/featExtractor/mrs_string/enableTDChild",
    marsyas.MarControlPtr.from_string("ZeroCrossings/zcrs"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableLPCChild",
    marsyas.MarControlPtr.from_string("Series/lspbranch"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableLPCChild",
    marsyas.MarControlPtr.from_string("Series/lpccbranch"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableSPChild",
```

```
      marsyas.MarControlPtr.from_string("MFCC/mfcc"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableSPChild",
      marsyas.MarControlPtr.from_string("SCF/scf"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableSPChild",
      marsyas.MarControlPtr.from_string("Rolloff/rf"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableSPChild",
      marsyas.MarControlPtr.from_string("Flux/flux"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableSPChild",
      marsyas.MarControlPtr.from_string("Centroid/cntrd"))
net.updControl("TimbreFeatures/featExtractor/mrs_string/enableSPChild",
      marsyas.MarControlPtr.from_string("Series/chromaPrSeries"))

...
```

O *bextract* recebe então como parâmetros de execução uma lista com os arquivos de áudio dos quais se deseja extrair as características e o tempo de cada música a ser processado, definido para este trabalho como 20 segundos. A fim de realizar uma melhor análise dos sinais de áudio, os primeiros 30 segundos de cada música são ignorados, evitando assim momentos de silêncio anteriores ao início do som e também eliminando possíveis introduções que não representem bem o gênero da música.

Como resultado da execução do *bextract* com estas configurações será gerado um arquivo em formato ARFF (*Attribute-Relation File Format*), contendo 58 atributos, sendo estes o valor médio e o desvio padrão de cada uma das 29 características extraídas.

Após esta etapa o conjunto de dados já apresenta um formato semelhante ao final, faltando adicionar cada rótulo relacionado a gênero como um novo atributo e atribuir, para cada entrada do conjunto de dados, o valor 1 caso o rótulo pertença àquela entrada, 0 caso contrário. No total, dos 1360 arquivos de áudio são criadas 1761172 entradas no conjunto de dados, cada uma contendo os respectivos valores para os 58 atributos citados acima. A média de amostras por arquivo de áudio é de aproximadamente 1295.

Em seguida, são obtidos os rótulos que serão associados às entradas do conjunto de dados. Para isso foi utilizada a biblioteca *pylast* (HASSAN, 2014), em conjunto com a linguagem Python, para o acesso às funcionalidades da *API* da rede social musical Last.fm (LAST.FM, 2013).

De cada artista entre os 104 disponíveis foram obtidas as 100 *tags* (rótulos) mais populares, que serão associados à cada uma de suas músicas. O processo é feito através do código especificado na Listagem 3.2. O código gera arquivos já em formato ARFF para cada um dos artistas, preenchendo com 1's a linha destinada aos dados, indicando que cada artista possui todos os rótulos a ele associados. Essa etapa facilita a fusão com o restante do conjunto de

dados.

Listagem de Código 3.2: Obtenção dos rótulos

```
import pylast

# E' necessario obter os proprios valores para API_KEY e API_SECRET
# Possivel em: http://www.last.fm/api/account
API_KEY = "********************************"
API_SECRET = "********************************"

username = "vitords"
password_hash = pylast.md5("*****")

network = pylast.LastFMNetwork(api_key = API_KEY, api_secret =
    API_SECRET, username = username, password_hash = password_hash)

# Pega o nome de cada artista de uma lista em 'artists.txt'
artist_file = open("artists.txt", "r")
artist_list = artist_file.readlines()
artist_list = [item.rstrip('\n') for item in artist_list]

for artist in artist_list:
    artist_object = network.get_artist(artist)
    # Procura por uma correcao do nome do artista
    artist_object.get_correction()
    # Cria um arquivo .arff para receber as tags de cada artista
    arff = open('data/' + artist + '_tags.arff', 'w')
    arff.write('@relation \"' + artist_object.get_name() +
       '_tags.arff\"\n')
    # Pega as 100 tags mais populares associadas ao artista
    top_tags = artist_object.get_top_tags()

    total_tags = 0
    for tag in top_tags:
        # Escreve as tags ja em um formato apropriado
        # 'a classificacao multirrotulo
        arff.write('@attribute \"' + tag.item.get_name() + '\" {0,1}\n')
        total_tags += total_tags

    arff.write('\n\n@data\n')
    for i in range(total_tags - 1):
        arff.write('1,')
    arff.write('1\n')

