UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

ONDE É QUE EU JÁ OUVI ISTO?

Eduardo José Ribeiro Duarte

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

ONDE É QUE EU JÁ OUVI ISTO?

Eduardo José Ribeiro Duarte

DISSERTAÇÃO

Trabalho orientado pelo Prof. Doutor Thibault Nicolas Langlois

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

2012

Agradecimentos

Começo por agradecer ao professor e orientador Thibault Nicolas Langlois pela dedicação e disponibilidade ao longo desta tese para que conseguisse atingir os meus objectivos.

Agradeço ainda à Faculdade de Ciências da Universidade de Lisboa, à Fundação para a Ciência e Tecnologia, e em particular ao grupo do projecto VIRUS, LaSIGE e DI, pelas condições disponibilizadas para a realização deste trabalho.

Não menos importante, agradeço aos meus pais, irmão e avós, que me mostraram a maior riqueza de uma família: paz e harmonia, à minha namorada (обичам те Манка!) e aos amigos que sempre me apoiaram nos bons e maus momentos, um muito OBRIGADO a todos!

Para os meus pais

Resumo

Nos últimos anos, avanços tecnológicos a nível de compressão de áudio e redes de computadores tem solicitado um aumento gigante na disponibilidade e partilha de música digital.

O objectivo fundamental deste projecto é desenvolver um protótipo, pelo qual a semelhança entre várias peças de áudio possa ser medida, exclusivamente, no conteúdo do áudio em si, isto é, a partir das suas propriedades e características mais básicas.

Este protótipo irá analisar as características inerentes de cada peça de áudio e usar os dados provenientes dessa análise para comparar músicas, independentemente de qualquer metadata que possa existir. A base para essa comparação consiste numa impressão digital do áudio em si, que tem como objectivo gerar uma assinatura que identifica um pedaço de áudio. Esta assinatura, transforma o sinal de áudio numa sequência de vectores sendo esta sequência de vectores, um conjunto de características espectrais, representadas como: Zero-Crossings, Spectral Centroid, Rolloff, Flux e Mel-Frequency Cepstral Coeficientes (MFCC) do sinal de áudio. Mais especificamente, o sinal de áudio é convertido numa sequência de símbolos, que correspondem às características de uma peça de áudio. Esta "impressão digital" do áudio, não só identifica uma peça musical, mas também fornece informações sobre suas características musicais.

Usando este protótipo, será possível uma selecção de filmes com base na semelhança entre as peças de áudio, ou seja, será possível exibir ao usuário uma série de filmes, que possuam sequências de áudio semelhante a um tipo de áudio escolhido pelo mesmo permitindo, por isso, pesquisar numa base de documentos de vídeo através, apenas, de peças de áudio.

O trabalho insere-se numa das tarefas do projecto VIRUS (Video Information Retrieval Using Subtitles), financiado pela FCT, para a qual as técnicas foram, grande parte, já desenvolvidas.

Palavras-chave: filme, áudio, características espectrais, similaridade, pesquisa

Abstract

Over de last ten years, technological advances at the level of compression of audio and computer networks has prompted a huge increase in the availability and sharing of digital music.

The main purpose of this project is to develop a prototype, for which the similarity between various pieces of audio can be measured, exclusively on the audio content itself, that is, from their most basic properties and characteristics.

This prototype will analyze the inherent characteristics of each piece of audio and use the data from this analysis to compare music regardless, of any metadata that may exist. The basis for this comparison is a fingerprint of the audio itself, which aims to generate a signature that identifies the piece of audio. This signature, transform the audio signal is a sequence of vectors witch is, a set of spectral *features*, represented as: Zero-Crossings, Spectral Centroid, Rolloff, Flux and Mel-Frequency Cepstral Coefficients (MFCC) audio signal. More specifically, the audio signal is converted into a sequence of symbols that correspond to the characteristics of a piece of audio. This "fingerprint" of the audio, not only identifies a piece of music, but also provides information on its musical characteristics.

Using this prototype, it’s possible to select a movie based on the similarity between pieces of audio specified by the user, or the user can a series of films that have audio similar to a type of audio selected by de user. Through this prototype is also possible to search in a database of video, by specifying only pieces of audio.

The work is part of a project's tasks VIRUS (Video Information Retrieval Using Subtitles), funded by FCT, for which the techniques were largely already developed.

Keywords: film, audio, spectral characteristics, similarity, search

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este primeiro capítulo da tese destina-se a fornecer uma definição clara do tema escolhido para este projecto. Em primeiro lugar, é descrita a motivação desta tese, onde basicamente é explicado o contexto onde se insere. Neste ponto, é exposto detalhadamente as metas e motivações do projecto descrevendo a forma como, estes, surgiram. Em segundo lugar, são explicadas as contribuições do trabalho. Neste ponto, é explicado quais as utilidades do protótipo, ou seja, o que é possível alcançar como o mesmo. Em terceiro, e último lugar, é descrita a estrutura da tese, onde será dada uma visão geral do que se irá discutir em cada capítulo desta tese.

1.1 Motivação

Os filmes são, por excelência, a forma de arte que explora a nossa actividade, afectiva e intelectual combinando diversos sistemas como texto, imagem e som. Em 1895, os irmãos Lumière levaram ao público o novo evento (cinematógrafo), no Grand Café, em Paris, e a partir daí, o cinema deu mostras do seu fascínio sendo capaz de retractar a realidade em movimento e de possibilitar a construção de um imaginário necessário à sociedade.

Nos últimos anos, bastante trabalho tem sido desenvolvido na área de segmentação de vídeo [1], [2] como um requisito básico para enfrentar a resolução de problemas como indexação e recuperação por conteúdo. Esta segmentação implica, num estado inicial, que o vídeo já esteja particionado em segmentos de imagens, sendo estes últimos sequências de *frames* obtidos através de quebras de imagem, tipicamente associados a mudanças de câmara.

No entanto, a utilização do áudio como uma fonte de informação para detecção de cenas em concreto tem tido cada vez mais relevância pois colmata algumas lacunas na detecção de cenas através de imagem. Considerando por exemplo duas pessoas a falarem: mesmo que existam quebras de imagem (mudança de câmara), a cena continua, na mesma, a

ser duas pessoas a falarem e não cenas diferentes devido a possíveis mudanças de câmara. À semelhança do vídeo também, nesta área, se tem verificado um grande avanço tecnológico. Algoritmos eficientes para compressão do áudio alteraram radicalmente a escala de armazenamento do áudio digital, possibilitando o armazenamento de músicas, no computador ou noutro dispositivo de armazenamento, o que proporciona, às pessoas, uma escolha selecção muito maior.

Algoritmos como o MP3 (MPEG 1 Audio Layer 3) - tipo de compressão de áudio com perdas quase imperceptíveis ao ouvido humano [3], teve um surto gigante de popularidade nos últimos anos possibilitando, assim, às pessoas aceder mais facilmente à música e armazenar uma quantidade maior da mesma. No entanto, questões como direitos de autor e protecção têm sido cada vez mais faladas e discutidas [4].

Por outro lado, a Internet influenciou, também, a maneira pela qual a música digital é obtida [5], [6]. Nos últimos anos, mais concretamente na última década, as restrições de largura de banda impediram que a distribuição de música digital prolifera-se rapidamente através da Internet [7]. No entanto, devido aos avanços na tecnologia a nível de comunicações e redes, como o aumento significativo de largura de banda, estas restrições têm vindo a desaparecer gradualmente, possibilitando a transferência de arquivos de forma rápida e confiável por toda a parte do mundo, o que teve um grande impacto sobre a forma em que a música é distribuída.

1.2 Contribuições do Trabalho

Neste trabalho, as contribuições baseiam-se, além de um estudo de outros trabalhos em áreas relacionadas, num desenvolvimento de um protótipo que permita novas inovações podendo ser, também, utilizado em diferentes sistemas. Ao oferecer um processo para a medir a semelhança entre várias peças de áudio, e não apenas música, permite que possíveis aplicações ou programas, usando o protótipo, dando um episódio de uma série ou um filme consigam detectar cenas baseadas num determinado tipo de som (cão a ladrar, tiro de pistola, vidros a partir, objectos a arder, etc..) ou se as peças de áudio forem do tipo musical, a detecção de músicas semelhantes.

Para além das contribuições descritas a cima, colaborei ainda na escrita dum artigo, juntamente com os outros membros do projecto VIRUS, intitulado: *Going Through the Clouds: Search Overviews and Browsing of Movie*, que foi publicado na conferência internacional Academic MindTrek 2012 Conference.

1.3 Estrutura do documento

Esta tese está dividida em 6 capítulos; de seguida são descritos os cinco capítulos seguintes:

- Descrição do projecto – Neste segundo capítulo, é apresentada em pormenor os objectivos que deverão ser atingidos após o término da tese, bem como a metodologia aplicada à realização da mesma. Por último, é definido o planeamento da tese onde será descrito com rigor o enquadramento temporal do trabalho realizado.

- Trabalho Relacionado – No terceiro capítulo, é apresentado o estado da arte sobre as temáticas de extracção e cálculo de semelhança de áudio, bem como todo o trabalho relacionado, incluindo os trabalhos directamente relacionados com o projecto e outros trabalhos que forneceram informações importantes para a realização do protótipo.

- Trabalho Realizado – No quarto capítulo, é descrito os detalhes do desenvolvimento dos diferentes aspectos do protótipo e discute metodologias e alternativas à construção do mesmo.

- Testes e Resultados – No quinto capítulo, são realizados vários testes utilizando o protótipo, com diferentes tipos de peças de áudio como musicas ou series, onde os resultados dos mesmos são devidamente expostos e avaliados.

- Conclusões – No sexto, e último capítulo, é sumarizado o trabalho realizado sendo discutidas, também, possibilidades para o melhoramento do protótipo no futuro.

1.4 Equadramento Institucional

Este projecto foi realizado no contexto do Projecto em Engenharia Informática, para a obtenção do Mestrado em Engenharia Informática com especialização em Engenharia de Software, na Faculdade de Ciências da Universidade de Lisboa, num dos grupos de investigação do Departamento de Informática, *LASIGE - Large-Scale Informatics Systems Laboratory*. Mais concretamente, esta tese enquadra-se no projecto *VIRUS - Video Information Retrieval Using Subtitles*, financiado pela FCT, onde outros projectos estão a ser feitos, também na área de multimédia, mais concretamente, a nível de legendas e interfaces.

Capítulo 2

Descrição do Projecto

Este segundo capítulo da tese, tem como intenção a definição clara do tópico ou assunto escolhido para este projecto. Em primeiro lugar, são rigorosamente apresentados os objectivos da tese designando as principais etapas envolvidas na sua execução. Em segundo lugar, a metodologia utilizada no seu desenvolvimento é exposta sendo explicado o porquê da escolha efectuada no determinado modelo e não noutro. Em terceiro e último lugar, é apresentado o planeamento para a concretização dos objectivos.

2.1 Objectivos

O objectivo fundamental deste protótipo é desenvolver um sistema pelo qual, a semelhança do áudio, por exemplo de um filme ou uma peça musical, possa ser medido exclusivamente nas características do áudio em si. Para tal, a construção deste protótipo, foi baseada em três objectivos. O primeiro consiste no desenvolvimento de um mecanismo para atribuição de assinaturas a peças de áudio, com o intuito de identificar uma peça de áudio guardando a sua informação. Estas assinaturas ou identificações das peças de áudio são muito importantes pois é a partir destas que se vai calcular a medida de similaridade do áudio. Através do mecanismo de assinaturas, as peças de áudio são comparadas de modo a podermos calcular o grau de similaridade entre as mesmas. Cada assinatura actua portanto, como um sumário ou representação de uma peça de áudio e essas representações podem ser comparadas para obter uma "imagem geral" de como uma peça de áudio é semelhante a outra. Finalmente, o último dos grandes objectivos, é a combinação das duas técnicas numa aplicação que as permita usar de uma forma eficiente e a possibilidade desta aplicação integrar noutros sistemas.

2.1.1 Assinatura

O Audio Fingerprinting, consiste num mecanismo que permite gerar uma "assinatura", que identifica um pedaço de áudio. Esta assinatura deve ser derivada, exclusivamente, a partir do conteúdo do áudio em si, isto é, a partir da informação do sinal espectral do áudio. Essa será, então, a informação contida na assinatura que dominaremos de *features*. O método para gerar estas assinaturas tem como objectivos, em primeiro lugar, dividir o sinal do áudio num número de *frames* e, em segundo lugar, computar o conjunto das *features* para cada uma destas *frames* na peça de áudio [8].

A implementação do método utilizado na computação e análise destas *features* consiste numa das partes mais importantes na construção do mecanismo de assinaturas. *Mel-frequency cepstrum coefficients* (MFCC), *Sharpness*, *Spectral Flatness* entre outras, podem ser utilizadas para gerar a representação das *features* de uma peça musical [8], [9], [10].

A descrição acima, dita o método para a criação das assinaturas das peças de áudio neste projecto. É importante referir, também, que estas assinaturas não correspondem a valores do tipo *hash* da peça de áudio mas sim uma representação espectral das *features* da mesma. Desta maneira, estas permitem uma comparação entre elas com base em métodos numéricos, caso não seria possível se estas assinaturas fossem representadas como valores relativos ao tipo *hash*.

2.1.2 Similaridade do Áudio

As assinaturas, descritas na secção anterior, oferecem-nos a base para o cálculo de similaridade entre elas. Estas assinaturas podem ser comparadas através de diferentes técnicas e mecanismos, permitindo comparar duas peças de áudio. Para que duas peças de áudio possam ser comparadas relativamente ao seu grau de semelhança, as assinaturas correspondentes irão ser comparadas. *Pitch Histograms* [11], *Earth Mover´s Distance* [12] e *Euclidean Distance* são técnicas usadas para comparar assinaturas [13], [14]. Estas técnicas, além de outras, conseguem quantificar e combinar - *matching*, através de modelos matemáticos, as representações musicais fornecidas pelas assinaturas, calculando assim o grau de similaridade entre as mesmas.

No entanto, o grau de rigor deste match para produzir relações de similaridade entre peças de áudio é muito dependente da precisão do mecanismo de assinaturas, ou seja, da capacidade deste representar as diferentes *features* de uma peça de áudio [15]. As funções de

similaridade, “trabalham” com as assinaturas das peças de áudio, logo se estas assinaturas não forem rigorosamente representadas, isto é, se estas não corresponderem à simbolização exacta da peça de áudio, então as funções de cálculo de similaridade irão, certamente, produzir resultados erróneos.

2.1.3 Protótipo

Ao implementar o mecanismo de criação de assinaturas das peças de áudio e o sistema de cálculo da similaridade para as mesmas, temos a base do projecto. No entanto, o desenvolvimento de um protótipo que demonstre o funcionamento destas tecnologias é igualmente importante. Este protótipo consistirá, basicamente, na construção de um interpretador de comandos que providencie algumas funções como:

- Criar bases de dados de peças de áudio
- Adicionar peças de áudio as bases de dados
- Gerar assinaturas
- Examinar e comparar assinaturas.

Após o término do protótipo com estas, e outras funções adicionais, será possível a integração com outros sistemas externos.

2.2 Metodologia

Como referido anteriormente, o este trabalho tem como objectivo a construção de um protótipo que permita a extracção e cálculo da similaridade entre diversas peças de áudio. Este protótipo permite a comunicação com outras entidades (cliente ou outro sistema externo) através de um interpretador de comandos. Este interpretador de comandos disponibiliza um conjunto de operações que permite não só a extracção e cálculo de similaridade do áudio como a criação de bases de dados de peças de áudio, outras operações para testes e visualização de resultados.

Neste trabalho será, então, utilizado uma metodologia iterativa pois ao longo da execução do protótipo, à medida que serão necessários novos mecanismos para a extracção e cálculo de similaridade de áudio, serão adicionados novos comandos ao interpretador.

2.3 Planeamento

A realização desta tese foi desenvolvida seguindo o seguinte enquadramento temporal:

- Setembro: Estudo da arte da extracção e cálculo de similaridade entre peças de áudio.
- Outubro: Proposta de uma solução para o problema (familiarização com bibliotecas e aplicações). Definição da metodologia e preparação da escrita do relatório preliminar.
- Novembro-Dezembro: Início da construção do protótipo – interpretador de comandos.
- Janeiro-Fevereiro: Implementação dos algoritmos escolhidos e adição de novos comandos
- Março-Junho: Avaliação do interpretador de comandos e testes relativos à extracção e cálculo de similaridade entre peças de áudio.
- Julho: Escrita do relatório final

Capítulo 3

Trabalho Relacionado

Este capítulo fornecerá uma visão geral das informações onde esta de tese se baseia. Em primeiro lugar, é discutido, num contexto musical e não apenas peças de áudio, a importância dos sistemas de recomendação automáticos. Neste ponto, é exposto quais os mais utilizados nos dias de hoje e como, alguns deles, realizam as operações de cálculo de similaridade entre peças de áudio sendo o mecanismo destes comparado com o proposto do protótipo para o mesmo efeito. Em segundo lugar, são expostas as técnicas de processamento do áudio. Neste ponto, são mencionadas algumas das mais conhecidas e utilizadas técnicas para a extracção das *features* das peças de áudio. Por último e em terceiro lugar, são descritas algumas formas para o cálculo de similaridade entre as peças de áudio; entre as assinaturas das mesmas.

3.1 Sistemas de recomendação automática

De acordo com as estimativas mais recentes, a quantidade de música disponível está a crescer a uma velocidade extraordinária de aproximadamente cinco horas de música por hora. Como consequência, torna-se imperativo que qualquer utilizador utilize diferentes filtros de modo a providenciar uma selecção de acordo com os gostos do mesmo. Antes de existirem estes mecanismos, o utilizador quando pretendia obter outras músicas tinha pouca escolha e uma influência mínima nestes filtros: mudava de estação de rádio ou ir a uma loja de CDs onde os mesmos eram escolhidos por outros indivíduos, ou seja, estava amplamente depende das escolhas de outros.

