INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Classificação de sons urbanos usando motifs e MFCC

FABIO MIGUEL MOREIRA BATISTA
Outubro de 2015

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Classificação de sons urbanos usando motifs e MFCC

Fábio Miguel Moreira Batista

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientador: Elsa Maria de Carvalho Ferreira Gomes

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro 2015

Resumo

A classificação automática de sons urbanos é importante para o monitoramento ambiental. Este trabalho apresenta uma nova metodologia para classificar sons urbanos, que se baseia na descoberta de padrões frequentes (*motifs*) nos sinais sonoros e utiliza-los como atributos para a classificação. Para extrair os *motifs* é utilizado um método de descoberta multi-resolução baseada em SAX. Para a classificação são usadas árvores de decisão e SVMs. Esta nova metodologia é comparada com outra bastante utilizada baseada em MFCC. Para a realização de experiências foi utilizado o *dataset* UrbanSound disponível publicamente.

Realizadas as experiências, foi possível concluir que os atributos *motif* são melhores que os MFCC a discriminar sons com timbres semelhantes e que os melhores resultados são conseguidos com ambos os tipos de atributos combinados.

Neste trabalho foi também desenvolvida uma aplicação móvel para Android que permite utilizar os métodos de classificação desenvolvidos num contexto de vida real e expandir o *dataset*.

Palavras-chave: Classificação de sons urbanos, mineração de dados, *motifs*, MFCC.

Abstract

The automatic classification of urban sounds is important for environmental monitoring. This work presents a new method to classify urban sounds based on frequent patterns (motifs) in the audio signals and using them as classification attributes. To extract the motifs, a multi-resolution discovery based on SAX is used. For the classification itself, decision trees and SVMs are used. This new method is compared with another largely used based on MFCCs. For the experiments, the publicly available UrbanSound dataset was used.

After the experiments, it was concluded that motif attributes are better to discriminate sounds with similar timbre and better results are achieved with both attribute types combined.

In this work was also developed a mobile application for Android which allows the use of the developed classifications methods in a real life context and to expand the dataset.

Keywords: Urban Sound Classification, data mining, motifs, MFCC

Agradecimentos

Agradeço a todas as pessoas que de alguma forma contribuíram para a realização desta investigação.

Um agradecimento especial para a Prof. Elsa Gomes que me proporcionou a oportunidade de realizar este trabalho, que me acompanhou ao longo da realização do mesmo e que se mostrou sempre disponível para ajudar.

Agradeço também aos meus pais que sempre me apoiaram em tudo e que me incentivaram para a realização do curso de mestrado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

AASP	*Audio and Acoustic Signal Processing*
ADT	*Android Developer Tools*
DCT	*Discrete Cosine Transform*
DFT	*Discrete Fourier Transform*
GMM	*Gaussian mixture model*
HMM	*Hidden Markov Models*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
iSAX	*indexable Symbolic Aggregate approXimation*
kNN	k-Nearest Neighbor
MFCC	*Mel-frequency Cepstral Coefficient*
MP	*Matching Pursuit*
MPEG	*Moving Picture Experts Group*
NMF	*Non-negative Matrix Factorization*
NMF	*Non-negative Matrix Factorization*
PAA	*Piecewise Aggregate Approximation*
PCA	*Principal Component Analysis*
PCM	*Pulse Code Modulation*
PHP	*PHP: Hypertext Preprocessor*
SAX	*Symbolic Aggregate approXimation*

SMO	*Sequential Minimal Optimization*
SOAP	*Simple Object Access Protocol*
SOLAR	*Sound Object Localization And Retrieval*
SVM	*Support Vector Machines*
URL	*Uniform Resource Locator*
WSDL	*Web Services Description Language*
XML	*eXtensible Markup Language*

Lista de Símbolos

∈	Pertença a conjunto
e	Número de Euler
α	Alfa
Θ	Teta
π	Pi
Σ	Sigma (somatório)
ω	Ômega

1 Introdução

A deteção automática de sons é uma área em constante evolução desde a década de 1950. Começando pelo reconhecimento da fala até ao reconhecimento de sons acústicos, os trabalhos nesta área são inúmeros. No entanto, poucos trabalhos foram realizados na área do reconhecimento de sons urbanos [Salamon, et al., 2014].

O reconhecimento de sons urbanos é útil no sentido de detetar fontes de ruido numa zona urbana. Segundo a OMS (Organização Mundial de Saúde) [World Health Organization, 2011], a poluição sonora tem um grande impacto negativo na saúde pública, podendo provocar, a quem está diariamente exposto, perda de audição, doenças cardiovasculares, distúrbios do sono, aumento do *stress*, entre outros problemas. Para combater a poluição sonora numa cidade é importante identificar não só o nível de volume do som, mas também identificar a sua origem [Advanced Digital Sciences Center].

O reconhecimento de sons na cidade pode ser também importante no combate à criminalidade. Detetar disparos de uma arma de fogo, gritos de socorro, e outros sons inerentes a um ato criminoso e a localização exata destes, poderia dar uma ajuda preciosa às autoridades possibilitando uma atuação mais eficaz. Inclusive, já foram realizados alguns trabalhos nesse sentido, incluindo um sistema que permite detetar sons de disparos de armas de fogo utilizando a mesma tecnologia para detetar sismos [Earthquake Hazards Program].

Há 20 anos atrás, era impraticável montar um sistema de captação e processamento de som numa cidade. Atualmente, com tecnologia existente para captação de sons, mais barata e com melhor qualidade, com o aumento significativo do poder de processamento dos computadores e com as melhorias nos sistemas de *data mining* é possível montar um sistema de captação de som numa zona urbana que detete eventos sonoros, identifique qual a origem do evento e

inclusive que localize onde o evento ocorreu, num tempo bastante reduzido [Advanced Digital Sciences Center, 2015].

1.1 Objetivos

O principal objetivo deste trabalho é aplicar a metodologia de classificação de sons usando os *motifs* aos sons urbanos. A metodologia já foi aplicada com sucesso noutro tipo de sons, mas nunca tinha sido testada em sons urbanos, onde os sons têm características muito variadas e a presença de ruido é uma constante. É também objetivo deste trabalho comparar os resultados da aplicação desta metodologia com outra tecnologia, frequentemente utilizada neste tipo de problemas, baseada em *Mel-frequency Cepstrum Coefficients* (MFCC) [Davis and Mermelstein, 1980].

Por último, é também objetivo criar uma ferramenta que permita utilizar os modelos de classificação gerados num contexto de tempo real.

1.2 Abordagem

Este trabalho incide na identificação de sons urbanos usando técnicas de *data mining* para classificar o som quanto à sua origem, utilizando uma técnica para segmentar e sintetizar os sons através de *motifs* e vários algoritmos de classificação. Tanto quanto se sabe esta abordagem não tinha sido antes utilizada em sons urbanos. Neste trabalho, utilizamos o *dataset* UrbanSound contendo 1302 gravações de sons em ambientes urbanos obtidas por Justin Salamon, Christopher Jacoby e Juan Pablo Bello [Salamon et al., 2014]. Utilizamos também um segundo *dataset* UrbanSound8K derivado do primeiro.

1.3 Resultados e contribuições

As principais contribuições e resultados deste trabalho são:

- A demonstração do poder de discriminação de sons da abordagem usando *motifs*, no caso em que os sons apresentam uma semelhança de timbres (e em que a abordagem por MFCC falha);
- A demonstração de que a combinação das duas abordagens tem um maior poder discriminante;
- Uma aplicação móvel para Android que permite utilizar os modelos de classificação gerados num contexto de vida real e expandir o *dataset*;
- Os resultados obtidos neste trabalho foram publicados no artigo *Classifying Urban Sounds using Time Series Motifs*, apresentado em The 6th International

Conference on Security-enriched Urban Computing and Smart Grids (*SUComS 2015*) [Gomes and Batista, 2015a].

- Os resultados obtidos neste trabalho foram também recentemente publicados no artigo *Using Multiresolution Time Series Motifs to Classify Urban Sounds*, na revista International Journal of Software Engineering and Its Applications [Gomes and Batista, 2015b]. Este artigo encontra-se em anexado a este documento.

1.4 Estrutura da dissertação

Esta dissertação está composta do seguinte modo:

No capítulo 2 é apresentado o problema estudado neste trabalho. Neste capítulo, é também feito um levantamento do estado da arte com referências a trabalhos anteriores na área.

No capítulo 3 é feita uma introdução aos algoritmos de classificação utilizados assim como os métodos para extração de atributos dos sons utilizados neste trabalho.

No capítulo 4 são introduzidas as ferramentas e tecnologias utilizadas neste trabalho.

No capítulo 5 são descritas todas as experiências de classificação realizadas, incluindo todo o processo de realização das mesmas e os resultados obtidos.

No capítulo 6 é apresentada a aplicação móvel desenvolvida, intitulada “UrbanSound Classifier”.

2 Classificação de sons

Neste capítulo faz-se um levantamento do estado da arte relacionada com a classificação de sons.

Na subsecção 2.1 referem-se os estudos de vários autores, indicado o estado em que se encontra a pesquisa na área e qual o contributo que é dado com este trabalho.

Na subsecção 2.2 são mostradas abordagens anteriores para classificar sons e, mais especificamente, utilizações anteriores de *motifs* para o efeito.

2.1 Definição do problema

A área com maior relevo no reconhecimento de som sempre foi, e continua a ser, o reconhecimento da fala [Giannoulis et al., 2013]. Em 1952 foi criado o primeiro sistema de reconhecimento de fala intitulado "Audrey", que apenas permitia reconhecer números falados por uma única voz. Atualmente existem sistemas capazes de reconhecer quase todas as palavras em dezenas de línguas.

O reconhecimento automático de sons começou a ter algum relevo apenas nas últimas duas décadas. Este interesse tardio pelo tema deve-se à complexidade do problema e ser considerado de menor utilidade quando comparado com o reconhecimento da fala. Nos sons urbanos (sons acústicos produzidos num ambiente urbano) a complexidade é ainda maior devido ao ruido de fundo característico de um ambiente urbano presente nos sons.

Existem inúmeros trabalhos na área do reconhecimento de sons acústicos que utilizam diversos métodos para efetuar o reconhecimento. São utilizados inclusive, com sucesso, métodos anteriormente utilizados para reconhecimento da fala no reconhecimento de sons acústicos. No que diz respeito especificamente aos sons urbanos existem ainda poucos trabalhos realizados.

Apesar da variedade de métodos existentes para reconhecimento e classificação de sons acústicos e urbanos, qualquer um deles está longe de ser perfeito. Muito poucos são aqueles que conseguem classificar com uma exatidão superior a 90%.

De um modo geral, qualquer metodologia para classificar sons é composta por duas fases principais:

- Extração de atributos dos sons
- Criação de modelos de classificação utilizando os atributos extraídos

A qualidade dos modelos depende do *dataset* de sons utilizado, dos atributos extraídos e do algoritmo de classificação usado para criar o modelo.

Com este trabalho pretende-se testar uma nova metodologia para a classificação de sons urbanos baseada na extração de padrões (*motifs*) existentes nos sons. Para avaliação dos resultados, utiliza-se uma das abordagens mais referidas na literatura para o efeito que é baseada em *Mel-frequency Cepstum Coeficients* (MFCC).

2.2 Abordagens anteriores

2.2.1 Reconhecimento e classificação de sons

Como já foi referido, os primeiros trabalhos sobre reconhecimento e classificação de sons incidiu sobre o reconhecimento da fala. Apenas nas últimas duas décadas foram feitos progressos no que diz respeito ao reconhecimento de sons.

A Tabela 1 sintetiza alguns contributos na área do reconhecimento e classificação de sons.

Tabela 1 – Exemplos de contributos na área do reconhecimento e classificação de sons acústicos

Referência	***Dataset***	**Tecnologias/Metodos**
[Beritelli, 2008]	Contém 10 gravações, cada uma com duração de 90 segundos. Cada gravação corresponde a uma das classes considerada: *Bus, Car, Construction, Dump, Factory, Office, Pool, Station, Stadium and Train*	Classificador de ruído de fundo com base numa abordagem de reconhecimento de padrões utilizando uma rede neural. Os sinais enviados para a rede neural são caracterizados por um conjunto de 12 MFCC. A exatidão do resultado da classificação varia entre 73% e 95%, dependendo da duração da janela de decisão.

Referência	*Dataset*	Tecnologias/Metodos
[Couvreur and Laniray, 2004]	917 gravações em ambiente urbano. Cada gravação contém ruido de fundo e um único evento sonoro.	O sistema proposto é baseado em Redes Neurais Artificiais, e modelos de Markov. Primeiro o sistema extrai vetores de atributos dos sinais sonoros, que representam a evolução temporal do espectro, de forma compacta. Depois, cada um deles é classificado utilizando Redes Neuronais. Por fim, estas decisões locais são integradas na estrutura de modelos ocultos de Markov para obter uma decisão global sobre a localização e a natureza dos eventos de ruído. A exatidão dos modelos de classificação criados varia entre 85% a 95%.
[Ntalampiras et al, 2008]	439 sons divididos em seis classes: aircraft (110) ,motorcycle (46), car (81), crowd (60), thunder (60) and train (82)	Extração de atributos baseados em MFCC e MPEG 7 (*Moving Picture Experts Group*). Para classificação utilizam um método baseado em mistura Gaussiana. O número de misturas Gaussianas para todas as experiências foi de 4, enquanto cada uma das densidades é descrita por uma matriz de covariância de forma diagonal. Descritores MPEG-7 atingem uma exatidão de 65%, enquanto os MFCC alcançam 78,3%.
[Cavaco and Rodeia, 2010]	Contém um conjunto de 60 sons de impactos com hastes de diferentes materiais: madeira, alumínio, aço e aço banhado a zinco)	Apresentam um classificador para sons similares. Começa por representar os sons com espectrogramas (ou mais precisamente, a magnitude da transformada de Fourier de curto tempo). Em seguida, extrai atributos de som destes espectrogramas usando o método de Cavaco e Lewicki [Cavaco and Lewicki, 2007] para modelar as estruturas intrínsecas dos sons. O método permite extrair funções de tempo e frequência variável que podem ser usadas como atributos de classificação na última fase do classificador, onde é utilizado um algoritmo de Redes Neuronais. São obtidas exatidões de 98.33% e 100% para atributos espectrais e temporais, respetivamente.
[Valero and Alías, 2012]	90 sons por cada fonte de ruido considerada (*light vehicles, heavy vehicles, motorcycles, aircrafts, trains and industrial noise*) com a duração de 4 segundos cada	Descrevem uma metodologia que começa por dividir o sinal em *frames* e extrair os atributos do sinal a partir de cada uma. De seguida, os atributos de cada *frame* são concatenados num único vetor. É então utilizada a Análise de Componentes Principais (PCA) para compactar a informação. São extraídos vários tipos de atributos. O melhor resultado obtido foi 90% de exatidão utilizando atributos MFCC e MPEG 7 combinados e utilizando misturas gaussianas (GMM) para gerar o modelo.

Referência	*Dataset*	Tecnologias/Metodos
[Hoiem et al., s.d.]	Entre 15 e 80 exemplos positivos por cada classe (*car horns, doors closing, dog barks, door bells, explosions, gunshots, laser guns, light sabers, male laughs, meows, telephone rings, screams, and sword clashes*) e cerca de 1000 exemplos negativos.	Utilizam um sistema de deteção de eventos sonoros em filmes intitulado SOLAR (*Sound Object Localization and Retrieval*). SOLAR evita segmentação pela decomposição de sons em janelas pequenas do mesmo tamanho e com sobreposição. Cada janela é avaliada por um classificador baseado em árvore de decisão impulsionada (*boosted*). Finalmente, os clipes de som que contêm áudio identificado como pertencente à classe objetivo são retornados pelo sistema, ordenados por confiança.
[Uzkent et al, 2011]	258 sons distribuídos pelas seguintes classes: *gunshot, glass breaking, dog bark, scream, engine rain restaurant*.	Introduzem um novo conjunto de atributos 2D, usando um método de extração de atributos com base no *pitch range* dos sons e na função de auto correlação. Usam *Support Vector Machines* (SVM) e *Radial Basis Function Neural Network*. Obtida exatidão de 88.7% utilizando atributos *pitch range* e MFCC combinados.
[Al-Sarayreh et al., 2009]	250 sons de carros e muitos outros sons de outros domínios (*rain, thunder, and plane*).	Utilizam técnicas de reconhecimento de som, a fim de ligar as luzes somente quando há carros na estrada apenas por um determinado período de tempo. São utilizadas técnicas de estatística para extração de atributos e Codificação Linear Preditiva para reconhecimento dos sons. Além disso, é utilizada quantização vetorial para mapear os sons em grupos, a fim de comparar os sons testados. Media de exatidão de 92.5%.
[Chu et al., 2009]	Gravações obtidas da "The BBC Sound Effects Library-Original Series" e do "The Freesound Project".	Utilizam o algoritmo "*matching pursuit*" (MP) para obtenção de atributos tempo-frequência. O método baseado em MP utiliza um dicionário de átomos para a seleção de atributos, resultando em um conjunto de atributos flexível, intuitivo e fisicamente interpretável. Os atributos baseados em MP são usados para juntar aos atributos obtidos usando a abordagem MFCC, para conseguir melhores resultados na classificação. Utilizam misturas gaussianas (GMM) e o kNN (K-nearest neighbor).para a classificação. Melhor exatidão média obtida foi de 83.9%.

Referência	*Dataset*	Tecnologias/Metodos
[Dufaux, 2001]	1000 amostras de sons ligados à vigilância e segurança, divididas em 10 classes (*door slams, glass breaks, human screams, explosions, gunshots, dogs barking, low bangs, phone rings, children voices and machines running*). Foi construída uma classe suplementar com diferentes sons de fundo.	Os métodos de deteção apresentados baseiam-se numa medição contínua das variações de energia dos sons. Em seguida, métodos estatísticos robustos realizam uma análise de tempo-frequência. As distribuições estatísticas dos parâmetros extraídos são então modeladas utilizando misturas gaussianas (GMM), Hidden Markov Models (HMM), ou *Perceptron Neural Networks*. Exatidão de 97% obtida sem ruído, 80% com ruído.
[Heittola et al, 2011]	103 gravações, com durações entre 10 e 30 minutos, divididas pelas classes: *basketball game, beach, inside a bus, inside a car, hallway, office, restaurant, grocery shop, street and stadium*.	Apresentam um sistema de deteção de evento de som baseado em Hidden Markov Models (HMM) de densidade contínua. Além dos coeficientes de MFCC estáticos, os diferenciais de tempo de primeira e segunda ordem são usados para descrever as propriedades dinâmicas do *Cepstrum*. Na fase de deteção é usado o algoritmo de Viterbi [Forney Jr., 2005] para determinar a sequência de eventos mais provável e os respetivos tempos onde acontecem. Uma representação em histograma do resultado é utilizada para reconhecer o contexto onde o som foi gravado. A exatidão na deteção de evento foi de 30% e no reconhecimento do contexto foi de 92.4%.
[Mesaros et al., 2015]	Gravações em 10 contextos diferentes: *basketball, beach, bus, car, hallway, office, restaurant, shop, street and track&field*.	Apresentam um método que deteta eventos sobrepostos sem construir modelo separados para cada classe. O método utiliza matrizes não negativas (NMF) acopladas onde uma das matrizes é a representação espectral do som e a outra é constituída por indicadores binários de presença de classe obtido a partir da anotação dos dados de áudio. Os autores obtiveram 57.8% de exatidão nos resultados.

Em 2012, foi proposto pelo IEEE AASP (*Institute of Electrical and Electronics Engineers Audio and Acoustic Signal Processing*) um desafio cujo objetivo era fazer progredir a área de deteção de sons, dado o pouco desenvolvimento que tinha em relação à deteção da fala e da música [Giannoulis et al., 2013]. O desafio foi dividido em duas tarefas distintas:

- Classificação do cenário acústico – Consiste em classificar sons quanto ao tipo de local onde foram gravados. Para esta tarefa foi disponibilizado um *dataset* com 100 sons

divididos por 10 classes: bus (autocarro), "busystreet" (rua movimentada), "office" (escritório), "openairmarket" (mercado ao ar livre), "park" (parque), "quietstreet" (rua calma), restaurante (restaurante), "supermarket" (supermercado), "tube" (metro), "tubestation" (estação de metro).

- Deteção de eventos dentro de um cenário – Consiste em detetar eventos específicos dentro de um som. Esta tarefa foi dividida em duas sub-tarefas: deteção monofónica e deteção polifónica. Na primeira, os eventos a detetar encontram-se isolados, enquanto na segunda existe sobreposição de eventos. Para a deteção monofónica foi disponibilizado um *dataset* que contém gravações em ambiente de escritório, divididos pelas seguintes classes: "alert" (beep), "clearthroat" (limpar a garganta), "cough" (tosse), "doorslam" (porta a bater), "drawer" (gaveta), "keyboard" (teclado), "keys" (chaves), "knock" (bater à porta), "laughter" (riso), "mouse" (clique do rato), "pageturn" (virar página), "pendrop" (caneta a cair), "phone" (telefone), "printer" (impressora), "speech" (fala), "switch" (interruptor). Para a deteção polifónica foi gerado um outro *dataset* a partir da concatenação, através de um sintetizador, dos sons do *dataset* anterior.

Na Tabela 2 encontram-se alguns dos trabalhos submetidos para o desafio.

