Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática

Utilização da Tecnologia de Classificação de Sinais de Áudio para auxiliar os Deficientes Auditivos na Identificação de Eventos Sonoros

Luan Rodrigo Reis Pereira dos Santos

Recife

Janeiro de 2015

Luan Rodrigo Reis Pereira dos Santos

Utilização da Tecnologia de Classificação de Sinais de Áudio para auxiliar os Deficientes Auditivos na Identificação de Eventos Sonoros

Orientadora: Teresa M. Maciel
Co-orientador: Giordano R. E. Cabral

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife

Janeiro de 2015

À minha querida esposa, Jéssika
Ao meu pai, Luiz
À minha mãe, Fátima
Aos meus irmãos
Aos meus amigos

Agradecimentos

Agradeço a minha família, por terem me incentivado a buscar o conhecimento que fez de mim o que sou hoje.

À minha esposa, Jéssika, por ser tão especial, carinhosa, cuidar de mim e me fazer infinitamente feliz.

Ao José Menezes, mestrando em Informática Aplicada na UFRPE, que me ajudou desde o inicio a entender os conceitos necessários para o desenvolvimento deste trabalho.

Aos meus orientadores Teresa e Giordano, por terem me guiado na construção deste trabalho.

À minha amiga Manola (Thais), por ter enfretando ao meu lado muitos dos momentos que nos fez o que somos hoje.

Aos meus amigos que me trouxeram momentos inesquecíveis e que ainda o fazem.

Resumo

A Classificação de Sinais de Áudio (CSA) é uma área da computação em constante melhorias. Atualmente a CSA é utilizada de diversas maneiras, tais como na transcrição automática de música ou texto, no reconhecimento de voz, e na busca rápida em bancos de dados. Existem projetos como o jMIR (ferramenta de Recuperação de Informação Musical Musical Information Retrieve) que fornecem uma maneira de solucionar o problema da classificação de áudio, que consiste na identificação da classe em que um áudio está inserido. Essa classificação é feita por meio de diversos algoritmos já implementados e testados pela comunidade científica. Entretanto, fez-se necessário um estudo de alguns conceitos fundamentais, sendo eles: a extração de caracteristicas, os métodos de classificação e a taxonomia, que são essenciais para que se possa aplicar esta tecnologia na prática. Através da abordagem desses conceitos este trabalho propõe, por meio de um protótipo, uma alternativa de solucionar um problema real que consiste na dificuldade que as pessoas com deficiência auditiva têm de identificar os eventos sonoros que acontecem ao seu redor.

Palavras-chave: Classificação de sinais de áudio, deficientes auditivos, jMIR

Abstract

Audio Signal Classification (ASC) is an area of computing in constant improvements. Currently the ASC is used in various ways, such as in automatic transcription of music or text, speech recognition, and quick search databases. There are projects like jMIR (Musical Information Retrieve tool) that provide a way to solve the problem of audio classification, which is the identification of the class in which a audio is inserted. This classification is made through various algorithms already implemented and tested by the scientific community. However, it was necessary a study of some fundamental concepts, namely: feature extraction, classification methods and taxonomy, which are essential so that we can apply this technology in practice. By addressing these concepts this work proposes, through a prototype, an alternative to solve a real problem that is the difficulty that people with hearing loss have to identify the sound events that happen around.

Keywords: Audio signal classification, hearing impaired, jMIR

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Classificação de Sinais de Áudio (CSA) é um problema já consagrado na ciência da computação. Existem diversas implementações de algoritmos e aplicações para este fim. Tem-se como exemplo o jMIR (ferramenta de Recuperação de Informação Musical
- Musical Information Retrieve), que busca trazer de uma forma simples ferramentas que auxiliam tanto na classificação, como na análise e processamento de sinais de áudio.

Este trabalho traz a proposta de criar um protótipo, cujo objetivo é fazer uso da tecnologia classificação de sinais de áudio, por meio de ferramentas existentes, que traga alguma evidência de que é possível fazer uso desta tecnologia para ajudar pessoas com deficiência auditiva.

1.1 Motivação

Este trabalho foi impulsionado por uma ideia gerada a partir de um Brainstorming, cuja finalidade era de criar uma aplicação para dispositivos móveis que fosse capaz de ajudar pessoas com deficiência auditiva fazendo uso da tecnologia de classificação de sinais de áudio.

Segundo o censo de 2010 do IBGE cerca de 45.623.910 pessoas, que equivalia a 23,9% da população brasileira, declarou apresentar algum tipo de deficiência física. Sendo 9.722.163 de deficientes auditivos [2]. Parte disso é devido a intensidade de sons que as pessoas estão submetidadas no dia-a-dia. Sons emitidos durante o tráfego de automóveis, obras em andamento, shows, boates e as músicas em volume exagerado nos fones de ouvido, são alguns

exemplos que contribuem para a perda da audição [3]. Segundo especialistas o limite saudável à exposição aos sons é de 80 decibéis (dB). Uma exposição diária acima desse limite pode causar danos irreversíveis para o indivíduo [4].

1.2 Objetivo Geral

O objetivo deste trabalho é mostrar que é possível fazer uso da tecnologia de classificação de sinais de áudio para ajudar pessoas com deficiência auditiva.

1.3 Objetivos Específicos

É preciso assimilar uma série de conhecimentos para fazer uso de uma tecnologia complexa como a da classificação de sinais de áudio. Sendo assim, os objetivos específicos do trabalho, são:

- Compreender a metodologia de classificação de sinais de áudio;
- Identificar os recursos necessários para se utilizar a ferramenta adotada;
- Demonstrar, através de um protótipo, que a tecnologia pode resolver o problema da identificação de eventos sonoros;

1.4 Organização do Trabalho

A monografia está escrita em cinco capítulos, que visam abordar desde o problema da CSA, quanto a elaboração de uma possível solução que possa ajudar as pessoas com definicência auditiva.

No Capítulo 2 é descrito o problema da classificação de sinais de áudio, bem como alguma das áreas de aplicação. Além de abordar os desafios e os requisitos necessários para compreendê-la.

No Capítulo 3 são apresentados os fundamentos necessários para um melhor entendimento do problema da CSA. Estes conhecimentos são utilizados para gerar a solução proposta por este trabalho.

No Capítulo 4 é apresentada a metodologia utilizada e os resultados obtidos através de um experimento ao qual o protótipo foi submetido.

Por fim, tem-se as conclusões gerais e algumas sugestões do que pode ser feito no futuro para dar continuidade a este trabalho.

Capítulo 2

O Problema

A Classificação de Sinais de Áudio (CSA) é um problema que já existe há um certo tempo na área de computação. A CSA consiste em extrair informações de um sinal sonoro em forma de características, ou parâmetros, no qual podem ser utilizados para classifica-lo baseado em algum critério pré-estabelecido [9].

Existem diversos estudos desenvolvidos ao longo dos anos que contribuíram para que a CSA chegasse ao estágio atual. A CSA tem sido utilizada em diversas áreas, como a transcrição automática de música ou texto, o reconhecimento de voz, a busca rápida em bancos de dados. Entretanto, a CSA só tem alcançado resultados satisfatórios em aplicações bastante restritas [9].

2.1 Desafios

A CSA é uma área da computação que faz uso de mencanismos bastante complexos como a Inteligência Artificial e, principalmente uma de suas subáreas, que é o Aprendizado de Máquina [13]. Entretanto, como o foco desse trabalho é mostrar que a aplicação da técnica da CSA pode trazer algum benefício para o problema dos deficiêntes auditivos, serão utilizadas ferramentas e algorítmos existentes.

Ainda assim, faz-se necessário entender determinados fundamentos que são essenciais para utilizar esta técnologia. E também, é preciso entender o funcionamento das ferramentas adotadas, sendo elas: o jAudio e o ACE componentes do jMIR.

2.2 Requisitos

Primeiramente, é necessário que se tenha uma base de áudios para que seja possível treinar o classificador de forma que ele saiba como classificar um sinal sonoro. A qualidade do som interfere fortemente no resultado final, ou seja, se um som tiver sido gravado em um ambiente com o mínimo de ruídos, o classificador terá uma mair chance de classificar o som de forma correta. Caso contrário, a chance de errar na classificação será bastante elevada.

Além disso, é preciso encontrar uma ou mais ferramentas que forneçam as funcionalidades de extração de características e de classificação de áudio. No caso desse trabalho a ferramenta escolhida foi o jMIR.

O jMIR é uma suíte de software de código aberto implementado em Java, desenvolvido por Cory McKay, para uso na recuperação de informação de musical. Ele oferece suporte tanto para a manipulação de arquivos de áudio, quanto para arquivos de formato simbólico. Estão incluídos no jMIR softwares para extração de características de áudio, aplicação de algoritmos de aprendizado de máquina, mineração de metadados e análise de metadados [11].