As tecnologias de digitalização musical mudaram esta situação em pelo menos dois aspectos: empresas de distribuição de música digital como o a Amazon ou a Apple com o iTunes, oferecem aos utilizadores acesso rápido a milhões de musicas a preços relativamente baixos, o que torna mais imperativo a necessidade de filtros, e com o abandono da utilização

de música guardada em dispositivos físicos, como CDs e vinil, a granularidade foi mudada de álbum para faixas individuais, o que provoca uma dificuldade acrescida, para o utilizador, em termos de escolha a nível de peças musicais. Para colmatar esta lacuna de falta de filtros, mecanismos de recomendação automáticos foram surgindo, cujo principal objectivo é, então, oferecer acesso altamente individualizado a "toda" a música existente no mundo.

Em seguida, irá ser feita uma revisão de como estes sistemas ajudam nesta matéria e quais destes respondem mais satisfatoriamente às necessidades:

- **Amazon** – Sugere álbuns ou música baseado no que foi adquirido com uma determinada ordem ou pelos mesmos clientes ou, também, como itens procurados. Esta forma de filtragem, denominada de filtragem colaborativa, pressupõe que os utilizadores que "concordam" com o passado, isto é, que continuam a gostar das mesmas músicas ou álbuns de que gostavam em tempos passados, continuarão a gostar, também, no futuro.

  Este tipo de filtragem geralmente sofre de dois grandes problemas: um deles é o começar, ou seja, o inicio quando um utilizador faz parte do sistema. É um problema pois este não possui histórico do que gosta, logo não serão possíveis recomendações de possíveis produtos. O segundo problema consiste no grau de popularidade de um álbum ou música. Neste caso, os álbuns que ainda não tenham sido adquiridos por qualquer pessoa não podem ser sugeridos ao utilizador e os álbuns mais populares possuem uma probabilidade muito maior de serem sugeridos do que os menos populares. Como consequência, esta filtragem colaborativa é muito pouco eficaz no que diz respeito a sugerir novas músicas ou álbuns musicais. Uma outra falha da Amazon, especifica para este problema, reside na possibilidade de um utilizador comum poder fazer encomendas para outras pessoas (por exemplo um presente), o que provocará um processo de recomendação, tanto para o próprio utilizador como para o outro, incorrecto.

- **Spotify** – Este sistema de streaming de músicas, baseia as suas recomendações com base no comportamento do utilizador, ou seja, nos artistas em quais o utilizador ouve mais. Podendo, potencialmente, oferecer resultados positivos face á estratégia seguida pela *Amazon*, pois não existe um registo com o histórico de registos de um utilizador,

este sistema possui o mesmo problema do começo e da popularidade. Além disso, o Spotify só recomenda artistas relacionados, o que é revela muito vago.

- **Genius** – este sistema que consiste numa função do iTunes, da Apple, gera listas de músicas e recomendações baseadas em: bibliotecas de musicas, histórico de compras e listas de musicas de outros clientes, fazendo eventualmente, a integração de informação externa. Assumindo que essa informação externa não assuma um papel fulcral, este sistema irá sofrer dos mesmos pontos negativos da Amazon com a filtragem colaborativa.

- **Last.fm** – este sistema combina informações obtidas com base no comportamento dos utilizadores com a metadata que este coloca (palavras ou expressões que descrevem uma música, álbum ou artista musical). Esta metadata beneficia o processo de recomendação sendo transparente para os utilizadores pois se por exemplo, um utilizador estiver a ouvir uma canção romântica este possivelmente terá como recomendações músicas cuja metada será "slow" ou "canções de amor"

- **Pandora –** à semelhança do spotify, o Pandora é também um serviço de streaming de música. Recomenda músicas que possuem no sistema com base em comentários de especialistas na área. Este facto, permite mecanismos de recomendação muito precisos que soam, de facto, aquilo que o utilizador pretende explicando detalhadamente o porque das sugestões. Verificamos, então, que este sistema faz as suas recomendações musicais com base na análise do conteúdo do áudio em vez da filtragem colaborativa utilizada no Last.fm e no Spotify ou de tags, não sendo por isso afectado pelo problema da recomendação com base na popularidade. No entanto, as opiniões de especialistas possuem custos elevados em termos de tempo e dinheiro: cada música demora, em média, quarenta e cinco minutos a ser devidamente analisada e comentada, o que torna inviável novas adições de música ou álbuns tendo como consequência a elevada limitação de escolha por parte dos utilizadores

- **Shazam** – Este sistema analisa o conteúdo do áudio da música e, apesar de não efectuar nenhum processo de recomendação de outras músicas, encontra apenas a mesma música na sua base de dados fornecendo os seus dados associados como por exemplo: o artista, o título, o ano de lançamento, entre outros. As *features* extraídas

pelo sistema (os picos mais altos no spectograma analisado [16]) são utilizados para o cálculo de similaridade. No entanto, estas mesmas *features* são praticamente inúteis para a recomendação de novas peças musicais.

3.2 Técnicas de processamento de áudio

Dito em secções anteriores, um dos principais objectivos do trabalho, é encontrar um conjunto de características que descrevem uma peça de áudio, ou seja, encontrar um conjunto de parâmetros que possam ser estimados e computados através do sinal do áudio em questão. Parâmetros esses que, anteriormente dito, denominamos de *features* que permitem gerar a assinatura das peças de áudio.

Temos, então, como conjunto de *features* de possível extracção relativas ao timbre e ao espectro da peça de áudio de cada *frame*:

3.2.1 Zero-Crossing

Este tipo de *features* é tipicamente usado em matemática, processamento de imagem e processamento de áudio. Podemos designar "Zero-Crossing" como um ou vários pontos onde o sinal de uma função muda, isto é, quando um sinal de uma função muda de negativo para positivo. Em termos de processamento de áudio, é contada o número de vezes em que a onda de frequência muda de sinal. Na figura a baixo é explicado como é medido o zero-crossing a nível de medição de sinal.

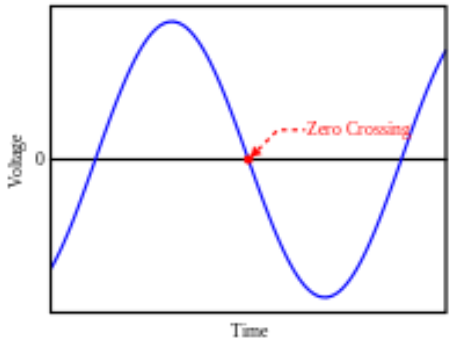

Figura 3.1: Visualização de Zero Crossings

3.2.2 Spectral Centroid

Este tipo de *features* é principalmente usado na análise do espectro do áudio. Estas *features* indicam onde se encontra a massa do espectro, ou seja, onde na peça de áudio existe maior potência em termos de sinal.

3.2.3 Spectral Flux

Este tipo de *features* é usado para calcular o quão rápido, a potência do sinal, muda consoante o tempo. Este cálculo é obtido calculando o poder do sinal de uma *frame*, comparando esse poder com o poder do sinal da *frame* anterior.

3.2.4 Mel-Frequency Cepstral Coefficients (MFCC)

Este tipo de *features* são muito utilizadas na análise de áudio pois permitem modelar as características espectrais do sinal de curta duração numa escala de frequência. Estas *features*, não explicitam claramente os aspectos temporais da peça de áudio em causa sendo, por isso, estas *features* associadas a um saco de *frames*. Então, peças de áudio com as mesmas *features* MFCC com diferente ordem, iriam produzir os mesmos resultados, ou seja, seriam duas peças de áudio semelhantes.

3.3 Representação de *features*

Como resultado da extracção das *features*, iremos ter quantidades elevadíssimas de informação, isto é, um grande número de vectores de *features* com várias dimensões, aproximadamente 1290 por cada 30 segundos de áudio. Ora isto torna, praticamente, impossível uma análise e manipulação da informação duma forma eficiente. Para tornar esta informação fácil de perceber e manipular, são necessárias técnicas de agregação de *features*: técnicas estatísticas ou de quantificação.

No que diz respeito a técnicas de agregação com base em termos estatísticos, a mais utilizada consiste em Gaussian Mixture Models [17]. Este modelo estatístico, utilizado para reconhecimento de áudio e segmentação de imagens, consiste basicamente numa função que lida com densidades a nível de probabilidades. Estes modelos são normalmente utilizados em funções de distribuição de probabilística em termos de *features* que resulta na criação de distribuições multidimensionais. Cada distribuição é representada através dum conjunto de

clusters. Estas distribuições multidimensionais podem ser estimadas através de dois algoritmos diferentes: Expectation-Maximization [18] e Maximum A Posteriori.

Em termos de técnicas de agregação quantitativa [19], o algoritmo *K-means* é um dos métodos mais convencionais e bem-sucedidos no que diz respeito a este aspecto [20]. Estudos mostram que o algoritmo *K-means* é relativamente eficiente no que diz respeito à simplificação das *features* usadas no trabalho. Para ser possível a sumarização desta grande quantidade de informação, este algoritmo vai, encontrar apenas os vectores mais representativos de cada peça, permitindo assim uma representação mais compacta melhorando a eficiência em termos de análise de informação. Estes vectores, que denominamos de *clusters* ou *centroids*, são usados como um dicionário ou *codebook* que nos permite transformar uma peça de áudio numa sequência de símbolos. Posteriormente, esta sequência de símbolos irá ser agregada num histograma, obtendo assim a assinatura de uma peça de áudio que pretendemos, como mostra a figura 3.2.

Após a obtenção destas assinaturas, estas serão passiveis de serem comparadas, através de diferentes técnicas, permitindo assim o cálculo de similaridade entre as mesmas. Estas técnicas serão discutidas no próximo ponto.

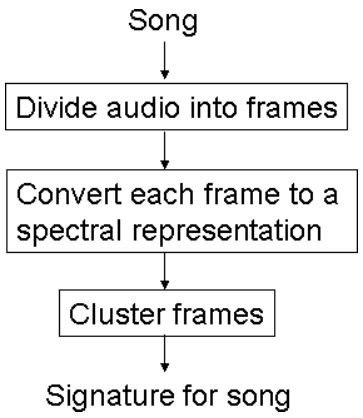

Figura 3.2: Cadeia de eventos para o cálculo de assinaturas

3.4 Técnicas de cálculo Similaridade áudio

Uma vez geradas as assinaturas das diferentes peças de áudio, estas precisam de ser comparadas. Estas assinaturas possuem uma quantidade enorme de informação e para serem comparadas, algoritmos eficientes de comparação devem ser empregues de modo a obtermos resultados satisfatórios.

Temos então, como mais conhecidas técnicas de cálculo de similaridade do áudio:

3.4.1 Earth Mover´s Distance

Este algoritmo consiste basicamente numa métrica entre modelos gaussianos, isto é, entre distribuições multidimensionais. Cada distribuição é representada através dum conjunto de *clusters* que irá servir como assinatura da mesma. Em termos de cálculo de similaridade de áudio, para comparar peças de áudio, este algoritmo irá calcular o custo mínimo necessário para transforma uma distribuição (neste caso, o conjunto de *features* de uma peça) numa outra distribuição relativa a outra peça de áudio, ou seja, o custo mínimo para transformar a assinatura duma peça de áudio noutra. Na figura em baixo mostra como são calculados estas distribuições multidimensionais com base na distribuição das *features* em termos de *clusters*.

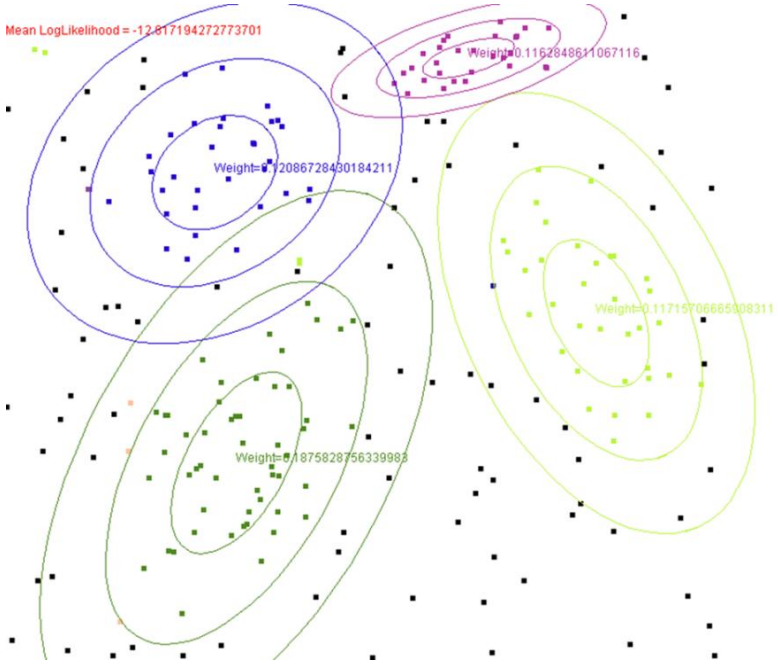

Figura 3.3:Criação de distribuições multidimensionais

3.4.2 Pitch Histograms

Este algoritmo, compara a similaridade entre duas peças de áudio através dos seus pitch, isto é, através dos seus tons mais representativos. Baseados em chroma vectors (vectores com doze elementos em cada dimensão que representa a intensidade associada a um tom musical), atribui um array de 128 valores indexados que mostram a frequência da ocorrência de cada nota de uma música o que permite capturar informações harmónicas de diferentes géneros musicais. O cálculo de similaridade é, então, feito entre os tons das várias

peças musicais onde, por exemplo, os géneros musicais com maior complexidade a nível de tom (como a música clássica ou jazz) irão apresentar uma maior variância a nível de tom e, portanto, possuem *pitchs* mais pronunciados comparados com outros géneros musicais como o Rock, Hip-Hop ou Música Electrónica que normalmente contêm progressões de acordes mais simples.

3.4.3 Euclidean Distance

A distância euclidiana, consiste numa medida entre objectos (normalmente pontos) num sistema de coordenadas cartesiano, facilmente visualizada através dum gráfico bidimensional. Para calcular a distância, tendo dois pontos, utilizamos a seguinte fórmula:

$$d = \sqrt{\sum_{i=1}^{N} \left| I_i - J_i \right|^2} .$$

Capítulo 4

Trabalho realizado

Este capítulo tem como principal objectivo a explicar o actual desenvolvimento do protótipo. Este capítulo abrange, a execução real do tema para esta tese e descreve os métodos implementados. Os desenvolvimentos dos conteúdos baseados em algoritmos de similaridade de áudio serão discutidos em primeiro lugar. Em segundo lugar, é discutida e explicada a integração desses algoritmos no protótipo, com detalhes fornecidos sobre as funcionalidades de cada comando.

4.1 Assinatura e Similaridade

Neste ponto, são discutidos os algoritmos para a gerar a assinatura das peças de áudio e algoritmos de cálculo de similaridade para que, as mesmas, possam ser comparadas. Uma assinatura, que representa uma peça de áudio, será gerada através da extracção das suas *features*. Estas assinaturas, depois de criadas, podem ser comparadas, através de uma medida de distância, de modo a calcular o quanto são próximas, isto é, o seu grau de semelhança [21]. Ao contrário de outros modelos relacionados, nenhum conjunto de *features* foi escolhido a priori para extracção, sabendo apenas qual o melhor conjunto de *features* após a realização de testes experimentais e análise dos resultados. No que diz respeito ao algoritmo escolhido para simplificação dos dados foi escolhido do algoritmo *K-means* que permite uma representação compacta das *features* extraídas através de *clusters*. Por último, será descrita a utilização da Distância Euclidiana associada ao algoritmo Nearest Neighbour para fazer, então, o cálculo de similaridade entre várias peças de áudio.

4.1.1 Criação assinatura – *cluestering*

Após a extracção destes diferentes tipos de *features*, estas serão computadas e agregadas de um modo compacto, de modo a ser possível a sua visualização de manipulação

de uma forma eficiente. Entre as técnicas de representação de *features*, mencionadas anteriormente, a utilizada para este projecto foi o algoritmo *K-means* uma vez que consiste no modelo mais convencional entre os modelos até hoje conhecidos. Este algoritmo irá encontrar as *k2 features* mais representativas de todas as que foram extraídas agrupando-as em conjuntos que denominamos de *clusters*. A partir deste conjunto de vectores mais representativos, ou *clusters*, irá ser criado um *codebook*, que funcionará como um dicionário, possibilitando transformar uma peça de áudio numa sequência de símbolos. Posteriormente, esta sequência de símbolos irá ser agregada num histograma, obtendo assim a assinatura de uma peça de áudio que pretendemos.

4.1.2 Criação assinatura – extracção *features*

Dito anteriormente, existe um conjunto de *features* possíveis de extracção (MFCC, flux..) para cada peça de áudio. Neste trabalho, ao contrário de outros projectos relacionados, não foram apenas retirados os MFCC mas um conjunto de *features* mais abrangente com o intuito de obtermos melhores resultados que os actuais. Nesta fase do trabalho, o mecanismo que permite: a extracção as *features* e a especificação das quais são extraídas, é de elevada importância, pois é a partir destas que se irá gerar a assinatura da mesma, e o cálculo de similaridade. Para não nos restringirmos apenas os MFCC foram criados diferentes conjuntos de *features* para extracção que denominamos de feature sets. Estes feature sets permitem declarar exactamente quais as *features* a extrair das peças de áudio. Mais especificamente um *feature set* permite:

- Escolher exactamente quais as *features* a extrair.
- Block Size – o número de *frames* que iram resultar do output da extracção.
- Step Size – o espaço temporal entre duas *frames* consecutivas.

Na figura 4.1, é descrito a cadeia de eventos que os mecanismos, usados para o trabalho, executam para obterem as *features* agregadas das peças de áudio.