Tabela 2 – Contributos submetidos no IEEE AASP *challange*

Referência	**Tarefa**	**Tecnologias/Metodos**
[Chum et al.,2013]	Classificação do cenário acústico	Utilizam HMM e modelos de mistura gaussiana. Os atributos utilizados foram: Transformada de Fourier, a sonoridade e dispersão espectral. Conseguida uma exatidão de 72% com validação cruzada de 10 *folds*.
[Elizalde et al., 2013]	Classificação do cenário acústico	Apresentam para classificação um algoritmo "*hand-tuned i-vector system*" e atributos baseados em MFCC. É conseguida uma exatidão de 65.8% utilizando validação cruzada com 5 *folds*.
[Geiger et al., 2013]	Classificação do cenário acústico	Utilizam a ferramenta de código aberto "openSMILE" para extração de atributos dos sons. Cada gravação é dividida em janelas que se sobrepõem (*overlapping*) com alguns segundos de duração. Usam os algoritmos de classificação SVM e NN. É conseguida uma exatidão de 73% utilizando validação cruzada com 5 *folds*.

Referência	Tarefa	Tecnologias/Metodos
[Gemmeke et al, 2013]	Deteção de eventos dentro de um cenário	Apresentam um método baseado em factorização de matrizes não negativas (NMF). Os eventos são modelados como uma combinação linear de átomos de dicionário e as misturas como uma combinação linear de sobreposição de eventos. Resultado de 65.2 de *F-Score*.
[Olivetti, 2013]	Classificação do cenário acústico	Apresentam um método eficaz para incorporar objetos genéricos, como amostras de áudio, num espaço vetorial de atributos, adequado para problemas de classificação. O método proposto baseia-se na utilização de uma representação vetorial compacta. Exatidão de 80% utilizando validação cruzada com 10 *folds*.
[Roma et al., 2013]	Classificação do cenário acústico	Exploram o uso de atributos de Análise de Quantificação de Recorrência (quantificar padrões que emergem em parcelas de recorrência). Esses recursos são calculados através de uma matriz de similaridade limiar calculado a partir de janelas de atributos obtidos por MFCC. Combinando com as estatísticas tradicionais MFCC, melhoram a exatidão quando se utiliza um classificador SVM. Obtida exatidão de 71%.
[Schröder et al., 2013]	Deteção de eventos dentro de um cenário	Descreve um sistema de classificação que tem por base atributos espectro-temporais dos sons e um modelo HMM de duas camadas como *back-end*. Obtido 85% de *F-Score*.

2.2.2 Utilização de *motifs*

O estudo dos *motifs* tem sido efetuado em várias áreas como na medicina [Gomes et al., 2013], na captura de movimentos [Minnen et al., 2006], na meteorologia [McGovern et al., 2007], na video-vigilância [Hamid et al., 2005], entre outras. Mas quanto à sua aplicação no reconhecimento de sons acústicos e urbanos, nunca foi utilizado.

Foi proposto em [Ferreira et al, 2006] um algoritmo para extração de *motifs* de series temporais. Mais tarde, foi proposto em [Castro and Azevedo, 2010] uma implementação melhorada do mesmo entitulada MrMotif.

O MrMotif, algoritmo de extração de *motif*s para classificação, foi utilizado na classificação de sons cardíacos com o objetivo de encontrar patologias cardíacas [Gomes et al., 2013]. Para a realização das experiências foi utilizado um *dataset* contituido por 312 sons cardíacos gravados com um dispositivo entitulado "DigiScope Collector". Os sons encontravam-se divididos em 3 classes: Normal (200 sons), Murmur (66 sons) e Extrasystole (46 sons). Os resultados conseguiram superar os obtidos pelo vencedor do concurso "PASCAL Classifying Heart Sounds Challenge" [Bentley] que se baseava na deteção de picos de som e na distância dos mesmos para a classificação [Gomes and Pereira, 2012]. Posteriormente, os resultados foram melhorados em [Gomes et al., 2014] e em [Oliveira et al., 2014] com algumas alterações na metodologia.

3 Data mining

Neste capítulo são apresentados as técnicas e algoritmos de *data mining* utilizados neste trabalho.

Na subsecção 3.1, são apresentados os algoritmos de classificação e o método de avaliação dos modelos.

Na subsecção 3.2, é feita uma introdução à metodologia utilizada neste trabalho para descobrir *motifs* nos ficheiros de som.

Na subsecção 3.3, descreve-se também a abordagem de classificação com base nos coeficientes MFCC para extração de atributos.

3.1 Algoritmos de classificação

Um algoritmo de classificação obtém modelos de classificação a partir de conjuntos de exemplos de treino etiquetados. Cada exemplo pertence a uma classe conhecida, sendo essa informação (etiqueta) fornecida ao algoritmo de classificação.

Um modelo de classificação é capaz de, dado um novo exemplo não etiquetado, determinar a sua classe, possivelmente com um grau de incerteza. A este processo de construção de um modelo a partir de dados previamente etiquetados chama-se também de "aprendizagem supervisionada" [Han e Kamber, 2006].

Nesta secção, iremos descrever os algoritmos de classificação utilizados neste trabalho: Decision Trees (J48), Random Forest e Support Vector Machine (SVM).

3.1.1 Decision Trees (J48)

O algoritmo de árvores de decisão (*decision trees*) utiliza a estratégia *dividir para conquistar*, onde um problema complexo é dividido em problemas mais simples. Trata-se de um método baseado em procura, que produz uma estrutura em árvore com base nos exemplos do conjunto de treino. A árvore de decisão é um grafo acíclico direcionado em que cada nó ou é um nó de divisão, com dois ou mais sucessores, ou um nó folha. Os nós de divisão representam um teste a um atributo. Os nós folha são portadores da informação da categoria. As árvores de decisão podem ser bastante precisas e são aplicáveis nas mais variadas áreas de atividade [Han e Kamber, 2006].

O J48 é uma implementação *open source* em Java do algoritmo de classificação C4.5 [Hall et al., 2009]. O J48 é utilizado para gerar árvores de decisão. Este algoritmo baseia-se no critério de ganho de informação para gerar as árvores, isto é, o atributo com maior ganho de informação

será a raiz da árvore, seguido dos restantes por ordem decrescente de ganho de informação [Quinlan, 1993].

Dado um conjunto *S* de exemplos, o processo usado pelo J48 para gerar as árvores de decisão pode ser descrito pelos seguintes passos:

1. Verificar os casos base
2. Identificar o atributo, *A*, com maior ganho de informação
3. Criar nó com o melhor atributo
4. Dividir *S* em subconjuntos $S_1,S_2,...S_n$ com base nos valores A_i do atributo *A* obtido no passo 2
5. Repetir os passos anteriores para $S_1,S_2,...,S_n$.

Em que os casos base são:

- Todas as instâncias do conjunto de exemplos pertencem à mesma classe (retorna uma folha etiquetada com essa classe).
- O conjunto de exemplos está vazio (retorna uma folha de falha).
- O conjunto de exemplos não tem atributos (retorna uma folha etiquetada com a classe mais frequente).

Para um atributo *A* com *m* valores A_i, o cálculo do ganho de informação é feito usando a seguinte formula:

$G(S,A) = E(S) - \sum_{i=1}^{m} \Pr(A_i) E(S_i)$	(1)

onde *E(S)* é a entropia do conjunto *S*, $Pr(A_i)$ é a probabilidade do atributo *A* ter o valor A_i em *S*, E(S_i) é a entropia do subconjunto de S com instâncias para as quais *A* tem o valor A_i.

A entropia (impureza) de um conjunto de exemplos *S* é calculada da seguinte forma:

$\sum_{i=1}^{n} - \Pr(C_i) * \log_2 \Pr(C_i)$	(2)

onde $Pr(C_i)$ é a probabilidade da classe C_i no conjunto *S*, sendo *n* o número de classes.

3.1.2 Random Forest

Segundo Breiman, o Random Forest é um classificador que consiste numa coleção de classificadores baseados em árvores de decisão $\{h(x, \Theta_k), k = 1, \dots\}$, onde $\{\Theta_k\}$ são vetores aleatórios, independentes e com distribuição idêntica, e cada arvore dá origem a um voto para a escolha da classe mais popular do input x [Breiman, 2001].

Ou seja, o Random Forest produz um modelo constituído por *N* árvores de decisão (*ensemble*), onde cada árvore é baseada num certo número de instâncias do conjunto de treino, escolhidas aleatoriamente. Cada nó de cada árvore é construído a partir de um subconjunto aleatório dos atributos. Ao receber uma instância de teste cada árvore irá decidir (votar) sobre qual a classe a que pertence. A classe mais votada será a classe prevista pelo modelo.

As árvores são construídas utilizando o índice Gini (Equação 3) para selecionar atributos em vez do Ganho de Informação (acima referido).

$Gini\_index(S) = 1 - \sum_{j=1}^{n} \Pr(C_j)^2$	(3)

sendo $Pr(C_j)$ a probabilidade de um exemplo do conjunto *S* ser da classe C_j e *n* o número de classes [Han e Kamber, 2006].

O Random Forest é extramente eficiente, mesmo com grandes quantidades de dados, conseguindo muito bons resultados em pouco tempo de processamento. Em diversas experiências, consegue melhor exatidão (*accuracy*) do que qualquer outro algoritmo e não existe sobreajustamento, independentemente do número de árvores geradas [Breiman, 2001].

A taxa de erro no Random Forest depende da correlação entre cada uma das árvores geradas e da força individual de cada árvore. Num modelo Random Forest, quanto menor for a correlação entre as árvores e menor a taxa de erro de cada uma, melhor é a exatidão.

3.1.3 Support Vector Machine

O SVM é um algoritmo de classificação e regressão que foi introduzido em 1992 por Bernhard E. Boser, Isabelle M. Guyon, e Vladimir N. Vapnik [Boser et al., 1992]. Nesta secção, abordaremos apenas o caso de SVMs lineares (com margens rígidas).

Para construir um modelo, o algoritmo SVM, representa as instâncias de treino como pontos no espaço. O objetivo é encontrar um hiperplano que melhor divida os dados. O melhor hiperplano será aquele cuja margem (distância entre o hiperplano e as instâncias de cada classe mais próximas do hiperplano) seja máxima. A cada lado do hiperplano irá corresponder uma classe. O modelo irá fazer a previsão de uma instância de teste mediante o lado do hiperplano em que fica a sua respetiva representação no espaço. Na Figura 1 encontra-se representado um exemplo de uma representação no espaço de várias instâncias com duas classes e o respetivo hiperplano de separação ótimo.

Seja *T* um conjunto de treino com *n* exemplos $x_i \in X$ e as respetivas etiquetas $y_i \in Y$, em que *X* constitui o espaço dos exemplos e $Y = \{-1, +1\}$. A função que define um hiperplano é apresentada na Equação 3, em que $w \cdot x$ é o produto escalar entre os vetores *w* e *x*, $w \in X$ é o vetor normal ao hiperplano descrito e $\frac{b}{||w||}$ corresponde à distância do hiperplano em relação à origem [Lorena and Carvalho, 2007].

$f(x) = w \bullet x + b = 0$	(4)

Para obter a classificação é utilizada a função sinal apresentada na Equação 4.

$g(\mathrm{x}) = \mathrm{sgn}(f(\mathrm{x})) = \begin{cases} +1 \text{ se } \mathrm{w} \cdot \mathrm{x} + b > 0 \\ -1 \text{ se } \mathrm{w} \cdot \mathrm{x} + b < 0 \end{cases}$	(5)

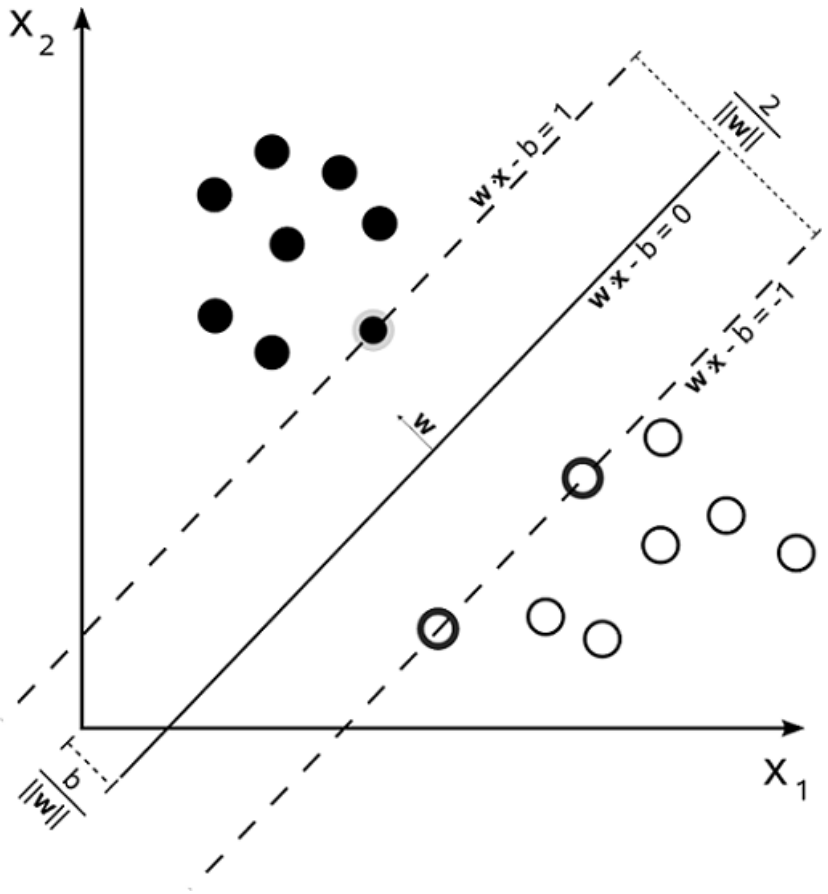

Figura 1 – Exemplo de um SVM linear com duas classes

Este método funciona apenas quando o conjunto de treino é linearmente separável. Porém, em situações reais, pode não ser fácil encontrar dados nesta situação por diversos motivos (presença de ruido, *outliers*, problema não linear) [Gama et al., 2012]. Existem ainda SVMs com margens suaves (extensão do caso linear) e SVMs não lineares.

O Weka disponibiliza o algoritmo SMO (Sequential Minimal Optimization). Trata-se de um algoritmo eficiente para a implementação da técnica SVM. Trata os valores em falta e transforma atributos nominais em binários. Normaliza, por omissão, todos os atributos [Hall et al., 2009].

3.1.4 Avaliação

Para avaliar a capacidade de previsão dos modelos foi usado o método de validação cruzada (*cross-validation*) com 10 subconjuntos (*folds*). O método consiste em dividir o conjunto de dados de entrada em 10 subconjuntos e construir 10 submodelos, onde em cada um são utilizados 9 conjuntos para treino e um conjunto para teste. Em cada submodelo, o conjunto de teste usado será diferente, assim como os dados de treino que não terão presentes os dados de teste. Para cada submodelo é então calculada a respetiva exatidão (*accuracy*) e o resultado da avaliação será a média das exatidões. Quanto maior for a exatidão, melhor será o modelo. Na Figura 2 encontra-se uma representação do processo de validação cruzada.

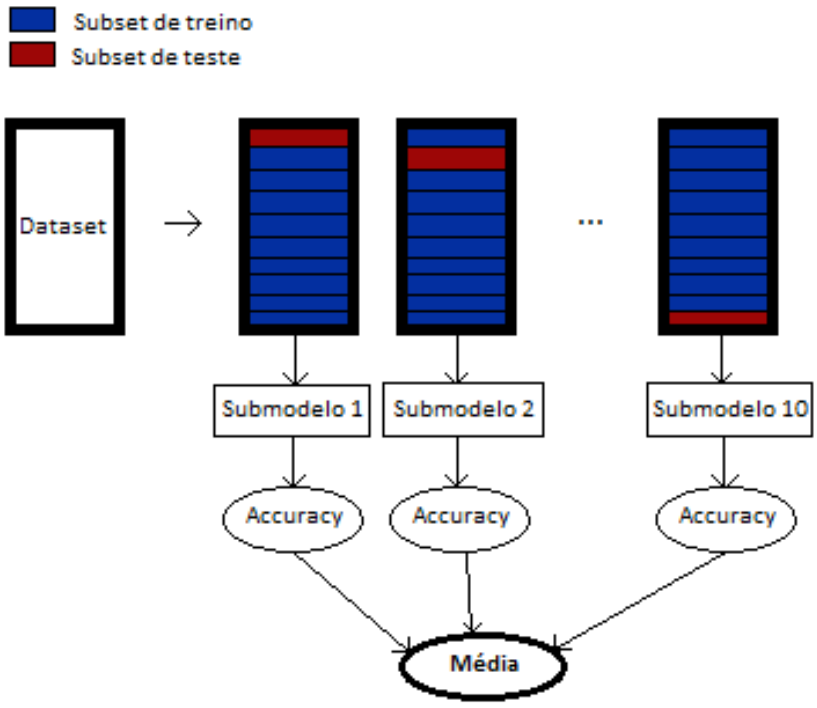

Figura 2 – 10 *fold cross-validation*

3.2 Descoberta de motifs em séries temporais

Um *motif* numa série temporal é um padrão frequente existente na série, isto é, uma repetição de uma determinada secção na série temporal. Na Figura 3 é apresentado um exemplo de um *motif* com três repetições.

Existem métodos/ferramentas para a extração de motifs em séries temporais. Uma dessas ferramentas, o MrMotif (Multiresolution Motif Discovery in Time Series) [Castro and Azevedo, 2010], foi utilizada neste trabalho. O código do MrMotif, desenvolvido em linguagem Java, é disponibilizado pelos autores.

A descoberta de *motifs* tem utilizações práticas na medicina [Gomes et al., 2013], na captura de movimentos [Minnen et al., 2006], na meteorologia [McGovern et al., 2007], na video-vigilância [Hamid et al., 2005], entre outras.

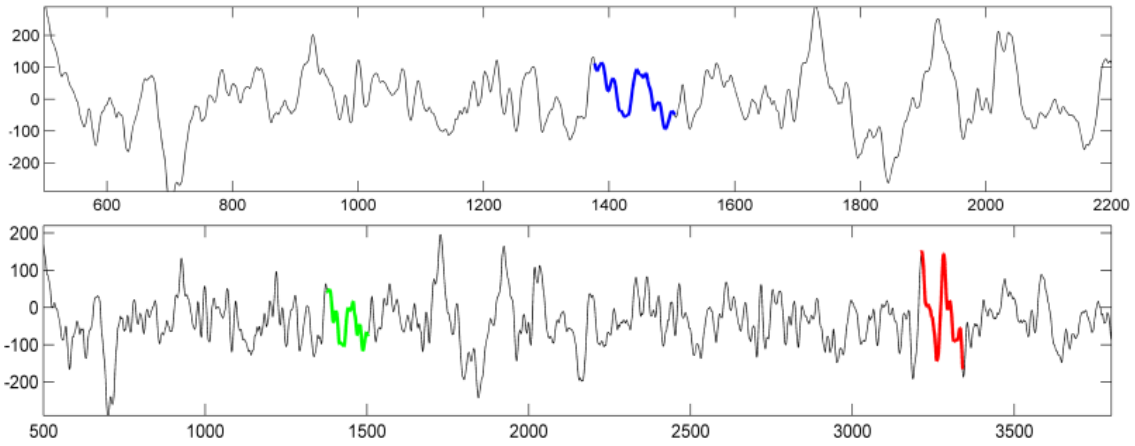

Figura 3 – Exemplo de um *motif* com 3 repetições (obtido de http://alfa.di.uminho.pt/~castro/mrmotif/)

3.3 MFCC

Os MFCC (Mel Frequency Cepstral Coefficients) são características de um som que são calculadas a partir da frequência e que têm em conta a perceção do ouvido humano.

Os MFCC foram introduzidos por Steven Davis e Paul Mermelstein em 1980 [Davis and Mermelstein, 1980]. Começaram por ser utilizados na área do reconhecimento automático da fala, mas já foram utilizados com sucesso para reconhecimento e classificação de sons acústicos [Geiger et al., 2013], [Elizalde et al., 2013], [Beritelli, 2008]. [Ntalampiras et al, 2008], [Salamon et al., 2014].

Na Figura 4 encontra-se representado o funcionamento de um algoritmo típico para obter os MFCC a partir de um som. Primeiro, o som é dividido em pequenas frames e é calculada a Tranformada Discreta de Fourier de cada frame. Depois, é aplicado o Mel Filterbank e é calculado o logaritmo das energias obtidas. Por fim, é calculada a Transformada Discreta de Cosseno dos logaritmos, que gera os MFCC.