Capítulo 3

Estado da arte

Este capítulo está estruturado em três segmentos: a seção 3.1 descreve o conceito de recuperação de informações musicais; a seção 3.2 descreve os conhecimentos necessários durante o processo de classificação de sinais de áudio; e por fim, na seção 3.3, são descritas algumas das ferramentas que auxiliam neste processo.

3.1 Recuperação de Informação Musical

A recuperação de informação musical, também denominada MIR (Musical Information Retrieve), é uma área de pesquisa associada à extração de informações ou meta-informações de sinais de áudio. O MIR tem por objetivo principal a classificação automática de sinais de áudio, mas não se limita apenas a esta funcionalidade. Ele é aplicado em diversas áreas, como por exemplo o aprendizado de máquina, mineração de dados, processamento de sinal digital, teoria musical, musicologia, educação musical e etc [11].

3.2 Classificação de Sinais de Áudio

Para que seja possível classificar um sinal de sonoro, é preciso primeiramente que haja uma definição dos componentes que serão utilizados no processo de classificação. Estes componentes são a taxonomia, as características a serem extraídas, e o método de classificação a ser utilizado.

3.2.1 Taxonomia

É preciso definir classes para os diferentes tipos de sinais de áudio que se deseja classificar, para que quando o classificador reconheça um som ele saiba dizer qual classe ele pertence. A definição das classes é feita através da taxonomia. A Figura 3.1 ilustra um exemplo de estrutura taxonômica.

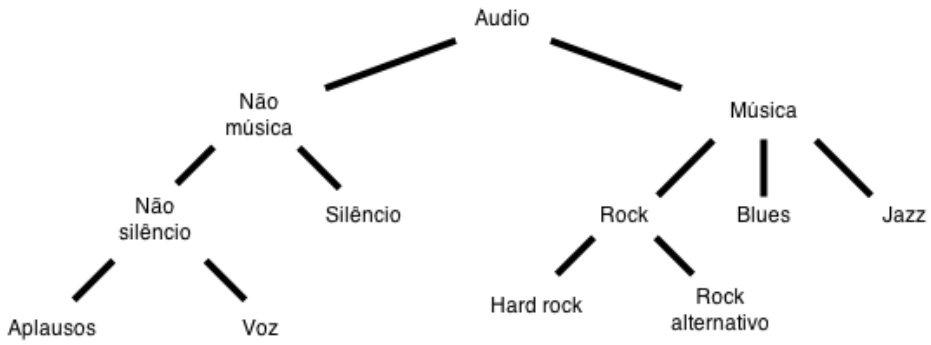

Figura 3.1: Exemplo de estrutura taxonômica.

A taxonomia é a maneira de se definir como um conjunto de determinados elementos será classificado. Podendo ser composta de diversos níveis hierárquicos, no qual os níveis mais baixos representam classificações mais restritas e específicas [9].

3.2.2 Extração de Características

A extração de características, também chamada de extração de parâmetros, é a primeira etapa da classificação de sinais [9]. Por meio deste processo é possível extrair informações de um arquivo de áudio que podem ser utilizadas na etapa de classificação, bem como na análise de áudio [11].

Existe uma enorme quantidade de possíveis características que podem ser extraídas de um sinal sonoro, fruto de diversas pesquisas na área. Dentre as mais utilizadas estão: Centróide Espectral, Taxa de Cruzamentos por Zero, Coeficientes Cepstrais, Pitch, Largura de Faixa, Fluxo Espectral, Proporção de Quadros com Baixa Energia, Sonoridade, Harmonicidade e o Ponto de Roll-Off [9].

3.2.3 Métodos de Classificação

Os métodos de classificação são o meio pelo qual se combina os valores extraídos das características de um sinal sonoro com a finalidade de produzir uma classificação confiável para o sinal analisado. Cada um deles implementa uma maneira diferente de manipular essas informações, nos quais podem gerar diferentes resultados na etapa de classificação. Alguns dos classificadores mais utilizados são: K Vizinhos mais Próximos, Modelos de Mistura Gaussiana, Modelos Ocultos de Markov, Máquinas de Vetores Suporte (Support Vector Machines) e o Discriminador Binário Simples [9].

3.3 Ferramentas

Como dito na seção 2.2 do Capítulo 2, o jMIR é uma suíte de softwares, portanto, ele possuí softwares para diversas necessidades relacionadas a extração e análise de áudio. Estão incluídos no jMIR os seguintes componentes: ACE, ACE XML, jMIRUtilities, jAudio, jSymbolic, jWebMiner, jLyrics, jProductionCritic, jSongMiner, jMusicMetaManager, Codaich, Bodhidharma MIDI, e o SLAC. [11].

Como o foco deste trabalho é apenas a classificação de sinais de áudio, nem todos os componentes incluídos no jMIR serão necessários. Dentre os que serão utilizados estão o ACE XML, o jAudio e ACE.

3.3.1 ACE XML

O ACE XML é o formato nativo utilizado por todos os componentes do jMIR, é através dele que seus componentes conseguem se comunicar uns com os outros [11]. Além do ACE XML, atualmente, não existe um formato padrão para o armazenamento de informações provenientes do MIR. Por conta disso, o formato Weka ARFF foi adotado como sendo o padrão, embora ele não tenha sido projetado com esse propósito [7]. O Weka ARFF possui uma série de limitações importantes, quando utilizado no estudo de MIR [12].

O ACE XML possui quatro formatos principais, são eles [11]:

- Feature Value (ou Feature Vector): Especificam os valores das características que foram

extraídos do áudio. A Figura 3.2 ilustra um exemplor de arquivo de *feature value*;

- Feature Definitions (ou Feature Key/Description): Descrevem, abstratamente, informações a cerca das característas que foram extraídas do áudio, assim como o processo utilizado na obtenção de tais características. A Figura 3.2 ilustra um exemplor de arquivo de *feature definitions*;

- Instance Label (ou Classifications): Estes arquivos têm como propósito etiquetar determinadas intâncias, podendo ser utilizadas para prever os resultados de saída de uma classificação, ou para expressar explicitamente que uma determináda instância pertence a uma determinada classe. A Figura 3.3 ilustra um exemplor de arquivo de *instance label*;

- Taxonomy (ou Class Ontology): São os arquivos que expressam as relações entre as classes. Podendo ser utilizados para especificar apenas possíveis classes, ou uma estrutura taxonômica completa. A Figura 3.3 ilustra um exemplor de arquivo de *taxonomy*.

Como o próprio nome já sugere, o padrão ACE XML foi construído utilizando a linguagem de marcação XML (eXtensible Markup Language) [15]. Os dados de um arquivo XML são estruturados em essencialmente duas partes: uma delas especifica a forma com a qual os dados serão estruturados, e a outra que consiste em armazenar os dados de acordo com a estrutura especificada [15].

```
<?xml version="1.0"?>
<!DOCTYPE feature_vector_file [
   <!ELEMENT feature_vector_file (comments, data_set+)>
   <!ELEMENT comments (#PCDATA)>
   <!ELEMENT data_set (data_set_id, section*, feature*)>
   <!ELEMENT data_set_id (#PCDATA)>
   <!ELEMENT section (feature+)>
   <!ATTLIST section start CDATA ""
                     stop CDATA "">
   <!ELEMENT feature (name, v+)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT v (#PCDATA)>
]>

<feature_vector_file>

   <data_set>
      <data_set_id>C:\Recordings\Handel_4.wav</data_set_id>
      <section start="0" stop="99">
      <feature>
       <name>Spectral Centroid</name>
       <v>0.0</v>
      </feature>
    </section>
      <section start="100" stop="199">
      <feature>
       <name>Spectral Centroid</name>
       <v>440.0</v>
      </feature>
    </section>
      <feature>
   </data_set>

   <data_set>

   <data_set>

</feature_vector_file>
```

(a) Arquivo de Faeture Value

```
<?xml version="1.0"?>
<!DOCTYPE feature_key_file [
   <!ELEMENT feature_key_file (comments, feature+)>
   <!ELEMENT comments (#PCDATA)>
   <!ELEMENT feature (name, description?, is_sequential, parallel_dimensions)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT description (#PCDATA)>
   <!ELEMENT is_sequential (#PCDATA)>
   <!ELEMENT parallel_dimensions (#PCDATA)>
]>

<feature_key_file>

   <feature>
      <name>Spectral Centroid</name>
      <description>The spectral centre of mass of a signal window.</description>
      <is_sequential>true</is_sequential>
      <parallel_dimensions>1</parallel_dimensions>
   </feature>

   <feature>
      <name>Duration</name>
      <description>The duration in ms of a recording.</description>
      <is_sequential>false</is_sequential>
      <parallel_dimensions>1</parallel_dimensions>
   </feature>

</feature_key_file>
```