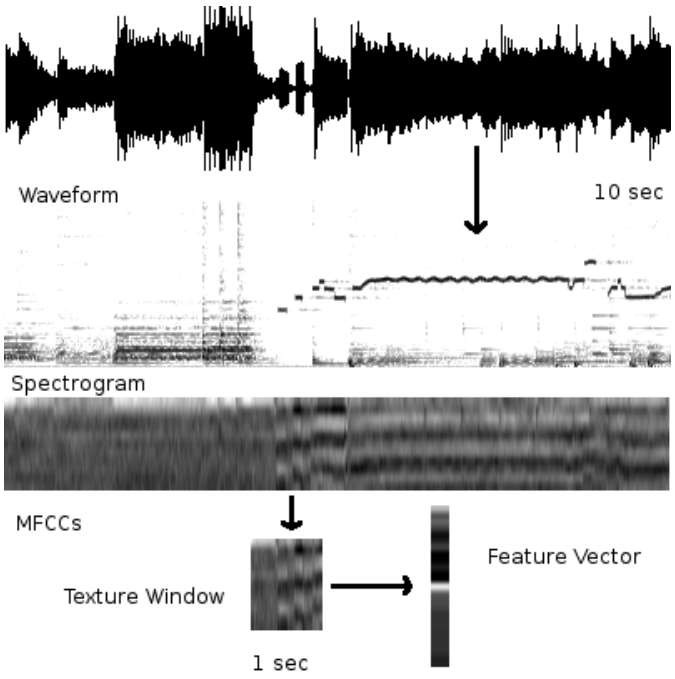

Figura 4.1: Cadeia de eventos para a extracção e agregação de *features* em *feature vectors*

4.1.3 Criação assinatura – implementação

No projecto, a extracção destas *features* é conseguida através de duas *frameworks*: *Marsyas* e *Yaafe*. Estas fornecem um conjunto de funções, através da linha de comandos, que permitem a extracção das *features*:

- **Marsyas** – Nesta *framework*, são utilizados dois métodos para gerar a assinatura de uma peça de áudio: `mkcollection` e `bextract`. A função `mkcollection`, cria uma base de dados de peças de áudio, cujas assinaturas pretendemos obter, recebendo um ou mais directórios como argumento, produzindo no final um ficheiro com a terminação .mf que indica a localização de cada uma das peças de áudio em questão. A função `bextract`, consiste na ferramenta mais importante do *Marsyas* pois é ela que vai extrair as *features* e produzir a assinatura de uma ou mais peças de áudio. Esta recebe como argumento: um ficheiro do tipo .mf, gerado através da execução do comando `mkcollection`, e o *feature set* que, como descrito anteriormente, especifica exactamente quais as *features* que queremos extrair e aquelas que não pretendemos. Como resultado desta extracção, é então criado ficheiro .arff que possui um grande conjunto de vectores de dimensão variável, dependendo do *feature set*, cuja representação é compactada através do *K-means*

num conjunto de símbolos. Feito isto, temos então concluído os mecanismos responsáveis para a criação de assinaturas para as peças de áudio.

- **Yaafe** – Nesta *framework*, ao contrário do *Marsyas*, é necessário apenas um método para a extracção e geração de assinatura: Feature Extraction Plan. Esta função, à semelhança do *Marsyas*, permitirá extrair as *features* e produzir a assinatura de uma ou mais peças de áudio. A figura 4.2, descreve detalhadamente como o *Yaafe* realiza a extracção das *features*: como argumentos recebe um conjunto de directórios de peças de áudio, cujas *features* pretendemos extrair, e um feature extraction plan que consiste num ficheiro de texto em que cada linha define o tipo de *feature* a ser extraída. No caso do *Yaafe*, o feature extraction plan determinará o feature set escolhido para extracção. Como resultado da extracção, é então criado ficheiro, não .arff como no Marsyas, .csv, individual para cada *feature* indicada, que possui um grande conjunto de vectores de dimensão variável, cuja representação é compactada, também, através do *K-means* num conjunto de símbolos.

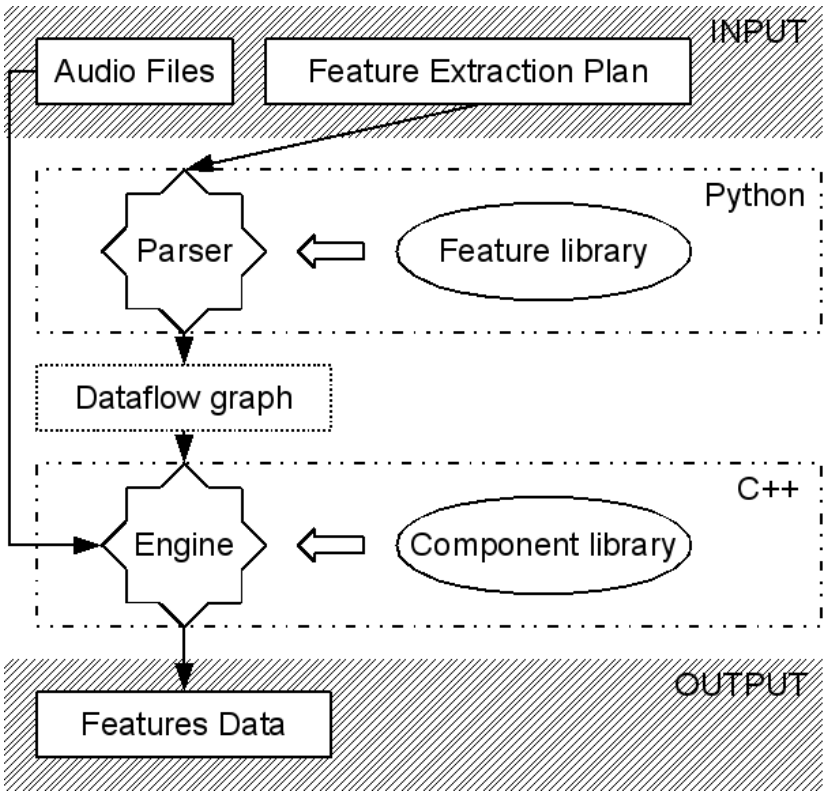

Figura 4.2: Estrutura do Yaafe

Como mencionado anteriormente, para não nos restringirmos a um determinado tipo *features*, foram criados *feature sets* que informam as *frame*works mencionadas a cima quais o tipo de *features* a extrair. Para este efeito, foram criadas cinco *feature sets*, que indicam qual a *frame*work desejada para extrair as *features* e quais destas irão ser extraídas:

- **Feature set 1 (fs1)**
  - **Framework -** *Marsyas*
  - **Features -** MFCC, Roll-Off, Flux e ZeroCrossings
  - **Block Size –** 512
  - **Step Size –** 512
- **Feature set 2 (fs2)**
  - **Framework -** *Marsyas*
  - **Features –** MFCC
  - **Block Size –** 512
  - **Step Size –** 512
- **Feature set 3 (fs3)**
  - **Framework -** *Yaafe*
  - **Features -** Energy, RollOf, MFCC, Flux e ZeroCrossings
  - **Block Size –** 1024
  - **Step Size –** 1024
- **Feature set 3 (fs4)**
  - **Framework -** *Yaafe*
  - **Features -** MFCC
  - **Block Size –** 1024
  - **Step Size –** 1024
- **Feature set 1 (fs5)**
  - **Framework -** *Marsyas*
  - **Features** - MFCC
  - **Block Size –** 1024
  - **Step Size –** 1024

Após a extracção das *features*, será agora necessário a representação das mesmas de uma forma compacta que permita a sua manipulação de uma forma eficiente. Esta agregação é realizada através do algoritmo *K-means* que, como dito anteriormente, através de um conjunto de *features*, este algoritmo irá encontrar as mais representativas e guarda-lo num ficheiro .arff ou .csv como *clusters*. Este algoritmo é implementado no projecto com a ajuda da ferramenta de machine learning WEKA. Esta ferramenta disponibiliza uma biblioteca que pode ser usada através da linguagem de programação Java. Uma vez que o interpretador de comandos está implementado em java, para utilizar este algoritmo de sumarização iremos recorrer à classe SimpleKMeans fornecida por esta ferramenta.

Criado o *codebook*, isto é, o conjunto de *features* mais representativas, as peças de áudio irão ser representadas numa sequência de símbolos. Para tal, iremos para cada *feature*, de cada peça de áudio, encontrar o *cluster* mais próximo e guardar essa informação no ficheiro que representa a sequência de símbolos. Para encontrar este *cluster* mais próximo, vamos usar o algoritmo Nearest Neighbour que, à semelhança do SimpleKMeans, podemos implementa-lo no interpretador de comandos através de classe LinearNNSearch. Após gerada a sequência de símbolos de cada peça de áudio, verificamos que a informação é, ainda, difícil de analisar e manipular uma vez que é atribuído o *cluster* mais próximo a cada *feature* (*feature*, *cluster*). Logo, esta sequência de símbolos de cada peça de áudio irá ser agregada em histogramas, onde é contabilizado quantas features estam mais próximas de cada *cluster*, representado as assinaturas para as peças de áudio.

4.1.4 Critério de medida de similaridade – distância euclidiana

A semelhança entre duas peças de áudio pode ser calculada medindo a distância, entre o conjunto das *features* extraídas, ou seja, entre as suas assinaturas. Técnicas como Earth Mover´s Distance e Euclidean Distance podem ser usadas, para cálculo de similaridade entre peças de áudio, que tem sido amplamente usadas noutros projectos do mesmo tipo. A Euclidean Distance será utilizada para este projecto, uma vez que consiste num tipo de medida eficaz, sendo, também, bastante simples de implementar. Para calcular o grau de similaridade entre peças de áudio, será aplicado o algoritmo da distância euclidiana entre os histogramas de cada peça de áudio. Estes histogramas consistem numa versão mais simplificada de um conjunto de símbolos, onde irão ser contabilizados quantos símbolos pertencem a cada *cluster* obtido.

4.1.5 Critério de medida de similaridade – implementação

O cálculo de distâncias utilizando a Euclidean Distance foi directamente implementado, de raiz, no interpretador de comandos. Para tal, foram utilizadas várias funções matemáticas fornecidas pelo java. Desta forma, é fornecido um maior nível de flexibilidade e um maior poder para o interpretador de comandos em questão.

4.1.6 Sumário Fingerprint e Similaridade

Os algoritmos utilizados, nesta fase do protótipo, fornecem a capacidade para produzir assinaturas musicais e calcular a semelhança entre elas. Estes algoritmos trabalham com ficheiros .wav e são implementados através de duas *frameworks*: *Marsyas* e Ya*a*fe. Estas assinaturas estão dispostas em ficheiros *.arff* e *.csv* correspondendo aos outputs provenientes do *Marsyas* e *Yaafe* respectivamente sendo a sua representação simplificada através do algoritmo *K-means* em ficheiros de símbolos.

4.2 Desenvolvimento Protótipo Alto Nível

Uma vez estudados e escolhidos os algoritmos para executar a geração de assinaturas e cálculo de similaridade entre peças de áudio, o passo seguinte foi integração destes algoritmos no interpretador de comandos, permitindo assim a sua usabilidade a alto nível. Neste ponto, é então discutido como foi feito o protótipo a alto nível, mais concretamente: a integração dos algoritmos assinatura e similaridade do *Marsyas* e do *Yaafe*, descritos na secção anterior, no interpretador de comandos e as várias funções ao dispor do utilizador.

4.2.1 Integração/Implementação algoritmos *Marsyas*

Para integrar os algoritmos de extracção e similaridade do Marsyas, no interpretador de comandos, foi necessário encontrar uma solução que não só permita o encapsulamento dos algoritmos no interpretador, mas que também os execute o número de vezes necessário. Uma vez que as funções do *Marsyas* são chamadas/executadas através de um terminal ou linha de comandos e visto que o interpretador de comandos irá ser desenvolvido na linguagem de programação Java, a forma para o encapsulamento das funções do *Marsyas* foi feita através do mecanismo Java Runtime, que permite ao Java aceder e executar comandos como se fosse um terminal.

Por exemplo: para executar o comando `mkcollection`, do *Marsyas*, pela linha de comandos temos:

- `mkcollection -c <ficheiro .mf> <dir>`

Este comando, executado na linha de comandos, recebe um directório, onde estão contidas uma ou mais peças de áudio, e um nome para o ficheiro .mf, onde ficam registadas as localizações das peças de áudio, que irá resultar como output do processo.

No entanto, como mencionado anteriormente, para executar este comando no interpretador de comandos é necessário utilizar o mecanismo do Java Runtime:

- `Runtime.getRuntime().exec(mkcollection -c <ficheiro .mf> <dir>)`

Desta forma, conseguimos encapsular a função `mkcollection`, do *Marsyas*, no interpretador de comandos, o que nos permite agora realizar diferentes operações sem ser necessário recorrer á linha de comandos.

4.2.2 Integração/Implementação algoritmos *Yaafe*

Para integrar os algoritmos de extracção e similaridade do *Yaafe*, no interpretador de comandos, foi utilizada a mesma estratégia descrita no *Marsyas* uma vez que as funções do *Yaafe* são, também, chamadas/executadas através de um terminal ou linha de comandos e visto que o interpretador de comandos irá ser desenvolvido na linguagem de programação Java, a forma para o encapsulamento das funções do *Yaafe* foi, também, feita através do mecanismo Java Runtime, que permite ao Java aceder e executar comandos como se fosse um terminal.

Contrariamente ao *Marsyas*, as funções de extracção de *features* do *Yaafe* encontram-se implantadas na linguagem de programação *Python*. Por exemplo, para extrairmos as *features* de uma peça de áudio temos o seguinte comando:

- `yaafe.py -c <featureplan.txt> -r <freq> <file.wav>`

Este comando, executado na linha de comandos, recebe um ficheiro .txt que determina o *feature set*, que como explicado em pontos anteriores, irá indicar quais os tipos de *features*

que irão ser extraídos pelo *Yaafe*. Os últimos dois parâmetros consistem na indicação da frequência do ficheiro, cujas *features* pretendemos extrair, e a localização do último.

4.2.3 Sumário Protótipo Alto Nível

O interpretador de comandos, desenvolvido nesta fase do projecto representa a ponte entre as técnicas de extracção e similaridade desenvolvidos, e as aplicações baseadas nessas tecnologias. Este interpretador de comandos, aproveita a os algoritmos de geração de assinaturas desenvolvida e cálculo de similaridade, do Marsyas e do Yaafe, encapsulando-os de forma a encorajar o seu uso para a execução de operações relacionadas. Os algoritmos, do Marsyas e do Yaafe, uma vez que são executados através da linha de comandos, foram encapsulados utilizando o mecanismos Java Runtime que permite executar comandos como se este fosse um terminal.

Uma vez estando concluídas estas funções no interpretador de comandos, temos então um conjunto de ferramentas (para maior detalhe destas ferramentas, consultar Apêndice E) necessárias para calcular distâncias entre peças de áudio. Para melhor percepção de como estas ferramentas são uteis, o próximo exemplo descreve a cadeia de comandos necessária para a extracção cálculo e de similaridade entre um conjunto de peças de piano e violino retirados de fontes externas:

```
1. create output directory /home/eduardo/Imagens/violinPianoTrocado/fs1/
2. set features fs1
3. register dir /home/eduardo/tese_mestrado/music/violin/
4. register dir /home/eduardo/tese_mestrado/music/piano/
5. merge collections violinAndPiano piano violin
6. make codebook violinAndPiano 800 2500
7. get symbols violinAndPiano
8. get histograms violinAndPiano
9. distance matrix violinAndPiano violinAndPiano
```

No ponto um, é indicado qual o directório cuja informação de output será gerada e disposta para análise. Em seguida, no ponto dois, é designado o *feature set*, sendo neste exemplo escolhido foi o feature set “fs1” para extracção. No ponto três e no ponto quatro, são criadas bases de dados de peças de áudio de violino e piano respectivamente, ou seja, colecções de música onde são registadas todos os wav contidos em cada um destes directórios. No ponto cinco, vamos juntar as duas colecções numa só uma vez que queremos

saber as distâncias entre todas as peças das duas colecções. No ponto seis, é realizada uma agregação das *features*, usando o algoritmo *K-means*, onde irão ser encontradas as *features* mais representativos de todas as peças de áudio na colecção que fizemos merge. Para tal, é criado um *codebook* com: *k2= 800* e *k1 = 2500*. Neste *codebook*, irão ser criados *800 clusters* num conjunto de *2500* de *features* por cada peça de áudio nas duas colecções

Tendo o nosso *codebook* concluído, no ponto sete e no ponto oito irá ser criada a assinatura de cada peça de áudio na colecção. Nestes dois passos, com base no *codebook* criado no ponto anterior, as *features* irão ser armazenadas numa sequência de símbolos, onde a última, para uma melhor percepção da informação e irá ser agregada num histograma obtendo assim a assinatura. Este processo irá ser repetido para cada peça de áudio da colecção, obtendo assim cada assinatura para cada peça de áudio que se encontra na colecção.

Capítulo 5

Testes e Resultados

Os capítulos anteriores desta tese, descreveram o planeamento e execução de todo o trabalho concluído deste projecto. Para averiguar o grau do sucesso de como o sistema calcula a similaridade entre peças de áudio, é imperativo que essa estimação seja empírica.

Este capítulo fornecerá uma revisão crítica deste trabalho, apresentando os resultados produzidos com base na execução de vários testes. Os resultados apresentados nesta secção, foram recolhidos ao longo do desenvolvimento do projecto e sua análise realizada gradualmente. Em primeiro lugar, serão descritas as formas possíveis para avaliar o grau de similaridade entre peças de áudio. Neste ponto, serão explicados os mecanismos construídos de forma a avaliar o grau de similaridade de uma forma rápida e eficaz. Na segunda secção, é descrita a informação, ou dados, que foram utilizados para o cálculo de similaridade. Neste ponto, é descrito como obtemos esses dados e as diversas formas possíveis de testes. Em terceiro, serão expostas as diversas experiências que fizemos com os dados descritos na secção anterior. Neste ponto, serão descritos com detalhe como foram feitas. Em quarto e último lugar, será feita uma sumarização dos resultados baseada na informação obtida.

5.1 Representação dos Resultados

Uma vez que lidamos com grandes quantidades de informação, torna-se necessário a construção de mecanismos de modo analisar e avaliar o output de uma forma rápida e intuitiva. Nesta secção, são descritos detalhadamente esses mecanismos e o porquê da sua criação, ou seja, em que medida estes nos são uteis na leitura dos resultados que obtivemos das experiências realizadas.