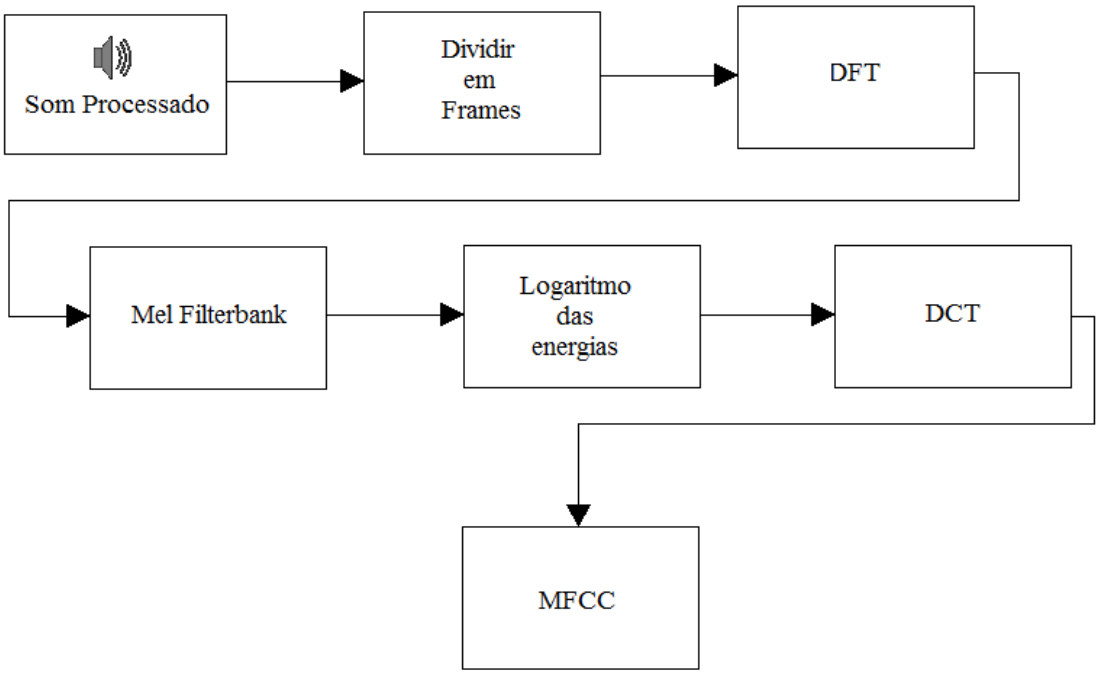

Figura 4 – Sequência de um algoritmo de extração de MFCC típico

4 Ferramentas e Tecnologias utilizadas

Neste capítulo apresentam-se as ferramentas e tecnologias utilizadas no desenvolvimento deste trabalho.

4.1 Linguagem de Programação

A linguagem de programação utilizada neste trabalho foi o Java. Esta escolha deve-se ao fato de este trabalho utilizar código já desenvolvido anteriormente nesta linguagem.

4.1.1 Java

Java é uma linguagem de programação e uma plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995 [Oracle, a]. Em 2010, a Sun Microsystems foi adquirida pela Oracle, e é a atual detentora da tecnologia [Oracle, c]. A 13 de Novembro de 2006 a Sun Microsystems lançou o Java sob os termos da *GNU General Public Licence*, tornando-o um *software* livre [Lee, 2006].

Os programas desenvolvidos em Java são independentes do sistema operativo. Estes são compilados para um *bytecode*, que é executado numa máquina virtual que permite que as aplicações desenvolvidas em Java possam ser executadas em qualquer sistema operativo suportado pela plataforma Java sem ser necessário quaisquer alterações [Brito].

Por ser multiplataforma, a tecnologia está presente em inúmeros tipos de dispositivos diferentes [Oracle, b].

4.1.2 NetBeans IDE

O NetBeans IDE é considerado o IDE oficial do Java. Gratuito e de código aberto, o NetBeans IDE é principalmente direcionado para o desenvolvimento em Java, mas também suporta outras linguagens de programação tais como PHP, C/C++ e HTML5. É codificado em Java e pode ser utilizado em qualquer sistema operativo suportado pelo Java [NetBeans].

4.1.3 Android Studio

O Android Studio é considerado atualmente o IDE oficial para desenvolvimento de aplicações para o sistema operativo móvel Android [Android Developers, a], e que veio substituir o desenvolvimento através do Eclipse ADT (Android Developer Tools) [Android Developers, b]. O Android Studio foi desenvolvido tendo por base o IDE InteliJ IDEA da JetBrains [JetBrains].

4.2 Data mining

O *data mining* (mineração de dados) consiste no processo de aplicação de algoritmos específicos de extração de padrões em grandes conjuntos de dados, com o objetivo de extrair informação útil.

4.2.1 Weka

O Weka [Witten, 2005] é um *software open source* de *data mining* codificado em Java. Suporta múltiplos algoritmos de *machine learning* direcionados para o *data mining*. Contém ferramentas para pré-processamento de dados, classificação, regressão, *clustering*, regras de associação, e visualização [Hall, et al., 2009]. As funcionalidades do Weka podem ser utilizadas através da sua UI, ou podem ser invocadas através de código Java.

5 Abordagem baseada em *motifs e MFCC*

Neste capítulo, descreve-se a abordagem de classificação usando atributos baseados em motifs e MFCC, as experiências realizadas e os resultados obtidos.

5.1 Conjunto de dados

Esta abordagem foi testada usando dois conjuntos de dados (*datasets*):

- UrbanSound – Contém 1302 gravações de ambientes urbanos obtidas em [Freesound]. Cada som contém uma ou mais ocorrências de um evento (classe), que pode ser de um dos seguintes tipos: *air_conditioner* (ar condicionado), *car_horn* (buzina de automóvel), *children_playing* (crianças a brincar), *dog_bark* (cão/cães a ladrar), *drilling* (berbequim), *engine_idling* (motor de automóvel), *gun_shot* (disparos de arma de fogo), *jackhammer* (martelo pneumático), *siren* (sirene) e *street_music* (musica de rua).
- UrbanSound8K – Contém apenas os eventos relevantes ocorridos nos sons do *dataset* UrbanSound (retirando os segmentos de som doutras naturezas) divididos em 8732 ficheiros áudio de aproximadamente 4 segundos de duração. Os ficheiros estão divididos em *10 folds*.

Os *datasets* foram construídos por Justin Salamon, Christopher Jacoby, Juan Pablo Bello [Salamon et al., 2014] e podem ser encontrados em [Salamon et al.].

5.2 Classificação com atributos baseados em motifs e MFCC

O principal objetivo do trabalho foi classificar sons urbanos através da descoberta de *motifs* nos sons. Foi também feita classificação com base nos coeficientes MFCC dos sons para comparação de abordagens. A escolha da abordagem por MFCC deve-se ao facto de ser muito comum na literatura.

Em qualquer dos casos, o primeiro passo consiste em efetuar o pré-processamento dos sons. Para tal, utiliza-se um *package* em Java criado anteriormente pela equipa que desenvolveu o projeto em que este trabalho se insere.

No pré-processamento o programa em java começa por ler os ficheiros “WAV” e transforma-os em dados do tipo *double*. Este código foi alterado a partir da classe WavReader.java do projeto webm, disponibilizado em [WebM].

Neste pré-processamento, os sons são filtrados, decimados e é construído o envelope de energia de Shannon. O resultado deste processamento serve como *input* para o passo seguinte.

O passo seguinte consiste em extrair atributos dos sons. Neste trabalho, foram utilizadas duas abordagens para extrair atributos: *motifs* e MFCC. No caso dos *motifs* os sons são vistos como séries temporais.

Para extração dos *motifs* é utilizada uma ferramenta, codificada em Java por Nuno Casto e Paulo Azevedo, intitulada Multiresolution Motif Discovery (MrMotif) [Castro and Azevedo, 2010].

O algoritmo do MrMotif começa por fazer a codificação da série temporal utilizando o algoritmo iSAX.

A metodologia SAX (Symbolic Aggregate approXimation) [Lin et al.,2002] transforma uma série temporal contínua numa sequência de símbolos discretos. Ou seja, o SAX (T,ω,α) converte uma serie temporal T numa sequência de palavras de tamanho ω, utilizando os símbolos do alfabeto α. O número de símbolos do alfabeto α designa-se por resolução. Para cada janela, o SAX começa por dividir a série temporal em ω segmentos utilizando o algoritmo PAA (Piecewise Aggregate Approximation) e de seguida atribuí um símbolo a cada segmento conforme a amplitude da serie temporal. Na Figura 5 está ilustrado um exemplo da aplicação do SAX a um segmento de uma serie temporal. Com um alfabeto {1,2,3,4}, aplicando o SAX ao segmento representado na figura resultaria a palavra {1,1,3,3,1,1,1,1} [Gomes et al., 2014].

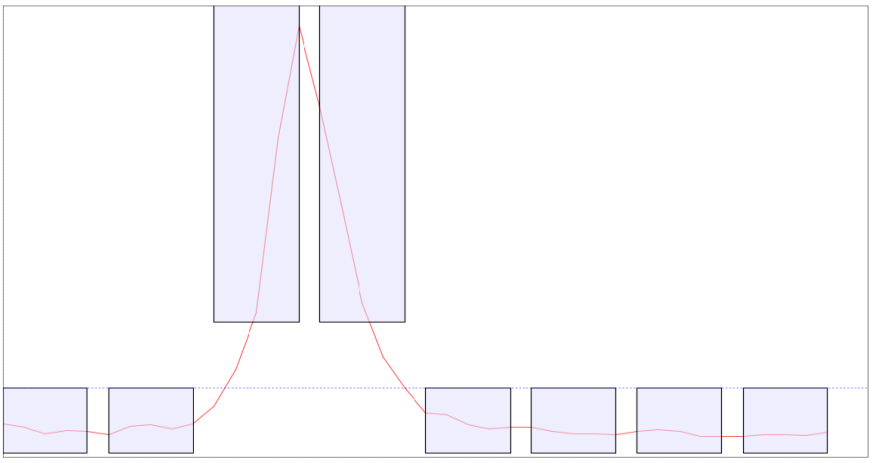

Figura 5 – SAX aplicado a um segmento de uma serie temporal – obtida com a ferramenta iMotifs

O iSAX (indexable Symbolic Aggregate approXimation) [Shieh and Keogh, 2008], utilizado no MrMotif, difere do SAX na medida em que permite diferentes resoluções da mesma palavra e é mais eficiente e escalável, conseguindo processar grandes *datasets*, com milhões de séries temporais e até um terabyte de tamanho.

O MrMotif começa por usar o iSAX com resoluções mais baixas e vai expandindo para resoluções mais altas. A resolução mínima é 2 e a máxima é 64, mas só os valores 2, 4, 8, 16, 32

e 64 são possíveis. Por cada resolução, converte as séries temporais em palavras iSAX e atualiza um contador dos top-*k motifs* atuais indexados pelo iSAX. Uma série temporal é considerada uma repetição de outra se tiverem a mesma representação simbólica. O resultado final será as top-*K* palavras de determinado tamanho (especificado por parâmetro) que mais se repetem (as mais frequentes).

Resumidamente, o objetivo do MrMotif é: dada uma base de dados de séries temporais *D*, para um tamanho de *motif m* e um parâmetro *K*, por cada resolução (g_{min},g_{min} x 2,..., g_{max}) encontrar os top-*K motifs*.

O MrMotif, para além de devolver as top-*K* palavras mais frequentes, devolve também um novo *dataset* onde cada linha corresponde a uma série temporal. Cada top-*K motif* é um atributo candidato para o novo *dataset*. O valor do atributo é a frequência do *motif* na série temporal correspondente. Esse novo *dataset* gerado é depois usado para a classificação. Na Figura 6 está representado um excerto de um *dataset* gerado pelo MrMotif.

```
P1R4,  P2R4,  P3R4,  P4R4,  P5R4,  P6R4,  P7R4,  P8R4,  P9R4,  P10R4,  Classe
 0,     0,     0,     6,     1,     6,     8,     9,     7,     2,     air_conditioner
 0,     0,     2,     1,     0,     1,     5,     3,     2,     0,     air_conditioner
 0,     0,     1,     0,     0,     5,     2,     1,     2,     1,     air_conditioner
```

Figura 6 – Excerto de um *dataset* gerado pelo MrMotif

Para extração dos MFCC foi feita uma implementação em Java pela orientadora deste trabalho a partir de um algoritmo originalmente codificado em Python por James Lyons [Lyons].

O algoritmo para extração dos MFCC funciona da seguinte forma:

1. Divide a série temporal em pequenas *frames*. No caso concreto deste trabalho as séries são divididas em *frames* de 40 milissegundos.
2. Calcula a transformada discreta de Fourier (DFT) e o periodograma de cada *frame*. Para calcular a DFT é usada a seguinte formula:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \qquad 1 \leq k \leq K \tag{6}$$

, onde *si(n)* é o sinal da frame *n*, *h(n)* é o tamanho da janela *n* e *K* o tamanho da DFT. Para calcular o periodograma é utilizada a fórmula:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \quad (7)$$

3. É calculado e aplicado o Mel-Spaced filterbank. É composto por entre 20 a 40 filtros triangulares que são aplicados ao periodograma calculado no passo anterior.
4. Calcula o logaritmo das energias resultantes no passo 3
5. Obtém a transformada discreta de cosseno (DCT) do resultado obtido no passo anterior para obter os coeficientes. Apenas os primeiros *K* coeficientes são mantidos e os restantes são descartados.

Por fim, é gerado um *dataset*, onde cada linha corresponde a uma série temporal e que contém a média e desvio padrão dos coeficientes de todas as *frames* da série temporal. Na Figura 7 está representado um excerto de um *dataset* gerado pela ferramenta descrita.

```
media0,   media1,  media2,  ..., media12,  desvio0, desvio1, desvio2,..., desvio12,classe
-6.552110,0.044820,-0.804583,...,-0.078737,6.484418,0.357620,0.798442,...,0.291187,air_conditioner
6.253899,-1.107637,-1.057427,...,-0.063792,6.333682,1.108795,1.055182,...,0.323983,air_conditioner
-6.487397,0.065353,-0.790978,...,6.398688,0.378886,0.752062,0.386420,...,0.296983,air_conditioner
```

Figura 7 – Excerto de um *dataset* gerado pela ferramenta dos *MFCC*

Usando as metodologias descritas acima (MrMotif e MFCC), geram-se respetivamente dois *datasets* com todos os sons de todas as classes. Para estudar as abordagens com mais detalhe geramos também, para cada uma delas, um *dataset* para cada par de classes diferentes. Foram feitas combinações com todos os pares de classes possíveis e um *dataset* com todas as classes. Isto perfaz um total de 46 *dataset*s por abordagem.

Estes *datasets* foram utilizados para avaliar experimentalmente a capacidade discriminante das abordagens de construção de atributos. Neste processo foi utilizado o *software* de *data mining* Weka.

Ao longo deste trabalho, foram realizadas diversas experiências. Nessas experiências foram avaliadas diversas variações de parâmetros relacionados quer com os métodos de aquisição dos atributos quer nos algoritmos de classificação.

Nas experiências foi utilizada quer a interface gráfica do Weka quer a sua API, através de um programa em Java desenvolvido para o efeito.

Analisamos os resultados usando os mesmos métodos de avaliação nos dois *datasets* UrbanSound e UrbanSound8K, descritos anteriormente.

Nas secções seguintes serão descritas algumas das experiências realizadas, para cada um dos *datasets*, para todas as classes, para pares de classes e para um conjunto particular de classes.

5.3 Aplicação dos algoritmos ao dataset UrbanSound

Nesta secção, serão apresentadas experiências utilizando o *dataset* UrbanSound.

Devido a limitações da ferramenta WavReader, apenas foram utilizados os ficheiros que se encontravam no formato Wave PCM [RFC 2361]. A Tabela 3 mostra o número de instâncias de cada classe utilizada nas primeiras experiências.

Tabela 3 – Número de instâncias de cada classe (UrbanSound *dataset*)

air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
32	95	105	166	75	54	81	19	51	83

5.3.1 Todas as classes usando *motifs*

No primeiro conjunto de experiências, o conjunto de atributos extraído para classificação foi obtido usando a abordagem baseada em *motifs*. Foi utilizada a metodologia descrita anteriormente para extrair os *motifs*. Foram testadas variadas combinações de parâmetros de entrada do programa MrMotif. Os exemplos seguintes foram obtidos utilizando os parâmetros com os quais se conseguiram os melhores resultados em abordagens anteriores. A saber:

Resolução: 8
Top-k: 40
Tamanho dos *motifs*: 40
Overlap: 10

Para a classificação foi utilizado o método de validação cruzada com 10 *folds*. Nos primeiros testes foram construídos modelos utilizando os algoritmos J48 e RandomForest com 20 árvores e 6 *K-features*. Mais tarde, foi utilizado os SVM e foram feitas experiências com diferentes parâmetros no RandomForest: 100 árvores e 2 *K- features*, 100 árvores e 5 *K- features*, 200 árvores e 2 *K- features*, 200 árvores e 5 *K- features*. Como já referido anteriormente, em cada experiência são construídos 46 modelos que correspondem a 45 modelos com todos os pares de classes possíveis e 1 modelo onde são utilizadas as instâncias de todas as classes.

Na Tabela 4 está representada a *accuracy* dos modelos gerados com o algoritmo RandomForest com 200 árvores e 5 *K- features*, que foi o que obteve melhores resultados (segundo esta medida de avaliação).

Tabela 4 – Resultados (*accuracy*) utilizando *motifs* e o RandomForest com 200 árvores e 5 K-features (UrbanSound *dataset*)

Classes	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	Street_music
air_conditioner	72,44	70,07	81,31	70,09	60,47	69,91	50,98	74,70	69,57
car_horn	-	64,00	69,73	55,88	67,79	71,59	80,70	67,12	64,04
children_playing	-	-	74,17	53,33	67,92	62,90	82,26	77,56	60,64
dog_bark	-	-	-	78,01	77,27	75,71	89,73	77,88	79,12
drilling	-	-	-	-	66,67	66,67	79,79	76,98	57,59
engine_idling	-	-	-	-	-	61,48	72,60	71,43	58,39
gun_shot	-	-	-	-	-	-	80,00	76,52	66,46
jackhammer	-	-	-	-	-	-	-	75,71	82,35
siren	-	-	-	-	-	-	-	-	79,85
Media Pares	70,88%								
Todas as classes	30,88%								

5.3.2 Todas as classes usando MFCC

Na fase seguinte, foram realizadas as experiências utilizando a abordagem baseada em MFCC para criar os *datasets* de atributos. Nos exemplos apresentados a seguir foram usados os seguintes parâmetros:

Número de coeficientes: 13
Número de filtros: 40
Tamanho da janela: 1024

Foram utilizados os mesmos algoritmos de classificação referidos na secção 5.3.1.

Na Tabela 5 são mostradas as *accuracies* obtidas nos modelos gerados com MFCC utilizando o algoritmo RandomForest com 200 árvores e 5 *K- features*.

Tabela 5 – Resultados (accuracy) utilizando MFCC e o RandomForest com 200 àrvores e 5 K-features (UrbanSound *dataset*)

Classes	car_horn	children_playing	dog_bark	drilling	Engine_idling	gun_shot	jackhammer	Siren	Street_music
air_conditioner	75,59	85,40	85,86	83,18	68,60	82,30%	70,59	81,93	80,00
car_horn	-	82,00	79,69	77,06	76,51	82,95%	83,33	84,93	84,27
children_playing	-	-	81,55	83,89	75,47	84,95%	87,90	82,69	73,40
dog_bark	-	-	-	80,08	84,55	80,16%	92,97	84,79	85,14
drilling	-	-	-	-	82,95	83,97%	76,60	83,33	86,71

engine_idling	-	-	-	-	-	84,44%	84,93	79,05	79,56
gun_shot	-	-	-	-	-	-	91,00	84,85	87,80
jackhammer	-	-	-	-	-	-	-	91,43	87,25
siren	-	-	-	-	-	-	-	-	87,31
Media Pares			82,51						
Todas as classes			52,17						

5.3.3 Todas as classes: comparação *motifs* vs MFCC

Nesta subsecção apresentam-se e comparam-se os melhores resultados obtidos usando *motifs* e MFCC, para todas as classes.

Na Figura 8 é feita a comparação entre *motifs* e MFCC com todos os algoritmos testados, utilizando a *accuracy* do modelo com todas as classes, respetivamente.

Podemos observar que os resultados obtidos usando MFCC obteve uma melhor *accuracy* em todos os algoritmos usados, sendo com o Random Forest com 200 árvores e 5 *features* que obteve melhor resultado com 52,17% de *accuracy*. Já com os *motifs* o melhor resultado foi 31,27% de *accuracy* obtido com o Random Forest com 100 árvores e 5 *features*.

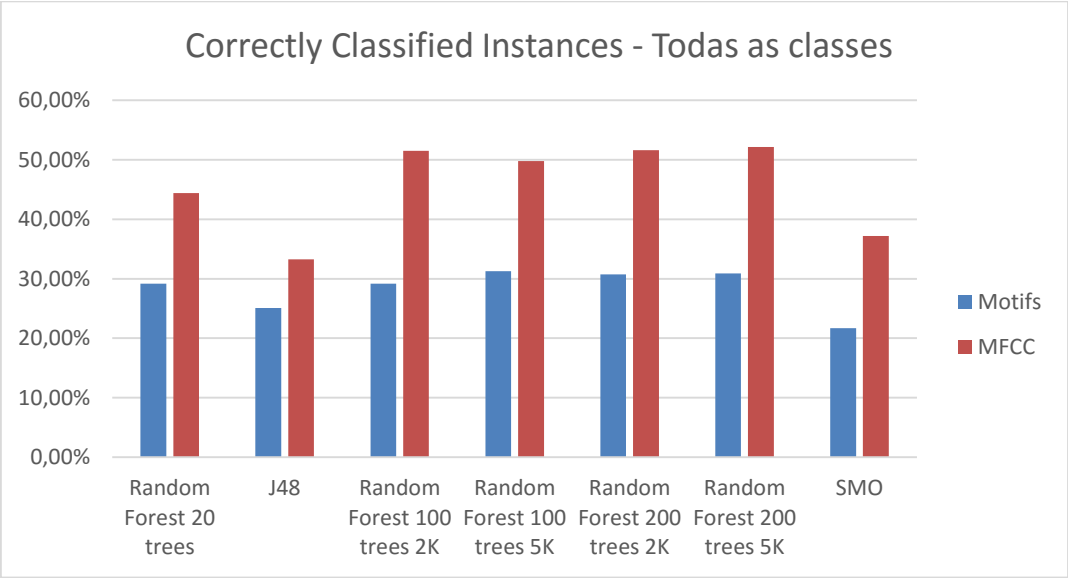

Figura 8 - *Motifs* vs MFCC – todas as classes (UrbanSound *dataset*)

5.3.4 Todos os pares de classes

Nesta subsecção, apresentam-se e comparam-se os melhores resultados obtidos usando *motifs* e MFCC, para todos os pares de classes.