(b) Arquivo de Feature Description

Figura 3.2: Exemplo de arquivos de *feature value* e *feature description* (feature definition).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE classifications_file [
<!ELEMENT classifications_file (comments,data_set+)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT data_set (data_set_id,misc_info*,role?,classification)>
<!ELEMENT data_set_id (#PCDATA)>
<!ELEMENT misc_info (#PCDATA)>
<!ATTLIST misc_info info_type CDATA "">
<!ELEMENT role (#PCDATA)>
<!ELEMENT classification (section*,class*)>
<!ELEMENT section (start,stop,class+)>
<!ELEMENT class (#PCDATA)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT stop (#PCDATA)>
]>

<classifications_file>
  <data_set>
    <data_set_id>C:\Recordings\Handel_4.wav</data_set_id>
    <misc_info info_type="Composer">Handel</misc_info>
    <role>training</role>
    <classification>
      <section>
        <start>0</start>
        <stop>89</stop>
        <class>Silence</class>
      </section>
      <section>
        <start>90</start>
        <stop>194</stop>
        <class>Music</class>
      </section>
      <section>
    </classification>
  </data_set>
  <data_set>
</classifications_file>
```

(a) Arquivo de Instance Label

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE taxonomy_file [
<!ELEMENT taxonomy_file (comments,parent_class+)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT parent_class (class_name,sub_class*)>
<!ELEMENT class_name (#PCDATA)>
<!ELEMENT sub_class (class_name,sub_class*)>
]>

<taxonomy_file>

  <parent_class>
    <class_name>Music</class_name>
  </parent_class>

  <parent_class>
    <class_name>NotMusic</class_name>

    <sub_class>
      <class_name>NotSilence</class_name>

      <sub_class>
        <class_name>Applause</class_name>
      </sub_class>
      <sub_class>
        <class_name>Speech</class_name>
      </sub_class>

    </sub_class>

    <sub_class>
      <class_name>Silence</class_name>
    </sub_class>
  </parent_class>

</taxonomy_file>
```

(b) Arquivo de Class Ontology

Figura 3.3: Exemplo de arquivos de *instance label* e *class ontology (taxonomy)*.

3.3.2 jAudio

O jAudio é o software responsável por extrair informações, em forma de valores, das características de um sinal sonoro. Ele pode ser utilizado através de uma interface gráfica ou por meio da linha de comando.

As características extraídas através do jAudio são exportadas para o formato ACE XML, mas também podem ser exportados para o formato ARFF [10]. O jAudio inclui 26 características principais, todas comprovadas em pesquisas de MIR, e algumas outras características experimentais [11].

Ao extrair as características dos arquivos de áudio, o jAudio exporta tanto um arquivo de *feature values* quanto um arquivo de *feature definitions*, ambos descritos na sub-seção 3.3.1. Também é possível extrair características de mais de um arquivo de áudio ao mesmo tempo, como ilustra a Figura 3.4.

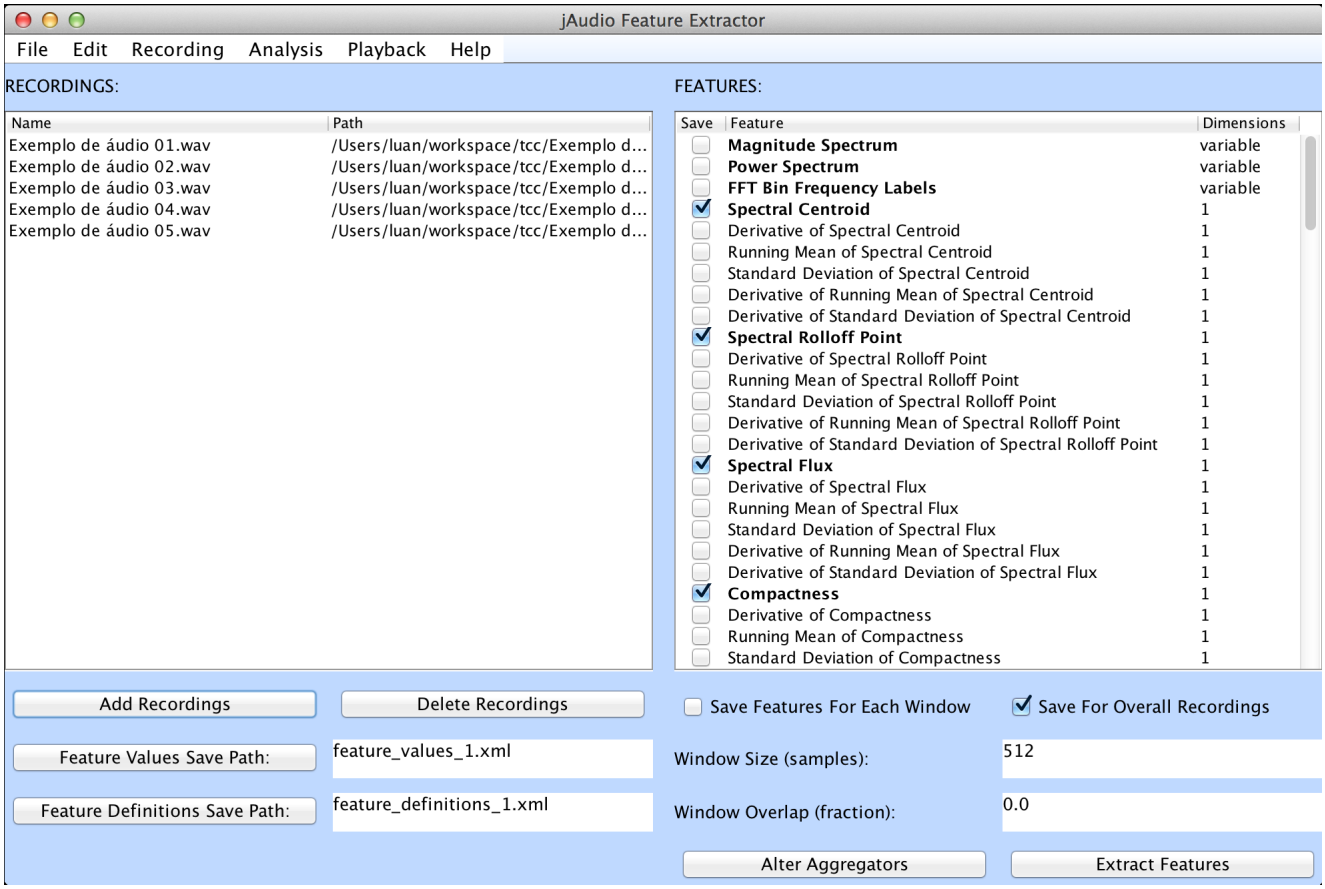

Figura 3.4: Interface gráfica do jAudio.

3.3.3 ACE

Pode-se dizer que o ACE (Autonomous Classification Engine) é o componente principal do jMIR. Ele é responsável por aplicar a aprendizagem de máquina, fazendo uso dos valores obtidos das características de um determinado sinal sonoro, com a finalidade de aprender automaticamente com esses dados e associá-los com as classes especificadas na taxonomia [11]. Além disso, o ACE é capaz de experimentar automaticamente uma variadade de diferentes algoritmos de aprendizagem de máquina e de redução de dimensionalidade, avaliando qual o mais indicado para um determinado problema de classificação [14].

Atualmente só é possível utilizar todas as fucionalidades do ACE através da linha de comando, pois sua interface gráfica ainda está em desenvolvimento. Portanto, até o momento em que esse trabalho foi escrito, só é possível gerenciar os arquivos de *feature value*, os de *feature definition*, os de *instance label* e os de *taxonomy* [14]. As Figuras 3.5, 3.6, 3.7 e 3.8 ilustram esses arquivos sendo gerenciados através da interface gráfica do ACE.