5.1.1 Distance Matrix

Uma matriz de distância é uma matriz bidimensional contendo as distâncias, aos pares, entre um conjunto de pontos. Neste projecto, para uma melhor leitura de resultado, esta matriz terá um tamanho de N x N, sendo N o número peças de áudio numa colecção ou data set. A figura a baixo descreve como serão apresentados os resultados através de uma matriz de distância.

Dist	A	B	C	D	E	F
A						
B	0.71					
C	5.66	4.95				
D	3.61	2.92	2.24			
E	4.24	3.54	1.41	1.00		
F	3.20	2.50	2.50	0.50	1.12	

Figura 5.1: Exemplo de uma matriz de distância

Como observado na figura, esta representa uma matriz de distância simétrica (a distância entre A-B é igual a B-A) quadrada, onde irá registar a distância entre todas as músicas de uma colecção. No entanto, se pretendermos medir a similaridade entre peças de áudio, não de uma mas várias colecções, obviamente a matriz não será uma matriz diagonal e só será quadrada se as colecções possuírem o mesmo número de peças de áudio.

Desta forma, conseguimos então, ter uma percepção rápida e intuitiva das distâncias entre as diversas peças de áudio.

5.1.2 Nearest Neighbour

Na secção anterior foi descrita uma matriz de distância para a observação dos resultados de uma forma generalizada. No entanto, esta medida não se revela muito eficaz quando pretendemos saber quais as peças mais próximas de uma determinada peça dada como input, pois esta mostra a distância entre todas as peças de áudio nas colecções e não uma peça individual, sendo no entanto, uma matriz de distâncias é bastante útil quando queremos comparar grandes quantidades de peças de áudio.

Para obtermos, então, as peças mais semelhantes de apenas uma peça de áudio, foi criado um mecanismo chamado nearest neighbour (comando ver no manual). Este mecanismo foi inspirado no algoritmo nearest neighbour, e possibilita-nos dar como input

uma peça de áudio e receber como output, as N mais próximas. Este mecanismo, revela-se muito proveitoso quando queremos medir o grau de similaridade de peças com eventos bem definidos como por exemplo: vidro a partir, fogo, tiros etc.

5.1.3 Comparação Visual

Os dois mecanismos, mencionados nas duas secções anteriores, são bastante intuitivos no que diz respeito à apresentação de resultados: uma matriz de distância quando queremos resultados mais quantitativos e o nearest neighbour quando queremos resultados mais detalhados, ou seja, resultados qualitativos. No entanto, outro mecanismo foi construído para a visualização e análise dos resultados: a comparação visual a partir de gráficos. Desta forma, será possível a construção de heurísticas de modo a ser possível a "visualização" de determinados sons e verificar se realmente estes foram registados nos instantes correctos.

5.2 Dados

Uma vez criado o interpretador de comandos, que integrou a geração de assinaturas de peças de áudio e técnicas de medição do grau de similaridade anteriormente desenvolvidos, grandes bases de dados de peças de áudio podem ser usadas para a comparação das suas assinaturas. A criação de bases de dados musicais são úteis na medida em que: o agrupamento de peças de áudio do mesmo tipo (podemos querer apenas bases de dados de musica clássica, ou jazz. Ou sons específicos como cães a ladrar, vidros a partir.) a serem recolhidos para calcular as assinaturas de um grande número de peças de áudio e em segundo lugar, proporciona um grande conjunto de base de dados que pode ser usado para testar as aplicações desenvolvidas.

De modo a não nos fixarmos apenas em música, foram criadas outras bases de dados de modo aos resultados serem mais abrangentes. As bases de dados utilizadas para as experiências foram as seguintes: peças musicais, instrumentos musicais, e episódios de séries. Posteriormente, será descrito com maior detalhe cada uma dessas bases de dados utilizadas para o cálculo de similaridade.

5.2.1 Músicas

Para testar o grau de similaridade entre peças musicais, foram criadas duas bases de dados: uma com trinta músicas (dez de rock, dez de musica pop/disco e dez de jazz) e uma

outra mais alargada, ou seja, com sessenta músicas (trinta de rock, trinta de musica pop/disco e trinta de jazz). Para maior detalhe destas bases de dados, estas encontram-se descritas nos Apêndice A e B respectivamente.

5.2.2 Instrumentos

Como dito anteriormente, ao contrário de outros projectos na área de extracção e cálculo de similaridade, onde o grau de semelhança entre as peças de áudio é calculado principalmente através do género musical de cada uma, neste projecto não foram apenas comparadas peças musicais. Foi então criada uma base de dados apenas com instrumentos musicais, isto é, uma base de dados onde as peças musicais contem apenas um determinado instrumento isolado, ou seja, apenas o som produzido pelo mesmo. Esta base de dados de instrumentos, que foram obtidos através do *Youtube* e *Freesounds*, contem: seis peças de áudio de xilofone, seis de flauta, seis de violino, seis de piano, seis de guitarra, seis de flauta e seis de clarinete. Para maior detalhe desta base de dados, esta encontra-se descrita no Apêndice C.

5.2.3 Séries

Um dos principais objectivos destrabalho será não só a possibilidade de medir a semelhança entre várias peças de áudio, não apenas música, mas também a detecção de determinados sons com por exemplo: cão a ladrar, tiro de pistola, vidros a partir, objectos a arder etc.

Para tal, foram criadas bases de dados de episódios: uma com *samples* da serie “Lost”, outra da serie “Prison Break”, uma da serie “Walking Dead” e uma da série “24”. De modo a capturar eventos específicos, mencionados no parágrafo anterior, o áudio dos episódios foi “cortado” em pequenas parcelas de três segundos com overlap, para aumentar a probabilidade de obter um som que corresponda a um evento reconhecido, de um e meio segundos (1ª peça: 0-3 seg, 2ª peça: 1.5-4.5 seg, …). Para maior detalhe desta base de dados, esta encontra-se descrita no Apêndice D.

5.2.4 Resultados

Uma vez criado o sistema, que integrou os mecanismos de geração de assinatura e de cálculo de similaridade de medida anteriormente desenvolvidos, grandes bases de dados de peças de áudio podem agora ser comparados para calcular o grau de similaridade entre elas.

Neste Ponto, irão ser revelados os resultados das experiências, mais concretamente, com as bases de dados expostas no ponto anterior. Em primeiro lugar, irão ser expostos os resultados de cálculo de similaridade utilizando as peças de áudio expostas no Apêndice A e B de modo a averiguar se as músicas mais próximas pertencem ao mesmo género musical. Em segundo lugar, irão ser feitos testes mais qualitativos, ou seja, testes com sons específicos onde se procura saber se o sistema é capaz de por exemplo: reconhecer um vidro a partir num episódio de uma série ou detectar instrumentos musicais isolados. Em terceiro lugar, são apresentados visualmente resultados, baseados em heurísticas, onde podemos perceber o sistema se comporta como o esperado no que diz respeito ao cálculo de similaridade entre peças de áudio.

5.2.5 Classificação de Género

No que diz respeito aos resultados relativos à classificação de género, como mencionado anteriormente, foram utilizadas as peças de áudio expostas no Apêndice A e B (peças musicais) e C (Instrumentos). Em primeiro lugar irão ser apresentados e discutidos os resultados das peças musicais e, então, dos Instrumentos isolados.

O facto de utilizarmos duas bases de dados de músicas, uma com o triplo das músicas da outra, permitiu-nos observar se os resultados seriam muito diferentes caso o número de músicas de cada género fosse diferente. Em ambas as bases de dados foram feitas três experiencias diferentes, isto é, para a mesma colecção foram calculadas as assinaturas de maneiras diferentes: com $k1$ = 1000 e $K2$ =5,10 e 100. No entanto, irá ser apresentado em detalhe (todas as tabelas de distância) apenas, entre as três experiências, a que apresentou melhores resultados. As restantes os resultados serão resumidas, de forma a termos uma visualização da comparação entre as mesmas de uma forma intuitiva, onde irá apenas ser apresentada a diferença entre: a distância média entre peças do mesmo género (Intra Class Average Distance) e a distância média entre peças de géneros diferentes (Inter Class Average Distance)

Para a colecção A (peças correspondentes no Apêndice A), na primeira experiência, foram utilizados os seguintes parâmetros:

- **Feature Set 1** – extraídas MFCC, Roll-Off, Flux e ZeroCrossings
- **K2 = 5** (número de centroids)
- **K1 = 1000** (número *features* extraídas por cada peça)

Para a primeira experiencia obtemos os seguintes resultados:

Jazz										
	Follow_Me	Are_We_There_Yet	And_Then_I_Knew	James	The_First_Circle	Dream_Of_The_Return	Every_Summer_Night	Red_Sky	Afternoon	If_I_could
Follow_Me	0	3814.62	7969.22	3135.97	8028.68	4027.18	6754.14	5060.7	4882.46	7879.12
Are_We_There_Yet	0	0	6633.04	1436.4	4736.85	3768.58	5610.22	4454.19	6053.28	5074.7
And_Then_I_Knew	0	0	0	6299.37	6716.61	5585.1	1505.45	3347.04	6060.88	8620.08
James	0	0	0	0	5231.38	2652.48	5115.75	4105.79	5106.33	5107.7
The_First_Circle	0	0	0	0	0	6889.17	6190.98	6518.67	9409.61	3926.7
Dream_Of_The_Return	0	0	0	0	0	0	4446.07	3911.76	2978.38	6370.57
Every_Summer_Night	0	0	0	0	0	0	0	2301.88	5098.47	7859.35
Red_Sky	0	0	0	0	0	0	0	0	4238.76	8139.94
Afternoon	0	0	0	0	0	0	0	0	0	9334.8
If_I_could	0	0	0	0	0	0	0	0	0	0
Average: 5386.409										

Tabela 5.1: Tabela 5.2: Distância entre peças da colecção de Jazz

Jazz&Rock										
	Another_Brick_In_The_Wall	Murders_In_The_Rue_Morgue	Black_Dog	Open_Your_Heart	Moonchild	Good_Times_Bad_Times	Wrathchild	The_Four_Horsemen	Money	Animal
Follow_Me	4495.42	7913.19	6591.32	7100.87	5638.32	7232.77	7537.03	12591.3	7862.09	7348.87
Are_We_There_Yet	5876.8	11234.65	9621.09	9527.64	8138.25	10217.14	11033.26	15188.18	8937.41	10742.81
And_Then_I_Knew	11599.81	15057.35	12473.6	12606.66	12931.46	13027.78	14279.4	19234.06	12930.1	13848.8
James	6130.32	10904.56	9255.74	9101.62	8041.38	9573.07	10451.48	15307.77	9099.7	10127.69
The_First_Circle	10201.62	15684.51	14194.41	14133.95	12746.07	14781.85	15535.08	19337.95	13390.25	15286.25
Dream_Of_The_Return	7471.73	11185.99	8947.76	8351.36	8572.54	8682.64	10176.55	16108.6	9031.24	9652.52
Every_Summer_Night	10464.1	13982.55	11495.97	11598.88	11826.02	12015.83	13217.54	18269.52	12026.55	12791.57
Red_Sky	8515.65	11981.82	9508.4	9875.11	9829.24	10414.86	11431.76	16027.34	10161.28	11053.18
Afternoon	8212.91	10361.38	7475.59	6993.46	8407.19	7236.84	9047.8	15344.07	8461.58	8398
If_I_could	9732.27	15217.5	14017.75	13357.5	12061.99	13813.15	14774.45	19494.09	12869.51	14461.66
Average: 11205.094										

Tabela 5.2: Distância entre peças da colecção de Jazz e Rock

Jazz&Disco										
	Seven_seconds_away	She_Drives_Me_Crazy	Chery_Chery_Lady	Drive	Nirvana	Let_a_Boy_Cry	Wake_Me_Up_Before_You_Go_Go	Live_Is_Life	Brother_Louie	Little_Bad_Gir
Follow_Me	10189.48	9270.67	6201.85	6522.51	3538.64	11405.51	11101.3	5580.31	8537.34	8487.88
Are_We_There_Yet	11845.5	11142.61	8205.68	9576.18	5800.4	12684.44	12394.2	8412.24	10577.44	10457.99
And_Then_I_Knew	14642.51	13484.97	9140.37	12969	10755.6	14416.93	14784.12	10736.72	12904.02	12023.84
James	11839.73	10743.68	7428.48	9104.19	5324.79	12465.92	12128.77	7786.3	10126.1	9868.18
The_First_Circle	16142.24	15546.13	12225.89	14240.8	10249.98	16868.28	16592.51	12895.08	15027.07	14701.95
Dream_Of_The_Return	11451.64	9563.34	5632.49	8691.41	5678.68	11161.79	10946.05	6827.63	8930.25	8310.09
Every_Summer_Night	14212.88	12615.38	8278.97	11955.69	9601.13	14029.64	13980.82	9746.68	12023.7	11191.4
Red_Sky	12647.68	11122.4	7083.2	10151.6	8066.29	12837.99	12620.44	8070.76	10514.74	9892.04
Afternoon	10743.39	7900.37	3271.51	7597.6	6395.31	9870.99	9528.74	5117.07	7232.76	6294.5
If_I_could	15401.1	14714.11	11748.08	13438.17	8893.24	15848.21	15617.23	12404.36	14206.45	13939.01
Average: 10670.953										

Tabela 5.3: Distância entre peças da colecção de Jazz e Disco/Pop

Rock										
	Another_Brick_In_The_Wall	Murders_In_The_Rue_Morgue	Black_Dog	Open_Your_Heart	Moonchild	Good_Times_Bad_Times	Wrathchild	The_Four_Horsemen	Money	Animal
Another_Brick_In_The_Wall	0	6596.46	6203.35	6466.6	3060.39	7292	7231.28	9946.5	5550.46	7171.54
Murders_In_The_Rue_Morgue	0	0	3841.78	5888.78	4026.78	5204.45	2496.68	6088.96	7725.26	3298.82
Black_Dog	0	0	0	3138.11	3892.17	3253.53	3323.81	8333.76	5461.14	2957.29
Open_Your_Heart	0	0	0	0	4355.87	2232.48	4694.28	10420.39	3967.22	3790.75
Moonchild	0	0	0	0	0	4860.01	4526.72	8219.44	4685.29	4468.87
Good_Times_Bad_Times	0	0	0	0	0	0	3224.7	10555.69	6009.45	2188.47
Wrathchild	0	0	0	0	0	0	0	8370.61	7645.62	1050.71
The_Four_Horsemen	0	0	0	0	0	0	0	0	10501.13	9021.07
Money	0	0	0	0	0	0	0	0	0	7050.79
Animal	0	0	0	0	0	0	0	0	0	0
Average: 5561.987										

Tabela 5.4: Distância entre peças da colecção de Rock

Rock&Disco										
	Seven_seconds_away	She_Drives_Me_Crazy	Chery_Chery_Lady	Drive	Nirvana	Let_a_Boy_Cry	Wake_Me_Up_Before_You_Go_Go	Live_Is_Life	Brother_Louie	Little_Bad_Girl
Another_Brick_In_The_Wall	9469.29	8737.09	8153.35	5996.05	3599.58	11213.77	10134.51	6486.62	8255.16	9003.48
Murders_In_The_Rue_Morgue	8719.33	8358.59	8775	3805.26	6753.83	11004.22	10432.28	5968.01	7783.22	8718.71
Black_Dog	7721.12	5424.72	5496.35	2352.03	5993.86	8614.89	7698.41	2812.39	4771.47	5474.3
Open_Your_Heart	7757.09	2750.75	4389.27	2644.19	5495.4	7021.52	4955.82	2473.23	2112.71	3204.64
Moonchild	7835.27	6897.89	7234.79	3324.71	3782.92	9570.92	8604.97	4835.05	6370.2	7386.53
Good_Times_Bad_Times	7616.1	4173.2	4630.9	1850.72	5549.74	7347.64	6488.35	2478.09	3393.86	3960.61
Wrathchild	8209.22	7041.15	7127.17	2446.3	6155.85	9668.29	9295.9	4490.65	6352.77	7042.49
The_Four_Horsemen	11816.16	12273.14	13797.03	9162.34	11801.99	14695.61	13754.36	11072.08	12019.86	13248.62
Money	8026.82	4630.52	6771.77	5484.82	6312.72	7626.27	5263.72	5540.37	4631.24	5810.83
Animal	8000.86	6069.14	6259.81	1796.38	5859.75	8926.91	8352.72	3676.99	5363.65	6020.63
Average: 6857.609										

Tabela 5.5: Distância entre peças da colecção de Rock e Disco/Pop

Disco										
	Seven_seconds_away	She_Drives_Me_Crazy	Chery_Chery_Lady	Drive	Nirvana	Let_a_Boy_Cry	Wake_Me_Up_Before_You_Go_Go	Live_Is_Life	Brother_Louie	Little_Bad_Girl
Seven_seconds_away	0	8609.38	9108.53	6836.23	8595.05	5287.29	9835.06	7912.82	8116.74	8832.16
She_Drives_Me_Crazy	0	0	4915.79	5161.23	7728.33	6099.59	2475.05	4432.86	928.21	2090.13
Chery_Chery_Lady	0	0	0	5478.67	6237.49	7547.14	6811.09	2830.32	4198.71	3138.43
Drive	0	0	0	0	4800.76	7753.22	7377.55	2816.39	4421.2	5254.88
Nirvana	0	0	0	0	0	9706.59	9367.49	5002.72	7054.24	7440.56
Let_a_Boy_Cry	0	0	0	0	0	0	6479.3	7631.42	5944.86	6215.22
Wake_Me_Up_Before_You_Go_Go	0	0	0	0	0	0	0	6783.18	3335.65	4011.29
Live_Is_Life	0	0	0	0	0	0	0	0	3568.56	3589.53
Brother_Louie	0	0	0	0	0	0	0	0	0	1630.36
Little_Bad_Girl_(feat._Taio_Cruz	0	0	0	0	0	0	0	0	0	0
Average: 5853.139										

Tabela 5.6: Distância entre peças da colecção de Disco/Pop

Resumidamente temos, então, os seguintes resultados:

Intra Class Average Distance	Inter Class Average Distance :	Inter - Intra
5600.511	9577.885	3977.373

Para a colecção A (peças correspondentes no Appendix A), na segunda experiência, foram utilizados os seguintes parâmetros:

- **Feature Set 1**
- **K2 = 10** (número de centroids)
- **K1 = 1000** (número *features* extraídas por cada peça)

Para esta, esta experiência, temos resumidamente os seguintes resultados:

Inter Class Average Distance	Intra Class Average Distance	Inter - Intra
6167.199	9238.538	3071.339

Para a colecção B (peças correspondentes no Appendix B), na primeira experiência, foram utilizados os seguintes parâmetros:

- **Feature Set 2**
- **K2 = 5** (número de centroids)
- **K1 = 1000** (número *features* extraídas por cada peça)

Para esta, esta experiência, temos resumidamente os seguintes resultados:

Intra Class Average Distance	Inter Class Average Distance :	Inter - Intra :
6965.363	9754.146	2788.783

Para a colecção B (peças correspondentes no Appendix B), na segunda experiência, foram utilizados os seguintes parâmetros:

- **Feature Set 2**
- **K2 = 10** (número de centroids)
- **K1 = 1000** (número *features* extraídas por cada peça)

Para esta, esta experiência, temos resumidamente os seguintes resultados:

Intra Class Average Distance	Inter Class Average Distance :	Inter - Intra :
6909.823	9576.051	2666.228

Para a colecção B (peças correspondentes no Appendix B), na segunda experiência, foram utilizados os seguintes parâmetros:

- **Feature Set 2**
- **K2 = 100** (número de centroids)
- **K1 = 1000** (número *features* extraídas por cada peça)

Para esta, esta experiência, temos resumidamente os seguintes resultados:

Intra Class Average Distance	Inter Class Average Distance :	Inter - Intra :
6078.984	5954.400	-124.583

5.2.6 Classificação de Género – Avaliação

A classificação do género musical foi avaliada com base nas bases de dados criadas anteriormente (Apêndice A e Apêndice B). Esta classificação, desenvolvida para este projecto, tenta "emergir" os diferentes géneros musicais a partir de uma base de dados de

músicas, com peças de áudio de diferentes géneros musicais, através do cálculo da similaridade entre as mesmas. Para testar se o sistema de cálculo de similaridade foi bem sucedido na sua função, foram extraídas as *features* e geradas as assinaturas para cada uma das peças de áudio para depois, serem comparadas consoante o seu grau de similaridade.