Na Figura 9 é feita a comparação entre *motifs* e MFCC para todos os algoritmos testados, utilizando como referência a média das *accuracies* dos pares de classes.

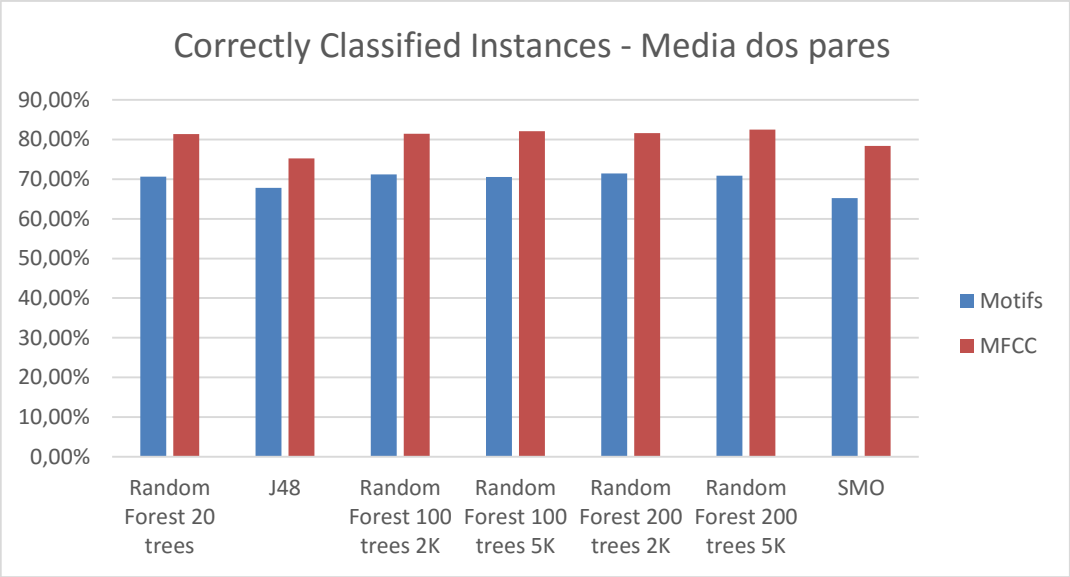

Figura 9 - *Motifs* vs MFCC – média dos pares (UrbanSound *dataset*)

Como se pode observar nos gráficos, a classificação tendo por base os MFCC consegue melhores resultados comparativamente à classificação baseada em *motifs*. No entanto, existem pares de classes que merecem especial atenção. Em [Salamon, et al., 2014] foi feita a classificação com base nos MFCC e é mencionado que os modelos construídos com os pares "air_conditioner + engine_idling", "drilling + jackhammer" e "children_playing + street_music" obtiveram piores resultados. Segundo os autores, este resultado deve-se ao facto de os MFCC se basearem no timbre, e os sons nesses pares de classes terem um timbre semelhante. Esse problema já não acontece quando se utilizam os *motifs*, pois não são baseados no timbre mas sim nos padrões encontrados nos sons.

5.3.5 Pares de classes difíceis

Nesta secção, apresentam-se resultados obtidos apenas nos pares de classes em que a abordagem usando MFCC tem mais dificuldade em discriminar. Como foi referido na secção

anterior, trata-se dos pares pares “air_conditioner + engine_idling”, “drilling + jackhammer” e “children_playing + street_music”. A este conjunto juntámos o par “siren + street_music”, onde se observou comportamento semelhante. Nessas experiências tentou-se melhorar os resultados da classificação baseada em *motifs*, fazendo variar os parâmetros no MrMotif.

Na Figura 10 podem ser observados os resultados obtidos. Os primeiros dois conjuntos de barras no gráfico dizem respeito aos resultados obtidos na experiência anterior com *motifs* e MFCC e os restantes são os resultados obtidos com diferentes parâmetros no MrMotif. Os parâmetros utilizados nas várias experiências são apresentados na Tabela 6.

Tabela 6 – Parâmetros usados no MrMotif nas experiências com pares de classes difíceis

	Resolução	Tamanho dos *motifs*	Overlap	Top-k
Expriência 1	16	40	10	40
Expriência 2	4	40	10	40
Expriência 3	8	40	20	40
Expriência 4	8	40	5	40
Expriência 5	8	80	10	40
Expriência 6	8	20	10	40
Expriência 7	4	40	5	40

Os valores apresentados no gráfico correspondem ao resultado obtido utilizando o algoritmo de classificação Random Forest com 200 árvores e 5 *K-features* e foi usada validação cruzada.

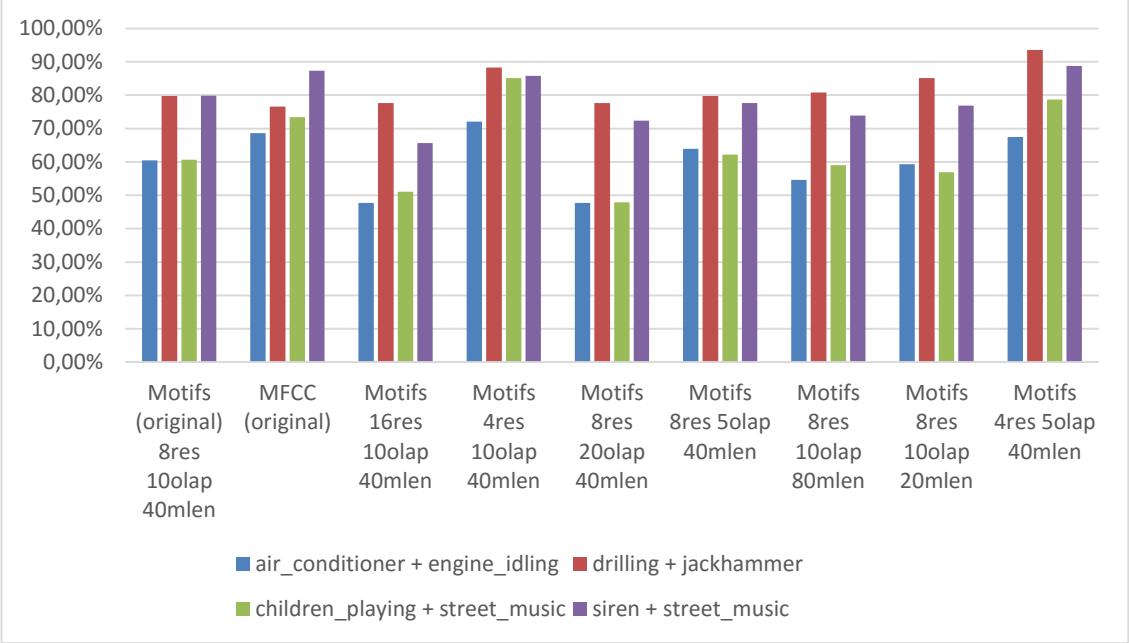

Figura 10 - Resultados dos testes a pares de classes "especiais" efetuados com diferentes parâmetros no MrMotif (UrbanSound *dataset*)

Como se pode observar na Figura 10, os resultados significativamente melhores são observados para a resolução 4, ultrapassando os resultados obtidos com os MFCC. Na Tabela 7 podem ser observados os melhores valores obtidos.

Tabela 7 – MFCC vs *motifs* de resolução 4 em pares difíceis (UrbanSound *dataset*)

	MFCC	*Motifs* 4res 10olap 40mlen	*Motifs* 4res 10olap 40mlen
air_conditioner + engine_idling	68,60	**72,09**	67,44
drilling + jackhammer	76,60	88,30	**93,62**
children_playing + street_music	73,40	**85,11**	78,72
siren + street_music	87,31	85,82	**88,81**

Observa-se também uma melhoria nos resultados quando se faz uma redução do tamanho dos *motifs* e do *overlap*. Porém, estas melhorias nos resultados não são significativas.

5.3.6 Combinação dos atributos *motifs* e MFCC

As experiências realizadas, descritas nas secções anteriores, indicam o seguinte:

- A classificação através da abordagem MFCC é melhor para sons com timbres distintos (buzina de automóvel, musica de rua, etc.).
- A classificação utilizando a abordagem *motifs* é melhor para sons com ritmo (motor de automóvel, martelo pneumático, etc.).

Feitas estas observações, foram feitas experiências onde foram criados modelos de classificação com os atributos dos MFCC e *motifs* combinados no mesmo *dataset*. Isto é, juntou-se a capacidade de classificar sons de timbres distintos com a capacidade de classificar sons de diferentes ritmos, com o objetivo de melhorar os resultados da classificação. Na Figura 11 está representado um excerto de um *dataset* com atributos dos MFCC e *motifs*.

```
  media0,..., media12,  desvio0,...,desvio12,P1R4,...,P40R4, Classe
8.997569,...,-0.201543,9.311436,...,0.354275,  5, ...,  0,   car_horn
8.684243,...,-0.157421,8.971719,...,0.364006,  5, ...,  2,   car_horn
8.283401,...,-0.207454,8.602608,...,0.332724,  6, ...,  1,   car_horn
```

Figura 11 – Excerto de *dataset* com atributos combinados de MFCC e *Motifs*

Na Figura 12 são apresentados os resultados da experiência efetuada conforme descrita no parágrafo anterior para os pares de classes "especiais", utilizando o *dataset* UrbanSound. Foram feitos testes com diferentes parâmetros no MrMotif. Os parâmetros utilizados são apresentados na Tabela 8.

Tabela 8 – Parâmetros utlizados no MrMotif nas experiências com atributos combinados (UrbanSound dataset)

	Resolução	Tamanho dos *motifs*	Overlap	Top-k
Expriência 1	8	40	10	40
Expriência 2	4	40	5	40
Expriência 3	4	40	10	40

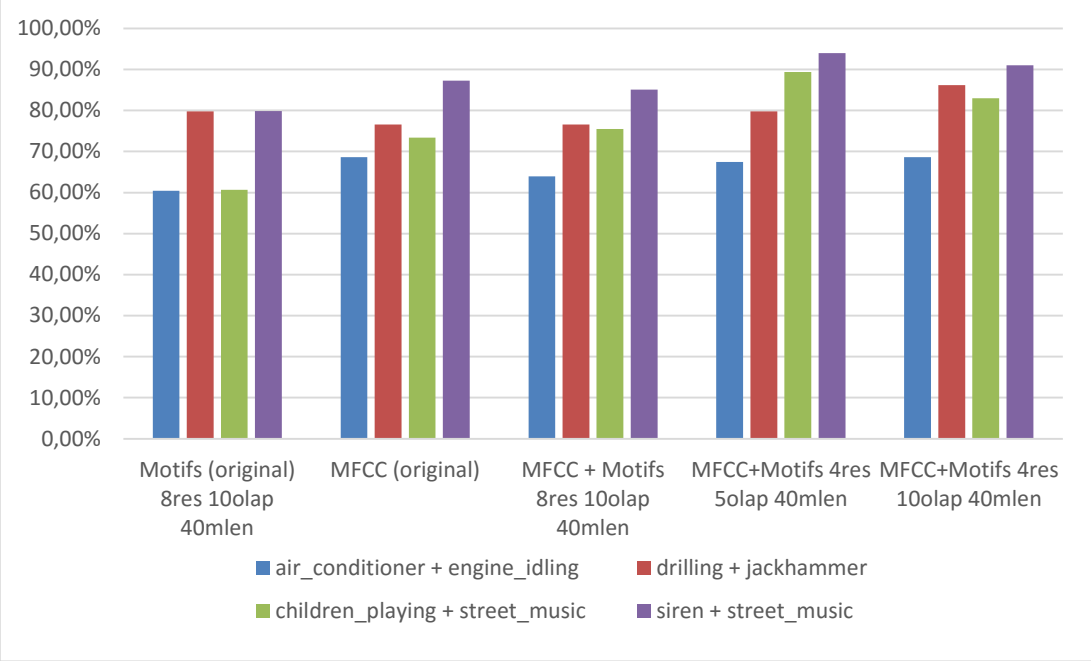

Figura 12 - Resultados dos testes a pares de classes "especiais" efetuados com MFCC e *motifs* combinados no mesmo *dataset* (UrbanSound *dataset*)

Como se pode observar na Figura 12, utilizar MFCC e *motifs* com resolução 4 combinados melhorou os resultados de uma forma geral.

A Tabela 9 contém um resumo dos melhores resultados. Os *motifs* obtiveram melhores resultados no caso dos pares de classes air conditioner + engine_idling (72.09) e air_conditioner + engine_idling (93.62) com resolução 4 (overlap 5 and 10 respetivamente). No caso do par drilling + jackhammer and siren + street_music os melhores resultados foram conseguidos combinando os atributos.

Tabela 9 – Melhores resultados obtidos em pares de classe difíceis (UrbanSound *dataset*)

Classes	MFCCs	MotifsR4O10	MotifsR4O5	MFCCs+MotifsR4O5
air_conditioner+eng_idling	68.60	**72.09**	67.44	67.44
drilling+jackhammer	76.60	88.30	**93.62**	79.79
child_playing+street_music	73.40	85.11	78.72	**89.36**
siren+street_music	87.31	85.82	88.81	**94.03**

5.4 Aplicação dos algoritmos ao dataset UrbanSound8K

Nesta secção, apresenta-se o conjunto de experiências utilizando o dataset UrbanSound8K. Teoricamente, devido aos sons deste *dataset* conterem apenas os eventos relevantes contidos nos sons do UrbanSound (secção 5.1), a classificação deveria apresentar melhores resultados.

Na Tabela 10 encontra-se mencionado o número e instâncias de cada classe utilizadas na experiência.

Tabela 10 – Número de instâncias de cada classe (UrbanSound8K *dataset*)

air_conditioner	**car_horn**	**children_playing**	**dog_bark**	**drilling**
974	429	1000	999	978
engine_idling	**gun_shot**	**jackhammer**	**siren**	**street_music**
1000	374	1000	920	1027

O *dataset* UrbanSound8K exige especial atenção, dado que os eventos sonoros relevantes, retirados dos sons do UrbanSound, estão divididos em vários ficheiros de som com duração aproximada de 4 segundos. É pois necessário ter o cuidado de não incluir excertos do mesmo evento no conjunto de treino e no conjunto de teste pois poderá haver um sobreajustamento dos resultados [Salamon, et al., 2014].

5.4.1 Comparação dos motifs com MFCC

Ao contrário do que foi feito no primeiro conjunto de experiências realizadas com o *dataset* UrbanSound, não foi utilizada validação cruzada com 10 *folds* automaticamente pelo Weka, devido ao problema acima descrito.

Para evitar o sobreajustamento, o *dataset* foi dividido em 10 *folds* onde não existem excertos do mesmo evento em *folds* diferentes. Assim, foi necessário adaptar o *script* de invocação do Weka de modo a que a seleção dos *folds* fosse feita manualmente. Foi também criado um novo *script* para efetuar a combinação das classes de modo a que fossem criados conjuntos de treino e de teste. Para efetuar a validação cruzada foram criados 10 conjuntos de treino e 10 de teste para cada par de classes e para o conjunto de todas as classes, onde o 1º conjunto de teste era composto pelas instâncias do *fold* 1 e o 1 º conjunto de treino pelas instâncias dos restantes *folds*, 2º conjunto de teste pelas instâncias do *fold* 2 e o 2 º conjunto de treino pelas instâncias dos restantes *fold* e assim sucessivamente.

Os parâmetros utilizados no MrMotif para esta experiência foram os seguintes:

Resolução: 4
Top-k: 40

Tamanho dos *motifs*: 20
Overlap: 10

Relativamente aos MFCC foram utilizados os mesmos parâmetros que nas experiências anteriores com o *dataset* UrbanSound.

Na Figura 13 e na Figura 14 encontram-se os resultados obtidos usando a abordagem descrita no parágrafo anterior. Os parâmetros utilizados, tanto no MrMotif como no programa MFCC, foram os mesmos que na experiência anterior.

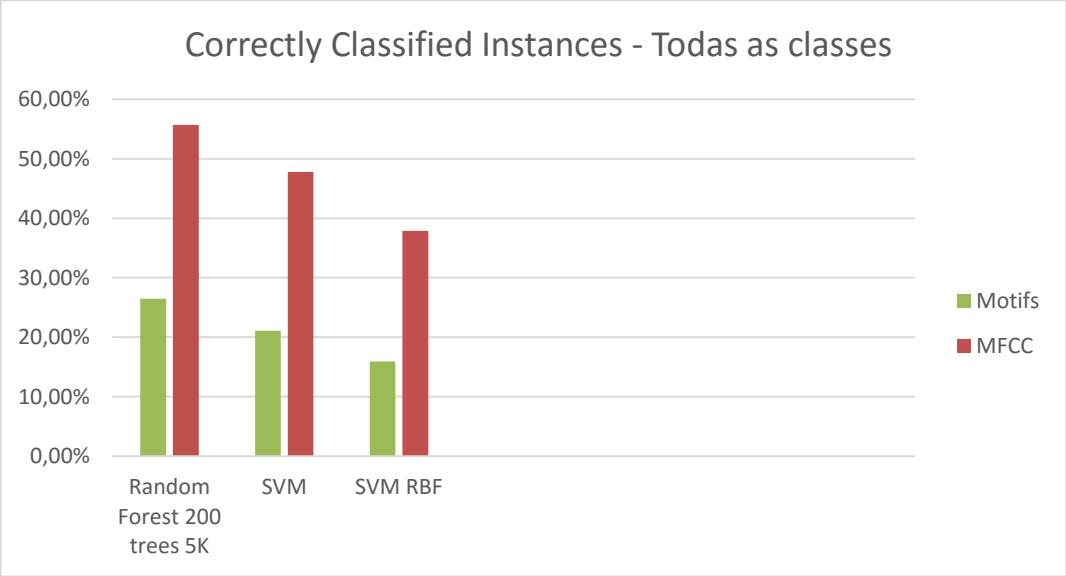

Figura 13 - *Motifs* vs MFCC – todas as classes (UrbanSound8K *dataset*)

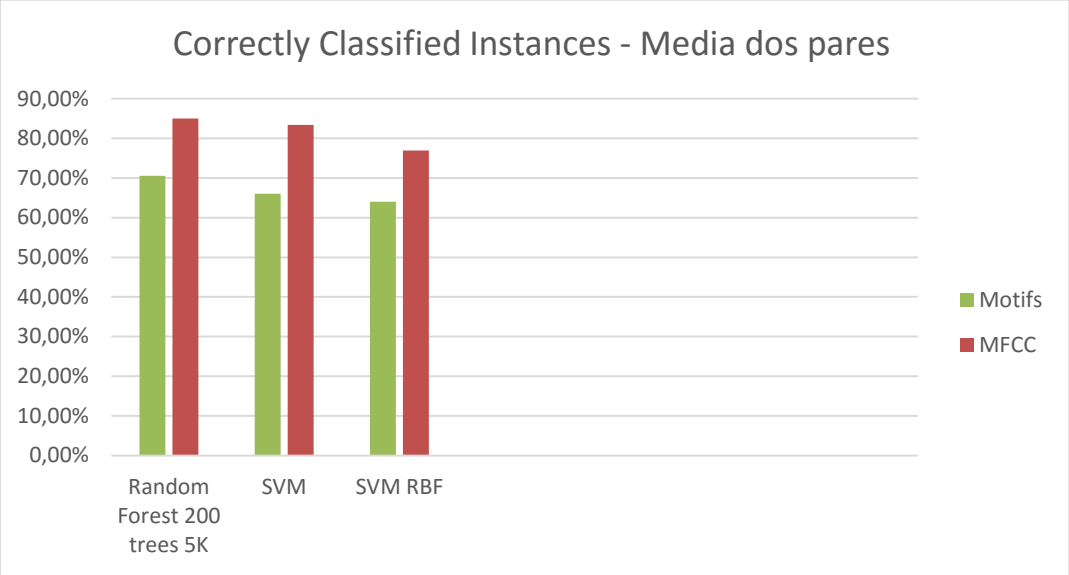

Figura 14 - *Motifs* vs MFCC – Média dos pares (UrbanSound8K *dataset*)

Como se pode observar na Figura 13 e na Figura 14, os modelos construídos tendo por base os MFCC conseguiram melhores resultados (85% contra 70,55% obtido com os *motifs*). Também se pode observar que os melhores resultados foram obtidos com o algoritmo Random Forest.

5.4.2 Combinação dos atributos *motifs* e MFCC

Nesta secção apresenta-se o resultado de experiências efetuadas combinando os *motifs* com MFCC, para o *dataset* UrbanSound8K, à semelhança do que foi feito para o *dataset* UrbanSound (descrito na 5.3.6). Foram também efetuadas várias experiências onde se testaram resultados da variação de diferentes parâmetros no MrMotif com o objetivo de melhorar os resultados. Os parâmetros utilizados são apresentados na Tabela 11.

Tabela 11 - Parâmetros utlizados no MrMotif nas experiências com atributos combinados (UrbanSound8K dataset)

	Resolução	Tamanho dos *motifs*	Overlap	Top-k
Expriência 1	4	10	5	40
Expriência 2	4	20	10	40
Expriência 3	8	10	5	40
Expriência 4	4	10	10	40

Na Tabela 12 e na Figura 15 apresentam-se os resultados dos testes a pares de classes "especiais" efetuados com os atributos MFCC e *Motifs* combinados no mesmo dataset. Como se pode observar, a junção de atributos melhora substancialmente os resultados. Pode-se observar também que utilizar diferentes parâmetros nos *motifs* afeta de forma diferente cada par de classes.