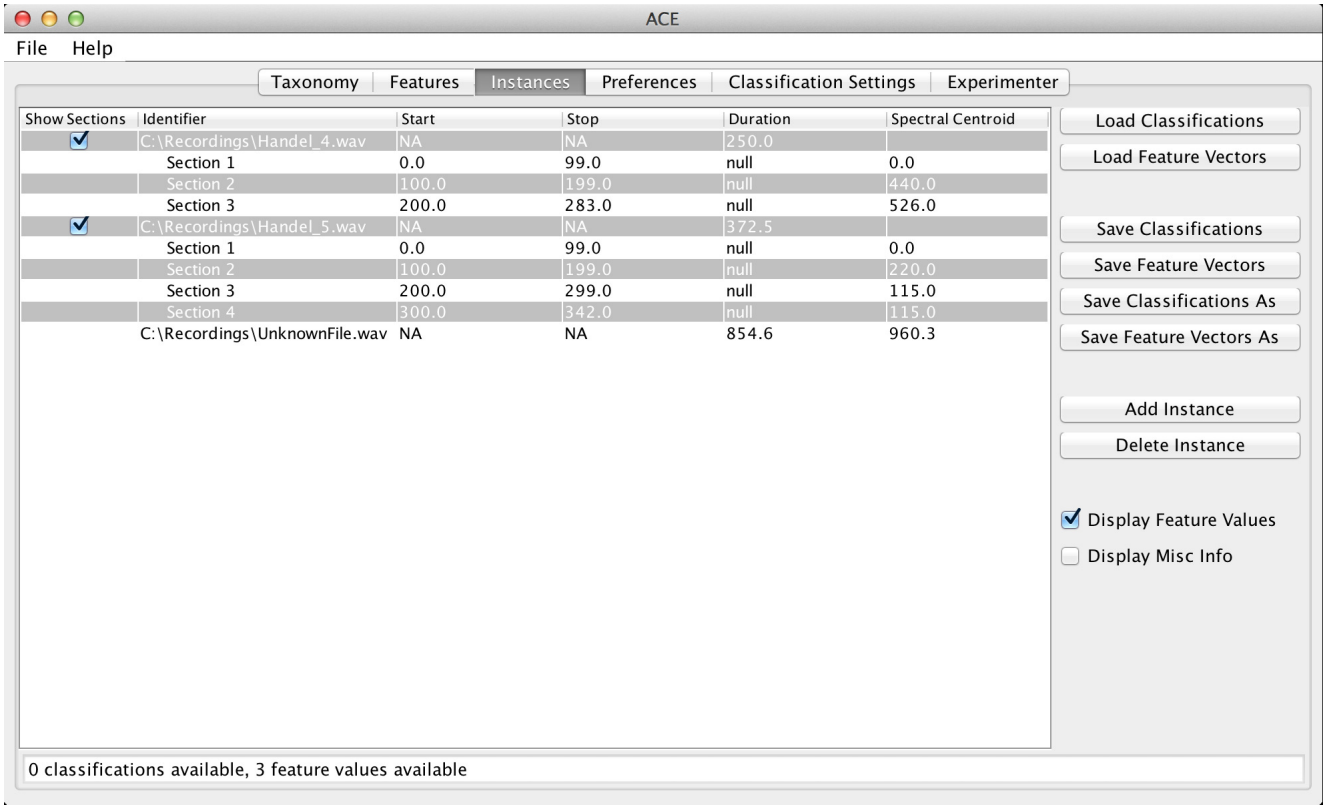

Figura 3.5: Arquivo de *feature value* carregado na interface gráfica do ACE.

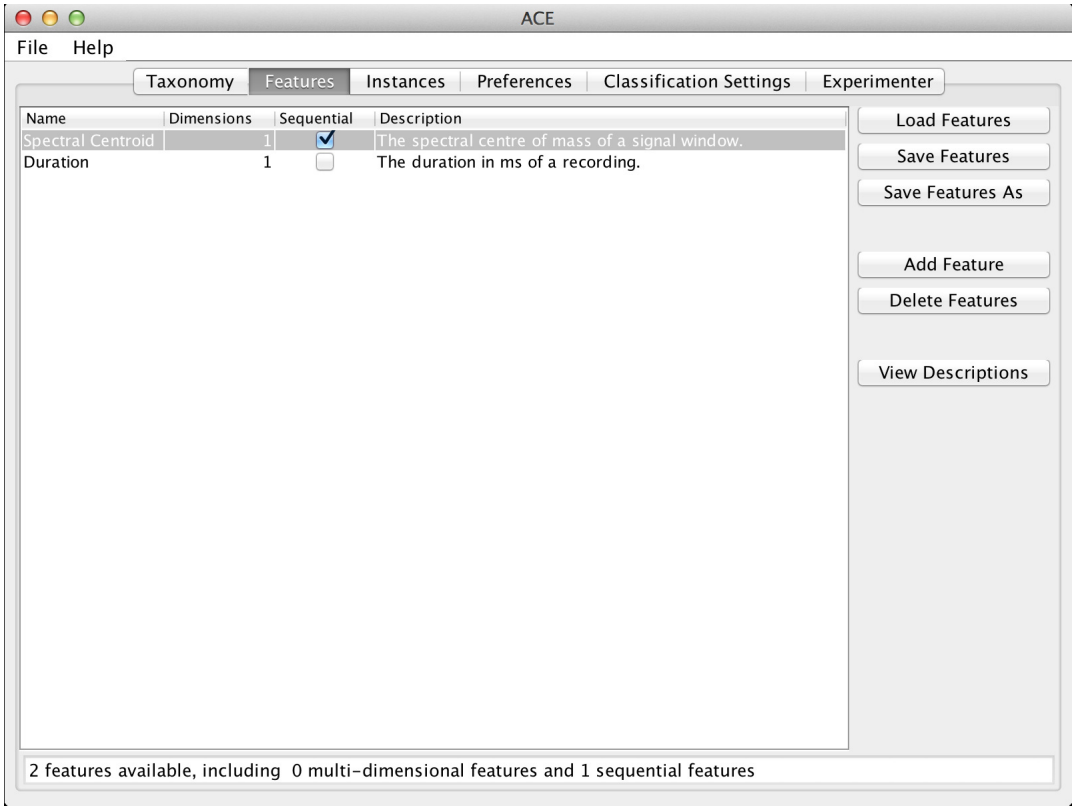

Figura 3.6: Arquivo de *feature definition* carregado na interface gráfica do ACE.

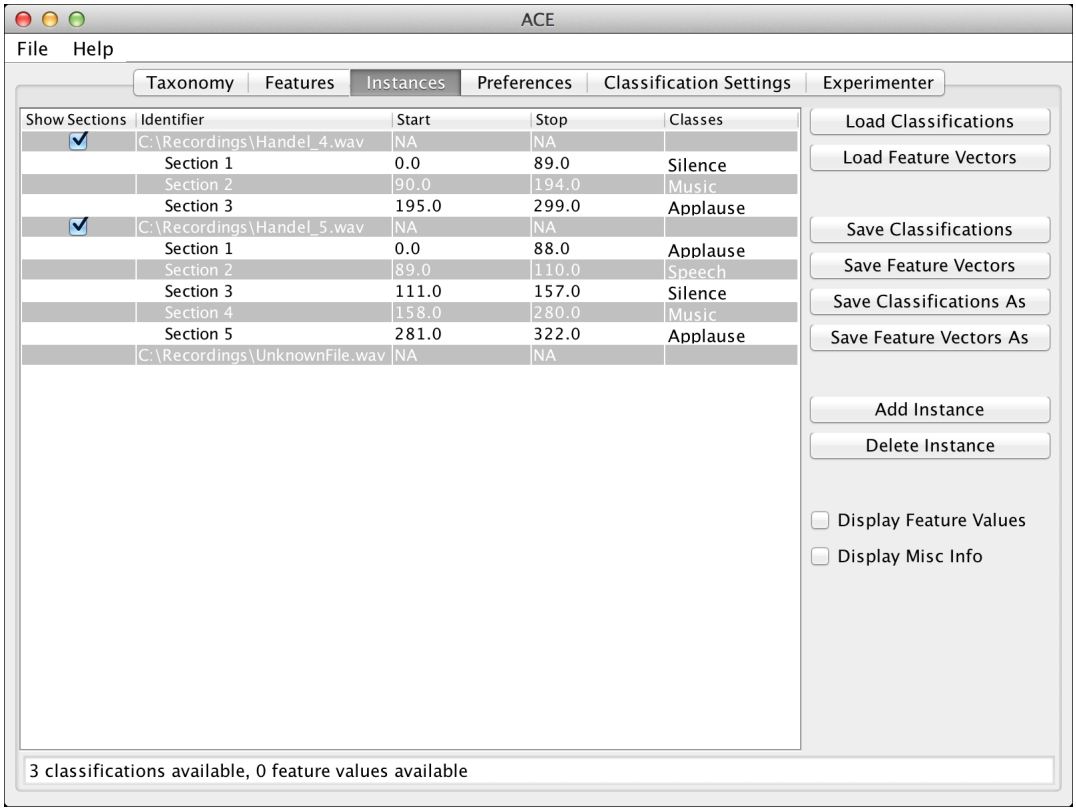

Figura 3.7: Arquivo de *instance label* carregado na interface gráfica do ACE.