No ponto anterior, foram disponibilizados os resultados. Com a excepção da primeira experiência, onde foi detalhadamente exposto todas as distâncias entre todas as peças de todas as colecções, nas restantes, apesar terem sido calculados na mesma todas as distâncias entre todas as peças de todas as colecções, apenas a diferença entre as médias das peças do mesmo género musical e as médias entre peças de géneros diferentes foram expostas. Para os resultados serem satisfatórios, a distância média entre peças do mesmo género deve ser, obviamente, inferior à distância média entre peças géneros diferentes, por exemplo: as distâncias entre as peças de Jazz devem inferiores às distancias entre as peças de Jazz e Disco/Pop, ou seja, quanto menor forem essas distâncias, maior é, então, o grau de similaridade.

Concluímos então, através do gráfico 5.2, que as distâncias médias entre peças do mesmo género são bastante inferiores que as distâncias médias entre peças de géneros diferentes, o que significa que o sistema de cálculo de similaridade construído conseguiu reconhecer peças do mesmo género musical entre vários géneros, sendo por isso, um bom resultado.

Em relação à experiencias com instrumentos, realizadas exactamente com os parâmetros iguais às experiências feitas com peças musicais, os resultados a comparação entre as médias (distância média entre peças do instrumento e a distância média entre peças de instrumentos diferentes) encontram-se descritos no gráfico 5.3. Neste caso, os resultados não são tão satisfatórios como nas peças de áudio musicais, tirando o caso das peças de clarinete, violino e Flauta cujas médias, entre as distâncias de peças de áudio da mesma colecção, como indica no gráfico, são inferiores às médias entre peças de áudio de colecções diferentes.

Apesar deste resultado, a média final (Inter Class Average Distance -Intra Class Average Distance), é positiva, o que contudo indica um resultado satisfatório.

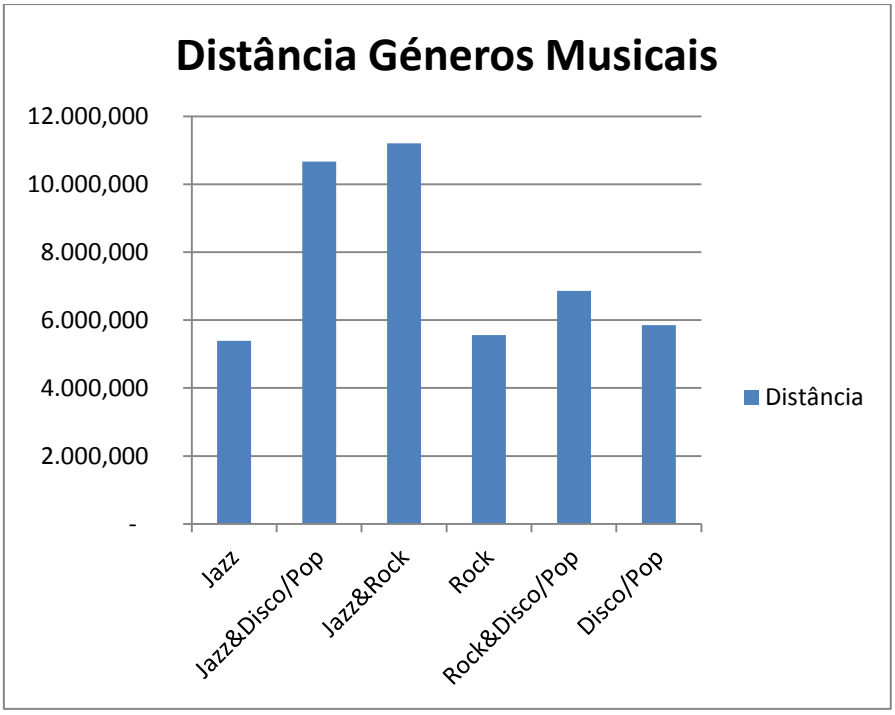

Figura 5.2: Média de distâncias entre géneros musica

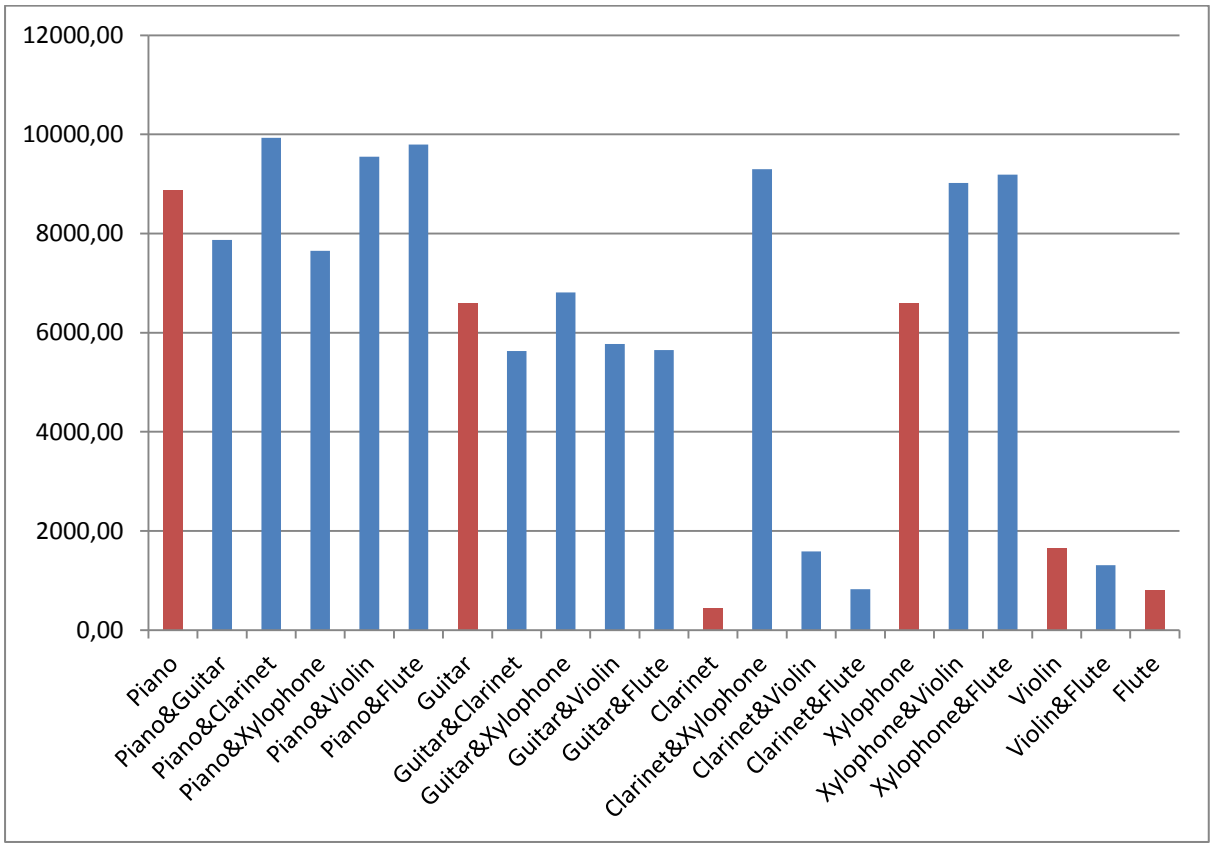

Figura 5.3: Média de distâncias entre Instrumentos

5.2.7 Especificidade de Sons

Dito anteriormente, de modo a não nos fixarmos apenas em música, foram criadas outras bases de dados de modo a testar o sistema noutros tipos de áudio para calcular a sua semelhança. Neste ponto, ao contrário do anterior, pretendemos avaliar o sistema no que diz respeito ao cálculo de similaridade entre peças de áudio a nível não quantitativo mas sim qualitativo. Esta avaliação envolve seleccionar apenas uma peça de áudio, e ordenar as peças mais próximas com base no grau de semelhança entre as mesmas.

Nesta avaliação, irão ser utilizados quatro episódios (Ver Apêndice D) e para cada um será mostrado: o Top20 peças de áudio mais próximas e o Top10 de peças mais próximas de uma peça determinada de áudio que corresponde a um evento conhecido. Os resultados sereão apresentados em tabelas onde ficam descritos: os dois *samples* a comparar, o instante inicial de cada um, a distância entre os mesmos e a descrição do som de cada um, indicando com uma cor diferente consoante o quanto são semelhantes:

- **Verde** – os samples, após a sua audição, possuem um elevado número de semelhanças entre eles.
- **Amarelo** - os samples, após a sua audição, possuem umas algumas semelhanças entre eles.
- **Vermelho** - os samples, após a sua audição, não possuem qualquer tipo de semelhança.

Após esta avaliação, individual, a cada episódio por samples iguais, irá ser feita uma experiência de modo a avaliar se existem sons semelhantes entre os três episódios: `The.Walking.Dead.S02E07, Lost.S04E11 e Prison.BreakS02E21.`

Para cada episódio os parâmetros de extracção e agregação de *features* foram os seguintes:

- **Feature Set 1**
- **K2 = 400** (número de centroids)
- **K1 = 200** (número *features* extraídas por cada peça)

Temos então os resultados para o episódio - The.Walking.Dead.S02E07:

- **Top 20**

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
TWD.S02E07.947.wav	39:25	TWD.S02E07.948.wav	39:27.5	2.45	(musica de fundo-musica de fundo)
TWD.S02E07.107.wav	4:25	TWD.S02E07.108.wav	4:27.5	2.83	(musica de fundo-musica de fundo)
TWD.S02E07.802.wav	33:22.5	TWD.S02E07.803.wav	33:25	2.83	(musica de fundo-musica de fundo)
TWD.S02E07.373.wav	15:30	TWD.S02E07.372.wav	15:27.5	4.24	(musica de fundo-musica de fundo)
TWD.S02E07.407.wav	16:55	TWD.S02E07.372.wav	15:27.5	6.48	(musica de fundo-musica de fundo)
TWD.S02E07.374.wav	15:32.5	TWD.S02E07.372.wav	15:27.5	6.93	(musica de fundo-musica de fundo)
TWD.S02E07.374.wav	15:32.5	TWD.S02E07.407.wav	16:55	7.75	(musica de fundo-musica de fundo)
TWD.S02E07.374.wav	15:32.5	TWD.S02E07.373.wav	15:30	8.83	(musica-musica)
TWD.S02E07.373.wav	15:30	TWD.S02E07.407.wav	16:55	9.49	(musica-musica)
TWD.S02E07.889.wav	37:0	TWD.S02E07.890.wav	37:2.5	11.4	(pessoa a falar-pessoa a falar)
TWD.S02E07.563.wav	23:22.5	TWD.S02E07.578.wav	23:37.5	12.58	(musica de fundo-musica de fundo)
TWD.S02E07.1008.wav	41:57.5	TWD.S02E07.984.wav	40:57.5	13.27	(musica de fundo-musica de fundo)
TWD.S02E07.110.wav	4:32.5	TWD.S02E07.109.wav	4:30	15.36	(musica de fundo-musica de fundo)
TWD.S02E07.597.wav	24:50	TWD.S02E07.598.wav	24:52.5	15.43	(musica de fundo-musica de fundo)
TWD.S02E07.51.wav	2:5	TWD.S02E07.890.wav	37:2.5	16.79	(pessoa a falar-musica de fundo)
TWD.S02E07.51.wav	2:5	TWD.S02E07.889.wav	37:0	17.66	(pessoa a falar-musica de fundo)
TWD.S02E07.106.wav	4:22.5	TWD.S02E07.107.wav	4:25	17.66	(musica de fundo-musica de fundo)
TWD.S02E07.597.wav	24:50	TWD.S02E07.578.wav	23:37.5	18.65	(musica de fundo-musica de fundo)
TWD.S02E07.510.wav	21:12.5	TWD.S02E07.507.wav	21:5	18.65	(musica de fundo-musica de fundo)
TWD.S02E07.597.wav	24:50	TWD.S02E07.563.wav	23:25	18.81	(musica de fundo-musica de fundo)

Tabela 5.7: Top 20 pares de samples mais semelhantes episódio The.Walking.Dead.S02E07

- **Top 10 mais próximas da peça** `The.Walking.Dead.S02E07.917.wav`

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
TWD.S02E07.57.wav	2:20	TWD.S02E07.917.wav	38:10	21.7	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.56.wav	2:17.5	TWD.S02E07.917.wav	38:10	31.62	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.102.wav	4:12.5	TWD.S02E07.917.wav	38:10	50.16	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.103.wav	4:15	TWD.S02E07.917.wav	38:10	77.36	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.943.wav	39:15	TWD.S02E07.917.wav	38:10	83.55	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.917.wav	38:10	TWD.S02E07.941.wav	39:10	86.52	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.917.wav	38:10	TWD.S02E07.61.wav	2:30	92.96	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.917.wav	38:10	TWD.S02E07.920.wav	38:17.5	101.22	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.101.wav	4:10	TWD.S02E07.917.wav	38:10	106 6	(grunhidos Zombie-grunhidos Zombie)
TWD.S02E07.917.wav	38:10	TWD.S02E07.1038.wav	43:12.5	110.3	(grunhidos Zombie-musica fundo)

Tabela 5.8: Top 10 samples mais semelhantes do sample The.Walking.Dead.S02E07.917.wav

Para a o episódio `L.S04E11` temos os seguintes resultados:

- **Top 20**

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
L.S04E11.360.wav	14:57.5	L.S04E11.371.wav	15:25	5.1	(musica de fundo - musica de fundo)
L.S04E11.946.wav	39:22.5	L.S04E11.941.wav	39:10	6.48	(musica de fundo - musica de fundo)
L.S04E11.858.wav	35:42.5	L.S04E11.856.wav	35:37.5	8.49	(musica de fundo - musica de fundo)
L.S04E11.858.wav	35:42.5	L.S04E11.857.wav	35:40	9.8	(musica de fundo - musica de fundo)
L.S04E11.990.wav	41:12.5	L.S04E11.989.wav	41:10	11.14	(insecto - insecto)
L.S04E11.366.wav	15:12.5	L.S04E11.371.wav	15:25	11.58	(musica de fundo - musica de fundo)
L.S04E11.366.wav	15:12.5	L.S04E11.360.wav	14:57.5	11.58	(musica de fundo - musica de fundo)
L.S04E11.856.wav	35:37.5	L.S04E11.857.wav	35:40	11.75	(musica de fundo - musica de fundo)
L.S04E11.150.wav	6:12.5	L.S04E11.151.wav	6:15	12.33	(musica de fundo - musica de fundo)
L.S04E11.886.wav	36:52.5	L.S04E11.887.wav	36:55	12.65	(musica de fundo - musica de fundo)
L.S04E11.10.wav	0:22.5	L.S04E11.11.wav	0:25	13.49	(musica de fundo - musica de fundo)
L.S04E11.1003.wav	41:45	L.S04E11.1001.wav	41:40	13.93	(musica de fundo - musica de fundo)
L.S04E11.942.wav	39:12.5	L.S04E11.941.wav	39:10	15.43	(musica de fundo - musica de fundo)
L.S04E11.857.wav	35:40	L.S04E11.855.wav	35:35	15.87	(musica de fundo - musica de fundo)
L.S04E11.361.wav	15:0	L.S04E11.371.wav	15:25	15.94	(musica de fundo - musica de fundo)
L.S04E11.998.wav	41:32.5	L.S04E11.1000.wav	41:37.5	16.31	(musica de fundo - musica de fundo)
L.S04E11.361.wav	15:0	L.S04E11.365.wav	15:10	16.49	(musica de fundo - musica de fundo)
L.S04E11.784.wav	32:37.5	L.S04E11.887.wav	36:55	16.79	(musica de fundo - musica de fundo)