arff.close()
artist_file.close()
```

Em seguida é executada a união dos arquivos de rótulos individuais dos artistas. Essa etapa gera um arquivo contento o conjunto L de todos os rótulos disponíveis. Lê-se o conjunto de rótulos de cada artista e, ao realizar a união com os demais, adiciona-se o número zero (0) para cada rótulo que não esteja presente no conjunto de rótulos pertinentes ao artista, evitando assim que artistas sejam associados aos rótulos errados.

Após a filtragem das *tags*, sobram 474 rótulos que possuem relação com gêneros musicais. Destes, são removidos 229 por possuírem apenas uma banda representante, 71 por terem apenas duas bandas representantes e 10 por existirem rótulos duplicados, totalizando 164 rótulos distintos com informações sobre gêneros. A união de todos os arquivos ARFF remoção dos atributos é trivial com a utilização do *software* WEKA (HALL et al., 2009).

Como cada banda selecionada para o trabalho tem em média 13 músicas, rótulos representados por duas bandas ou mais estão presentes em pelo menos 26 músicas, resultando em cerca de 35360 amostras. Se cada amostra possui 20 milissegundos, um rótulo terá, assim, ao menos 70 segundos de áudio como exemplo.

O cabeçalho final, incluindo os 58 atributos relacionados à características do arquivo de áudio e os 164 atributos relacionados aos rótulos, bem como exemplos de entradas do conjunto de dados, encontra-se na Listagem A.1.

3.3 Classificação

Para a classificação, foram testados os *frameworks* scikit-learn (PEDREGOSA et al., 2011), Orange (DEMSAR et al., 2013), MULAN (TSOUMAKAS et al., 2011) e MEKA (READ; REUTEMANN, 2014).

Orange e scikit-learn possuem uma pequena quantidade de algoritmos de classificação multirrótulo implementados quando comparados com os *frameworks* MULAN e MEKA. A decisão para o uso do MEKA se deu pela extensa quantidade de algoritmos disponíveis, pela sua fácil utilização e por servir também como um *wrapper* para o MULAN, podendo-se utilizar as funcionalidades dos dois *frameworks* em conjunto.

Para a adequação do conjunto de dados ao MEKA, basta que no início do cabeçalho sejam indicadas quantas classes (rótulos) existem para a tarefa de classificação, como indicado na Listagem 3.3. O valor negativo antes do número de rótulos indica que estes estão após as características, no final da lista de atributos.

Listagem de Código 3.3: Adequação do cabeçalho para uso com o MEKA