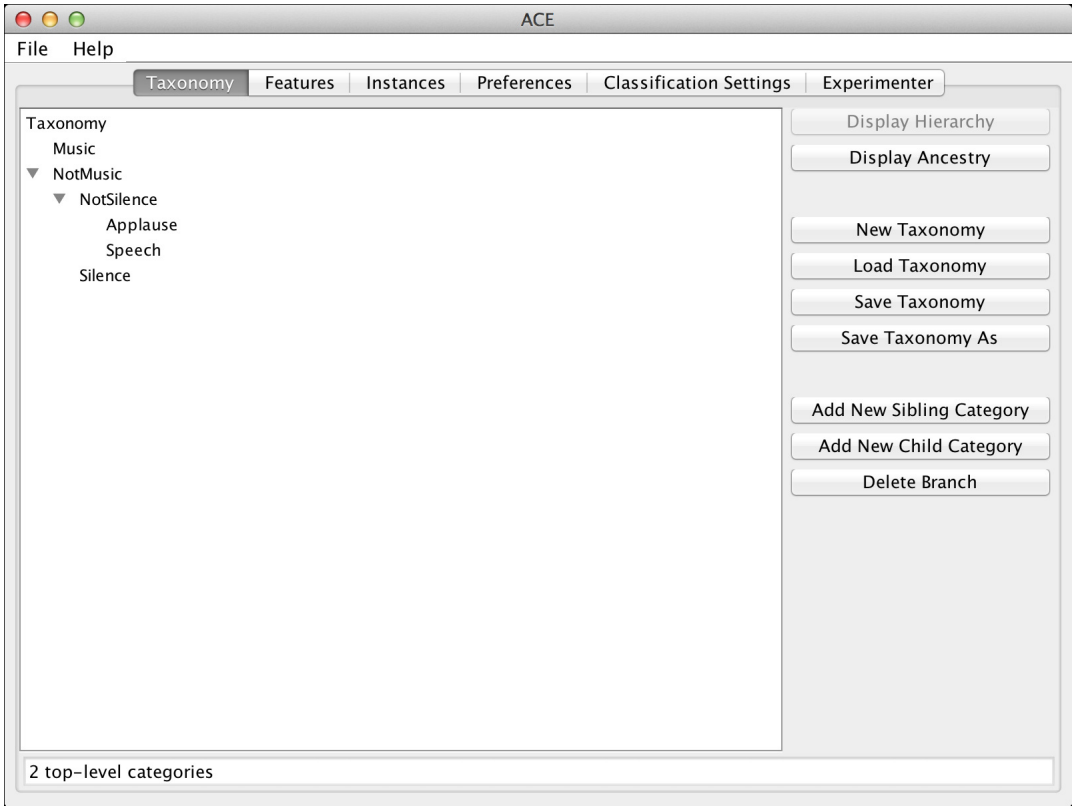

Figura 3.8: Arquivo de *taxonomy* carregado na interface gráfica do ACE.

Carregar e gerenciar estes arquivos é só uma pequena parte do que o ACE fornece. Dentre as suas principais funcionalidades estão:

- Training: Por meio de um arquivo de *feature value*, *feature definition* e *instance label*, é possível treinar um classificador utilizando um dos classificadores que o ACE fornece.
- Classifying: Dado que existe um classificador previamente treinado, é possível fazer a classificação de um áudio, por meio dos arquivos de *feature value* e *feature definitions* extraídos através de algum software como o jAudio, e também, é necessário um arquivo de *taxonomy*.
- Cross Validating: O ACE fornece a funcionalidade de executar o método de cross-validating para avaliar a performance de um classificador. Este método consiste na formação de pares, escolhidos de forma aleatória, para cada instância contida no arquivo de *instance label*, além do classificador a ser avaliado. Em seguida será exibida a taxa de sucesso de classificação entre cada par de instâncias, bem como uma matriz de confusão.
- Experimenting: Consiste em fazer uso de diversas técnicas de classificação a fim de encontrar o melhor classificador para o tipo de problema. As instâncias serão submetidas a quatro diferentes tipos de redutores de dimensionalidade, que são o Principal Components Analysis (PCA), Exhaustive search utilizando o classificador naive Bayesian, Genetic search utilizando o classificador naive Bayesian, e a seleção de nenhuma característica. Para cada cada um dos redutores de dimensionalidade serão testados todos os classificadores disponíveis no ACE utilizando o método de cross-validating. Em seguida é criado um novo classificador, cujos resultados foram os mais assertivos entre os demais, e treinado com as instâncias iniciais.

É interessante notar que como o ACE foi construído sobre o padrão de infraestrutura de aprendizagem de máquina do Weka, sempre que novos algoritmos forem produzidos pela comunidade do Weka eles serão imediatamente incorporados ao ACE [14].

Capítulo 4

Metodologia e Resultados

Este capítulo descreve a metodologia utilizada para criação do protótipo mencionado no capítulo introdutório, e os resultados dos experimentos realizados por meio de sua execução utilizando áudios externos referentes aos eventos escolhidos para o experimento.

4.1 Metodologia

A proposta de solução deste trabalho consiste em mostrar que é possível utilizar a classificação de sinais de áudio por meio da criação de um protótipo visando oferecer uma forma de auxiliar as pessoas que possuam deficiência auditiva. O processo seguido para este fim é ilustrado na Figura 4.1.

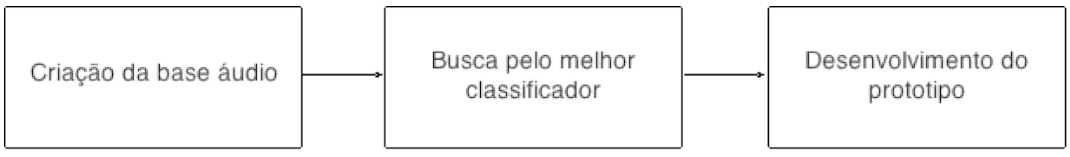

Figura 4.1: Metodologia adotada.

Inicialmente foi preciso conseguir uma base de áudios, contendo um número razoável de amostras para os tipos de eventos que foram selecionados. Em seguida foi escolhido um classificador através do método de experimenting, descrito na subseção 3.3.3, com o propósito de garantir uma maior taxa de acerto durante a classificação dos áudios que foram testados.

Em seguida, foi criado um protótipo responsável por fazer uso deste classificador e classificar o arquivo de áudio de entrada em uma das classes definidas na taxonomia. Para isso, o protótipo teve que se comunicar tanto com o jAudio, pois é necessário que o arquivo de entrada passe pelo processo de extração de características; e também com o ACE, que faz a classificação utilizando os arquivos gerados pelo jAudio.

Finalmente, foram testados alguns arquivos de áudio de entrada, obtidos de uma fonte externa, para validar que o protótipo consegue operar corretamente e exibir o resultado da classificação.

4.1.1 Criação da base de áudio

Como mencionado na seção 2.2, é preciso ter uma base de áudios de qualidade para que o classificador consiga classificar a entrada recebida de modo mais assertivo. Sendo assim, os arquivos da base de áudio usados para construir o protótipo foram fornecidos pela empresa Escribo S.A. (antiga Daccord Music Software S.A.), cujos arquivos foram utilizados em um de seus produtos, o Áudio Alerta [1].

O Áudio Alerta é fruto de um projeto com incentivos da Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) e Financiadora de Estudos e Projetos (FINEP). O seu propósito é monitorar ambientes por meio da detecção e localização de sons de eventos como; disparo de tiro, batidas, explosões, e quedas [1]. A Figura 4.2 ilustra o Áudio Alerta instalado e em uso.

Figura 4.2: Áudio Alerta.

A base de áudios do Áudio Alerta possui 56 tipos de eventos diferentes, totalizando

4899 amostras de áudio. Ela possui áudios de eventos, como buzina, alarme, sirene, grito, latido, freio, furadeira, batida, som emitido pelos automóveis, som de ambiente, disparo de armas de fogo de diversos calibres e etc.

Em geral, um mesmo evento sonoro possui diferentes valores para suas características, visto que existem diversas variáveis que podem influenciar na captação do áudio, como por exemplo a distância; o ambiente; a acústica; os equipamentos utilizados; ruídos e etc. Portanto, a quantidade de amostras de um mesmo evento ajuda o classificador a entender melhor o tipo de som.

Como a proposta desse trabalho se baseia na criação de um protótipo, os tipos de eventos utilizados para mostrar que é possível classificar sinais de áudio foram limitados a apenas dois, que foram os áudios de buzina e de alarme, cujo número de amostras são de 44 e 43 respectivamente.

4.1.2 Busca pelo melhor classificador

Primeiramente foi utilizado o jAudio para extrair as características dos arquivos de áudio. As características selecionadas foram as que o próprio jAudio propõe a serem utilizadas por padrão, que são: Spectral Centroid, Spectral Rolloff Point, Spectral Flux, Compactness, Spectral Variability, Root Mean Square, Fraction Of Low Energy Windows, Zero Crossings, Strongest Beat, Beat Sum, Strength Of Strongest Beat, MFCC (Mel-Frequency Cepstral Coefficients), LPC (Linear Predictive Coding) e Method of Moments. A Figura 4.3 ilustra a extração das características dos arquivos da base de áudio atravez do jAudio.