L.S04E11.835.wav	34:45	L.S04E11.834.wav	34:42.5	17.49	(musica de fundo - musica de fundo)
L.S04E11.163.wav	6:45	L.S04E11.162.wav	6:42.5	17.66	(musica de fundo - musica de fundo)
L.S04E11.366.wav	15:12.5	L.S04E11.361.wav	15:0	17.72	(musica de fundo - musica de fundo)

Tabela 5.9: Top 20 pares de samples mais semelhantes do episódio Lost.S04E11

- **Top 10 mais próximas da peça** `Lost.S04E11.989.wav`:

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
L.S04E11.999.wav	41:35	L.S04E11.1003.wav	41:45	24.86	(insecto-insecto)
L.S04E11.999.wav	41:35	L.S04E11.1001.wav	41:40	33.62	(insecto-insecto)
L.S04E11.999.wav	41:35	L.S04E11.996.wav	41:27.5	64.51	(insecto-insecto)
L.S04E11.999.wav	41:35	L.S04E11.1000.wav	41:37.5	65.77	(insecto-insecto)
L.S04E11.1002.wav	41:42.5	L.S04E11.999.wav	41:35	66.59	(insecto-insecto)
L.S04E11.998.wav	41:32.5	L.S04E11.999.wav	41:35	81.47	(insecto-insecto)
L.S04E11.999.wav	41:35	L.S04E11.997.wav	41:30	142 5	(insecto-insecto)
L.S04E11.999.wav	41:35	L.S04E11.740.wav	30:47.5	145.96	(insecto-pessoa a falar)
L.S04E11.79.wav	3:15	L.S04E11.999.wav	41:35	146.3	(insecto-insecto)
L.S04E11.78.wav	3:12.5	L.S04E11.999.wav	41:35	149.32	(insecto-insecto)

Tabela 5.10: Top 10 samples mais semelhantes do sample Lost.S04E11.989.wav

Para a o episódio `Prison.BreakS02E21` temos os seguintes resultados:

- **Top 20**

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
PB.S02E21.347.wav	14:25	PB.S02E21.337.wav	14:0	1.41	(musica de fundo-musica de fundo)
PB.S02E21.706.wav	29:22.5	PB.S02E21.707.wav	29:25	2.10	(musica de fundo-musica de fundo)
PB.S02E21.1025.wav	42:40	PB.S02E21.1023.wav	42:35	21.54	(musica de fundo-musica de fundo)
PB.S02E21.347.wav	14:25	PB.S02E21.346.wav	14:22.5	21.63	(musica de fundo-musica de fundo)
PB.S02E21.346.wav	14:22.5	PB.S02E21.337.wav	14:0	21.77	(musica de fundo-musica de fundo)
PB.S02E21.269.wav	11:10	PB.S02E21.268.wav	11:7.5	23.49	(musica de fundo-musica de fundo)
PB.S02E21.948.wav	39:27.5	PB.S02E21.949.wav	39:30	25.41	(musica de fundo-musica de fundo)
PB.S02E21.334.wav	13:52.5	PB.S02E21.339.wav	14:5	26.8	(musica de fundo-musica de fundo)
PB.S02E21.472.wav	19:37.5	PB.S02E21.474.wav	19:42.5	29.26	(musica de fundo-musica de fundo)
PB.S02E21.341.wav	14:10	PB.S02E21.342.wav	14:12.5	29.63	(musica de fundo-musica de fundo)
PB.S02E21.453.wav	18:50	PB.S02E21.454.wav	18:52.5	30.72	(musica de fundo-musica de fundo)

PB.S02E21.155.wav	6:25	PB.S02E21.156.wav	6:27.5	32.59	(musica de fundo-musica de fundo)
PB.S02E21.262.wav	10:52.5	PB.S02E21.264.wav	10:57.5	32.68	(musica de fundo-musica de fundo)
PB.S02E21.475.wav	19:45	PB.S02E21.472.wav	19:37.5	32.77	(musica de fundo-musica de fundo)
PB.S02E21.818.wav	34:2.5	PB.S02E21.809.wav	33:40	33.17	(musica de fundo-musica de fundo)
PB.S02E21.365.wav	15:10	PB.S02E21.367.wav	15:15	34.67	(musica de fundo-musica de fundo)
PB.S02E21.112.wav	4:37.5	PB.S02E21.656.wav	27:17.5	35.13	(musica de fundo-musica de fundo)
PB.S02E21.274.wav	11:22.5	PB.S02E21.277.wav	11:30	35.24	(musica de fundo-musica de fundo)
PB.S02E21.332.wav	13:47.5	PB.S02E21.340.wav	14:7.5	35.33	(musica de fundo-musica de fundo)
PB.S02E21.351.wav	14:35	PB.S02E21.352.wav	14:37.5	35.81	(musica de fundo-musica de fundo)

Tabela 5.11: Top 20 pares de samples mais semelhantes do episódio Prison.BreakS02E21

- **Top 10 mais próximas da peça** `Prison.BreakS02E21.399.wav`:

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
PB.S02E21.399.wav	16:35	PB.S02E21.778.wav	32:22.5	38.24	(silencio&som forte - silencio&som forte)
PB.S02E21.399.wav	16:35	PB.S02E21.1018.wav	42:22.5	40.12	(silencio&som forte - silencio&som forte)
PB.S02E21.399.wav	16:35	PB.S02E21.566.wav	23:7.5	47.24	(silencio&som forte - silencio&som forte)
PB.S02E21.1033.wav	43:0	PB.S02E21.399.wav	16:35	49 2	(silencio&som forte - silencio&som forte)
PB.S02E21.1034.wav	43:2.5	PB.S02E21.399.wav	16:35	53.2	(silencio&som forte - silencio&som forte)
PB.S02E21.399.wav	16:35	PB.S02E21.219.wav	9:5	55.12	(silencio&som forte - silencio&som forte)
PB.S02E21.398.wav	16:32.5	PB.S02E21.399.wav	16:35	64.54	(silencio&som forte - silencio&som forte)
PB.S02E21.567.wav	23:35	PB.S02E21.399.wav	16:35	65.13	(silencio&som forte - silencio&som forte)
PB.S02E21.779.wav	32:25	PB.S02E21.399.wav	16:35	72	(musica de fundo - silencio&som forte)
PB.S02E21.399.wav	16:35	PB.S02E21.218.wav	9:2.5	75.74	(silencio&som forte - silencio&som forte)

Tabela 5.12: Top 10 samples mais semelhantes do sample Prison.BreakS02E21.399.wav

Para a o episódio `24.S01E01` temos os seguintes resultados:

- **Top 20**

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
24.S01E01.325.wav	8:6	24.S01E01.326.wav	8:7.5	5.48	(musica-musica)
24.S01E01.324.wav	8:4.5	24.S01E01.326.wav	8:7.5	6.50	(musica-musica)
24.S01E01.155.wav	3:51	24.S01E01.154.wav	3:49.5	7.71	(pessoa a falar-pessoa a falar)
24.S01E01.1115.wav	27:51	24.S01E01.1119.wav	27:57	7.48	(musica de fundo-musica de fundo)
24.S01E01.1120.wav	27:58.5	24.S01E01.1115.wav	27:51	8.12	(musica de fundo-musica de fundo)
24.S01E01.1078.wav	26:55.5	24.S01E01.826.wav	20:37.5	8.60	(musica de fundo-musica de fundo)
24.S01E01.935.wav	23:21	24.S01E01.934.wav	23:19.5	9.38	(musica de fundo-musica de fundo)

24.S01E01.1195.wav	29:51	24.S01E01.348.wav	8:40.5	9.90	(musica de fundo-musica de fundo)
24.S01E01.325.wav	8:6	24.S01E01.324.wav	8:4.5	10.58	(musica-musica)
24.S01E01.893.wav	22:18	24.S01E01.460.wav	11:28.5	11.10	(musica de fundo-musica de fundo)
24.S01E01.1123.wav	28:3	24.S01E01.1119.wav	27:57	11.83	(musica de fundo-musica de fundo)
24.S01E01.1123.wav	28:3	24.S01E01.1120.wav	27:58.5	12 8	(musica de fundo-musica de fundo)
24.S01E01.1120.wav	27:58.5	24.S01E01.1119.wav	27:57	12.41	(musica de fundo-musica de fundo)
24.S01E01.1123.wav	28:3	24.S01E01.1115.wav	27:51	13.27	(musica de fundo-musica de fundo)
24.S01E01.644.wav	16:4.5	24.S01E01.643.wav	16:3	14.14	(musica-musica)
24.S01E01.307.wav	7:39	24.S01E01.315.wav	7:51	14.70	(musica-musica)
24.S01E01.643.wav	16:3	24.S01E01.642.wav	16:1.5	15.17	(musica-musica)
24.S01E01.1102.wav	27:31.5	24.S01E01.1100.wav	27:28.5	15.87	(musica de fundo-musica de fundo)
24.S01E01.309.wav	7:42	24.S01E01.315.wav	7:51	16.12	(musica-musica)
24.S01E01.1203.wav	30:3	24.S01E01.1202.wav	30:1.5	17.44	(musica de fundo-musica de fundo)

Tabela 5.13: Top 20 pares de samples mais semelhantes do episódio 24.S01E01

- **Top 10 mais próximas da peça** `24.S01E01.460.wav`:

Sample1	Inst Ini	Sample2	Inst Ini	Dist	Descrição
24.S01E01.893.wav	22:18.0	24.S01E01.460.wav	11:28.5	14.35	(contagem - contagem)
24.S01E01.896.wav	22:22.5	24.S01E01.460.wav	11:28.5	52.02	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.459.wav	11:27.0	53.5	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.892.wav	22:16.5	65.05	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.464.wav	11:34.5	68.13	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.894.wav	22:19.5	69.44	(contagem - contagem)
24.S01E01.895.wav	22:21.0	24.S01E01.460.wav	11:28.5	72.47	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.1209.wav	30:12.0	72.62	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.1230.wav	30:43.5	73.8	(contagem - contagem)
24.S01E01.460.wav	11:28.5	24.S01E01.851.wav	21:15.0	80.06	(contagem - pessoa a falar)

Tabela 5.14: Top 10 samples mais semelhantes do sample 24.S01E01.460.wav

Após a exposição dos resultados individuais de cada episódio, temos então os resultados para o *merge* dos episódios: `The.Walking.Dead.S02E0`, `Prison.BreakS02E21` e `The.Walking.Dead.S02E07`.

- **Top 20**

Sample1	Sample2	Distância
Lost.S04E11.927.wav	Lost.S04E11.933.wav	1.41
Lost.S04E11.1000.wav	Lost.S04E11.998.wav	1.43
The.Walking.Dead.S02E07.802.wav	The.Walking.Dead.S02E07.803.wav	1.45
Lost.S04E11.989.wav	Lost.S04E11.990.wav	2.14
Lost.S04E11.998.wav	Lost.S04E11.1003.wav	2.41
Lost.S04E11.1000.wav	Lost.S04E11.1003.wav	2.72
Lost.S04E11.997.wav	Lost.S04E11.998.wav	2.44
Lost.S04E11.1000.wav	Lost.S04E11.997.wav	2.48
Lost.S04E11.930.wav	Lost.S04E11.933.wav	2.82
Lost.S04E11.929.wav	Lost.S04E11.931.wav	3.16
The.Walking.Dead.S02E07.543.wav	The.Walking.Dead.S02E07.560.wav	3.16
Lost.S04E11.997.wav	Lost.S04E11.1003.wav	3.74
Lost.S04E11.927.wav	Lost.S04E11.930.wav	4.24
Lost.S04E11.1002.wav	Lost.S04E11.995.wav	4.69
The.Walking.Dead.S02E07.374.wav	The.Walking.Dead.S02E07.372.wav	4.89
Lost.S04E11.934.wav	Lost.S04E11.933.wav	5.09
Lost.S04E11.930.wav	Lost.S04E11.934.wav	5.09
The.Walking.Dead.S02E07.543.wav	The.Walking.Dead.S02E07.542.wav	5.47
Lost.S04E11.927.wav	Lost.S04E11.934.wav	5.65
Lost.S04E11.991.wav	Lost.S04E11.990.wav	5.83
The.Walking.Dead.S02E07.372.wav	The.Walking.Dead.S02E07.373.wav	5.83
The.Walking.Dead.S02E07.804.wav	The.Walking.Dead.S02E07.797.wav	5.83
The.Walking.Dead.S02E07.560.wav	The.Walking.Dead.S02E07.557.wav	6.16

Tabela 5.15: Top 20 pares de samples mais semelhantes dos três episódios

- **Top 50 mais próximas entre os episódios:** `The.Walking.Dead.S02E0, Prison.BreakS02E21 e The.Walking.Dead.S02E07.`

Sample1	Sample2	Dist	Descrição
Prison.BreakS02E21.691.wav	Lost.S04E11.653.wav	11.832	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.1001.wav	Lost.S04E11.839.wav	37.067	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.603.wav	Lost.S04E11.843.wav	39.648	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.603.wav	Lost.S04E11.842.wav	40.199	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.603.wav	Lost.S04E11.844.wav	41.109	(musica de fundo-musica de fundo)
Prison.BreakS02E21.659.wav	Lost.S04E11.616.wav	41.133	(musica de fundo-musica de fundo)
Prison.BreakS02E21.688.wav	Lost.S04E11.210.wav	41.205	(musica de fundo-musica de fundo)
Prison.BreakS02E21.277.wav	Lost.S04E11.684.wav	41.376	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.999.wav	Lost.S04E11.476.wav	41.593	(musica de fundo-musica de fundo)
Prison.BreakS02E21.5.wav	Lost.S04E11.384.wav	41.809	(pessoa a falar-pessoa a falar)
The.Walking.Dead.S02E07.957.wav	Lost.S04E11.115.wav	42.544	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.328.wav	Lost.S04E11.190.wav	42.661	(objecto a rachar - objecto a rachar)
Prison.BreakS02E21.11.wav	Lost.S04E11.949.wav	43.543	(diálogo - pessoa a falar)
The.Walking.Dead.S02E07.1001.wav	Lost.S04E11.840.wav	43.657	(musica de fundo-musica de fundo)
Prison.BreakS02E21.649.wav	Lost.S04E11.855.wav	43.931	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.148.wav	Lost.S04E11.314.wav	44.226	(musica de fundo-musica de fundo)
Prison.BreakS02E21.98.wav	Lost.S04E11.840.wav	45.033	(musica de fundo-musica de fundo)
Prison.BreakS02E21.524.wav	Lost.S04E11.297.wav	45.299	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.964.wav	Lost.S04E11.292.wav	45.607	(musica de fundo-pessoa a falar)
Prison.BreakS02E21.649.wav	Lost.S04E11.858.wav	45.912	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.1001.wav	Prison.BreakS02E21.98.wav	46.021	(musica de fundo-musica de fundo)
Prison.BreakS02E21.808.wav	Lost.S04E11.399.wav	46.151	(musica de fundo-musica de fundo)
Prison.BreakS02E21.922.wav	Lost.S04E11.580.wav	46.216	(pessoa a falar-pessoa a falar)
The.Walking.Dead.S02E07.286.wav	Prison.BreakS02E21.840.wav	46.411	(musica de fundo-musica de fundo)
Prison.BreakS02E21.918.wav	Lost.S04E11.581.wav	46.432	(pessoa a falar-pessoa a falar)
Prison.BreakS02E21.898.wav	Lost.S04E11.61.wav	46.454	(pessoa a falar-pessoa a falar)
Prison.BreakS02E21.74.wav	Lost.S04E11.981.wav	46.647	(musica de fundo-pessoa a falar)
The.Walking.Dead.S02E07.957.wav	Lost.S04E11.265.wav	47.116	(musica de fundo-musica de fundo)
Prison.BreakS02E21.641.wav	Lost.S04E11.628.wav	47.581	(pessoa a falar-diálogo)
Prison.BreakS02E21.1028.wav	Lost.S04E11.865.wav	47.791	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.961.wav	Lost.S04E11.616.wav	48.104	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.979.wav	Lost.S04E11.161.wav	48.165	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.950.wav	Prison.BreakS02E21.907.wav	48.249	(pessoa a falar - musica de fundo)
The.Walking.Dead.S02E07.551.wav	Lost.S04E11.779.wav	48.518	(pessoa a falar-diálogo)

The.Walking.Dead.S02E07.201.wav	Lost.S04E11.3.wav	48.559	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.926.wav	Prison.BreakS02E21.154.wav	48.662	(tiros - tiros)
Prison.BreakS02E21.91.wav	Lost.S04E11.654.wav	48.703	(silencio&som forte - silencio&som forte)
The.Walking.Dead.S02E07.148.wav	Lost.S04E11.319.wav	48.764	(pessoa a falar - musica de fundo)
Prison.BreakS02E21.640.wav	Lost.S04E11.505.wav	48.764	(pessoa a falar-pessoa a falar)
The.Walking.Dead.S02E07.604.wav	Lost.S04E11.844.wav	48.785	(musica de fundo-musica de fundo)
Prison.BreakS02E21.760.wav	Lost.S04E11.771.wav	48.844	(diálogo - diálogo)
Prison.BreakS02E21.116.wav	Lost.S04E11.616.wav	48.867	(musica de fundo-musica de fundo)
Prison.BreakS02E21.649.wav	Lost.S04E11.856.wav	48.867	(musica de fundo-musica de fundo)
The.Walking.Dead.S02E07.830.wav	Lost.S04E11.309.wav	49.173	(pessoa a falar-pessoa a falar)
Prison.BreakS02E21.670.wav	Lost.S04E11.126.wav	49.213	(pessoa a falar-pessoa a falar)
The.Walking.Dead.S02E07.493.wav	Lost.S04E11.217.wav	49.396	(pessoa a falar-diálogo)
Prison.BreakS02E21.759.wav	Lost.S04E11.113.wav	49.411	(diálogo - diálogo)
Prison.BreakS02E21.995.wav	Lost.S04E11.421.wav	49.537	(pessoa a falar-pessoa a falar)
The.Walking.Dead.S02E07.187.wav	Lost.S04E11.266.wav	49.578	(pessoa a falar - musica de fundo)
The.Walking.Dead.S02E07.189.wav	Lost.S04E11.845.wav	49.658	(pessoa a falar-diálogo)

Tabela 5.16: Top 50 pares de samples mais semelhantes entre os três episódios

5.2.8 Especificidade de Sons – Avaliação

Ao analisar os resultados de uma maneira geral, concluímos que foram muito satisfatórios. Em relação ao top20 peças de áudio com menor distância, ou seja, mais semelhantes verificamos que, na maior parte das vezes, as peças encontram-se muito próximas temporalmente. É aparentemente lógico que assim aconteça uma vez que todas as peças possuem um overlap de um e meio segundos, o que faz com que dada uma peça de áudio a próxima irá ter os últimos um e meio segundos da dada peça.