```
@relation 'genres-multilabel: -C -164'
...
```

Os algoritmos disponíveis no MEKA utilizados para a comparação são:

- *Binary Relevance* (BR);

- *Binary Relevance - Random Subspace* (BRq);
- *Classifier Chains* (CC);
- *Label Powerset* (LP);
- *Majority Labelset*;
- *Multi-Label k-Nearest Neighbor* (MLkNN);
- *Random k-Labelsets* (RAkEL).

Estes sete algoritmos foram selecionados devido à constantes menções dos mesmos na bibliografia existente (como em (READ et al., 2011), (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008), (READ et al., 2010) e (MADJAROV et al., 2012)). Para todos foi utilizado o método de divisão do conjunto de dados em um conjunto de treino (60% dos dados) e um conjunto de testes (40% dos dados), sem validação cruzada devido à limitações de *hardware*.

Os algoritmos *Binary Relevance*, *Label Powerset* e *Majority Labelset* não necessitam de parâmetros. Para o BRq foi utilizado o fator de redução (*downsampling ratio*) igual a 0.75 e a semente aleatória (*random seed*) igual a zero. Para o algoritmo *Classifier Chains* foi utilizada a semente aleatória também igual a zero. O MLkNN foi executado com 10 vizinhos (*k-Nearest Neighbors = 10*) e com o fator de suavização igual a um. O RAkEL foi executado com semente aleatória igual a zero, número de modelos igual à três e o tamanho dos subconjuntos igual à metade da quantidade de rótulos do conjunto ($\frac{L}{2}$).

Para os algoritmos BR, BRq, CC, LP e RAkEL, que exigem um classificador base, foram utilizados os seguintes classificadores:

- ZeroR;
- C4.5;
- *Naive Bayes*.

Estes algoritmos base tem o trabalho de realizar a classificação dos dados após os algoritmos multirrótulo dividirem o problema em tarefas menores. Estas classificações são retornadas ao algoritmo multirrótulo, que faz a união dos resultados em um só, gerando um resultado efetivamente multirrótulo. O principal motivo da escolha desses algoritmos como classificadores base foi o de proporcionar um meio de avaliar o desempenho de classificadores probabilísticos

(*Naive Bayes*), de árvores de decisão (C4.5) e regras de associação (ZeroR), três áreas bastante comuns do aprendizado de máquina, como classificadores base para algoritmos multirrótulo,.

Todos os testes foram executados nas mesmas configurações de *hardware*, dispondo de 8 núcleos para o processamento e 16GB de memória RAM, 14 desses dedicados à Máquina Virtual Java, sobre a qual é desenvolvido e executado o *framework* MEKA.

Devido à limitações no hardware, o conjunto de dados criado para esse trabalho necessita passar por uma etapa de amostragem, onde a ordem das entradas é embaralhada aleatoriamente e 10% do total de entradas e reservado para os testes. Assim, cada algoritmo é executado sobre as mesmas 176117 instâncias, preservando a quantidade de rótulos.

4 RESULTADOS

Neste capítulo são apresentadas tabelas com métricas calculadas ao final da execução dos testes utilizando o *dataset* (conjunto de dados) cuja geração foi detalhada anteriormente, chamado aqui de *genres-ml*. Cada tabela representa os valores resultantes da execução dos algoritmos com um classificador base, quando necessário, sendo a tabela 4.1 referente ao classificador base ZeroR, a tabela 4.2 ao classificador base C4.5 e a tabela 4.3 ao *Naive Bayes*. Quando presente, o asterisco (*) ao lado do nome do algoritmo denota que o mesmo está na tabela para fins de comparação, já que não faz uso do classificador base indicado.

Tabela 4.1: Métricas com o classificador base ZeroR

genres-ml: Métricas com o classificador base ZeroR						
Algoritmo	H. Loss	Accuracy	0-1 Loss	1-Error	Precision	Recall
BR	0.15	0.348	1	**0**	0.509	0.527
BRq	0.134	0.328	1	0.618	0.589	0.422
CC	0.126	0.294	1	0.492	0.689	0.329
LP	0.19	0.237	0.976	0.925	0.373	0.356
Maj. Labelset*	0.19	0.237	0.976	0.925	0.373	0.356
MLkNN*	**0.067**	**0.685**	**0.64**	**0**	**0.78**	**0.782**
RAkEL	0.188	0.232	1	0.925	0.376	0.344

Tabela 4.2: Métricas com o classificador base C4.5

genres-ml: Métricas com o classificador base C4.5						
Algoritmo	H. Loss	Accuracy	0-1 Loss	1-Error	Precision	Recall
BR	0.105	0.499	0.994	0.48	0.659	0.659
BRq	0.114	0.488	0.997	0.553	0.611	0.7
CC	0.119	0.509	0.664	0.627	0.611	0.607
LP	0.101	0.593	0.541	0.516	0.671	0.673
Maj. Labelset*	0.19	0.237	0.976	0.925	0.373	0.356
MLkNN*	**0.067**	**0.685**	**0.64**	**0**	**0.78**	**0.782**
RAkEL	0.101	0.589	0.738	0.439	0.667	0.681

Pode-se notar pelos valores exibidas nas tabelas 4.1, 4.2 e 4.3 que o *Multi-Label k-Nearest Neighbor* (MLkNN) obteve os melhores resultados, apresentando um melhor desempenho em todas as métricas quando comparado aos outros seis algoritmos, independentemente do classificador base utilizado. Estes resultados contrastam com a pesquisa apresentada por (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008), onde o algoritmo RAkEL obteve os melho-

Tabela 4.3: Métricas com o classificador base Naive Bayes

genres-ml: Métricas com o classificador base Naive Bayes						
Algoritmo	H. Loss	Accuracy	0-1 Loss	1-Error	Precision	Recall
BR	0.253	0.094	1	0.081	0.175	0.175
BRq	0.462	0.17	1	0.966	0.186	0.6
CC	0.253	0.259	1	0.948	0.312	0.536
LP	0.167	0.286	0.937	0.895	0.447	0.389
Maj. Labelset*	0.19	0.237	0.976	0.925	0.373	0.356
MLkNN*	**0.067**	**0.685**	**0.64**	**0**	**0.78**	**0.782**
RAkEL	0.168	0.285	0.944	0.895	0.446	0.391

res valores nas métricas, porém assemelham-se bastante aos resultados obtidos por (TAWIAH; SHENG, 2013), onde o MLkNN também demonstrou o melhor desempenho.

Analisando os testes realizados nesses dois casos, (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008) utiliza uma *Support Vector Machine* como classificador base, enquanto (TAWIAH; SHENG, 2013) utiliza o algoritmo C4.5 e parâmetros de execução bastante semelhantes com o desse trabalho.

Outros resultados interessantes podem ser observados: Com o classificador base ZeroR (Tabela 4.1) os algoritmos *Label Powerset* e *Majority Labelset* obtiveram os mesmos resultados para todas as métricas, porém, como pode ser verificado na tabelas 4.7 e 4.8, o *Majority Labelset* é entre 52,11% e 61,13% mais rápido na execução da tarefa de classificação.

A troca do classificador base ZeroR para o C4.5 aumenta significativamente o desempenho de todos os algoritmos multirrótulo, porém faz com que o tempo de execução cresça exponencialmente. Tomando como exemplo o caso do algoritmo *Binary Relevance*, essa troca adiciona mais de oito horas ao processo.

Em geral, o BR se mostra o algoritmo de classificação multirrótulo com execução mais demorada, seguido pelo BRq, MLkNN, *Classifier Chains*, RAkEL, *Label Powerset* e *Majority Labelset*. Este último, apesar da velocidade de execução, mostrou-se o algoritmo com o pior desempenho nas métricas.

4.1 Comparação com Outro Conjunto de Dados

Sabendo que o conjunto de dados criado para este trabalho é bastante extenso, uma comparação das métricas e tempos de execução com um *dataset* menor é interessante. A fim de realizar esta comparação de desempenho, foi utilizado o conjunto de dados *emotions*, elaborado

por (TSOUMAKAS; KALLIRIS; VLAHAVAS, 2008) a partir de dados musicais. Entretanto, enquanto o conjunto de dados apresentado por este trabalho se preocupa com a classificação de gêneros musicais, este outro tem a finalidade de classificação multirrótulo de humor.