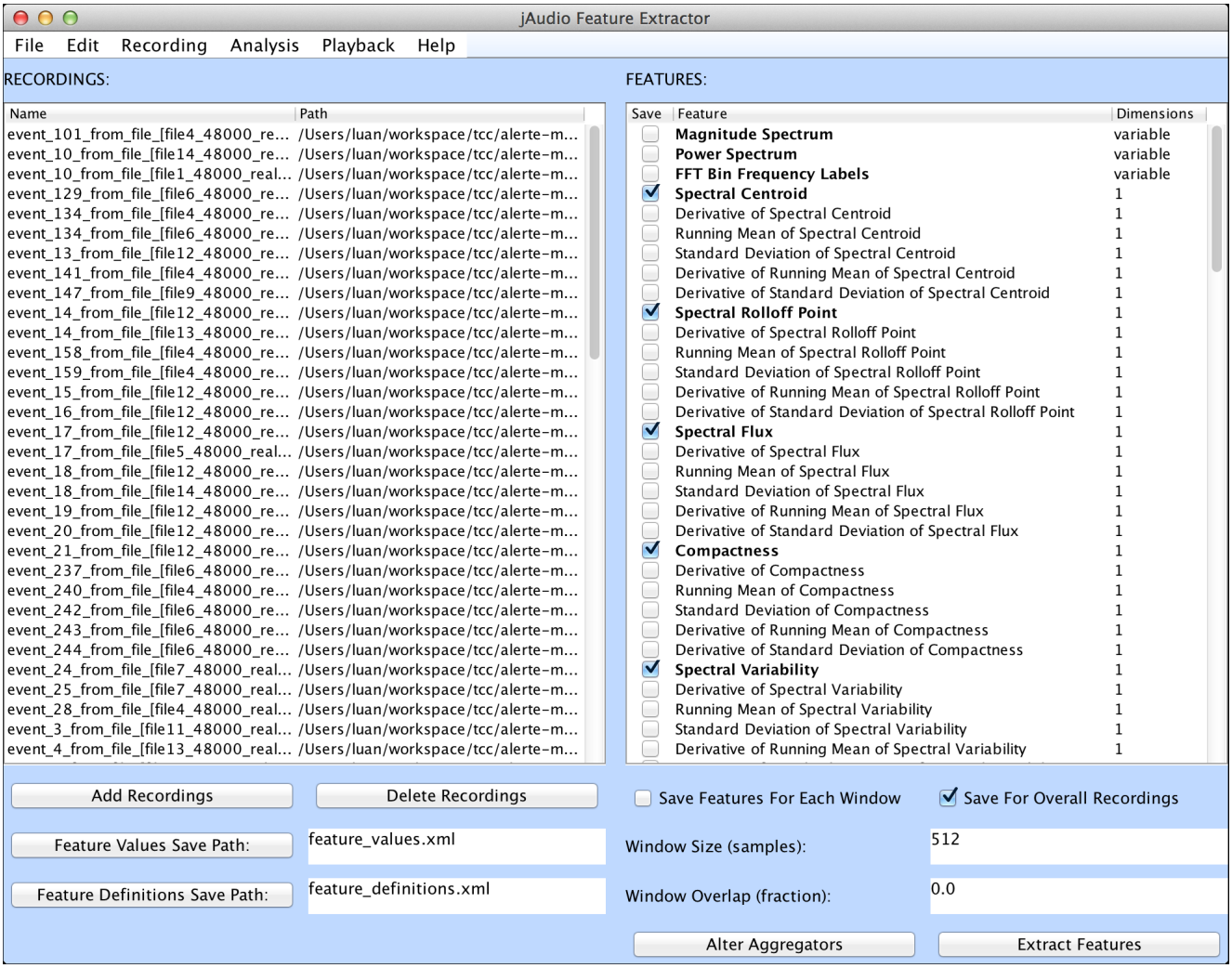

Figura 4.3: Extração de características das amostras utilizando o jAudio.

Além dos arquivos de *feature values* e *feature definitions* gerados pelo jAudio após a extração de características, também é necessário haver um arquivo de i*nstance label* para que seja possível executar a função de *experimenting* do ACE. A Figura 4.4 apresenta o arquivo de *instance label* das amostras. E a Figura 4.5 apresenta o arquivo de *taxonomy* utilizado.

```
<?xml version="1.0"?>
<!DOCTYPE classifications_file [
   <!ELEMENT classifications_file (comments, data_set+)>
   <!ELEMENT comments (#PCDATA)>
   <!ELEMENT data_set (data_set_id, misc_info*, role?, classification)>
   <!ELEMENT data_set_id (#PCDATA)>
   <!ELEMENT misc_info (#PCDATA)>
   <!ATTLIST misc_info info_type CDATA "">
   <!ELEMENT role (#PCDATA)>
   <!ELEMENT classification (section*, class*)>
   <!ELEMENT section (start, stop, class+)>
   <!ELEMENT class (#PCDATA)>
   <!ELEMENT start (#PCDATA)>
   <!ELEMENT stop (#PCDATA)>
]>

<classifications_file>

  <comments></comments>

  <data_set>
    <data_set_id>
      /Users/luan/workspace/tcc/alerte-me/src/main/resources/audio_db/48000/
      108_buzina/event_101_from_file_[file4_48000_realTimeRecording_ch0.bin].wav
    </data_set_id>
    <role>training</role>
    <classification>

      <class>Buzina</class>

    </classification>
  </data_set>

  <data_set>
    <data_set_id>
      /Users/luan/workspace/tcc/alerte-me/src/main/resources/audio_db/48000/
      108_buzina/event_10_from_file_[file14_48000_realTimeRecording_ch0.bin].wav
    </data_set_id>
```

Figura 4.4: Arquivo de *instance label* das amostras.

```
<?xml version="1.0"?>
<!DOCTYPE taxonomy_file [
   <!ELEMENT taxonomy_file (comments, parent_class+)>
   <!ELEMENT comments (#PCDATA)>
   <!ELEMENT parent_class (class_name, sub_class*)>
   <!ELEMENT class_name (#PCDATA)>
   <!ELEMENT sub_class (class_name, sub_class*)>
]>

<taxonomy_file>

  <comments></comments>

  <parent_class>
    <class_name>Buzina</class_name>
  </parent_class>

  <parent_class>
    <class_name>Alarme</class_name>
  </parent_class>

</taxonomy_file>
```

Figura 4.5: Arquivo de *taxonomy* utilizado.

Quando executado o comando de *experimenting* do ACE é necessário informar o número de subconjuntos a serem utilizadas pela *cross validating*. Neste caso foi utilizado o

número total de amostras, que é 87, para garantir uma melhor taxa de acerto. A Figura 4.6 apresenta o o classificador que obteve os melhores resultados durante o *experimenting*. E a Figura 4.7 apresenta os resultados do *cross validating* executado nesse classificador.

```
------------------------------------------------------------------
--------------------------BEST RESULTS----------------------------
------------------------------------------------------------------

BEST CLASSIFIER: k-NN  (k = 1, unweighted)
BEST DIMENSIONALITY REDUCTION: Principal Components Analysis (PCA)
TIME TAKEN: 0.009383333333333334 minutes
```

Figura 4.6: Melhor resultado de classificador através do método de experimenting do ACE.

```
BEST SUCCESS RATE: ?%
BEST ERROR RATE: ?%
BEST STANDARD DEVIATION: ?%
NUMBER OF FOLDS: 87

CONFUSION MATRIX:
  a  b   <-- classified as
 30 14 |  a = Buzina
  8 35 |  b = Alarme
```

Figura 4.7: Resultados do método de experimenting do ACE.

O classificador que obteve os melhores resultados através do método de *experimenting* foi o k-NN (k = 1, unweigthed) utilizando o Principal Components Analysis (PCA) para executar a tarefa de redução de dimensionalidade. Verifica-se que as taxas de sucesso, erro e de desvio padrão não foram exibidas corretamente. Isso ocorre quando há uma falha no cálculo de pelo menos um dos resultados das *cross validanting* executadas durante o *experimenting*. Entretanto, com a matriz de confusão foi possível obter os dados através de um cálculo simples.

A matriz de confusão representa como um determinado número de amostras foi classificado durante a *cross-validating*. Os valores na diagonal primária da matriz representam as classificações que obtiveram sucesso, e os valores da diagonal secundária representam as falhas. No caso do evento buzina, por exemplo, temos 44 amostras deste mesmo evento, logo apenas 30 foram classificadas corretamente e as outras 14 foram classificadas como sendo alarme. Já no o evento de alarme, onde temos 43 amostras, 35 delas foram classificadas corretamente, e as outras 8 foram classificadas como buzina. Sendo assim, para obtermos a

taxa de acerto, basta somar os valores da diagonal primária, que é 65, e dividir pelo número total de amostras, que é 87. Logo a taxa de acerto do classificador que obteve os melhores resultados foi de aproximadamente 74,7%, e a taxa de erro ficou em torno de 25,3%.

Tendo obtido qual o classificador mais indicado para o problema em questão, ele foi então treinado através do método de *training* do ACE, utilizando o PCA para a redução de dimensionalidade. Assim como o método de *experimenting*, o training requer que sejam passados como parâmetro os arquivos de *feature definitions*, *feature values* e *instance label*. Além disso, é necessário especificar o nome do arquivo que representa o classificador treinado, bem como o código que representa o classificador, neste caso foi IBk para o k-NN (k = 1, unweigthed), e também, se houver, o tipo de redução de dimensionalidade, que foi o PCA. A Figura 4.8 apresenta o comando utilizado para treinar o classificador.

```
java -jar ACE.jar -lfkey ../../feature_definitions.xml -lfvec ../../feature_
values.xml -lmclas ../../instance.xml -train -sres ../../machine.model -lear
ner IBk -fs PCA
save_file../../machine.model

Classifier succesfully trained.
```

Figura 4.8: Treinamento do classificador utilizando o ACE pela linha de comando.