Tabela 12 - Resultados dos testes a pares de classes "especiais" efetuados com MFCC e *Motifs* combinados no mesmo *dataset* (UrbanSound8K *dataset*)

	MFCC	MFCC + *Motifs* 4res 10wlen 5olap	MFCC + *Motifs* 4res 20wlen 10olap	MFCC + *Motifs* 8res 10wlen 5olap	MFCC + *Motifs* 4res 10wlen 10olap
air_conditioner + engine_idling	60,66	66,78	**68,53**	66,67	66,78
children_playing + street_music	53,70	79,00	78,30	**79,25**	79,00
drilling + jackhammer	64,86	**72,65**	72,14	69,27	**72,65**
siren + street_music	69,99	**91,06**	89,73	89,75	**91,06**

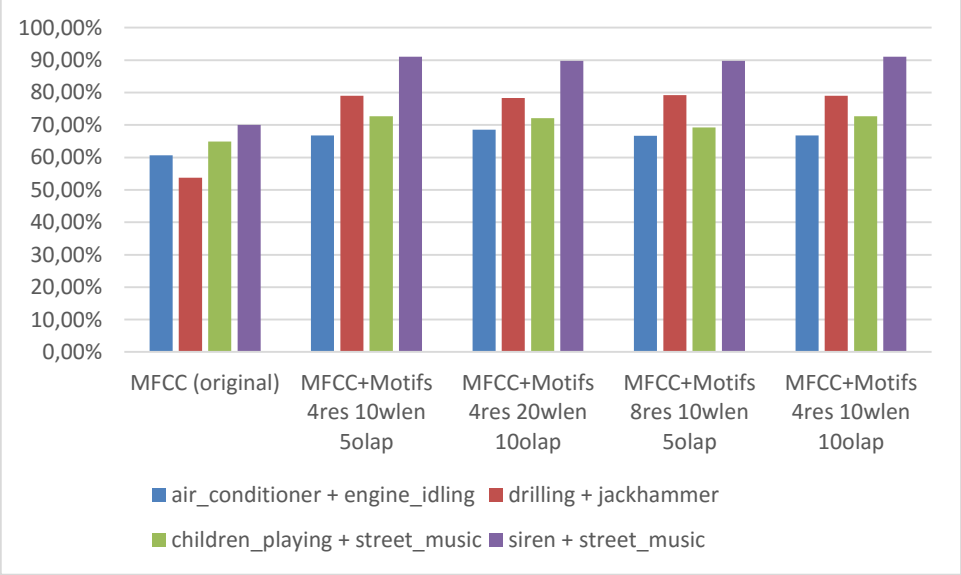

Figura 15 - Resultados dos testes a pares de classes "especiais" efetuados com MFCC e *Motifs* combinados no mesmo *dataset* (UrbanSound8K *dataset*)

5.4.3 Obtenção de diferentes atributos MFCC

Depois de efetuar vários conjuntos de experiências, relatadas nas secções anteriores, foram feitas experiências onde se variaram os parâmetros do algoritmo de obtenção dos atributos MFCC. Os resultados podem ser visualizados na Figura 16 e Figura 17, onde se pode observar que a um aumento do número de coeficientes incluídos no *dataset* gerado corresponde um efeito positivo nos resultados e uma diminuição do número de *filterbanks* tem um efeito negativo.

Observou-se que os melhores resultados, para todas as classes, foram obtidos com 25 coeficientes, com uma *accuracy* de 58,92%, uma melhoria de 3,24% em relação ao original (55,68 com 12 coeficientes). Observou-se também que para esses 25 coeficientes a *accuracy* aumentava também no caso da média dos pares de classes obtendo-se 87,78% contra os 85,00% obtidos inicialmente com 12 coeficientes.

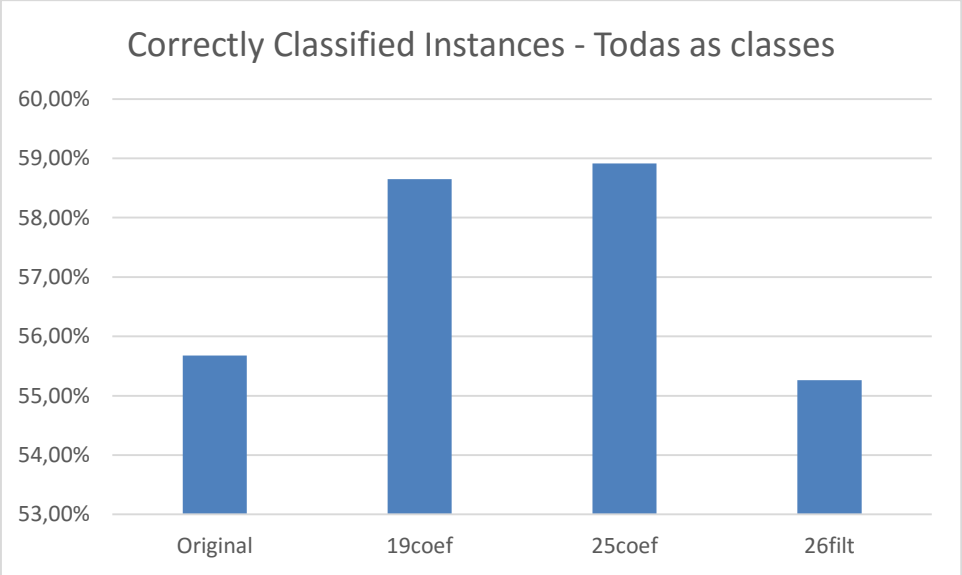

Figura 16 - Resultados dos testes de variação de parâmetros no algoritmo dos MFCC – todas as classes (UrbanSound8K *dataset*)

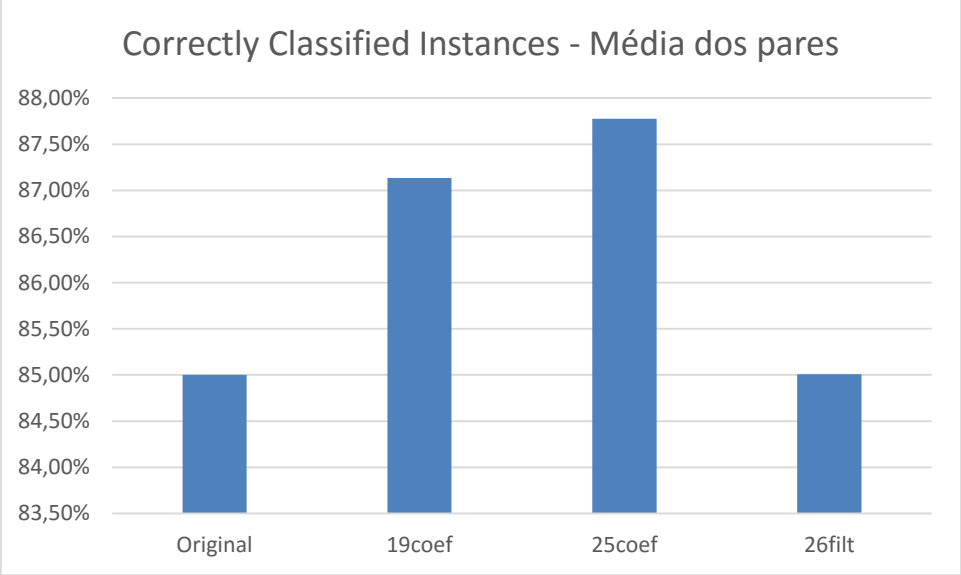

Figura 17 - Resultados dos testes de variação de parâmetros no algoritmo dos MFCC – média dos pares (UrbanSound8K *dataset*)

5.4.4 Combinação dos MFCC melhorados com motifs

Nesta secção apresentam-se os melhores resultados obtidos resultantes da combinação dos *motifs* com os MFCC melhorados. Como se pode observar pela Figura 18, os resultados melhoraram ligeiramente, obtendo-se no modelo com todas as classes uma *accuracy* de 58,88% e nos pares uma média de 86,89%.

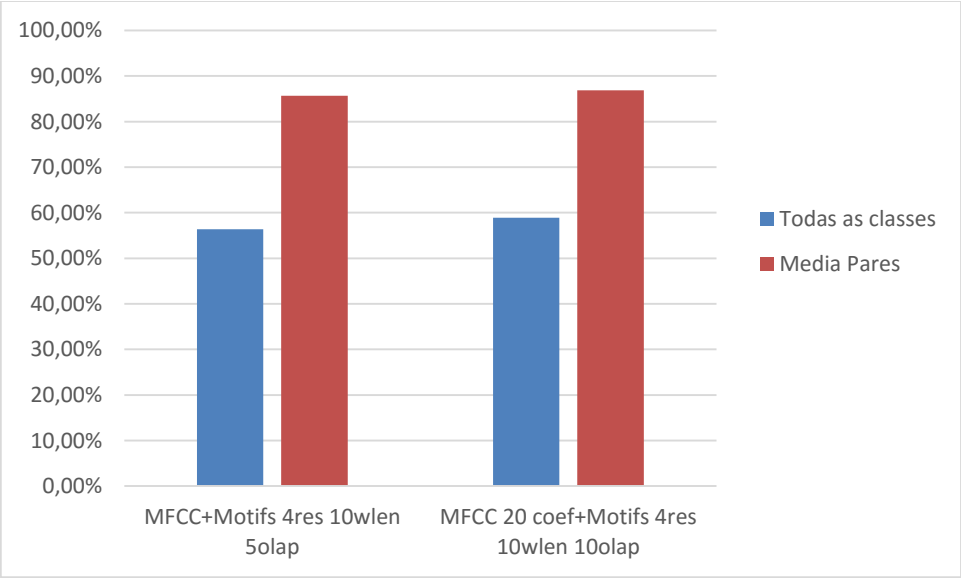

Figura 18 - Resultados obtidos com a combinação de atributos dos *Motifs* com os dos MFCC (anterior vs melhorado) (UrbanSound8K *dataset*)

Quanto às classes “especiais” (Figura 19) existe uma melhoria ligeira em alguns casos apenas. Nos pares “air_conditioner + engine idling” e “drilling + jackhammer” consegue-se uma pequena melhoria, obtendo-se 66,82% e 80,45% de *accuracy* respetivamente contra 66,78% e 79,00% obtidos originalmente. Já nos pares “children_playing + street_music” e “siren + street_music” a *accuracy* piora ligeiramente passando o primeiro de 72,65% para 71,79% e o ultimo de 91,06% para 89,34%.

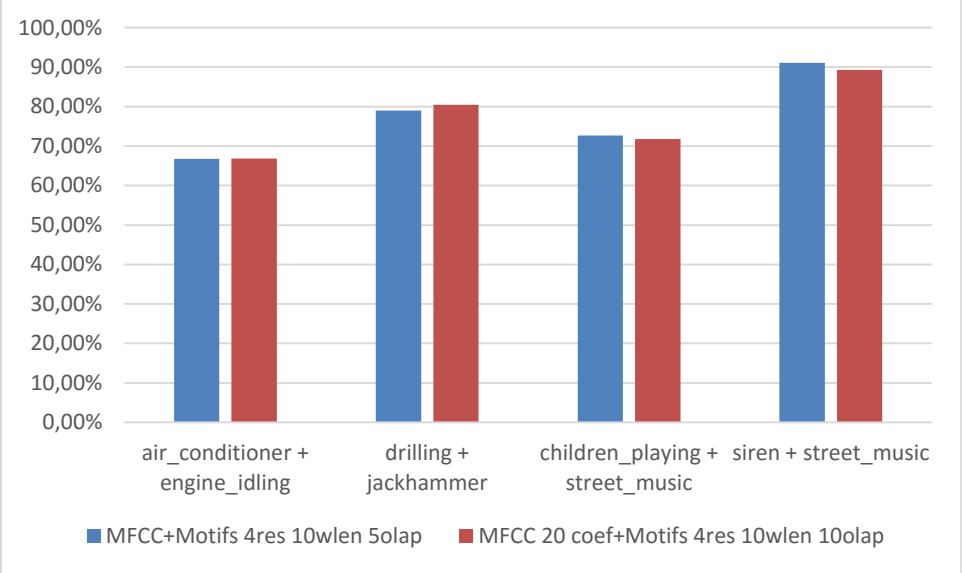

Figura 19 - Resultados obtidos com a combinação de atributos dos *Motifs* com os dos MFCC em pares de classes “especiais” (anterior vs melhorado) (UrbanSound8K *dataset*)

Como se pode observar, a utilização de mais coeficientes MFCC aumentou apenas em dois casos a accuracy do classificador. Portanto, não parece útil a utilização de mais coeficientes.

5.5 Discussão dos resultados

As experiências foram realizadas usando os conjuntos de dados UrbanSound e UrbanSound8K.

No caso do UrbanSound as nossas experiências preliminares sugerem que os atributos *motifs* têm um bom poder de discriminação no que diz respeito aos sons urbanos que não são captados pelos atributos MFCC básicos. Observou-se também que resoluções mais pequenas nos *motifs* produzem melhores resultados, por exemplo no par “children_playing + street_music” obteve-se 51,06% com 16 de resolução e 85,11% com 4 de resolução. A combinação de atributos *motif* e MFCC também resultou num melhor desempenho, sendo 94,03% o valor de accuracy maior (no par “siren + street_music”).

No caso do UrbanSound8K, as *accuracies* são, no geral, mais altas que na UrbanSound, o que leva à conclusão que o isolar dos eventos (retirar ruido antes e depois do evento) a etiquetar/classificar aumenta substancialmente a performance do classificador. No caso dos MFCC o aumento da *accuracy* em relação à obtida com a UrbanSound é substancial, (de 52,17% para 55,68% com todas as classes e 82,51% para 85,00% na média dos pares). No caso dos *motifs* há uma ligeira diminuição (de 30,88% para 26,45% com todas as classes e de 70,88% para 70,55% na média dos pares). Mas foi a junção dos atributos MFCC e *motifs* que produziu os melhores resultados (58,88% com todas as classes e 86,89% na média dos pares).

6 Aplicação Móvel

Depois das experiências realizadas para encontrar o melhor modelo de classificação, era importante saber como se comportaria o modelo numa situação de vida real. Era também importante que o modelo pudesse classificar mais do que 10 classes. Para cumprir esses objetivos foi criado o *UrbanSound Classifier*.

O *UrbanSound Classifier* é uma aplicação móvel que permite classificar sons urbanos e expandir o *dataset* com novos sons e novas classes. É constituída por dois componentes:

- Cliente (aplicação móvel para Android)
- Servidor (aplicação em Java)

Na subsecção 6.1 são descritos os casos de uso, as funcionalidades da aplicação e como utiliza-las.

Na subsecção 6.2 é mostrada a arquitetura do sistema e como os componentes comunicam entre si.

Nas subsecções 6.3 e 6.4 são mostrados alguns detalhes da implementação do componente de servidor e da aplicação móvel, respetivamente.

Por fim, na subsecção 6.5 são mostrados alguns testes efetuados à aplicação.

6.1 Funcionalidades

O *UrbanSound Classifier* pode ser usado por qualquer pessoa. No entanto, foi concebido com o propósito de ser usado por investigadores que pretendam testar a capacidade do modelo desenvolvido e alargar o *dataset* UrbanSound8K.

As principais funcionalidades do *UrbanSound Classifier* são:

- Gravar sons com o dispositivo móvel e geri-los
- Efetuar a classificação dos sons gravados ou outros e dar feedback sobre a mesma para expandir o *dataset*

6.1.1 Gravar sons e geri-los

Acedendo à aplicação móvel é possível gravar sons clicando no botão “New Recording” no ecrã principal e clicando depois no símbolo de gravação no ecrã seguinte para começar a gravar. Clicando novamente no mesmo sítio para parar a gravação e guardar o ficheiro (Figura 20). É possível reproduzir o som gravado clicando no botão de *Play*.

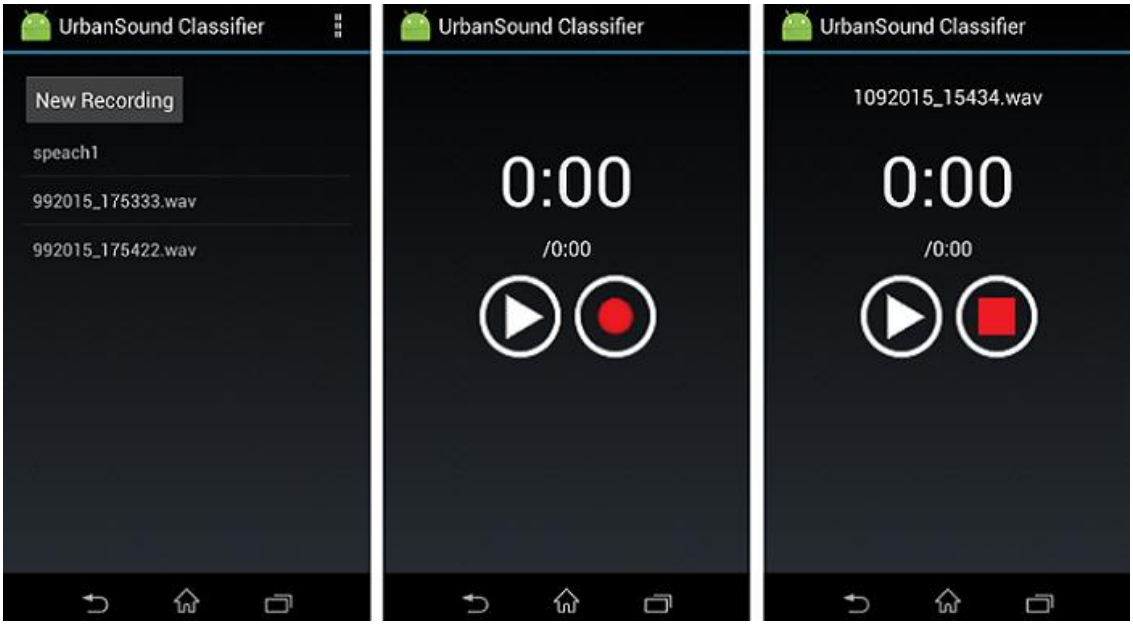

Figura 20 – Passos para gravar um som no UrbanSound Classifier

Os ficheiros de som gerados pelas gravações são em formato Wave PCM e por defeito ficam com o nome no formato “DataGravação_HoraGravação.wav”. Os sons são guardados no armazenamento interno reservado à aplicação.

No ecrã principal pode ser visualizada uma lista com as gravações já efetuadas. Clicando numa gravação, é possível reproduzi-la e visualizar detalhes sobre a classificação e se mantiver premido é possível renomear ou apagar a gravação (Figura 21).

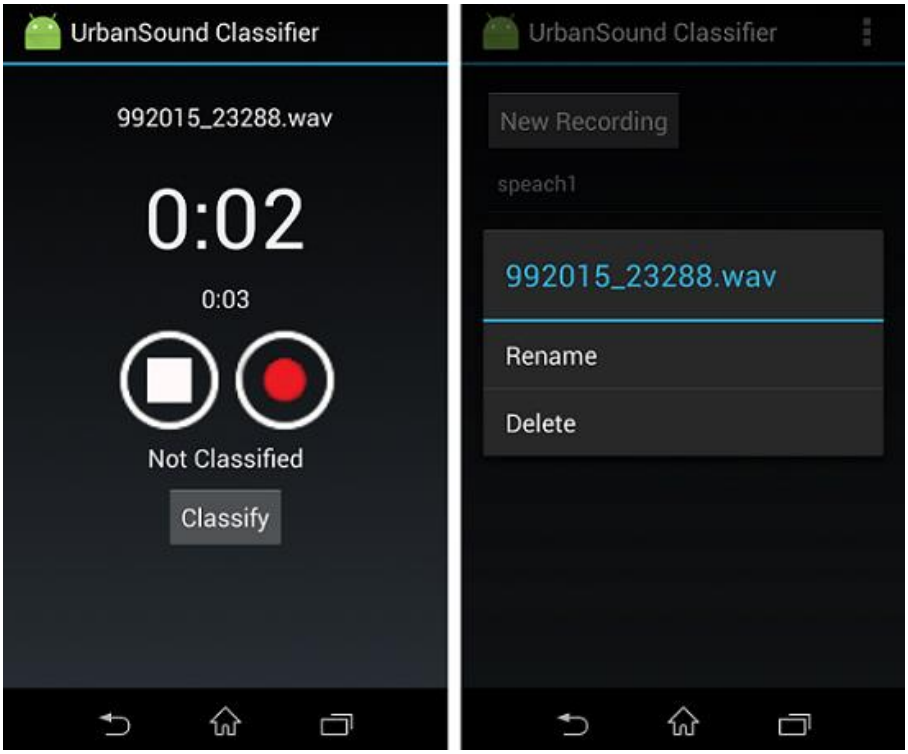

Figura 21 – Reproduzir, renomear e eliminar ficheiro de som no UrbanSound Classifier

É possível adicionar gravações manualmente, adicionando ficheiros de som ao armazenamento interno da aplicação. Qualquer ficheiro de som adicionado nesse espaço é adicionado à aplicação como uma nova gravação aquando do arranque da mesma.

6.1.2 Classificar e dar feedback

Para executar esta funcionalidade é necessário existir conexão entre os componentes cliente e servidor. Para mais detalhes sobre a conexão ver a secção 6.2.