O *dataset* emotions possui 592 instâncias e 77 atributos, sendo 6 deles rótulos e 71 relacionados à características dos sinais de áudio. Esta pequena quantidade de rótulos e instâncias faz com que o tempo de execução dos algoritmos seja bastante menor, como observado nas tabelas 4.7 e 4.8.

Novamente o MLkNN apresenta o melhor desempenho, como é possível notar através das métricas apresentadas nas tabelas 4.4, 4.5 e 4.6. Diferentemente do conjunto de dados *genres-ml*, em algumas situações, outros algoritmos obtém um único valor melhor que o MLkNN, como e o caso da Perda Zero-Um com classificador base diferente do ZeroR.

Tabela 4.4: Testes no *dataset emotions* com o classificador base ZeroR

emotions: Métricas com o classificador base ZeroR						
Algoritmo	H. Loss	Accuracy	0-1 Loss	1-Error	Precision	Recall
BR	0.428	0.229	0.975	0.604	0.314	0.344
BRq	0.339	0.21	0.936	0.604	0.396	0.217
CC	0.304	0	1	0.683	0	0
LP	0.413	0.271	0.881	0.723	0.337	0.369
Maj. Labelset*	0.514	0.272	0.916	0.604	0.29	0.477
MLkNN*	**0.203**	**0.549**	**0.743**	**0.287**	**0.657**	**0.696**
RAkEL	0.385	0.277	1	0.604	0.379	0.415

Tabela 4.5: Testes no *dataset emotions* com o classificador base C4.5

emotions: Métricas com o classificador base C4.5						
Algoritmo	H. Loss	Accuracy	0-1 Loss	1-Error	Precision	Recall
BR	0.318	0.39	0.926	0.49	0.482	0.596
BRq	0.274	0.21	0.876	0.51	0.541	0.659
CC	0.273	0.442	0.817	0.485	0.551	0.561
LP	0.278	0.442	0.782	0.49	0.542	0.561
Maj. Labelset*	0.514	0.272	0.916	0.604	0.29	0.477
MLkNN*	**0.203**	**0.549**	0.743	**0.287**	**0.657**	**0.696**
RAkEL	0.219	0.532	**0.733**	0.302	0.631	0.672

Quando o classificador utilizado é o C4.5, o algoritmo RAkEL obtém uma Perda Zero-Um de 0,733, contra 0,743 do MLkNN, significando que, nessas condições, o RAkEL classifica 26,7% das instâncias com o conjunto exato de rótulos, enquanto o MLkNN atinge 25,7%. Já

com o classificador base *Naive Bayes*, o algoritmo *Label Powerset* obtém uma Perda 0-1 de 0,703 contra 0,743 do MLkNN. Com este mesmo classificador base o valor de Exaustividade do MLkNN é 0,696 contra 0,783 do algoritmo *Binary Relevance - Random Subspace* (BRq), significando que 78,3% dos rótulos preditos pelo BRq para as instâncias do conjunto de dados durante o teste são considerados relevantes.

Nota-se então que com um *dataset* significativamente menor os resultados tendem a variar mais, evento que podemos notar nas variações comentadas anteriormente, onde o MLkNN teve um desempenho um pouco abaixo de algoritmos como o RAkEL e o BRq. Também é possível observar que as métricas, quando calculadas em um conjunto de dados maior, apresentam melhores valores. Isso se da pelo fato de existirem mais exemplos para cada um dos rótulos, melhorando a eficiência dos algoritmos na tarefa de classificação.

Tabela 4.6: Testes no *dataset emotions* com o classificador base Naive Bayes

emotions: Métricas com o classificador base Naive Bayes						
Algoritmo	H. Loss	Accuracy	0-1 Loss	1-Error	Precision	Recall
BR	0.231	0.503	0.723	0.337	0.617	0.637
BRq	0.284	0.504	0.832	0.46	0.523	**0.783**
CC	0.269	0.513	0.797	0.485	0.542	0.759
LP	0.229	0.526	**0.703**	0.371	0.625	0.623
Maj. Labelset*	0.514	0.272	0.916	0.604	0.29	0.477
MLkNN*	**0.203**	**0.549**	0.743	**0.287**	**0.657**	0.696
RAkEL	0.251	0.485	0.772	0.332	0.584	0.615

Tabela 4.7: Comparação do tempo de execução dos algoritmos com classificador base

Tempo de execução dos testes com classificador base (em segundos)						
	genres-ml			*emotions*		
Algoritmo	ZeroR	C4.5	Naive Bayes	ZeroR	C4.5	Naive Bayes
BR	1789.481	32110.714	3069.427	0.082	0.773	0.187
BRq	419.275	21724.509	1339.239	0.05	0.489	0.089
CC	1024.853	7460.306	2510.181	0.021	0.569	0.135
LP	7.512	765.087	134.164	0.013	0.413	0.168
RAkEL	97.682	8003.25	1342.024	0.22	2.731	0.542

Tabela 4.8: Comparação do tempo de execução dos algoritmos sem classificador base

Tempo de execução dos testes sem classificador base (em segundos)				
	genres-ml		*emotions*	
Algoritmo	Mínimo	Máximo	Mínimo	Máximo
Maj. Labelset	3.915	4.592	0.001	0.004
MLkNN	10748.049	10945.153	0.273	0.656

5 CONSIDERAÇÕES FINAIS

A classificação multirrótulo ainda é bastante inexplorada por pesquisadores, mas vem se consolidando como uma ferramenta útil na área de aprendizado de máquina. Sua principal vantagem é a forma como representa informações do mundo real, possibilitando que a um mesmo objeto de interesse possam ser descritas diversas características distintas, as quais podem ser utilizadas simultaneamente para a resolução dos mais variados problemas.

Este trabalho mostra que os diferentes algoritmos de classificação multirrótulo produzem resultados consideravelmente distintos quando são utilizados diferentes classificadores base, e que alguns algoritmos produzem melhores resultados para grandes quantidade de dados, como o MLkNN. Um estudo mais aprofundado do impacto desses classificadores pode beneficiar essa área de pesquisa.

O algoritmo MLkNN se destaca na maior parte dos testes, mostrando-se uma boa escolha para se trabalhar com a tarefa de classificação de dados musicais, porém o seu tempo de execução pode inviabilizar o seu uso em algumas situações. Melhorias nesse ponto do algoritmo seriam de grande utilidade para o campo da classificação multirrótulo e, principalmente, para a aplicação desse algoritmo em aplicações voltadas ao usuário final.

Também é possível observar que não só no MLkNN, mas em todos os algoritmos de classificação multirrótulo testados, o tempo de execução em relação ao tamanho do conjunto de dados tem um crescimento exponencial. Um caso notável é o do algoritmo *Binary Relevance*, o qual aumenta seu tempo de execução em mais de 40 mil vezes quando o número de entradas do conjunto de dados é multiplicado por pouco menos de 300. Esse fator, somado à grande exigência de recursos de *hardware* para a realização das tarefas, pode ser proibitivo para aplicações reais.

Por fim, espera-se que o conjunto de dados criado para esse trabalho possa ser utilizado em pesquisas futuras e assim auxiliar no aprimoramento dos algoritmos de classificação multirrótulo existentes, na confecção de novos algoritmos e, principalmente, na utilização desse tipo de classificação para a resolução de problemas que se encaixam na descrição de uma tarefa de classificação multirrótulo.

5.1 Trabalhos Futuros

Para grandes quantidades de dados, a paralelização dos algoritmos de classificação multirrótulo pode ser uma alternativa viável, possibilitando assim o uso dos algoritmos em aplicações voltadas ao usuário final, como sistemas automatizados de recomendação ou categorização não supervisionada de textos.

Mudanças nos próprios algoritmos também podem ser realizadas, adaptando-os para problemas específicos, considerando o tipo de dado, a quantidade de rótulos e a quantidade instâncias do conjunto de dados com que se está trabalhando. Testes com uma maior quantidade de bases de dados, de diferentes domínios, podem identificar quais são os pontos de um algoritmo que necessitam de adaptações.

Finalmente, a elaboração de um *framework* para classificação multirrótulo escrito em linguagens de mais baixo nível, como C ou C++, pode ter um impacto positivo nessa área de pesquisa, tanto por ter melhor controle da utilização do hardware quanto por expandir a base de aplicações que tem acesso às funcionalidades, dada a maior quantidade de usuários destas linguagens, como mostrado em (TIOBE, 2014).

ANEXOS

ANEXO A – Formato do Conjunto de Dados

Listagem de Código A.1: Formato final do conjunto de dados