4.2 Resultados

4.2.1 Desenvolvimento do protótipo

O protótipo foi criado utilizando a linguagem de programação Java, visto que tanto o jAudio quanto o ACE foram também escritos nesta mesma linguagem. Portanto, a comunicação entre o código do protótipo e o destes dois softwares se deu de forma compatível.

Foi dado ao projeto o nome de “Alerte-me”, e sua estrutura foi definida conforme ilustrado nas Figuras 4.9 e 4.10.

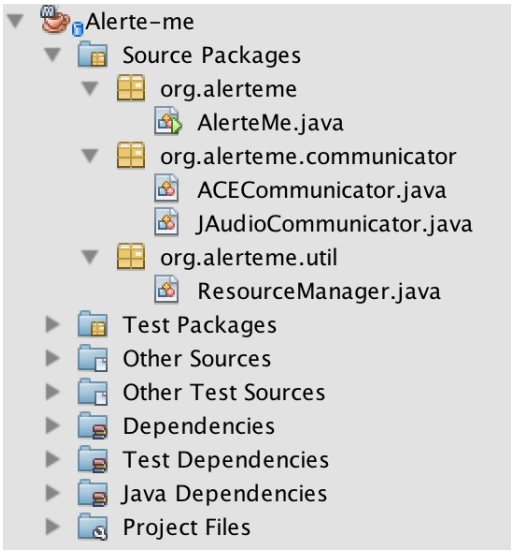

Figura 4.9: Estruturação do projeto.

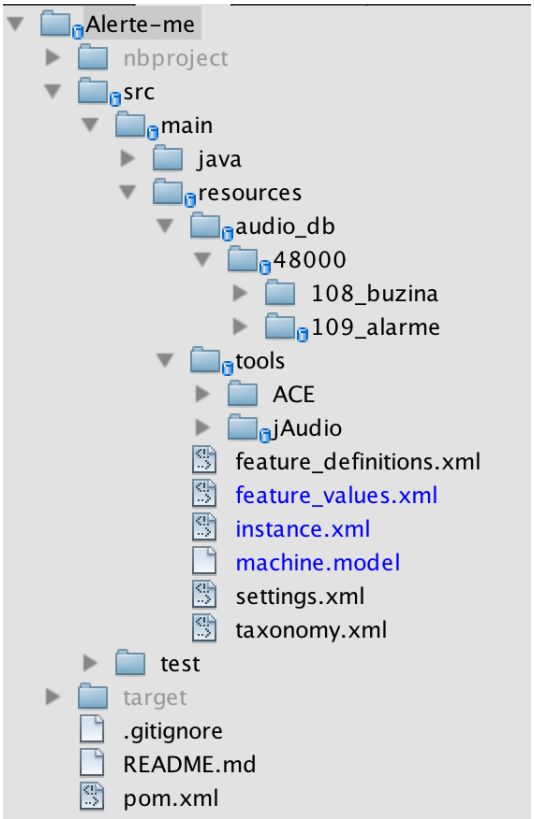

Figura 4.10: Estruturação dos recursos utilizados pelo projeto.

Como ilustrado na Figura 4.9 o projeto é composto de 4 classes, e o papel de cada uma delas é definido da seguinte maneira:

- JAudioCommunicator: É a classe responsável por se comunicar com o jAudio. Ela possui um método chamado *extractFeatures* que espera receber como parâmetro um arquivo de áudio; um arquivo de settings.xml, que representa quais características serão extraídas deste áudio; e um nome para os arquivos de *feature values* e *feature definitions* que serão gerados após o termino da execução do método.

- ACECommunicator: Esta classe, por sua vez, é responsável por se comunicar com o ACE. Ela possui um método chamado *classify*. Os parâmetros que este método espera são um arquivo de *taxonomy*, *feature definitions*, *feature values* e o arquivo do classificador treinado. Ao final da execução do método, ele irá retornar um array do tipo String contendo o resultado da classificação.

- AlerteMe: Esta é a classe principal do programa, ela possui um método main onde são criadas as instâncias das classes JAudioCommunicator e ACECommunicator e são atribuídas as configurações necessárias para que o programa consiga receber um arquivo de áudio e classifique-o de acordo com as classes definidas no arquivo de *taxonomy*.

- ResourceManager: Esta classe foi criada apenas para facilitar a maneira como os arquivos gerados apartir do JAudioCommunicator e do ACECommunicator são lidos pela classe principal.

A Figura 4.11 ilustra a relação das classes do protótipo por meio de um diagrama de classes.

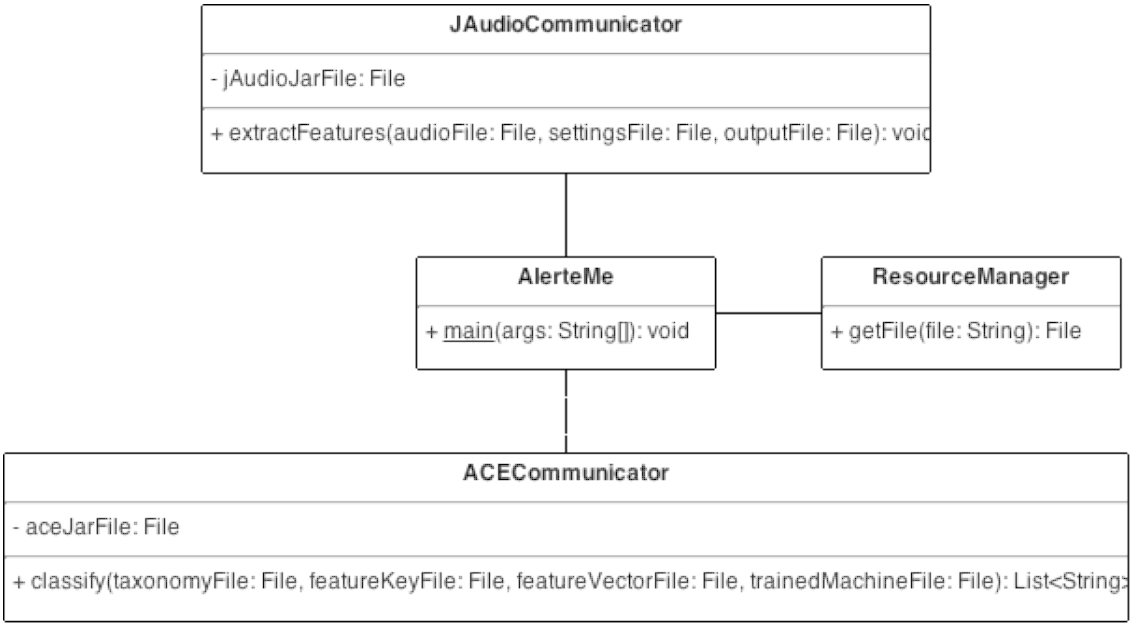

Figura 4.11: Diagrama de classes do protótipo.

Não foi criado nenhum tipo de interface gráfica para o protótipo, podendo esta funcionalidade ser desenvolvida como trabalho futuro. Portanto, o programa só pode ser executado por meio da linha de comando. Como o projeto foi criado utilizando a estrutura do Maven [5] ele deve ser executado de uma forma diferente da convencional, como mostra a Figura 4.12.

```
luan @ LAlreis-3: [~]: mvn exec:java -Dexec.mainClass="org.alerteme.AlerteMe"
-Dexec.args="exemplo_de_audio.wav"
```

Figura 4.12: Comando para execução do programa.

O fluxo do programa se inicia ao receber um arquivo de áudio, e em seguida é estabelecida a comunicação com o jAudio através do JAudioCommunicator que faz uma chamada para o jAudio realizar a extração das características deste áudio utilizando suas configurações padrões. Com isso, um arquivo contendo os valores destas características é gerado. Então, o programa inicia a comunicação com o ACE por meio do ACECommunicator, que fará uma chamada para o ACE realizar uma classificação utilizando o arquivo gerado anteriormente. Além disso, ele também faz uso dos arquivos de *feature definitions*, *taxonomy* e do classificador treinado que foram pré-definidos na etapa de treinamento do classificador. E por fim, tem-se como resposta o resultado da classificação que é exibido no console da linha de comando, conforme ilustra a Figura 4.13.

```
luan @ LAlreis-3: [~/workspace/tcc/alerte-me]: mvn exec:java -Dexec.mainClass="org.alerteme.AlerteMe"
 -Dexec.args="src/main/resources/audio_db/48000/109_alarme/event_117_from_file_[file3_48000_realTimeR
ecording_ch0.bin].wav"
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Alerte-me 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.3.2:java (default-cli) @ Alerte-me ---
[WARNING] Warning: killAfter is now deprecated. Do you need it ? Please comment on MEXEC-6.
[, , INSTANCE: /Users/luan/workspace/tcc/alerte-me/src/main/resources/audio_db/48000/109_alarme/event
_117_from_file_[file3_48000_realTimeRecording_ch0.bin].wav ,    CLASSIFICATION: Alarme]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.161 s
[INFO] Finished at: 2015-01-19T01:40:05-03:00
[INFO] Final Memory: 7M/81M
[INFO] ------------------------------------------------------------------------
```

Figura 4.13: Exemplo de classificação.