No entanto, só estes resultados não são por si só suficientemente satisfatórios. De modo a testar o sistema ainda com mais detalhe, previamente ouviram-se as peças de áudio, de cada episódio, e foi seleccionada uma peça de áudio, por cada episódio, na qual reconhecemos um som ou um padrão específico: no episódio da série Walking dead escolheu-se um som dos "grunhidos" dos zombies, no episódio da série Lost escolheu-se o som do barulho dos "insectos à noite", no episódio da série Prison Break foi escolhido um padrão de som onde existe silencio seguido de um forte som e para o episódio de 24 a "contagem", muito comum em todos os episódios da série. Para todos estes sons, como é visível nas tabelas de top 10 mais próximas dos três episódios, a percentagem de sucesso no que diz respeito ao grau de similaridade é quase 100%. De notar que, das dez peças mais próximas da peça dada como

input, algumas, encontram-se em diferentes instantes do episódio e não apenas próxima da peça dada como input o que significa que o sistema encontra sons, bem definidos, em diferentes intervalos de tempo

5.2.9 Simulações

Com base nos resultados anteriores, apesar de no caso do episódio da série Walking Dead e no episódio da série Prison Break os dez sons mais próximos estejam presentes em vários instantes do episódio, verificamos que no caso do episódio da série Lost apenas existe um som distante (Lost.S04E11.**78**.wav - Lost.S04E11.**999**.wav) do som dado como input.

Para termos mais certezas que o sistema reconhece correctamente determinados sons, foram construídas duas amostras de som `LostGunshoot.wav` e `Glass24.wav`, respectivamente:

1. Oito amostras de trinta segundos do episódio de `Lost`, intercaladas com oito amostras de três segundos de tiros de pistola.
2. Oito amostras de trinta segundos do episódio de `24`, intercaladas com oito amostras de três segundos de vidro a partir.

Estas amostras foram subdivididas em várias peças, cada uma com três segundos e overlap de um e meio segundos, e comparadas com outras duas colecções: a primeira foi comparada com uma colecção de vinte peças de áudio de vidro a partir, a segunda com uma colecção de vinte peças de áudio de tiros de armas.

Para cada colecção foram utilizados os seguintes parâmetros de extracção:

- **Feature Set 1**
- **K2 = 100** (número de centroids)
- **K1 = 1000** (número *features* extraídas por cada peça)

Para a colecção 1, obteve-se o seguinte gráfico:

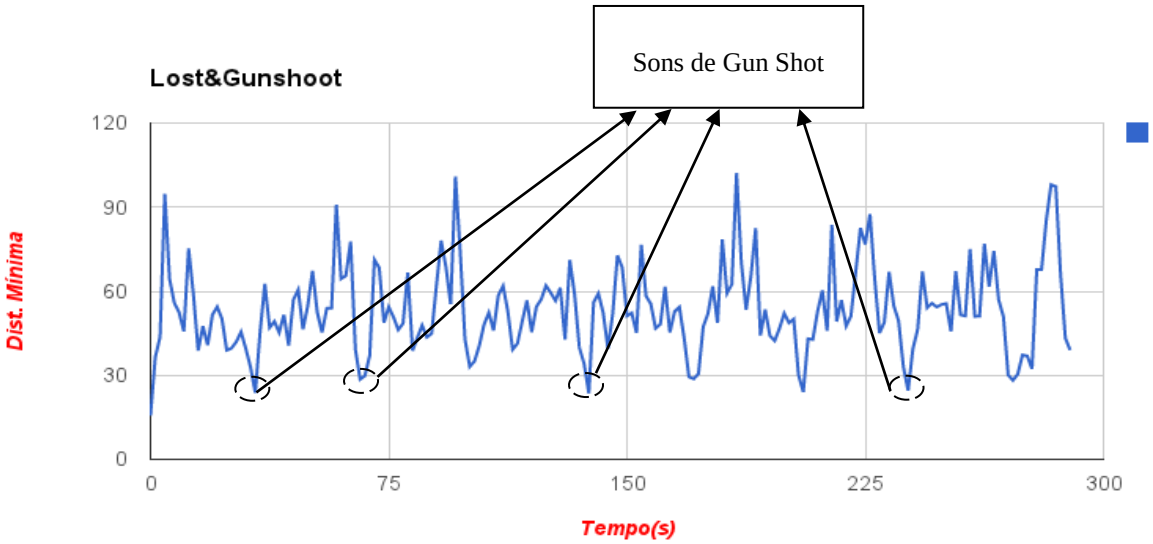

Figura 5.4: Variação dos samples de Gun Shots através do tempo

Para a colecção 2, obteve-se o seguinte gráfico:

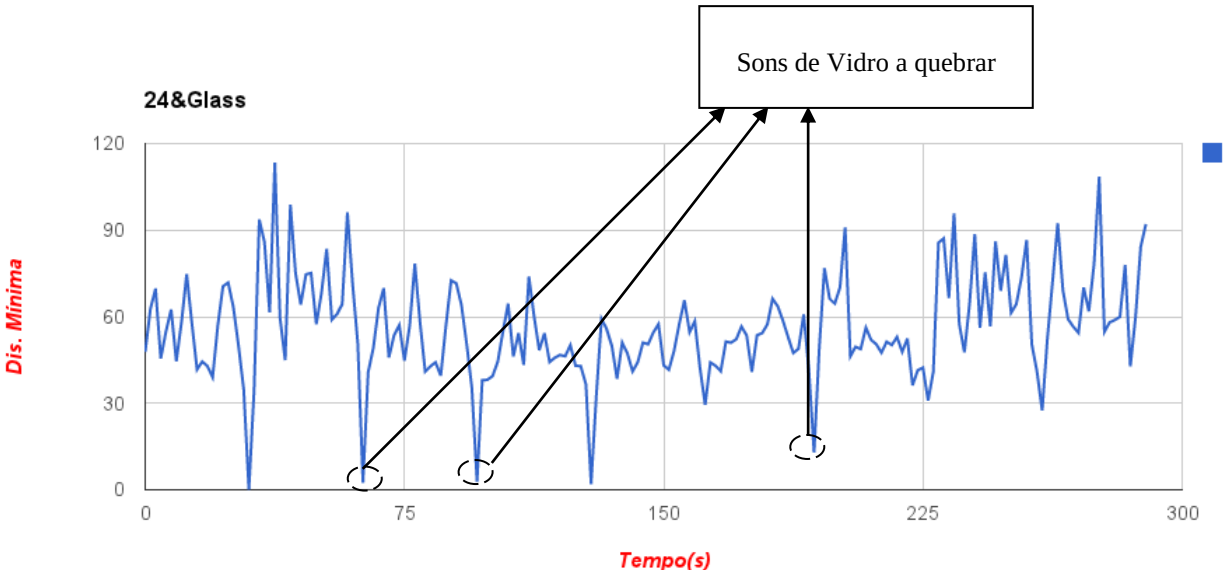

Figura 5.5Variação dos samples de vidro através do tempo

5.2.10 Simulações - Avaliação

Com base nos resultados obtidos, o sistema claramente reconheceu os sons de tiros e de vidro a partir. Nestes gráficos, observa-se que quando o sistema detecta o vidro a partir ou o tiro, intercalado entre trinta segundos de episódio, a distância é muito mais baixa pois estes instantes são comparados com colecções do mesmo tipo (vinte amostras de vidro a partir e vinte amostras de tiros).

Visto isto, conclui-se que o sistema reconhece com algum rigor eventos reconhecidos entre outro tipo de áudio.

Capítulo 6

Conclusões

Este capítulo culmina no término desta tese. Por fim, é então feito um resumo do trabalho concluído sendo as conclusões tiradas sobre o sucesso global do mesmo. Após este resumo, possíveis questões para futuras pesquisas são discutidas, finalizando esta tese.

6.1 Sumário e conclusões

No decorrer desta tese foi desenvolvido um interpretador de comandos, que é capaz de extrair informação de várias peças de áudio e de medir com precisão a semelhança dessas mesmas peças. Esse sistema analisa as características espectrais das peças de áudio e é capaz de extrair as *features*, das mesmas, a fim de comparar e de as classificar independentemente de quaisquer metadados que possam existir.

A sua implementação foi um sucesso, e permitiu desenvolver os dois principais mecanismos: extracção e cálculo de similaridade, com o auxilio de ferramentas como o Marsyas. O grau de sucesso destes mecanismos foi testado com vários tipos de peças de áudio, não apenas músicas, e os resultados revelaram-se prometedores. Tanto a nos testes de cálculo de género como na identificação de sons reconhecidos em séries, os resultados mostram que o sistema é bastante preciso no cálculo da similaridade.

6.2 Trabalho futuro

Existem vários possíveis caminhos para novas pesquisas em relação ao sistema desenvolvido. Embora, globalmente, o sistema atinja todas as suas metas, podem, obviamente, serem introduzidas novas melhorias em várias áreas. A adição de novas aplicações para a extensão das aplicações desenvolvidas, ou a integração deste sistema noutro sistema externo, são ambas possibilidades.

Concretamente, um dos maiores obstáculos para a usabilidade do sistema é o facto de que este só é capaz de analisar peças musicais em formato WAV. Um outro aspecto que pode, futuramente, ser alterado será o mecanismo de *clustering*, pois o WEKA que é utilizado revela-se lento quando em estão causa muitas peças de áudio.

Bibliografia

[1] Arman, F., Depommier, R., Hsun A. and Chiu, M.Y. "Content-based browsing of video sequences." *Proceedings of the second ACM international conference on Multimedia.* 1994 pp 97-103

[2] Wu, Y., Zhuang, Y. and Pan, Y. "Content-based video similarity model." *Proceedings of the eighth ACM international conference on Multimedia.* 2000. pp 465-467

[3] ISO/IEC 11172-3:1993, "*Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s - Part 3: Audio*."

[4] Gang, L., Akansu, A.N., Ramkumar, M. and Xuefei X. "On-line music protection and MP3 compression" *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, 2001.

[5] Lam, Calvin K. M. and Tan, Bernard C. Y. "The Internet is changing the music industry." *Communications of the ACM*, Volume: 44 Issue: 8. pp 62-68. August 2001

[6] Premkumar, G. Prem. "Alternate distribution strategies for digital music." *Communications of the ACM*, Volume 46 Issue 9 pp 89-95. September 2003

[7] Chen, E.Y. "Digital audio radio-an application of audio compression technology." *Proceedings of The IEEE International Conference on Industrial Technology*, 2-6 Dec 1996.

[8] Haitsma, J and Kalker, T. "A Highly Robust Audio Fingerprinting System." *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, October 2002.

[9] Foote, J. and Cooper, M. "Automatic Music Summarization via Similarity Analysis." *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, October 2002.

[10] Foote, J. and Cooper, M. "Audio Retrieval by Rhythmic Similarity." *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, October 2002.

[11] Tzanetakis, G., Ermolinskiy, A. and Cook, P. "Pitch Histograms in Audio and Symbolic Music Information Retrieval". *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, October 2002.

[12] Rubner, Y., Tomasi, C., and Guibas. L. “The Earth Mover's Distance as a metric for image retrieval.” Technical report, Stanford University, 1998.

[13] Foote, J. and Cooper, M. "Automatic Music Summarization via Similarity Analysis." Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002), October 2002.

[14] Foote, J. and Cooper, M. "Audio Retrieval by Rhythmic Similarity." *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, October 2002.

[15] Logan, B. and Salomon, A. "A Content-Based Music Similarity Function". *Proceedings of IEEE International on Multimedia and Expo (ICME)*, August 2001.

[16] Avery Li-Chun Wang. An Industrial-Strength Audio Search Algorithm. In *4th International Society of Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA, 2003.

[17] David Pye. "Content-Based Methods for the Management of Digital Music". *Proceedings of ICASSP, 2000*.

[18] Geoffrey McLachlan and Thriyambakam Krishnan. The EM Algorithm and Extensions. John Wiley & Sons, New York, 1996.

[19] Cheung, E.S.H. and Constantinides, A.G. "Fast nearest neighbour search algorithms for self-organising map and vector quantisation." *Conference Record of The Twenty Seventh Asilomar Conference on Signals, Systems and Computers, 1-3 Nov 1993. Pp 946 -950.*

[20] Beth Logan and Stephen Chu, "Music Summarization Using Key Phrases", Cambridge Research Laboratories Technical Report 2000/1, 2000

[21] Langlois, T. and Marques, G *"A Music Classification Method Based On Timbral Features"* (ISMIR 2009)

Apêndice A – Lista 10 Músicas, 3 Géneros

A lista, a baixo, representa dez músicas de três géneros Pop, Rock e Jazz respectivamente para a avaliação e teste do sistema de similaridade baseada em conteúdo musical.

- Modern Talking - Brother_Louie.wav
- Modern Talking - Chery_Chery_Lady.wav
- Cars - Drive.wav
- Gala - Let_a_Boy_Cry.wav
- David Guetta - Little_Bad_Girl_(feat._Taio_Cruz).wav
- Opus - Live_Is_Life.wav
- El Bosco - Nirvana.wav
- Youssou N'dour - Seven_seconds_away.wav
- Fine Young Cannibals - She_Drives_Me_Crazy.wav
- Wham - Wake_Me_Up_Before_You_Go_Go.wav

- Pearl Jam - Animal.wav
- Iron Maiden - Moonchild.wav
- Pink Floyd - Another_Brick_In_The_Wall.wav
- Iron Maiden - Murders_In_The_Rue_Morgue.wav
- Led Zepplin - Black_Dog.wav
- Europe - Open_Your_Heart.wav
- Led Zepplin - Good_Times_Bad_Times.wav
- Metallica - The_Four_Horsemen.wav
- Pink Floyd - Money.wav
- Iron Maiden - Wrathchild.wav

- Pat Metheny Group - Afternoon.wav
- Pat Metheny Group - And_Then_I_Knew.wav
- Pat Metheny Group - Are_We_There_Yet.wav
- Pat Metheny Group - Dream_Of_The_Return.wav
- Pat Metheny Group - Every_Summer_Night.wav

- Pat Metheny Group - Follow_Me.wav
- Pat Metheny Group - If_I_could.wav
- Pat Metheny Group - James.wav
- Pat Metheny Group - Red_Sky.wav
- Pat Metheny Group - The_First_Circle.wav

Apêndice B – Lista 30 Músicas, 3 Géneros

A lista, a baixo, representa dez músicas de três géneros Pop, Rock e Jazz respectivamente para a avaliação e teste do sistema de similaridade baseada em conteúdo musical.

- Ace of Base - All_That_She_Wants.wav
- BackStreet Boys - Back_Street_Back.wav
- Modern Talking - Brother_Louie.wav
- Justin Timberlake - Carry_Out.wav
- Modern Talking - Chery_Chery_Lady.wav
- Bob Mcferin - Don_t_Worry_Be_Happy.wav
- Cars - Drive.wav
- 2 Unlimited - Free_Tonight.wav
- Ray Parker Jr. - Ghostbusters.wav
- Culture Club Karma_Chameleon.wav
- Kevin_Lyttle_Feat_Spragga_Benz_-_Tur.wav
- Madonna - La_Isla_Bonita.wav
- Kaoma - Lambada.wav
- Gala - Let_a_Boy_Cry.wav
- David Guetta - Little_Bad_Girl_(feat._Taio_Cruz.wav
- Opus - Live_Is_Life.wav
- Cinnamon Chasers - Luv_Deluxe.wav
- Eiffel 65 - Move_Your_Body.wav
- El Bosco - Nirvana.wav
- Antique - Opa_Opa.wav
- Madonna - Papa_Dont_Preach.wav
- Stevie Wonder - Part_time_lover.wav
- Fairground Attraction - Perfect.wav
- Moon Razors - Rise_Up_(Club_Mix).wav
- Youssou N'dour - Seven_seconds_away.wav

- Fine Young Cannibals - She_Drives_Me_Crazy.wav
- Shely_Bane – Summer Love.wav
- Stero Mcs - Step_It_Up.wav
- Corona - The_Summer_Is_Magic.wav
- Wham - Wake_Me_Up_Before_You_Go_Go.wav

- Pearl Jam - Alive.wav
- Pearl Jam - Animal.wav
- Pink Floyd - Another_Brick_In_The_Wall.wav
- ACDC - Back_In_Black.wav
- ACDC - Big_Gun.wav
- Led Zepplin - Black_Dog.wav
- Pink Floyd - Breathe.wav
- Led Zepplin - Communication_Breakdown.wav
- Fight_Fire_With_Fire.wav
- Flight_Of_Icarus.wav
- Led Zepplin - Good_Times_Bad_Times.wav
- Led Zepplin - Immigrant_Song.wav
- Pearl Jam - Jeremy.wav
- Led Zepplin - Kashmir.wav
- Metallica - Master_of_Puppets.wav
- Pink Floyd - Money.wav
- Iron Maiden - Moonchild.wav
- Metallica - Motorbreath.wav
- Iron Maiden - Murders_In_The_Rue_Morgue.wav
- Europe - Open_Your_Heart.wav
- Metallica - Ride_The_Lightning.wav
- Led Zepplin - Rock_and_Roll.wav
- Europe - Seven_Doors_Hotel.wav
- Pearl Jam - State_Of_Love_And_Trust.wav
- Led Zepplin - Ten_Years_Gone.wav
- Metallica - The_Four_Horsemen.wav
- Led Zepplin - The_Song_Remains_The_Same.wav
- ACDC - Thunderstruck.wav
- Led Zepplin - Whole_Lotta_Love.wav

- Iron Maiden - Wrathchild.wav

- Pat Metheny Group - 45_8.wav
- Pat Metheny Group - Afternoon.wav
- Pat Metheny Group - And_Then_I_Knew.wav
- Pat Metheny Group - Another_Life.wav
- Pat Metheny Group - Are_We_There_Yet.wav
- Pat Metheny Group - Are_You_Going_With_Me.wav
- Pat Metheny Group - As_It_Is.wav
- Pat Metheny Group - Barcarole.wav
- Pat Metheny Group - Better_Days_Ahead.wav
- Pat Metheny Group - Daulton_Lee.wav
- Pat Metheny Group - Dream_Of_The_Return.wav
- Pat Metheny Group - Eighteen.wav
- Pat Metheny Group - Every_Summer_Night.wav
- Pat Metheny Group - Follow_Me.wav
- Pat Metheny Group - Here_To_Stay.wav
- Pat Metheny Group - If_I_could.wav
- Pat Metheny Group - James.wav
- Pat Metheny Group - Lone_Jack.wav
- Pat Metheny Group - Offramp.wav
- Pat Metheny Group - opening.wav
- Pat Metheny Group - part_three.wav
- Pat Metheny Group - part_two.wav
- Pat Metheny Group - Red_Sky.wav
- Pat Metheny Group - Slip_Away.wav
- Pat Metheny Group - Stranger_In_Town.wav
- Pat Metheny Group - The_Bat_part_II.wav
- Pat Metheny Group - The_First_Circle.wav
- Pat Metheny Group - The_Longest_Summer.wav
- Pat Metheny Group - Wherever_You_Go.wav
- Pat Metheny Group - You.wav

Apêndice C – Lista de Instrumentos Musicais

A lista, a baixo, representa seis músicas(todas retiradas do youtube e freesounds) de seis instrumentos musicais: clarinet, flute, guitar, piano, violin e xylophone, respectivamente, para a avaliação e teste do sistema de similaridade baseada em conteúdo musical.