```
@relation 'genres-multilabel'

@attribute Mean_Mem20_ZeroCrossings_HopSize512_WinSize512
  _Sum_AudioCh0 numeric
@attribute Mean_Mem20_Centroid_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_Rolloff_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_Flux_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC0_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC1_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC2_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC3_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC4_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC5_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC6_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC7_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC8_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC9_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC10_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC11_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_MFCC12_Power_powerFFT_WinHamming
  _HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_A_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_A#_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_B_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_C_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_C#_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_D_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_D#_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_E_Power_powerFFT
  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Mean_Mem20_PeakRatio_Chroma_F_Power_powerFFT
```

```
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Mean_Mem20_PeakRatio_Chroma_F#_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Mean_Mem20_PeakRatio_Chroma_G_Power_powerFFT
58  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Mean_Mem20_PeakRatio_Chroma_G#_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_ZeroCrossings_HopSize512_WinSize512
    _Sum_AudioCh0 numeric
63 @attribute Std_Mem20_Centroid_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_Rolloff_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_Flux_Power_powerFFT_WinHamming
68  _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC0_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC1_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
73 @attribute Std_Mem20_MFCC2_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC3_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC4_Power_powerFFT_WinHamming
78  _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC5_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC6_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
83 @attribute Std_Mem20_MFCC7_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC8_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC9_Power_powerFFT_WinHamming
88  _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC10_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_MFCC11_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
93 @attribute Std_Mem20_MFCC12_Power_powerFFT_WinHamming
    _HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_A_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_A#_Power_powerFFT
98  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_B_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_C_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
103 @attribute Std_Mem20_PeakRatio_Chroma_C#_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_D_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_D#_Power_powerFFT
108  _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_E_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
  @attribute Std_Mem20_PeakRatio_Chroma_F_Power_powerFFT
```