Para classificar um som que ainda não foi classificado, basta escolher o som a classificar no ecrã principal e clicar no botão “Classify” no ecrã seguinte. O som será então enviado para o servidor e este irá devolver o resultado da classificação que será apresentado no ecrã.

Depois da classificação efetuada é possível dar *feedback*, indicando se a classificação está correta ou não. Em caso afirmativo basta clicar no botão “Yes, it’s correct”. Em caso negativo deve-se clicar no botão “No, it’s not” que faz com que seja apresentado um menu onde se seleciona qual a classe correta (Figura 22). O servidor, recebendo o *feedback*, adiciona o som ao *dataset* associando ao mesmo a classe indicada no *feedback*.

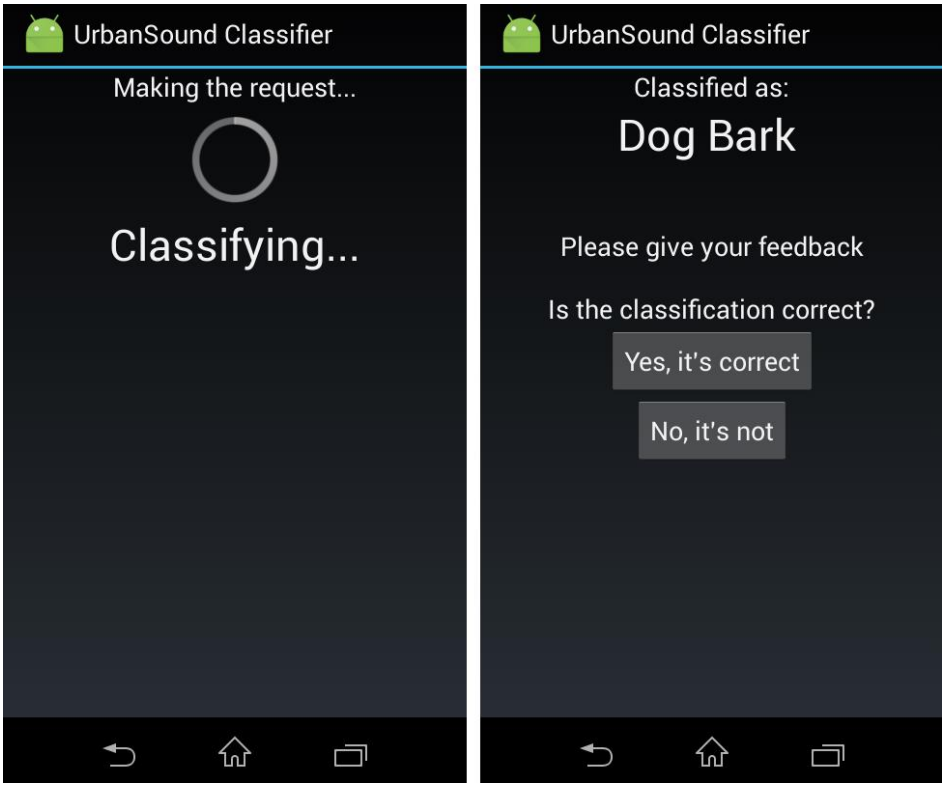

Figura 22 – Classificar e dar *feedback* no UrbanSound Classifier

6.2 Comunicação entre componentes

Como já foi referido o sistema é composto por dois componentes: o cliente e o servidor. Para estabelecer a comunicação entre ambos foi criado um *webservice* que utiliza o protocolo SOAP, usa como formato de mensagem o XML e utiliza o protocolo HTTP como meio de transmissão. Na Figura 23 estão representados os componentes do sistema e a forma como se ligam.

Figura 23 – Arquitetura do sistema

Para que ambos estabeleçam ligação, primeiro é necessário configurar o URL de ligação em ambos os componentes. No servidor deve-se editar o ficheiro "url.txt" que se encontra na raiz da aplicação. No ficheiro, deverá ser introduzido o endereço IP do servidor ou o domínio. Na componente móvel, o URL é configurável utilizando o menu de contexto do ecrã principal e selecionando a (única) opção "Set URL". Surgirá uma *Inputbox* onde deverá ser escrito o endereço IP do servidor ou domínio. O URL tem o seguinte formato: *http:///[IP ou domínio do servidor]/urbansound/?wsdl*.

Com o servidor ativo, se o URL for acedido pelo *browser*, poderá ser visualizado um ficheiro num formato específico de XML, o WSDL, que descreve as operações disponibilizadas pelo serviço. Essas operações são as seguintes:

`Int newRequest()` – Serve para efetuar um novo pedido de classificação. O servidor devolve o identificador do pedido.

`Void uploadFile(int requestid, String fileBase64)` – Para enviar o ficheiro de som a classificar. Deverá ser fornecido o identificador do pedido e o ficheiro que deverá vir codificado em base64.

`String processRequest (int requestid)` – Classificar o som enviado previamente. Deverá ser fornecido o identificador do pedido. Devolve a classe determinada pelo classificador.

`String getClassList()` – Devolve a lista das classes que podem ser atribuídas em formato XML.

`Void processFeedback(int requestid, String classid)` – Para dar *feedback*. Deverá ser fornecido o identificador do pedido e o identificador da classe correta dada como *feedback*.

Na Figura 24 é apresentada a sequência de pedidos efetuada pelo cliente ao serviço para classificar um som. Começa com a requisição de uma classificação usando a operação `newrequest`, ao que o servidor devolve um identificador do pedido. A seguir é enviado o ficheiro de som, codificado em base64, para o servidor através da operação `uploadfile`. Por fim, é pedida a execução da classificação usando a operação `processRequest` que efetua a classificação do som enviado e devolve a classe resultante.

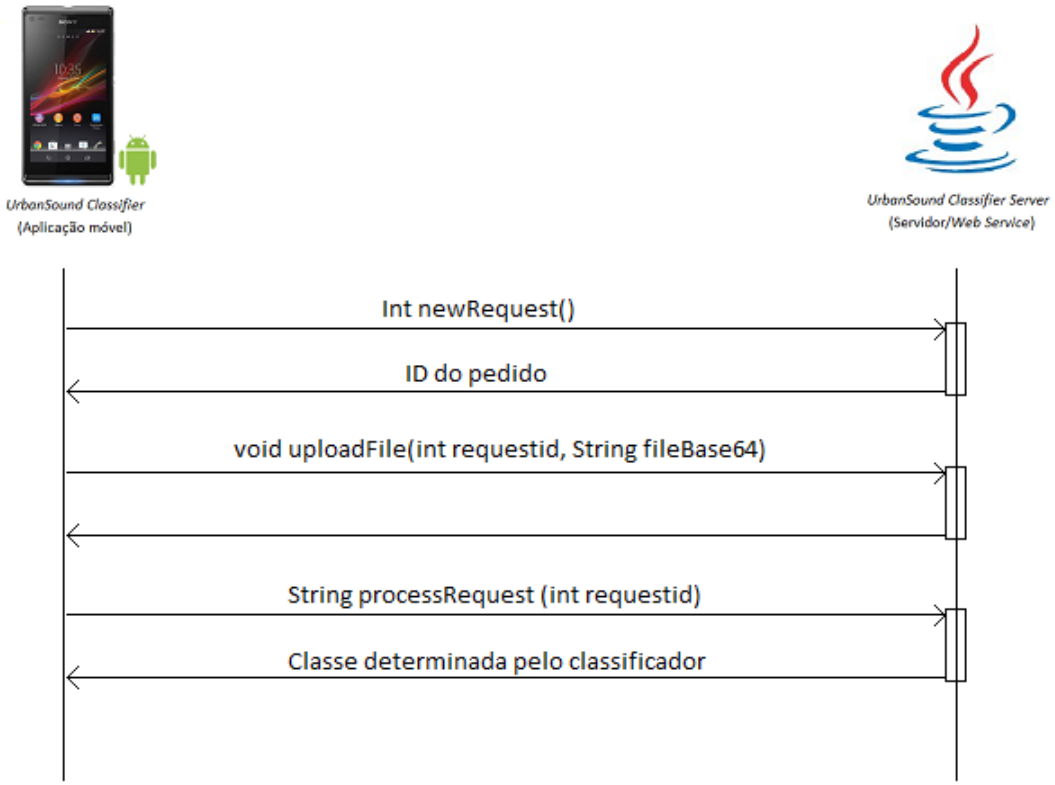

Figura 24 – Sequência de pedidos para classificar um som

Na Figura 25 é apresentada a sequência de pedidos efetuada para dar *feedback*, que apenas pode ser executada depois da tarefa anterior. Para dar o *feedback* basta fazer o pedido para executar a operação `processFeedback`, enviando o identificador do pedido (obtido aquando da classificação) e qual a classe que o utilizador indica como correta. Caso o *feedback* seja negativo, isto é, a classe obtida pelo classificador não é a correta, é pedido ao servidor através da operação `getClassList` a lista de classes disponíveis em XML para apresentar ao utilizador, para este escolher qual a correta.

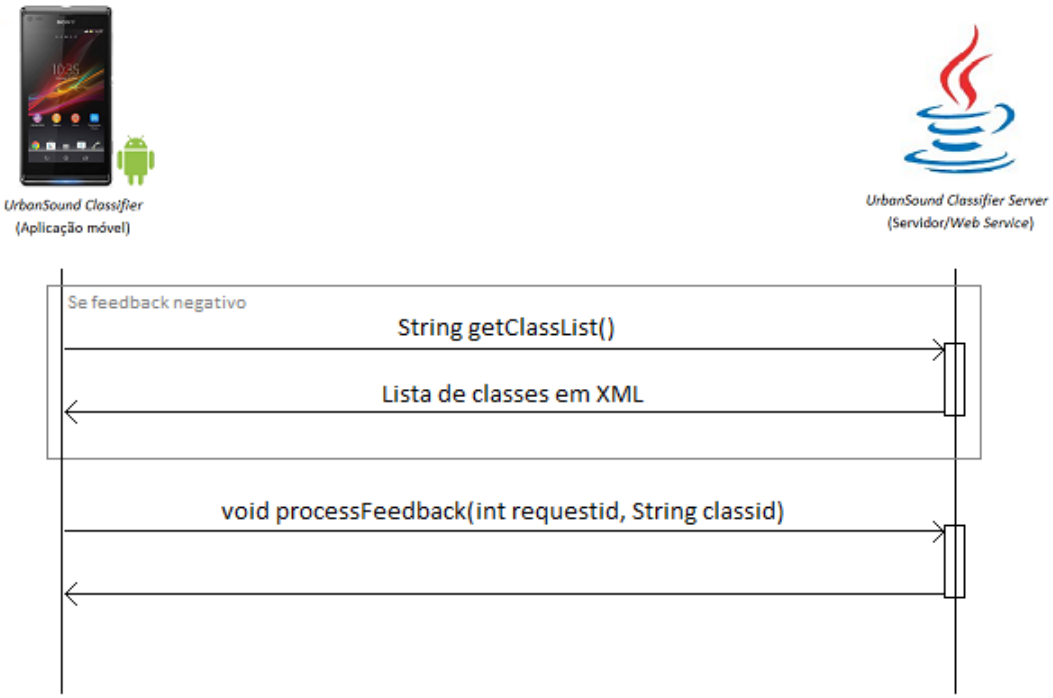

Figura 25 – Sequência de pedidos para dar *feedback*

6.3 UrbanSound Classifier Server

Para a implementação da componente de servidor foi efetuada utilizado o NetBeans IDE e a linguagem de programação Java.

Na Figura 26 é apresentado o diagrama de classes do *UrbanSound Classifier Server*.

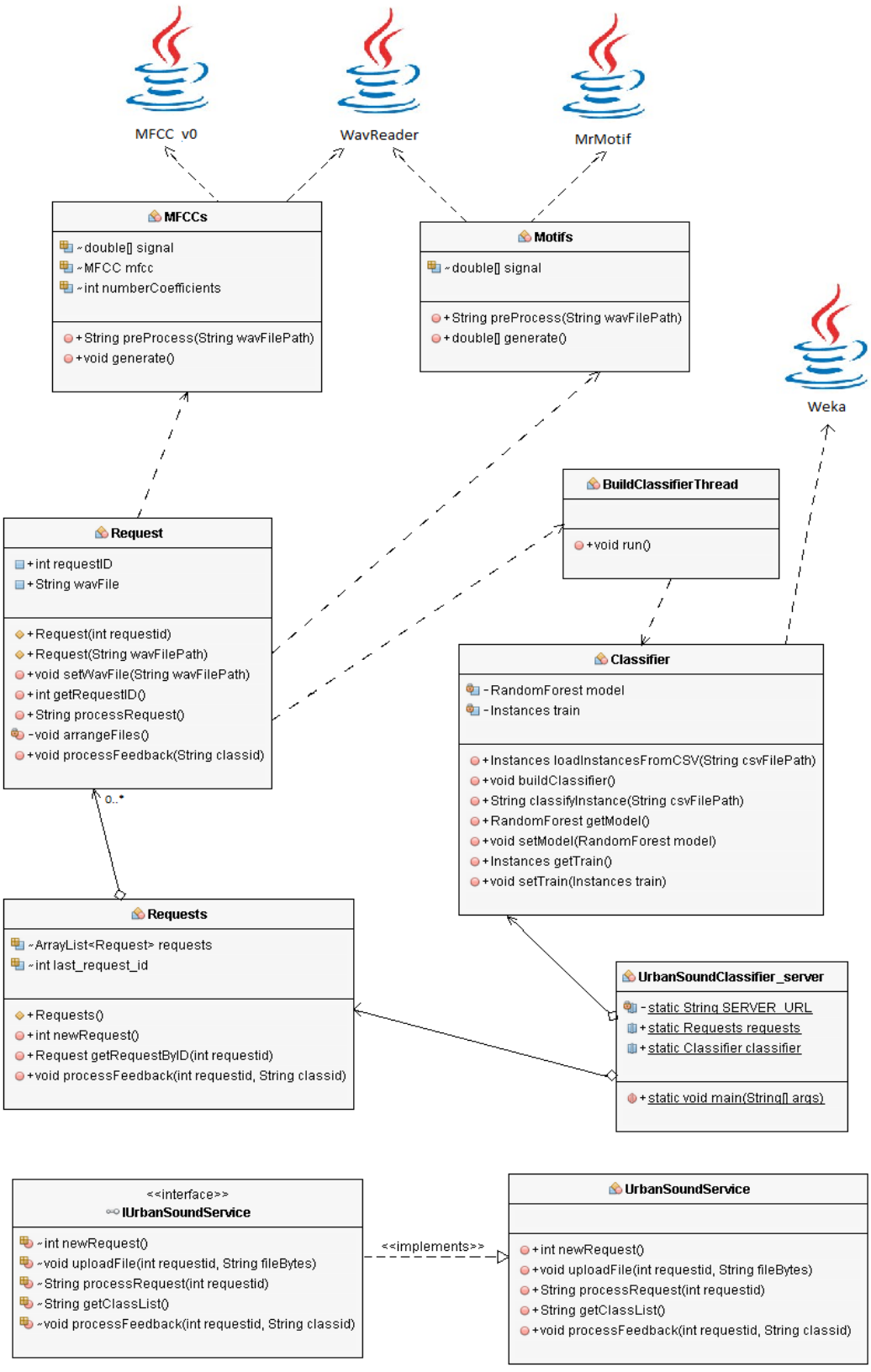

Figura 26 – Diagrama de classes do *UrbanSound Classifier Server*

Tal como nas experiências realizadas, para construção do modelo de classificação é utilizada a API do Weka, para o pré-processamento dos sons é utilizado o WavReader e para a extração de atributos é utilizado o MrMotif e o MFCC. Não foi necessário efetuar qualquer adaptação nestes componentes para funcionarem com o *UrbanSound Classifier*. Foram apenas adicionados a este por referência.

Quando se executa a aplicação, esta começa por construir o modelo com o Weka. É utilizado o algoritmo RandomForest com 200 árvores e 5 K-*features*, pois foi o algoritmo (e respetivos parâmetros) que produziu os melhores resultados nas experiências realizadas, e é utilizado como conjunto de treino inicial o *dataset* UrbanSound8K.

Depois de contruído o modelo é inicializado o *WebService*. Quando a inicialização estiver concluída, o componente está preparado para receber pedidos do(s) cliente(s). A implementação do *WebService* encontra-se na classe `UrbanSoundService`. A comunicação com o serviço é feita através da *interface* `IUrbanSoundService`.

Ao receber um pedido para classificação, a aplicação cria um novo objecto do tipo `Request` e gera um novo identificador para o pedido através do objeto `Requests`, que é responsável pela gestão da fila de pedidos.

Depois de rececionado o som a classificar pode então ser iniciada a classificação. Para isso, são utilizadas as classes `MFCCs` e `Motifs` que interagem com as ferramentas WavReader, MFCC_v0 e MrMotif para extrair os atributos do som. Os atributos MFCC e *motifs* gerados em ficheiros separados são depois concatenados num só para que seja utilizada a combinação dos atributos na classificação. Por fim, é utilizado o modelo criado para classificar o som usando os atributos gerados como conjunto de teste.

Depois da classificação efetuada, o cliente pode dar feedback sobre a classificação. O servidor, ao receber o *feedback* atribui a classe indicada ao novo som, os atributos do som são acrescentados ao conjunto de treino e o modelo é reconstruído utilizando um objeto `BuildClassifierThread`.

6.4 UrbanSound Classifier

Para a implementação da componente de cliente foram utilizados o IDE Android Studio e a linguagem de programação Java.

Na Figura 27 é apresentado o diagrama de classes do *UrbanSound Classifier*.

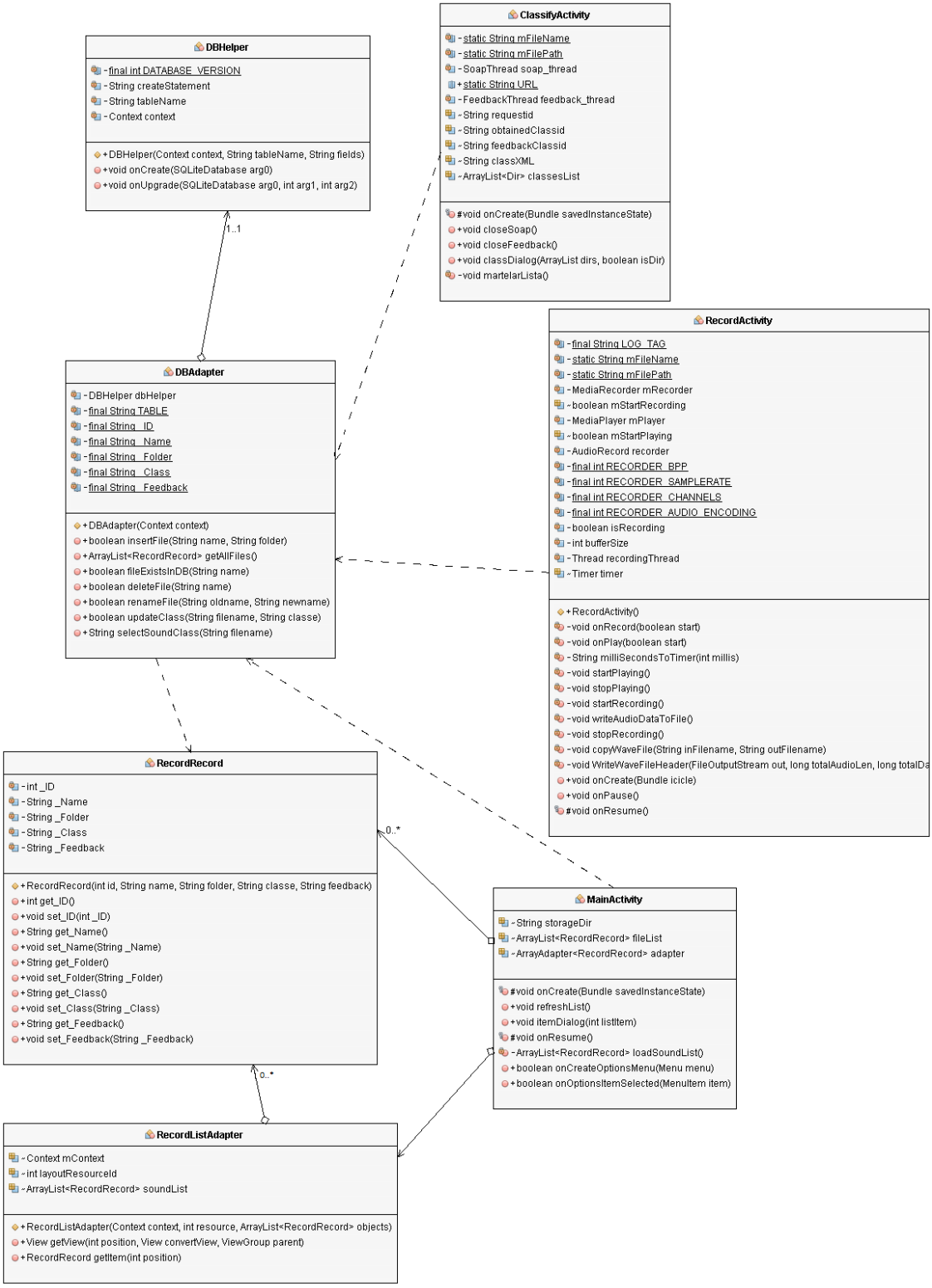

Figura 27 - Diagrama de classes do *UrbanSound Classifier*

O arranque da aplicação é efetuado na classe `MainActivity`. Nela é apresentada uma lista das gravações já efetuadas com a aplicação. Essa lista contém objetos do tipo `RecordRecord` que representam as gravações. A classe `RecordListAdapter` é responsável pela representação visual da lista.