- clarinet1.wav
- clarinet2.wav
- clarinet3.wav
- clarinet4.wav
- clarinet5.wav
- clarinet6.wav

- flute1.wav
- flute2.wav
- flute3.wav
- flute4.wav
- flute5.wav
- flute6.wav

- guitar1.wav
- guitar2.wav
- guitar3.wav
- guitar6.wav
- guitar7.wav
- guitar8.wav

- piano1.wav
- piano2.wav
- piano3.wav

- piano4.wav
- piano5.wav
- piano6.wav
- violin1.wav
- violin2.wav
- violin3.wav
- violin4.wav
- violin5.wav
- violin6.wav
- violin7.wav

- xylophone1.wav
- xylophone2.wav
- xylophone3.wav
- xylophone4.wav
- xylophone5.wav
- xylophone6.wav
- xylophone7.wav
- xylophone8.wav

Apêndice D – Lista de Episódios

A lista, a baixo, representa as quatro colecções relativas aos quatro episódios: `LostS04E11`, 24.`S01E01`, `PrisonBreakS02E21`, `24.S01E01` e `TheWalkingDeadS02E07` para a avaliação e teste do sistema de similaridade baseada em conteúdo do áudio. Todos os episódios estão cortados em pedaços de áudio de 3 segundos com overlap de 1.5 segundos.

Prison.BreakS02E21
Prison.BreakS02E21.1.wav
Prison.BreakS02E21.2.wav
Prison.BreakS02E21.3.wav
Prison.BreakS02E21.4.wav
Prison.BreakS02E21.5.wav
Prison.BreakS02E21.6.wav
Prison.BreakS02E21.7.wav
Prison.BreakS02E21.8.wav
Prison.BreakS02E21.9.wav
…
Prison.BreakS02E21.1034.wav

The.Walking.Dead.S02E07
The.Walking.Dead.S02E07.1.wav
The.Walking.Dead.S02E07.2.wav
The.Walking.Dead.S02E07.3.wav
The.Walking.Dead.S02E07.4.wav
The.Walking.Dead.S02E07.5.wav
The.Walking.Dead.S02E07.6.wav
The.Walking.Dead.S02E07.7.wav

The.Walking.Dead.S02E07.8.wav
The.Walking.Dead.S02E07.9.wav
…
The.Walking.Dead.S02E07.1038.wav

Lost.S04E11
Lost.S04E11.1
Lost.S04E11.2
Lost.S04E11.3
Lost.S04E11.4
Lost.S04E11.5
Lost.S04E11.6
Lost.S04E11.7
Lost.S04E11.8
Lost.S04E11.9
…
Lost.S04E11.1036

24.S01E01
24.S01E01_1.wav
24.S01E01_2.wav
24.S01E01_3.wav
24.S01E01_4.wav
24.S01E01_5.wav
24.S01E01_6.wav
24.S01E01_7.wav
24.S01E01_8.wav
24.S01E01_9.wav
…
24.S01E01_1135.wav

Apêndice E – Manual do Interpretador de Comandos

A lista, a baixo, expõe todos os comandos disponíveis no interpretador. A cada um, está associado um exemplo de forma a facilitar a percepção, também, a sua utilidade e manuseamento.

Arranque

Ao correr o programa obtemos o seguinte menu…
A primeira linha consiste no ficheiro que o programa precisa no arranque. Este ficheiro contem a localização das *features* do Marsyas (2ª linha "path to marsyas :

```
Reading configuration from file /home/eduardo/workspace/TeseMestrado/config-tl.txt
Path to marsyas : /home/eduardo/marsyas/build/bin/

******Choose menu item:******
create output directory <dir>
register file <path>
register dir <dir>
set features <string>
select collection <name of collection>
show
merge collections <name of merged Collection> <name of collection1> <name of collection> ... <name of collectionN>
make codebook <name of collection> <k2>
get symbols <name of collection>
get histograms <name of collection>
distance matrix <name of collection1> <name of collection2>
list AudioPieces <name of collection>
play <name of collection> <numero>
quit
>
```

/home/eduardo/marsyas/build/bin/) e pode ainda conter, ou não, uma série de comandos a serem processados pelo programa.

Comandos

create output directory <dir>

Este comando tem como objectivo a criação de um directório/pasta, escolhido pelo cliente, onde são guardados todos os ficheiros e testes criados pelo programa. Neste exemplo, este comando, irá criar uma pasta *fs1*, vazia, dentro da pasta *Imagens*.

```
> create output directory /home/eduardo/Imagens/fs1/

OPTION:create output directory /home/eduardo/Imagens/fs1/
OUTPUTDIR:/home/eduardo/Imagens/fs1/
Directorio criado com sucesso!
```

register dir <dir>

Este comando tem como objectivo a criação de uma collection de peças de áudio. Neste exemplo, este comando, irá criar uma collection, isto é, uma ficheiro .mf com as peças de áudio contidas no directório /home/eduardo/Documentos/samples1/piano/.

```
> register dir /home/eduardo/Documentos/samples1/piano/
Making the collection file pianoColec.mf
Here is the standard error of the AudioCollection(String directory, String collectionName) command (if any):

Here is the standard output of the AudioCollection(String directory, String collectionName) command:

Writing collectionName = /home/eduardo/Imagens/fs1/pianoColec into file /home/eduardo/Imagens/fs1/pianoColec.mf
Adding Contents of Directory = /home/eduardo/Documentos/samples1/piano/
 to collection /home/eduardo/Imagens/fs1/pianoColec.mf
Adding directory entry piano4Sample.wav
Adding directory entry piano2Sample.wav
Adding directory entry piano1Sample.wav
Adding directory entry piano3Sample.wav
Wrote collection /home/eduardo/Imagens/fs1/pianoColec.mf
-------------- end --------------
```

register file <path>

Este comando tem como objectivo a criação de uma AudioPiece correspondente a uma peça de áudio. Neste exemplo, este comando, irá criar uma AudioPiece e um ficheiro .arff correspondente que irá armazenar os diferentes vectores referentes às características expectrais da peças de áudio dada como path.

```
> register file /home/eduardo/marsyas/music_speech/musica1/news1.wav

File /home/eduardo/marsyas/music_speech/musica1/news1.wav registred!
```

set *features* <string>

Este comando tem como objectivo escolher qual forma serão extraídas as *features* das peças de áudio. Neste exemplo, foi escolhida a fs2 que extrai as *features* duma peça de áudio de forma distinta se usássemos a fs1, seleccionada por defeito.

```
> set features fs2
Feature set is now fs2!
```

select collection <name of collection>

Este comando tem como objectivo permitir escolher uma determinada collection, dando o nome da mesma no parametro <*name of collection.* Neste exemplo, é seleccionada a collection … onde posteriormente se podem aplicar outros comandos.

```
> select collection pianoColec
Selected collection is: pianoColec
```

show

Este comando tem como objectivo sumarizar os comandos já realizados. Como vemos neste exemplo, é mostrado: qual a collection seleccionada, quais as collections criadas, quais os codebooks criados e qual destes últimos está seleccionado.

```
> show
Feature set: fs1
Selected Collection: No Collection selected
AudioCollections in AudioCollectionSet:
-> guitarColec
-> pianoColec

Codebooks in CodebookCollection:
-> guitarColecCodBook.cbk8.fs1

Selected Codebook: guitarColecCodBook.cbk8.fs1
/home/eduardo/Imagens/fs1
/home/eduardo/Imagens/fs1
```

merge collections <name of merged Collection> <name of collection1> <name of collection> ... <name of collectionN>

Este comando tem como objectivo juntar duas collections previamente criadas numa única, nova, collection. Neste exemplo, são juntas as collections *pianoColec* e *guitarColec* numa nova collection – piano&guitar.

```
> merge collections guitar&piano guitarColec pianoColec
COLNAME:guitar&piano
guitarColec
pianoColec
Merged collection guitar&piano created
```

`make codebook <name of collection> <k2>`

Este comando tem como objectivo a criação de um ficheiro codebook - documento utilizado para reunir e armazenar códigos. No nosso caso, o codebook irá armazenar os vectores localização dos diferentes *clusters* numa collection. Neste exemplo, irão ser guardados num ficheiro codebook os *clusters* da collection guitarColec.

```
> make codebook guitarColec 8
Here is the standard output of the first command:


Window Size (in samples): 512
Hop Size (in samples): 512
Memory Size (in analysis windows):40
Accumulator size (in analysis windows):5000

Extractor = REFACTORED

collectionName = MARSYAS_EMPTY
classifierName = SVM
Downsampling factor = 1
Processing: 0 - /home/eduardo/Documentos/samples1/guitar/guitar2Sample.wav
Processing: 1 - /home/eduardo/Documentos/samples1/guitar/guitar1Sample.wav
Processing: 2 - /home/eduardo/Documentos/samples1/guitar/guitar3Sample.wav
Finished feature extraction
Finished classifier training
Here is the standard error of the first command (if any):

FILENAME:/home/eduardo/Imagens/fs1/guitarColec.mf
-------------- end --------------
Codebook file guitarColecCodBook.cbk8.fs1.arff does not exist.
Load data (/home/eduardo/Imagens/fs1/guitarColec.fs1.arff)
arffToInstances(/home/eduardo/Imagens/fs1/guitarColec.fs1.arff) -> 3876 instances read.
Start clustering process.
# of clusters: 8Result :
kMeans
======

Number of iterations: 23
```

```
output                                                                    /home

Clustered Instances

0       197 (  5%)
1       930 ( 24%)
2       217 (  6%)
3       351 (  9%)
4       790 ( 20%)
5       622 ( 16%)
6       308 (  8%)
7       461 ( 12%)
```

get symbols <name of collection>

Este comando tem como objectivo a criação dos ficheiros que contém, para cada vector, o centroid/cluster mais próximo. Para tal, é executado o comando `bextract` do Marsyas para extrair as *features*, fazendo posteriormente a comparação de distâncias entre os centroids contidos no ficheiro codebook calculado no comando anterior e os vectores calculados com o bextract. O resultado é guardado num ficheiro *.symb*.

```
> get symbols guitarColec
/home/eduardo/Documentos/samples1/guitar/guitar2Sample.wav
The file /home/eduardo/Imagens/fs1/guitar2Sample.mf does not exist.
The feature file /home/eduardo/Imagens/fs1/guitar2Sample.fs1.arff does not exist.
Extracting features with command : /home/eduardo/marsyas/build/bin/bextract /home/eduardo/
Here is the standard output of the first command:


Window Size (in samples): 512
Hop Size (in samples): 512
Memory Size (in analysis windows):40
Accumulator size (in analysis windows):5000

Extractor = REFACTORED

collectionName = MARSYAS_EMPTY
classifierName = SVM
Downsampling factor = 1
Processing: 0 - /home/eduardo/Documentos/samples1/guitar/guitar2Sample.wav
Finished feature extraction
Finished classifier training
Here is the standard error of the first command (if any):

-------------- end --------------
```

get histograms <name of collection>

Este comando tem como objectivo a simplificação de um ficheiro *.symb*. Este irá agrupar o numero de vectores por cluster, isto é, vai contabilizar quantos vectores estão mais próximos do cluster 1, cluster 2 … etc. O resultado irá ser guardado num ficheiro *.hist*.

```
> get histograms guitarColec
getSymbolFile : guitar2Sample.guitarColecCodBook.cbk8.fs1.symb
histogramFile /home/eduardo/Imagens/fs1/guitar2Sample.guitarColecCodBook.cbk8.fs1.hist does not exist.
getSymbolFile : guitar2Sample.guitarColecCodBook.cbk8.fs1.symb
symbolsFile exists.
getSymbolFile : guitar1Sample.guitarColecCodBook.cbk8.fs1.symb
histogramFile /home/eduardo/Imagens/fs1/guitar1Sample.guitarColecCodBook.cbk8.fs1.hist does not exist.
getSymbolFile : guitar1Sample.guitarColecCodBook.cbk8.fs1.symb
symbolsFile exists.
getSymbolFile : guitar3Sample.guitarColecCodBook.cbk8.fs1.symb
histogramFile /home/eduardo/Imagens/fs1/guitar3Sample.guitarColecCodBook.cbk8.fs1.hist does not exist.
getSymbolFile : guitar3Sample.guitarColecCodBook.cbk8.fs1.symb
symbolsFile exists.
```

list AudioPieces <name of collection>

Este comando tem como objectivo a listagem das audiopices de uma collection cujo nome é dado como argumento.Neste exemplo queremos ver as audiopieces que a collection *guitarColec* contem.

```
> list AudioPieces guitarColec
0 -> /home/eduardo/Documentos/samples1/guitar/guitar2Sample.wav
1 -> /home/eduardo/Documentos/samples1/guitar/guitar1Sample.wav
2 -> /home/eduardo/Documentos/samples1/guitar/guitar3Sample.wav
```

play <name of collection> <number>

Este comando tem como objectivo tocar a peça de áudio na collection dada como argumento e o seu número.

distance matrix <name of collection1> <name of collection2>

Este comando tem como objectivo a criação de uma matrix de distâncias entre duas collections, isto é, é calculada uma matrix que contem a distância entre as peças de áudio de uma collection1 e todas as peças de áudio de uma collection2 e vice-versa. Neste exemplo, é calculada a distance matrix de uma merged collection – piano&guitar&sax-trumpet.

```
pianoColec&sax-trumpetColec
 0.83 0.86 0.78
 0.82 0.86 0.77
 0.73 0.77 0.68
 0.63 0.69 0.47

LD:0.8648534911296548
N:12
AVERAGE
0.8569482175470075

guitarColec
 0 0.39 0.58
 0 0 0.86
 0 0 0

LD:0.8631700264770564
N:3
AVERAGE
0.7065924192987144

guitarColec&sax-trumpetColec
 0.59 0.56 0.48
 0.82 0.86 0.71
 0.67 0.28 0.61

LD:0.8569345421419597
N:9
AVERAGE
0.7214110060251907

sax-trumpetColec
 0 0.5 0.46
 0 0 0.48
```

nearest neighbours <n> <fich-audio> <dist-mat>

Este comando tem como objectivo a indicação das N peças mais próximas da peça de áudio dada como input.

Neste exemplo, é calculado os 10 mais próximos do sample `24.S01E01.460.wav`

```
> nearest neighbors 10 24.S01E01.460.wav 24S01E0124S01E01.cbk300-400.fs1.arff.matrix
AudioPiece: 24.S01E01.460.wav
Distance Matrix: 24S01E0124S01E01.cbk300-400.fs1.arff.matrix
N: 10
OUTPUTDIR:/home/eduardo/Imagens/24S01E01/fs1
/home/eduardo/Imagens/24S01E01/fs1/24S01E0124S01E01.cbk300-400.fs1.arff.distSet
DISTANCES
24.S01E01.893.wav,0.0,24.S01E01.460.wav,0.0,14.35
24.S01E01.896.wav,0.0,24.S01E01.460.wav,0.0,52.02
24.S01E01.460.wav,0.0,24.S01E01.459.wav,0.0,53.5
24.S01E01.460.wav,0.0,24.S01E01.892.wav,0.0,65.05
24.S01E01.460.wav,0.0,24.S01E01.464.wav,0.0,68.13
24.S01E01.460.wav,0.0,24.S01E01.894.wav,0.0,69.44
24.S01E01.895.wav,0.0,24.S01E01.460.wav,0.0,72.47
24.S01E01.460.wav,0.0,24.S01E01.1209.wav,0.0,72.62
24.S01E01.460.wav,0.0,24.S01E01.1230.wav,0.0,73.8
24.S01E01.460.wav,0.0,24.S01E01.851.wav,0.0,80.06
```

quit

Encerra e termina a execução do programa.