```
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Std_Mem20_PeakRatio_Chroma_F#_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Std_Mem20_PeakRatio_Chroma_G_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute Std_Mem20_PeakRatio_Chroma_G#_Power_powerFFT
    _WinHamming_HopSize512_WinSize512_Sum_AudioCh0 numeric
@attribute 'gothic doom metal' {0,1}
@attribute 'power folk metal' {0,1}
@attribute 'swedish death metal' {0,1}
@attribute 'gothenburg metal' {0,1}
@attribute 'dark rock' {0,1}
@attribute 'progressive death metal' {0,1}
@attribute 'funk rock' {0,1}
@attribute 'american heavy metal' {0,1}
@attribute 'classic heavy metal' {0,1}
@attribute 'progressive power metal' {0,1}
@attribute 'german power metal' {0,1}
@attribute 'power speed metal' {0,1}
@attribute 'american trad rock' {0,1}
@attribute 'hair rock' {0,1}
@attribute 'german heavy metal' {0,1}
@attribute 'irish folk' {0,1}
@attribute 'pub rock' {0,1}
@attribute 'pagan folk metal' {0,1}
@attribute 'italian metal' {0,1}
@attribute 'folk death metal' {0,1}
@attribute 'symphonic folk metal' {0,1}
@attribute 'punk pop' {0,1}
@attribute 'boogie rock' {0,1}
@attribute 'british trad rock' {0,1}
@attribute 'classic hard rock' {0,1}
@attribute 'dutch metal' {0,1}
@attribute 'celtic metal' {0,1}
@attribute 'grunge rock' {0,1}
@attribute 'sludge metal' {0,1}
@attribute 'brutal death metal' {0,1}
@attribute reggae {0,1}
@attribute 'alternative pop-rock' {0,1}
@attribute 'country rock' {0,1}
@attribute folk {0,1}
@attribute 'rock gaucho' {0,1}
@attribute 'rock brasileiro' {0,1}
@attribute 'ska punk' {0,1}
@attribute 'viking folk metal' {0,1}
@attribute 'german rock' {0,1}
@attribute 'irish folk punk' {0,1}
@attribute 'irish punk rock' {0,1}
@attribute 'opera metal' {0,1}
@attribute 'gothic symphonic metal' {0,1}
@attribute 'british metal' {0,1}
@attribute 'epic power metal' {0,1}
@attribute 'groove metal' {0,1}
@attribute 'stoner metal' {0,1}
@attribute rap {0,1}
@attribute synthpop {0,1}
@attribute 'medieval metal' {0,1}
@attribute rockabilly {0,1}
```

```
@attribute ebm {0,1}
@attribute 'folk punk' {0,1}
@attribute 'irish punk' {0,1}
@attribute 'celtic punk' {0,1}
@attribute 'finnish metal' {0,1}
@attribute 'melodic rock' {0,1}
@attribute 'extreme metal' {0,1}
@attribute hip-hop {0,1}
@attribute 'traditional metal' {0,1}
@attribute 'rock nacional' {0,1}
@attribute 'brit rock' {0,1}
@attribute mpb {0,1}
@attribute 'college rock' {0,1}
@attribute 'symphonic black metal' {0,1}
@attribute 'celtic rock' {0,1}
@attribute 'irish rock' {0,1}
@attribute 'alternative pop' {0,1}
@attribute 'atmospheric metal' {0,1}
@attribute 'swedish metal' {0,1}
@attribute 'nordic metal' {0,1}
@attribute 'dark metal' {0,1}
@attribute 'melodic power metal' {0,1}
@attribute 'brazilian rock' {0,1}
@attribute 'pagan metal' {0,1}
@attribute 'battle metal' {0,1}
@attribute 'rock n roll' {0,1}
@attribute 'art rock' {0,1}
@attribute 'industrial rock' {0,1}
@attribute post-rock {0,1}
@attribute 'british rock' {0,1}
@attribute 'american metal' {0,1}
@attribute 'arena rock' {0,1}
@attribute 'southern rock' {0,1}
@attribute 'american rock' {0,1}
@attribute 'symphonic gothic metal' {0,1}
@attribute 'symphonic rock' {0,1}
@attribute dance {0,1}
@attribute trip-hop {0,1}
@attribute 'fantasy metal' {0,1}
@attribute 'pop punk' {0,1}
@attribute 'glam metal' {0,1}
@attribute country {0,1}
@attribute 'garage rock' {0,1}
@attribute 'rock and roll' {0,1}
@attribute 'acoustic rock' {0,1}
@attribute 'real metal' {0,1}
@attribute 'heavy rock' {0,1}
@attribute 'hair metal' {0,1}
@attribute 'industrial metal' {0,1}
@attribute 'scandinavian metal' {0,1}
@attribute 'german metal' {0,1}
@attribute 'album rock' {0,1}
@attribute 'melodic black metal' {0,1}
@attribute emo {0,1}
@attribute 'power pop' {0,1}
@attribute 'pop metal' {0,1}
@attribute 'glam rock' {0,1}
@attribute darkwave {0,1}
```