A `RecordActivity` é onde se apresenta informação relativa a um som, isto é, nome do ficheiro de áudio, duração, classe resultante da classificação e *feedback* dado. É também possível reproduzir o som e efetuar uma nova gravação. As informações relativas às gravações são guardadas numa base de dados implementada através das classes `DBAdapter` e `DBHelper` com recurso à tecnologia SQLite [SQLite].

Por último a `ClassifyActivity` é responsável pela classificação e feedback. Efetua a comunicação com o servidor utilizando a biblioteca *ksoap2-android* [ksoap2-android].

6.5 Testes

Por fim, foram efetuados alguns testes para avaliar o desempenho do sistema.

No primeiro teste o objetivo foi testar o modelo de classificação e o tempo de resposta do servidor. Para isso foram retirados 20 sons do conjunto de treino e foi efetuada a classificação desses sons usando a aplicação móvel e obtendo a classificação do servidor.

A aplicação móvel encontrava-se instalada num *smartphone* Sony Xperia L e o componente de servidor encontrava-se instalado num desktop com o sistema operativo Windows 10. Ambos se encontravam conectados através de uma rede local sem fios.

Na Tabela 13 são apresentados os resultados do teste. A segunda coluna refere-se a classe real (etiquetada) do som, a terceira coluna refere a classe obtida pelo modelo do servidor e a quarta refere o tempo de resposta do servidor ao pedido de classificação O tempo de resposta depende principalmente da duração do som e da qualidade da conexão. No caso deste teste os sons duram entre 2 a 5 segundos.

Tabela 13 – Teste de *performance* efetuado ao UrbanSound Classifier

Nº Gravação	Classe Real	Classe Obtida	Tempo de resposta (milissegundos)
1	air_conditioner	jackhammer	2440
2	air_conditioner	jackhammer	2259
3	car_horn	car_horn	2176
4	car_horn	car_horn	2117
5	children_playing	drilling	2496
6	children_playing	children_playing	2312
7	dog_bark	dog_bark	2306
8	dog_bark	drilling	3370
9	drilling	jackhammer	2457
10	drilling	children_playing	2563
11	engine_idling	engine_idling	3059
12	engine_idling	gun_shot	2342
13	gun_shot	gun_shot	2155
14	gun_shot	gun_shot	920
15	jackhammer	jackhammer	2575
16	jackhammer	drilling	2757
17	siren	siren	2454
18	siren	siren	2481
19	street_music	jackhammer	2442
20	street_music	street_music	2547

Como se pode observar na Tabela 13, o classificador acertou na classe de 11 sons de 20. De relembrar que a exatidão do modelo classificação é de 58.88%. Os tempos de execução rondaram os 1 e 3 segundos.

Posteriormente foi efetuado outro teste para verificar se o classificador tinha a capacidade de aprender novas classes através da funcionalidade de *feedback* da aplicação. Para isso foram obtidos 15 sons de sinos a tocar em [Freesound]. Utilizou-se a aplicação para classificar um de cada vez e em cada um dar o feedback com pertencente à classe *bell*, classe que antes de efetuado este teste não existia no conjunto de treino. Cada um dos sons foi introduzido no conjunto antes do seguinte ser classificado. Na Tabela 14 encontram-se listadas as classes obtidas pelo modelo de classificação.

Tabela 14 – Classes obtidas no teste à funcionalidade de *feedback* do UrbanSound Classifier

Nº Gravação	Classe obtida
1	dog_bark
2	dog_bark
3	dog_bark
4	dog_bark
5	engine_idling
6	dog_bark
7	dog_bark
8	siren
9	dog_bark
10	drilling
11	bell
12	bell
13	siren
14	bell
15	jackhammer

Como se pode observar na Tabela 14, os 10 primeiros sons não foram classificados com *bell* pelo modelo. No entanto, 3 dos sons seguintes foram classificados como pertencentes à nova classe, o que permite concluir que é possível o classificador aprender novas classes através da funcionalidade de *feedback*.

7 Conclusões

Com este trabalho foi feito um contributo para a área de reconhecimento e classificação de sons urbanos apresentando uma nova metodologia.

Através das experiências realizadas, foi possível observar que os atributos baseados em *motifs* foram melhores para discriminar sons com timbres semelhantes em comparação com os baseados em MFCC. Foi possível observar também que os atributos MFCC e *motifs* quando combinados, produziram modelos com melhor exatidão do que separados.

Foi também desenvolvida uma aplicação móvel e respetivo servidor que poderá servir de apoio a novos investigadores na área.

Para trabalho futuro propomos as seguintes linhas de desenvolvimento:

- Expansão do *dataset* UrbanSound, que poderá ser feita usando a ferramenta UrbanSound Classifier.
- Encontrar um método que permita isolar eventos relevantes nos sons a fim de melhorar a exatidão da classificação.
- Encontrar um método que permita discriminar melhor as classes e obter melhor exatidão na classificação quando existe um grande número de classes a tratar.
- Desenvolvimento de um sistema de deteção e classificação de sons urbanos em larga escala para deteção de fontes de poluição sonora.

8 Referências

[Abe and Yamaguchi, 2005] Abe, H. & Yamaguchi, T., 2005. *Implementing an Integrated Time-Series Data Mining Environment - A Case Study of Medical KDD on Chronic Hepatitis.* s.l., 1st Internacional Conference on Complex Medical Engineering (CME 2005).

[Al-Sarayreh et al., 2009] Al-Sarayreh, K. T., Al-Qutaish, R. E. & Al-Kasasbeh, B. M., 2009. Using the Sound Recognition Techniques to Reduce the Electricity Consumption in Highways. *Journal of American Science*.

[Beritelli, 2008] Beritelli, F., 2008. *A Pattern Recognition System for Environmental Sound Classification based on MFCCs and Neural Networks.* s.l., ICSPCS.

[Boser et al., 1992] Boser, B., Guyon, I. & Vapnik, V., 1992. *A Training Algorithm for Optimal Margin Classiers.* s.l., COLT '92 Proceedings of the fifth annual workshop on Computational learning theory, pp. 144-152.

[Breiman, 2001] Breiman, L., 2001. Random Forests. *Machine Learning*, pp. 5-32.

[Castro and Azevedo, 2010] Castro, N. & Azevedo, P., 2010. *Multiresolution Motif Discovery in Time Series.* Columbus, Ohio, s.n., pp. 665-676.

[Cavaco and Lewicki, 2007] Cavaco, S. & Lewicki, M. S., 2007. Statistical modeling of intrinsic structures in impacts sounds. *J. Acoust. Soc. Am., Vol. 121, No. 6*, Junho, p. 3558–3568.

[Cavaco and Rodeia, 2010] Cavaco, S. & Rodeia, J., 2010. Classification of Similar Impact Sounds. *ICISP 2010, LNCS 6134*, pp. 307-314.

[Chu et al., 2009] Chu, S., Narayanan, S. & Kuo, C.-C. J., 2009. Environmental Sound Recognition With Time–Frequency Audio Features. *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 17, NO. 6*, August.

[Chum et al.,2013] Chum, M., Habshush, A., Rahman, A. & Sang, C., 2013. *IEEE AASP SCENE CLASSIFICATION CHALLENGE USING HIDDEN MARKOV MODELS AND FRAME BASED CLASSIFICATION.* s.l., s.n.

[Couvreur and Laniray, 2004] Couvreur, L. & Laniray, M., 2004. *Automatic Noise Recognition in Urban Environments Based on Artificial Neural Networks and Hidden Markov Models.* Prague, s.n.

[Couvreur, s.d.] Couvreur, C., s.d. *Environmental Sound Recognition: A Statistical Approach,* s.l.: s.n.

[Davis and Mermelstein, 1980] Davis, S. & Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on ASSP 28*, pp. 357-366.

[Domingues et al., 2012] Domingues, M. A. et al., 2012. *Combining Usage and Content in an Online Music Recommendation System for Music in the Long-Tail.* Lyon, s.n., pp. 925-929.

[Dufaux, 2001] Dufaux, A., 2001. Detection and Recognition of Impulsive Sound Signals. January.

[Elizalde et al., 2013] Elizalde, B., Lei, H., Friedland, G. & Peters, N., 2013. *CAPTURING THE ACOUSTIC SCENE CHARACTERISTICS FOR AUDIO SCENE DETECTION.* s.l., IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.

[Ferreira et al, 2006] Ferreira, P. G., Azevedo, P. J., Silva, C. G. & Brito, R. M., 2006. Mining Approximate Motifs in Time Series. *Discovery Science*, pp. 89-101

[Forney Jr., 2005] Forney Jr., G. D., 2005. The Viterbi Algorithm: A Personal History.

[Gama et al., 2012] Gama, J. et al., 2012. *Extração Conhecimento de Dados.* ed. 1, 1 vol., ISBN: 978-972-618-698-4 ed. Lisboa: Silabo.

[Geiger et al., 2013] Geiger, J. T., Schuller, B. & Rigoll, G., 2013. *LARGE-SCALE AUDIO FEATURE EXTRACTION AND SVM FOR ACOUSTIC SCENE CLASSIFICATION.* s.l., IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.

[Gemmeke et al, 2013] Gemmeke, J. F., Vuegen, L., Vanrumste, B. & hamme, H. V., 2013. AN EXEMPLAR-BASED NMF APPROACH FOR AUDIO EVENT DETECTION.

[Giannoulis et al., 2013] Giannoulis, D. et al., 2013. *Detection and classification of acoustic scenes and events: an IEEE AASP challenge.* s.l., s.n.

[Gomes and Batista, 2015a] Gomes, E. F. & Batista, F., 2015. Classifying Urban Sounds using Time Series Motifs. *Advanced Science and Technology Letters,(SUComS),97*, pp. 52-57.

[Gomes and Batista, 2015b] Gomes, E. F. & Batista, F., 2015. Using Multiresolution Time Series Motifs to Classify Urban Sounds. *International Journal*

of Software Engineering and Its Applications,9,8, pp. 189-196.

[Gomes and Pereira, 2012] Gomes, E. F. & Pereira, E., 2012. *Classifying heart sounds using peak location for segmentation and feature construction.* s.l., AISTATS

[Gomes et al., 2013] Gomes, E., Jorge, A. & Azevedo, P., 2013. *Classifying heart sounds using multiresolution time series motifs: an exploratory study.* s.l., Proceedings of the International C* Conference on Computer Science and Software Engineering. ACM.

[Gomes et al., 2014] Gomes, E. F., Jorge, A. M. & Azevedo, P. J., 2014. Classifying heart sounds using SAX motifs, random forests and text mining techniques. *In 18th International Database Engineering & Applications Symposium, IDEAS 2014*, 7-9 July, pp. 334-337.

[Hall et al., 2009] Hall, M. et al., 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations, Volume 11, Issue 1*.

[Hamid et al., 2005] Hamid, R. et al., 2005. *Unsupervised Activity Discovery and Characterization From Event-Streams.* s.l., Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI05).

[Han and Kamber, 2006] Han, J. & Kamber, M., 2006. *Data Mining: Concepts and Techniques.* 2nd Edition s.l.:ELSEVIER.

[Heittola et al, 2011] Heittola, T., Mesaros, A., Virtanen, T. & Eronen, A., 2011. *SOUND EVENT DETECTION AND CONTEXT RECOGNITION.* s.l., s.n.

[Hoiem et al., s.d.] Hoiem, D., Ke, Y. & Sukthankar, R., s.d. SOLAR: Sound Object Localization and Retrieval in Complex Audio Environments.

[Lin et al.,2002] Lin, J., Keogh, E., Patel, P. & Lonardi, S., 2002. *Finding motifs in time series.* Edmonton, Alberta, Canada, Proceedings of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[McGovern et al., 2007] McGovern, A. et al., 2007. *Anticipating the formation of tornadoes through data mining.* s.l., Conference on Artificial Intelligence and its Applications to Environmental Sciences at the American Meteorological Society.

[Mesaros et al., 2015] Mesaros, A., Heittola, T., Dikmen, O. & Virtanen, T., 2015. *SOUND EVENT DETECTION IN REAL LIFE RECORDINGS USING COUPLED MATRIX FACTORIZATION OF SPECTRAL REPRESENTATIONS AND CLASS ACTIVITY.* s.l., s.n.

[Minnen et al., 2006] Minnen, D., Starner, T., Essa, I. & Isbell, C., 2006. *Discovering Characteristic Actions from On-Body Sensor Data.* s.l., 10th International Symposium on Wearable Computers (ISWC06).

[Ntalampiras et al, 2008] Ntalampiras, S., Potamitis, I. & Fakotakis, N., 2008. Automatic Recognition of Urban Soundscenes. *New Direct. in Intel. Interac. Multimedia, SCI 142*, pp. 147-153.

[Oliveira et al., 2014] Oliveira, S. C., Gomes, E. F. & Jorge, A. M., 2014. *Heart Sounds Classification using Motif based Segmentation.* IDEAS '14 Proceedings of the 18th International Database Engineering & Applications Symposium, s.n., pp. 370-371.

[Olivetti, 2013] Olivetti, E., 2013. *THE WONDERS OF THE NORMALIZED COMPRESSION DISSIMILARITY REPRESENTATION.* s.l., s.n.

[Roma et al., 2013] Roma, G., Nogueira, W. & Herrera, P., 2013. *RECURRENCE QUANTIFICATION ANALYSIS FEATURES FOR AUDITORY SCENE CLASSIFICATION.* s.l., s.n.

[Salamon et al., 2014] Salamon, J., Jacoby, C. & Bello, J. P., 2014. *A Dataset and Taxonomy for Urban Sound Research.* s.l., 22nd ACM International Conference on Multimedia, Orlando USA, Nov. 2014.

[Schröder et al., 2013] Schröder, J. et al., 2013. *ON THE USE OF SPECTRO-TEMPORAL FEATURES FOR THE IEEE AASP CHALLENGE 'DETECTION AND CLASSIFICATION OF ACOUSTIC SCENES AND EVENTS'.* s.l., IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.

[Shieh and Keogh, 2008] Shieh, J. & Keogh, E., 2008. *iSAX: Indexing and Mining Terabyte Sized Time Series.* s.l., s.n.

[Tzanetakis and Cook, 2002] Tzanetakis, G. & Cook, P., 2002. Musical Genre Classification of Audio Signals. *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, VOL. 10, NO. 5*, JULY.

[Uzkent et al, 2011] Uzkent, B., Barkana, B. D. & Cevikalp, H., 2011. NON-SPEECH ENVIRONMENTAL SOUND CLASSIFICATION USING SVMS WITH A NEW SET OF FEATURES. *International Journal of Innovative Computing, Information and Control Volume 8, Number 5(B)*, p. 3511–3524.

[Valero and Alías, 2012] Valero, X. & Alías, F., 2012. Hierarchical Classification of Environmental Noise Sources Considering the Acoustic Signature of Vehicle Pass-Bys. *ARCHIVES OF ACOUSTICS Vol. 37, No. 4*, pp. 423-434.

[Witten, 2005] Witten, I. H. & Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques.

[World Health Organization, 2011] World Health Organization, 2011. *Burden of disease from environmental noise: Quantification of healthy life years lost in Europe,* s.l.: s.n.

Referências URL

[Advanced Digital Sciences Center] Advanced Digital Sciences Center, 2015. [Online] Available at: http://publish.illinois.edu/audioanalytics/

[Android Developers, a] Android Developers, s.d. *Android Studio.* [Online] Available at: https://developer.android.com/sdk/index.html [Acedido em 13 Outubro 2015].

[Android Developers, b] Android Developers, s.d. *Android Developer Tools.* [Online] Available at: https://developer.android.com/tools/help/adt.html [Acedido em 13 Outubro 2015].

[Bentley] Bentley, P., Nordehn, G., Coimbra, M. & Mannor, S., 2011. *Classifying Heart Sounds Challenge.* [Online] Available at: http://www.peterjbentley.com/heartchallenge/

[Brito] Brito, E., 2015. *Java: Entenda para que serve o software e os problemas da sua ausência.* [Online] Available at: http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/11/java-entenda-para-que-serve-o-software-e-os-problemas-da-sua-ausencia.html

[Earthquake Hazards Program] Earthquake Hazards Program, 2015. *Earthquake Technology Fights Crime.* [Online] Available at: http://earthquake.usgs.gov/learn/publications/gunshots/

[Freesound] Freesound, s.d. *Freesound.* [Online] Available at: https://www.freesound.org/ [Acedido em 5 Agosto 2015].

[JetBrains] JetBrains, s.d. *IntelliJ IDEA.* [Online] Available at: https://www.jetbrains.com/idea/ [Acedido em 13 Outubro 2015].

[ksoap2-android] ksoap2-android, s.d. *ksoap2-android Project.* [Online] Available at: http://simpligility.github.io/ksoap2-

android/index.html
[Acedido em 14 Setembro 2015].

[Lee] Lee, M., 2006. *Sun begins releasing Java under the GPL.* [Online]
Available at: http://www.fsf.org/news/fsf-welcomes-gpl-java.html

[Lyons] Lyons, J., 2015. *Mel Frequency Cepstral Coefficient (MFCC) tutorial.* [Online]
Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[NetBeans] NetBeans, 2015. *NetBeans IDE - Overview.* [Online]
Available at: https://netbeans.org/features/index.html

[Oracle, a] Oracle, 2015. *O que é a Tecnologia Java e porque preciso dela?.* [Online]
Available at: http://www.java.com/pt_BR/download/faq/whatis_java.xml

[Oracle, b] Oracle, 2015. *Obtenha Informações sobre a Tecnologia Java.* [Online]
Available at: http://www.java.com/pt_BR/about/

[Oracle, c] Oracle, 2015. *Oracle and Sun Microsystems.* [Online]
Available at: http://www.oracle.com/us/sun/index.html

[RFC 2361] RFC 2361, s.d. *RFC 2361 - WAVE and AVI Codec Registries.* [Online]
Available at: http://www.rfc-base.org/rfc-2361.html
[Acedido em 5 Agosto 2015]

[Salamon et al.] Salamon, J., Jacoby, C. & Bello, J., s.d. *Urban sound datasets.* [Online]
Available at: https://serv.cusp.nyu.edu/projects/urbansounddataset/
[Acedido em 2 Novembro 2014]

[SQLite] SQLite, s.d. *SQLite.* [Online]
Available at: https://www.sqlite.org/
[Acedido em 11 Setembro 2015].

[WebM] WebM, s.d. *WebM - an open web media project.* [Online]
Available at: https://code.google.com/p/webm/
[Acedido em 5 Agosto 2015].

Anexos

Using Multiresolution Time Series Motifs to Classify Urban Sounds

Elsa Ferreira Gomes[1,2] and Fábio Batista[2]

[1] *GECAD- Knowledge Engineering and Decision Support Research Center*

[2] *ISEP-School of Engineering, Polytechnic of Porto, Portugal*

efg@isep.ipp.pt

Abstract

The automatic classification of urban sounds is important for environmental monitoring. In this work we employ SAX-based Multiresolution Motif Discovery to generate features for Urban Sound Classification. Our approach consists in the discovery of relevant frequent motifs in the audio signals and use the frequency of discovered motifs as characterizing attributes. We explore and evaluate different configurations of motif discovery for defining attributes. In the automatic classification step we use a decision tree based algorithm, random forests and SVM. Results obtained are compared with the ones using Mel-Frequency Cepstral Coefficients (MFCC) as features. MFCCs are commonly used in environmental sound analysis, as well as in other sound classification tasks. Experiments were performed on the Urban Sound dataset, which is publicly available. Our results indicate that we can separate difficult pairs of classes (where MFCC fails) using the motif approach for feature construction.

Keywords: *Urban sound classification, motif discovery, time series analysis, SAX, MFCC.*

1. Introduction

Many areas, such as health (e.g. cardiac sounds), ambient intelligence or environmental monitoring, can largely benefit from the automatic classification of sounds. Automatic sound classifiers can be learnt from sets of labeled sounds using machine learning algorithms.

One important step of the process is the representation of sounds using a set of features. A standard approach for generating the features is based on the Mel-Frequency Cepstral Coefficients (MFCC). In this paper we propose a SAX-based Multiresolution Motif Discovery and compare it to the MFCC approach specifically on the challenging Urban Sound Dataset [1]. This technique, as far as we know, has never been used on this problem. In our previous work, we have applied motif based features to heart sound classification and compared it with a peak-detection based approach [13], [14]. Urban sounds, however, are much more complex and irregular than heart sounds. Our rationale is the following. A time series motif is a sound segment that is frequently observed. Different classes of sounds should correspond to different motifs found. The idea of our work is to discover relevant frequent motifs in the audio signals and use the discovered motifs and their frequency as characterizing attributes. We validate our proposal on the Urban Sound Dataset using different configurations of MrMotif, a motif discovery algorithm, to generate features. We compare our approach with the Mel-Frequency Cepstral Coefficients (MFCC). We have also assessed the possible advantage of combining attributes from

both origins. We use a Decision tree based algorithm, Random Forest and SVM. Finally we focus on pairs of Urban Sound classes that are particularly challenging for the MFCC approach and obtain good results where MFCC fails.