4.2.2 Experimento

Para avaliar os resultados foram utilizadas 40 amostras de áudio diferentes das que o classificador foi treinado. Estes áudios foram obtidos através dos websites GRSites [8], WavSource.com [16] e freeSFX [6] que oferecem diversas amostras de áudios para vários tipos de eventos. As Tabelas 4.1 e 4.2 ilustram os resultados obtidos após a execução do programa, utilizando cada um dos áudios.

Nome do arquivo	**Tipo de evento**	**Resultado da classificação**
emergency028.wav	Alarme	Buzina
emergency029.wav	Alarme	Alarme
emergency030.wav	Alarme	Alarme
fire_alarm_with_fade_out.mp3	Alarme	Buzina
fire_alarm.mp3	Alarme	Buzina
german_fire_siren_calls _fire_department_rain _in_background_.mp3	Alarme	Buzina
sci_fi_alarm_loop_1.mp3	Alarme	Buzina
alarm_sample_1.wav	Alarme	Alarme
alarm_sample_3.wav	Alarme	Alarme
alarm_sample_4.wav	Alarme	Alarme
alarm_sample_5.wav	Alarme	Alarme
alarm_sample_8.wav	Alarme	Alarme
alarm_sample_10.wav	Alarme	Alarme
alarm_sample_11.wav	Alarme	Alarme
alarm_sample_12.wav	Alarme	Alarme
alarm_sample_16.wav	Alarme	Alarme
alarm_sample_17.wav	Alarme	Alarme
alarm_sample_21.wav	Alarme	Alarme
alarm_sample_22.wav	Alarme	Alarme
alarm_sample_24	Alarme	Alarme
vehicle041.wav	Buzina	Buzina
vehicle042.wav	Buzina	Buzina

Tabela 4.1: Resultado da classificação através do protótipo.

Nome do arquivo	**Tipo de evento**	**Resultado da classificação**
vehicle043.wav	Buzina	Buzina
vehicle046.wav	Buzina	Buzina
vehicle047.wav	Buzina	Buzina
car_horn01.wav	Buzina	Buzina
car_horn02.wav	Buzina	Buzina
car_horn03.wav	Buzina	Buzina
car_horn04.wav	Buzina	Buzina
car_horn05.wav	Buzina	Buzina
car_horn06.wav	Buzina	Buzina
car_horn07.wav	Buzina	Buzina
auto_horn_1972 _toyota_single _short_blast.mp3	Buzina	Buzina
auto_horn_1972 _vw_three _short_blast.mp3	Buzina	Buzina
auto_horn_1972 _vw_three _short_blasts.mp3	Buzina	Buzina
auto_horn_1972 _toyota_single _long_blast.mp3	Buzina	Buzina
auto_horn_1972 _toyota_three _short_blasts.mp3	Buzina	Buzina
auto_horn_1972 _vw_short _blast.mp3	Buzina	Buzina
car_horn_x.wav	Buzina	Buzina
honk_honk_x.wav	Buzina	Alarme

Tabela 4.2: Resultado da classificação através do protótipo.

Os áudios utilizados neste experimentos eram de desconhecimento total do classificador, logo observou-se que os resultados foram bastante satisfatórios. Apesar de que em um

dos casos o áudio não foi classificado corretamente, porém isto é esperado dado que existe uma taxa de erro de 25,3% do classificador.

Capítulo 5

Conclusões

A classificação de sinais de áudio é uma área que está em constante aprimoramento. Apesar dos testes realizados sobre o protótipo terem sido limitados, foi possível perceber que, apesar do classificador ter conseguido acertar qual era a classe das amostras na maioria dos casos, a taxa de acerto ainda está distante de chegar a um valor próximo de 100%. Ainda assim, considerando que existem certos tipos de eventos sonoros que até mesmo o ouvido humano também pode se enganar, pode-se concluir que é interessante tentar utilizar a tecnologia de classificação de sinais de áudio no dia a dia das pessoas que possuam deficiência auditiva, e avaliar se o resultado correto das classificações se sobressai aos resultados incorretos. Já que na vida real existe uma série de fatores, como ruídos, que podem interferir na qualidade do som captado.

5.1 Trabalhos futuros

Com todo o conhecimento adquirido por meio deste trabalho, a ideia é dar continuidade ao desenvolvimento do protótipo a fim de torná-lo uma aplicação concreta. Dado que já se tem uma base de áudios definida, pode-se utilizar mais tipos de eventos e criar uma estrutura taxonômica mais elaborada.

Um outro fator muito importante e necessário para concretizar a ideia de criar uma aplicação, consiste na captação do áudio em tempo real. Visto que as pessoas que possuem deficiência auditiva não capazes de saber quando determinados tipos de evento ocorre.

Referências Bibliográficas

[1] Daccord. A tecnologia, 2013. Disponível em: http://www.audioalerta.com.br/. Acessado em: 12/12/2014.

[2] Instituto Brasileiro de Geografia e Estatística IBGE. Censo demográfico 2010 - resultados preliminares da amostra, 2010. Disponível em: http://www.ibge.gov.br/home/estatistica/populacao/censo2010/. Acessado em: 10/12/2014.

[3] Assessoria de imprensa da Telex Soluções Auditivas. Barulho, inimigo da boa audição. Jornal Dia Dia - Ciência e Tecnologia, 2014. Disponível em: http://www.jornaldiadia.com.br/news/noticia.php?Id=49258#.VKG078AAA. Acessado em: 11/12/2014.

[4] Marie Camilleri e Nuno Trigueiros-Cunha. Ruido: Atenção perigo ! protecção, 2014. Disponível em: http://www.cochlea.org/po/ruido. Acessado em: 11/12/2014.

[5] The Apache Software Foundation. Introduction to the standard directory layout, 2015. Disponível em: http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html. Acessado em: 21/12/2014.

[6] freeSFX. Download free sound effects, 2014. Disponível em: http://www.freesfx.co.uk/. Acessado em: 19/12/2014.

[7] Eibe Frank Gordon Paynter, Len Trigg. Attribute-relation file format (arff), 2008. Disponível em: http://www.cs.waikato.ac.nz/~ml/weka/arff.html. Acessado em: 13/11/2014.

[8] GRSites. Free sound effects archive, 2014. Disponível em: http://www.grsites.com/archive/sounds/. Acessado em: 28/12/2014.

[9] Amauri Lopes Jayme Garcia Arnal Barbedo. Estado da arte da classificação de sinais de Áudio. In *Anais do II Congresso Brasileiro de Engenharia de Áudio da AES-Brasil, apresentado na VIII Convenção Nacional da AES-Brasil*, pages 21–26, SÃO PAULO, SP, BRASIL, Jul 2004. Engenharia de Áudio da AES Brasil.

[10] C. McKay I. Fujinaga McEnnis, D. and P. Depalle. jaudio: A feature extraction library. In *Proceedings of the International Conference on Music Information Retrieval.*, pages 600–603, 2005.

[11] C. McKay. *Automatic music classification with jMIR.* PhD thesis, Department of Music Research, McGill University, 845 Rue Sherbrooke Ouest, Montraal, QC H3A 0G4, Canada, Jan 2010. http://jmir.sourceforge.net/publications/PhD_Dissertation_2010.pdf.

[12] J. A. Burgoyne J. Thompson McKay, C. and I. Fujinaga. Using ace xml 2.0 to store and share feature, instance and class data for musical classification. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 303–308, Japan, Oct 2009. Kobe.

[13] P. Russell, S.; Norvig. *Artificial Intelligence: A Modern Approach. 3rd Edition.* Prentice Hall, New Jersey, 2009.

[14] C. McKay J. A. Burgoyne Thompson, J. and I. Fujinaga. Additions and improvements to the ace 2.0 music classifier. In *Proceedings of the International Society for Music Information Retrieval Conference.*, pages 435–440, Japan, Oct 2009. Kobe.

[15] C. M. Sperberg-McQueen Eve Maler François Yergeau John Cowan Tim Bray, Jean Paoli. Extensible markup language (xml) 1.1 (second edition). In *W3C Recommendation.* Aug 2006.

[16] WavSource.com. The source for free sound files and reviews, 2014. Disponível em: `http://www.wavsource.com/`. Acessado em: 19/01/2015.