```
@attribute 'true metal' {0,1}
@attribute post-grunge {0,1}
@attribute ska {0,1}
@attribute 'symphonic power metal' {0,1}
@attribute 'modern rock' {0,1}
@attribute 'viking metal' {0,1}
@attribute 'psychedelic rock' {0,1}
@attribute metalcore {0,1}
@attribute 'alt rock' {0,1}
@attribute post-punk {0,1}
@attribute 'indie pop' {0,1}
@attribute 'classic metal' {0,1}
@attribute 'nu metal' {0,1}
@attribute 'blues rock' {0,1}
@attribute 'gothic rock' {0,1}
@attribute 'stoner rock' {0,1}
@attribute industrial {0,1}
@attribute 'epic metal' {0,1}
@attribute 'folk rock' {0,1}
@attribute 'soft rock' {0,1}
@attribute 'folk metal' {0,1}
@attribute 'speed metal' {0,1}
@attribute britpop {0,1}
@attribute soul {0,1}
@attribute 'alternative metal' {0,1}
@attribute 'doom metal' {0,1}
@attribute 'new wave' {0,1}
@attribute funk {0,1}
@attribute blues {0,1}
@attribute 'melodic death metal' {0,1}
@attribute 'melodic metal' {0,1}
@attribute electronic {0,1}
@attribute jazz {0,1}
@attribute 'gothic metal' {0,1}
@attribute 'black metal' {0,1}
@attribute 'symphonic metal' {0,1}
@attribute 'pop rock' {0,1}
@attribute grunge {0,1}
@attribute 'thrash metal' {0,1}
@attribute hardcore {0,1}
@attribute 'progressive metal' {0,1}
@attribute 'power metal' {0,1}
@attribute 'death metal' {0,1}
@attribute 'indie rock' {0,1}
@attribute 'punk rock' {0,1}
@attribute pop {0,1}
@attribute indie {0,1}
@attribute 'heavy metal' {0,1}
@attribute punk {0,1}
@attribute 'progressive rock' {0,1}
@attribute 'classic rock' {0,1}
@attribute 'alternative rock' {0,1}
@attribute 'hard rock' {0,1}
@attribute metal {0,1}
@attribute rock {0,1}