2. Background

2.1 Feature extraction using MFCC

A widely used feature set, found in literature, is known as Mel Frequency Cepstral Coefficients (MFCCs). MFCCs are commonly used in environmental sound analysis and as a competitive baseline to benchmark new approaches. In order to achieve better result rates it is common to use other features together with MFCCs.
The Mel scale relates perceived frequency, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Using this scale makes our features match more closely what humans hear [2], [19].
The formula for converting from frequency (f) to Mel scale (m) is:

$$M(f) = 1125\ln(1 + f/700)$$

To go from Mel scale back to frequency:

$$M^{-1}(f) = 700exp(m/1125) - 1$$

2.2 Multiresolution Motif Discovery

Motif discovery has been used in different areas namely in health and medicine and in particular in EEG time series a motif may be a pattern that usually precedes a seizure [24], [12]. One recent trend in time series analysis is to use SAX [17]. SAX is a symbolic approach for time series that represents the continuous series as a discretized one. It allows for dimensionality reduction and indexing. In classic data mining tasks such as clustering or classification SAX is as good as well-known representations such as Discrete Wavelet Transform (DWT) and Discrete Fourier Transform (DFT), while requiring less storage space. The representation allows researchers to avail of the wealth of data structures and algorithms in bioinformatics or text mining, and also provides solutions to data mining tasks, such as motif discovery [17]. As a symbolic approximation, SAX(T,w,a) converts the original real time series T of length n into a sequence of w (word size) symbols belonging to an alphabet of size a. In Figure 1 we see how a time series segment can be discretized into a sequence of SAX symbols {1,1,3,3,1,1,1,1}. The alphabet size is called resolution. The Sax algorithm starts by reducing the dimensionality of the time series by dividing it into w segments (word length) with the same length using the Piecewise Aggregate Approximation (PAA) algorithm. This algorithm assigns to each segment its average value. Then, the amplitude of the time series is divided into intervals and then a symbol can be assigned to each interval. To generate equiprobable intervals it uses a-1 breakpoints, producing the same area under the Normal curve, and symbols are obtained from the intervals. The segments below the smallest breakpoint are assigned the 0 symbol, the segments between the first and second breakpoints the symbol 1, and so forth.
The iSAX representation extends classic SAX by allowing different simultaneous resolutions for the same word. The main idea of the MrMotif algorithm is to start from a low iSAX resolution and then expand to higher resolutions. As this expansion is performed, the number of instances of a given cluster reduces as each cluster is split into several of the next resolution. At the highest resolution, a cluster is formed only if the subsequences in that cluster are very similar, as each iSAX symbol covers only a narrow interval in the amplitude of the time series. The minimum possible

resolution gmin in iSAX is 2 and the maximum resolution gmax is assigned to 64 (it uses 2, 4, 8, 16, 32 and 64 resolutions).
Castro el al. [7] have proposed a Multiresolution Motif Discovery in Time Series algorithm (MrMotif) based on the iSAX representation. This algorithm solves the motif discovery problem as an approximate Top-K frequent subsequence discovery problem. The aim of MrMotif algorithm is to find the solution for the problem: given a time series database D, a motif length m and a K parameter, for each resolution in (gmin, gmin×2, …, gmax) find the top-K frequent motifs. In this work we will use MrMotif for generating motif-based features for heart sound classification.

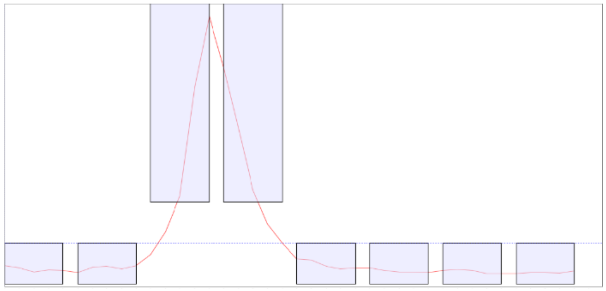

Figure 1: SAX discretization of a time series segment - image obtained with the MrMotif tool [6]

3. Dataset

In this study we use the UrbanSound dataset [1] which contains the 761 labeled (WAV) sound recordings (table 1). Each recording is labeled with the start and end times of sound events from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren and street_music. Each recording may contain multiple sound events, but for each file only events from a single class are labeled. The classes are drawn from the urban sound taxonomy. This dataset contains sounds that occur in an urban environment, all recordings are real field-recordings and has a large variety of sounds and recording conditions. The duration of source occurrences in the set can vary from 1-2 s (e.g. gun_shot sounds) to over 30 s (continuous sounds such as jackhammers or idling engines) [19].

Class	air_cd	car_hr	child_pll	dog_brk	drilling	eng_id	gun_sh	jackham	Siren	str_mu
Number	32	95	105	166	75	54	81	19	51	83

Table 1: Class distribution

4. MFCCs based Classification

MFCCs are broadly used in environmental sound analysis. Namely, in the detection and classification of acoustic scenes [20], [11], [4], urban soundscenes [18], in sound event detection [16], [15] or environmental sound classification [22], in sound recognition [9], [8] in audio classification [21], [10]. We have implemented MFCCs in Java [25], from an existing version in Python, available on [3].

4.1 Methodology

We used frames with length 1024 (window size) with 50% of overleap. For each of this frames we extract 13 features, as described below.

1. The process starts by calculating the Discrete Fourier Transform of each frame.
2. Then, the Mel-spaced filterbank is calculated. We apply 40 triangular filters to the periodogram estimate of the power spectrum (the square of absolute value of the complex Fourier transform). Our filterbank comes in the form of 40 vectors. Each vector is mostly zeros, but is non-zero for a certain section of the spectrum. To calculate filterbank energies we multiply each filterbank with the power spectrum, then add up the coefficients. Thus, we have an indication of how much energy was in each filterbank.
3. Take the log of each of the 40 energies from step 2.
4. Take the Discrete Cosine Transform (DCT) of the 40 log filterbank energies to give 40 cepstral coefficients. Only the lower 13 of the 40 coefficients are kept.
5. The resulting features (13 numbers for each frame) are called Mel Frequency Cepstral Coefficients.

Then, we consider the average and the standard deviation of total frames as features (26).

5 Motif based Classification

In a previous work, we have used motifs as features in cardiac audio time series [13], [14]. The general idea is to find frequent motifs in the audio time series using a frequent pattern mining algorithm. Such discovered motifs are regarded as features. Our hypothesis is that these features contain valuable information for discrimination tasks. To test this hypothesis we first identify relevant motifs in the original dataset and build a new dataset where each relevant motif is an attribute. Then, we compare the results of using these features with the results obtained with the MFCCs features.

5.1 Methodology

A motif in a time series is a frequent pattern, i.e. a repetition of a particular segment of the series. Figure 1 has an example of a motif. We use the Multiresolution Motif Discovery (in Time Series) algorithm [6] to detect the common (and relevant) patterns. This algorithm uses the SAX methodology to discretize the continuous

signals and looks for patterns in the resulting discrete sequences. In particular, we have used the MrMotif algorithm as implemented by its authors [25].
For our experimental exploration, we followed the steps below.

1. Apply to the original audio dataset the pre-processing (filters and normalized average Shannon energy) used in the previous approach.
2. Apply the MrMotif algorithm to the resulting time series. In this step we have to choose concrete values to MrMotif's parameters. We have tried different combinations.
3. Build a dataset where each line corresponds to a line of the original dataset. Each relevant motif found in the previous step is a candidate attribute for this new dataset. The value of such attribute is the frequency of the motif in the corresponding time series.
4. Run a machine learning algorithm on the resulting dataset and estimate the predictive ability of the obtained classifier. The parameters of the MrMotif implementation are the following: motif length *m* - the length of the sliding window that contains the section to discretize in the original time series; number of motifs generated *K* - these are the top-*K* relevant motifs for each resolution from 4 to 64; word size *w* - this is the number of discrete symbols of the iSAX word; and overlap *o* the extent to which two windows can overlap. In Figure 2 we can see an example of a motif with 2 repetitions for one audio file from the dataset.

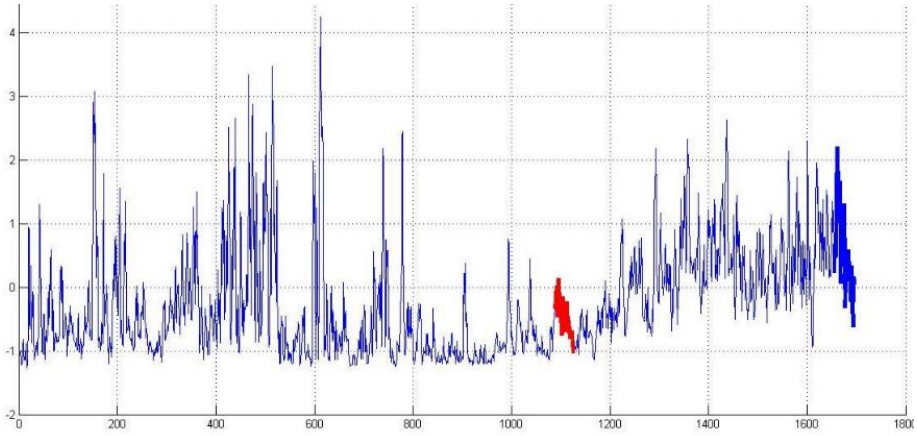

Figure 2:Motif with 2 repetitions of length 40 for one audio file

In Figure 3, a sample of one dataset is described. Each attribute Mi is a top-10 motif of resolution 4. Values are frequencies of motifs. The last column is class (CH, Car Horn and DB, Dog Bark). Classifiers were evaluated using the Weka data mining suite [23]. We proceed by describing the conducted experiments.

M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	Class
1	0	5	2	0	0	3	1	2	0	CH
2	2	7	4	3	0	5	0	0	0	CH
5	4	5	7	5	5	7	3	7	6	DB

Figure 3: Sample of a resulting dataset of resolution 4, Top-10.

6. Experiments

We have conducted experiments to compare the ability of the motifs approach and the MFCCs approach to discriminate the different sound classes. We have used different classification algorithms, from Weka data mining software [23] with 10-fold cross validation.
For each experiment, we report the average accuracy across all 10 folds. The algorithms used were J48, Random Forest, and SMO (a Support Vector Machine algorithm available in Weka).
In the first set of experiments we compare both approaches on the complete data set with all the classes simultaneously. The second set of experiments is composed of classification tasks for each pair of classes separately. Finally, the third set of experiments focus on four specific pairs of classes which are more challenging to discriminate. Three were identified in [19]: air_conditioner + engine_idling, drilling + jackhammer and siren + street_music. These authors explain that the difficulty may arise from the fact that the timbre of each pair is quite similar. A fourth pair, children and street music, was added since it had similar behavior.

6.1 Generating features

To obtain the motifs based features we have used the above described MrMotif algorithm. We varied the parameter K corresponding to the number of motifs selected as attributes. The motif resolution R was fixed with value 4. For window size we have used 40, and for overlap 20. The size of the SAX word is 8 symbols. In the experiments described below these are the default values. The MFCCs based features were obtained as described above. In these experiments we have used these two types of features in separate and combined.

6.2 Results

In Figure 4, at the left, we can see the Accuracy results for both approaches obtained using the classifiers on the complete data set (all the classes). In Figure 4, at the right, we see how the approaches perform, in average, for all pairs of classes. As we can see, in both experiments, MFCC features reveal better discriminant power.

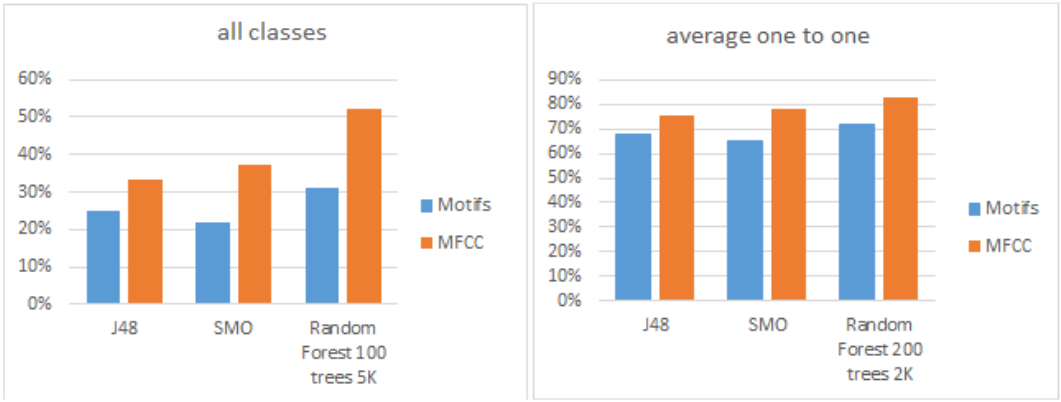

Figure 4:Motif and MFCCs classification for all classes (left) and for each pair of classes (right)

Now we focus on the above mentioned three difficult pairs of classes: air_conditioner + engine_idling, drilling + jackhammer, children_playing + street_music and also for siren + street_music. In these experiments we varied the motif parameters and used random forests [5] because, in previous experiments, we achieved better results with this classifier. This algorithm builds multiple decision trees with different attribute sets sampled from the original dataset. This results in complementary models that together have an effect similar to a probabilistic disjunctive model. The important parameters for the random forest algorithm are the number of models I and the number of attributes sampled for each model (in the tables we use the letter K for this parameter as originally used by the random forest Weka implementation, albeit the collision with the motif K parameter). In this experiments we used I = 200 and K = 5. For these four pairs, the motifs approach obtained visibly better results than MFCCs. Results were still improved combining features from MFCCs and motifs. In Figure 5 we can see the results using different parameter configurations.

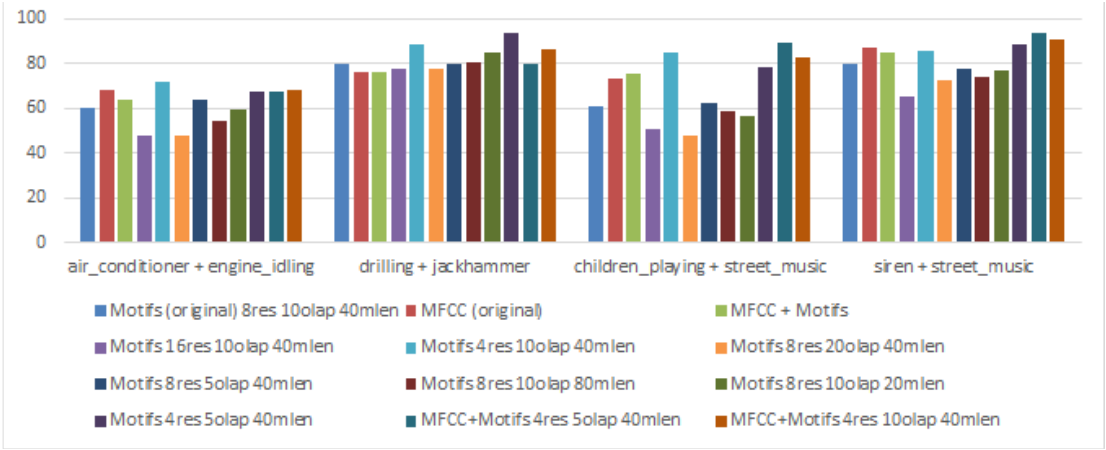

Figure 5: Motif and MFCCs classification for each pair of the four classes

Table 2 contains a selection of the best results. Motifs, in the case of air conditioner + engine_idling (72.09) and in the case air_conditioner + engine_idling (93.62) with resolution 4 (overlap 5 and 10 respectively). In the case of drilling + jackhammer and siren + street_music we achieved better results combining features.

Classes	MFCCs	MotifsR4O10	MotifsR4O5	MFCCs+MotifsR4O5
air_conditioner+eng_idling	68.60	**72.09**	67.44	67.44
drilling+jackhammer	76.60	88.30	**93.62**	79.79
child_playing+street_music	73.40	85.11	78.72	**89.36**
siren+street_music	87.31	85.82	88.81	**94.03**

Table 2: Comparison with the MFCCs approach and Motifs (varying number of motifs)

7. Conclusions and Future Work

In this paper we have presented results of the application of SAX-based motif discovery in time series to urban sound classification. We compare the results obtained with the results using MFCC because they are commonly used in environmental sound analysis.

Experiments were performed using the UrbanSound dataset. Our preliminary experiments suggest that motif features have good discriminating power with respect to urban audio sounds not captured in basic MFCCs. The combination of different resolutions can yield better results. The combination of motifs and MFCCs attributes also resulted in better performance. Our next step will be to explore with more detail these classes where these MFCCs features fail. We will also make a thorough sensitivity analysis of the approach with respect to the value of motif generation parameters such as window length, overlap, resolution and classification algorithm parameters. We can also use other features from MFCCs and try different combinations with motifs features. It is also important to study the influence of background noise when dealing with real world urban sounds [19]. To further improve results we intend to combine approaches to retrieve the characteristics of sound such as timbral texture, rhythmic content and pitch content [21].

8. ACKNOWLEDGEMENTS

This work is supported by FEDER Funds through the Programa Operacional Factores de Competitividade - COMPETE program and by National Funds through FCT-Fundação para a Ciência e a Tecnologia (project: FCOMP-01-0124-FEDERPEst-OE/EEI/UI0760/2014).

This paper is a revised and expanded version of a paper entitled “Classifying Urban Sounds using Multiresolution Time Series Motifs” presented at SUComS 2015, Porto, Portugal, June 21-23, 2015.

9. References

[1] https://serv.cusp.nyu.edu/projects/urbansounddataset/urbansound.html **(2015)**.

[2] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequencycepstral-coefficients-mfccs/ **(2015)**.

[3] https://github.com/jameslyons/python speech features **(2015)**.

[4] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. IEEE Signal Process. Mag., 32(3):16–34 **(2015)**.

[5] L. Breiman. Random forests. Machine Learning, 45(1):5–32 **(2001)**.

[6] N. Castro. Multiresolution motif discovery in time series website.http://www.di.uminho.pt/ castro/mrmotif.

[7] N. Castro and P. J. Azevedo. Multiresolution Motif Discovery in Time Series. In SDM, pages 665–676 **(2010)**.

[8] Y. Chen and Z. Tang. Research on noise processing and speech frame classification model of noisy short utterance signal processing. International Journal of Signal Processing, Image Processing and Pattern Recognition, 7(4):145–154 **(2014)**.

[9] S. Chu, S. S. Narayanan, and C. C. J. Kuo. Environmental sound recognition with time frequency audio features. IEEE Transactions on Audio, Speech & Language Processing,17(6):1142–1158, **(2009)**.

[10] M. A. Domingues, F. Gouyon, A. M. Jorge, J. P. Leal, J. Vinagre, L. Lemos, and M. Sordo. Combining usage and content in an online music recommendation system for music in the long-tail. In Proceedings of the 21stWorldWideWeb Conference,WWW2012 **(2012)** April 16-20, Lyon, France, (Companion Volume), pages 925–930, 2012.

[11] B. Elizalde, H. Lei, G. Friedland, and N. Peters. Capturing the acoustic scene characteristics for audio scene detection.

[12] P. G. Ferreira, P. J. Azevedo, C. G. Silva, and R. M. M. Brito. Mining approximate motifs in time series. In Discovery Science, pages 89–101 **(2006)**.

[13] E. F. Gomes, A. M. Jorge, and P. J. Azevedo. Classifying heart sounds using multiresolution time series motifs: an exploratory study. In C3S2E, pages 23–30 **(2013)**.

[14] E. F. Gomes, A. M. Jorge, and P. J. Azevedo. Classifying heart sounds using SAX motifs, random forests and text mining techniques. In 18th International Database Engineering & Applications Symposium, IDEAS 2014 **(2014)** July 7-9,Porto, Portugal, pages 334–337.

[15] T. Heittola, A. Mesaros, A. J. Eronen, and T. Virtanen. Context-dependent sound event detection. EURASIP J. Audio, Speech and Music Processing, 2013:1 **(2013)**.

[16] D. Hoiem, Y. Ke, and R. Sukthankar. SOLAR: sound object localization and retrieval in complex audio environments. In 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP ’05 **(2005),** March 18-23, Philadelphia, Pennsylvania, USA, pages 429–432.

[17] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series **(2002)** In Proceedings of the 2nd Workshop on Temporal Data Mining, pages 53–68.

[18] S. Ntalampiras, I. Potamitis, and N. Fakotakis. Automatic recognition of urban soundscenes. In New Directions in Intelligent Interactive Multimedia, pages 147–153 **(2008)**.

[19] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. **(2014)** Nov. In 22st ACM International Conference on Multimedia (ACM-MM’14), Orlando, FL, USA.

[20] J. Schroder, N. Moritz, M. R. Schadler, B. Cauchi, K. Adiloglu, J. Anem¨uller, S. Doclo, B. Kollmeier, and S. Goetze. On the use of spectro-temporal features for the IEEE AASP challenge detection and classification of acoustic scenes and events’ **(2013)** October 20-23. In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2013, New Paltz, NY, USA, pages 1–4.

[21] G. Tzanetakis, A. Ermolinskiy, and P. R. Cook. Pitch histograms in audio and symbolic music information retrieval **(2002)** October 13-17. In ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France.

[22] B. Uzkent, B. D. Barkana, and H. Cevikalp. Non-speech environmental sound classification using SVMs with a new set of features. International Journal of Innovative Computing, Information and Control ICIC International **(2012)**.

[23] I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques **(2005)**.

[24] D. Yankov, E. J. Keogh, J. Medina, B. Y. chi Chiu, and V. B. Zordan. Detecting time series motifs under uniform scaling. In P. Berkhin, R. Caruana, and X. Wu, editors, KDD, pages 844–853. ACM **(2007)**.

[25] E. F. Gomes, F. Batista. Classifying Urban Sounds using Time Series Motifs. Advanced Science and Technology Letters.Vol.97 (SUComS 2015), pp.52-57 **(2015)**.