@data
...
```

```
  0.072266,0.061495,0.131712,0.098371,-39.946773,3.193712,-1.377289,
    1.740213,-0.632698,0.251692,-0.868285,-0.208399,-0.122129,0.866163,
288   0.077341,0.041194,-0.251412,0.000644,0.000637,0.000684,0.000697,
    0.00079,0.000934,0.001045,0.001053,0.000956,0.001077,0.00126,0.000919,
    0.010054,0.007254,0.010092,0.064792,0.445012,0.375154,0.433374,
    0.536164,0.528963,0.565918,0.625175,0.717758,0.471846,0.463688,
    0.348481,0.341128,0.413616,0.000224,0.000255,0.000252,0.000254,
293   0.000251,0.000222,0.000249,0.000247,0.000233,0.000267,0.000496,
    0.00026,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
    0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
    0,1,1,0,0,0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,
298   0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,1,0,1,1,1,0,0,0,0,1,0,1,1,0,
    1,1,1
  0.03418,0.010515,0.857782,0.088443,-52.079057,2.531097,1.365156,1.685998,
    1.373958,1.173761,0.505331,0.567892,0.67679,0.357963,0.646324,
    0.366753,0.459318,0.00132,0.001173,0.001114,0.001276,0.001435,
303   0.001555,0.00165,0.001707,0.001699,0.001665,0.00157,0.001448,
    0.015373,0.007163,0.339083,0.047643,4.70981,0.639164,0.951529,
    0.555038,0.589676,0.457344,0.518579,0.723201,0.671292,0.56721,
    0.602106,0.557932,0.510414,0.002509,0.002223,0.002104,0.002405,
    0.002703,0.002938,0.003123,0.003231,0.003225,0.003164,0.002991,
308   0.002761,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,
    0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,
    0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,1,1,1,0,0,1,
313   0,1,0,1,1,0,1,1,1
  0.080762,0.047017,0.103891,0.122163,-43.223026,3.414603,-0.140636, (...)
  0.122266,0.067051,0.178794,0.084796,-43.877965,3.533039,0.960345, (...)
  0.011328,0.006226,0.603113,0.093321,-64.353193,6.685266,0.331146, (...)
  0.091699,0.071482,0.131907,0.082599,-38.152759,3.074381,-1.06654, (...)
318 ...
```

REFERÊNCIAS

CARBONELL, J. G.; MICHALSKI, R. S.; MITCHELL, T. M. An overview of machine learning. In: **Machine learning**. [S.l.]: Springer, 1983. p.3–23.

CARRILLO, D.; LÓPEZ, V. F.; MORENO, M. N. Multi-label Classification for Recommender Systems. In: **Trends in Practical Applications of Agents and Multiagent Systems**. [S.l.]: Springer, 2013. p.181–188.

CBS. **MP3.com**. Disponível em <http://www.mp3.com/>, Acesso em: 20 out. 2013.

DEMSAR, J. et al. Orange: data mining toolbox in python. **Journal of Machine Learning Research**, [S.l.], v.14, p.2349–2353, 2013.

HALL, M. et al. The WEKA data mining software: an update. **ACM SIGKDD Explorations Newsletter**, [S.l.], v.11, n.1, p.10–18, 2009.

HASSAN, A. **pylast**: a python interface to last.fm (and other api-compatible websites). Disponível em <http://code.google.com/p/pylast/>, Acesso em: 07 jan. 2014.

KATAKIS, I.; TSOUMAKAS, G.; VLAHAVAS, I. Multilabel text classification for automated tag suggestion. In: EUROPEAN CONFERENCE ON MACHINE LEARNING AND PRINCIPLES AND PRACTICE OF KNOWLEDGE DISCOVERY IN DATABASES (ECML/PKDD). **Proceedings...** [S.l.: s.n.], 2008.

KHANAL, N. et al. **Analytics**: supervised vs. unsupervised learning. Disponível em <https://monk.library.illinois.edu/cic/public/analytics/clusterclassification.html>, Acesso em: 05 nov. 2013.

KIM, Y. E. et al. Music emotion recognition: a state of the art review. In: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL. **Proceedings...** [S.l.: s.n.], 2010. p.255–266.

LAST.FM. **Last.fm**. Disponível em <http://www.last.fm/>, Acesso em: 03 set. 2013.

LI, T.; OGIHARA, M. Detecting emotion in music. In: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL (ISMIR). **Proceedings...** [S.l.: s.n.], 2003. v.3, p.239–240.

MADJAROV, G. et al. An extensive experimental comparison of methods for multi-label learning. **Pattern Recognition**, [S.l.], v.45, n.9, p.3084–3104, 2012.

MAIMON, O.; ROKACH, L. Data Mining and Knowledge Discovery Handbook. , [S.l.], 2010.

MERRIAM-WEBSTER. **"music"**. Disponível em <http://www.merriam-webster.com/dictionary/music>, Acesso em: 20 out. 2013.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-Hill, 1997.

PACHET, F.; ROY, P. Improving multilabel analysis of music titles: a large-scale validation of the correction approach. **Audio, Speech, and Language Processing, IEEE Transactions on**, [S.l.], v.17, n.2, p.335–343, 2009.

PEDREGOSA, F. et al. Scikit-learn: machine learning in python. **The Journal of Machine Learning Research**, [S.l.], v.12, p.2825–2830, 2011.

READ, J. et al. Efficient multi-label classification for evolving data streams. , [S.l.], 2010.

READ, J. et al. Classifier chains for multi-label classification. **Machine Learning**, [S.l.], v.85, n.3, p.333–359, 2011.

READ, J.; REUTEMANN, P. **MEKA**: multi-label extension of weka). Disponível em <http://meka.sourceforge.net/>, Acesso em: 07 jan. 2014.

SANDEN, C.; ZHANG, J. Z. Algorithmic Multi-Genre Classification of Music: an empirical study. In: INTERNATIONAL COMPUTER MUSIC CONFERENCE. **Proceedings...** [S.l.: s.n.], 2011.

SHAO, B. User-centric music information retrieval. , [S.l.], 2011.

SORDO, M. et al. The quest for musical genres: do the experts and the wisdom of crowds agree? In: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL (ISMIR). **Proceedings...** [S.l.: s.n.], 2008. p.255–260.

TAWIAH, C. A.; SHENG, V. S. Empirical Comparison of Multi-Label Classification Algorithms. In: AAAI CONFERENCE ON ARTICIFIAL INTELLIGENCE, 27. **Proceedings...** [S.l.: s.n.], 2013. p.1645–1646.

TIOBE. **TIOBE Index for January 2014)**. Disponível em < http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>, Acesso em: 15 jan. 2014.

TSOUMAKAS, G. et al. MULAN: a java library for multi-label learning. **Journal of Machine Learning Research**, [S.l.], v.12, n.7, p.2411–2414, 2011.

TSOUMAKAS, G.; KATAKIS, I. Multi-label classification: an overview. **International Journal of Data Warehousing and Mining (IJDWM)**, [S.l.], v.3, n.3, p.1–13, 2007.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Mining multi-label data. In: **Data mining and knowledge discovery handbook**. [S.l.]: Springer, 2010. p.667–685.

TSOUMAKAS, K. T. G.; KALLIRIS, G.; VLAHAVAS, I. Multi-label classification of music into emotions. In: ISMIR 2008: PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE OF MUSIC INFORMATION RETRIEVAL. **Anais...** [S.l.: s.n.], 2008. p.325.

TZANETAKIS, G. **Marsyas**: music analysis, retrieval and synthesis for audio signals. Disponível em <http://marsyas.info/>, Acesso em: 05 nov. 2013.

TZANETAKIS, G.; COOK, P. Musical genre classification of audio signals. **Speech and Audio Processing, IEEE Transactions on**, [S.l.], v.10, n.5, p.293–302, 2002.

TZANETAKIS, G. et al. **Marsyas Official Repository**. Disponível em <https://github.com/marsyas/marsyas/>, Acesso em: 06 jan. 2014.

WANG, F. et al. Tag Integrated Multi-Label Music Style Classification with Hypergraph. In: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL (ISMIR). **Proceedings...** [S.l.: s.n.], 2009. p.363–368.

WIECZORKOWSKA, A.; SYNAK, P.; RAŚ, Z. W. Multi-label classification of emotions in music. In: **Intelligent Information Processing and Web Mining**. [S.l.]: Springer, 2006. p.307–315.

XU, M. et al. HMM-based audio keyword generation. In: **Advances in Multimedia Information Processing-PCM 2004**. [S.l.]: Springer, 2005. p.566–574.

YING, T. C.; DORAISAMY, S.; ABDULLAH, L. N. Genre and mood classification using lyric features. In: INTERNATIONAL CONFERENCE ON INFORMATION RETRIEVAL & KNOWLEDGE MANAGEMENT (CAMP), 2012. **Anais...** [S.l.: s.n.], 2012. p.260–263.