UNIVERSIDADE FEDERAL DO ABC

EDUARDO PETECOF MATTOSO

Desenvolvimento de uma ferramenta computacional para o reconhecimento automático de locutores baseado nos coeficientes mel cepstrais e nas misturas de Gaussianas

Santo André

2019

EDUARDO PETECOF MATTOSO

Desenvolvimento de uma ferramenta computacional para o reconhecimento automático de locutores baseado nos coeficientes mel cepstrais e nas misturas de Gaussianas

Trabalho apresentado como requisito parcial para a Conclusão do Curso de Engenharia de Informação na Universidade Federal do ABC.

Orientador: Prof. Dr. Kenji Nose Filho

Santo André

2019

AGRADECIMENTOS

Agradeço à Deus, à minha família e a todos os colegas, professores, técnicos e colaboradores que encontrei durante minha graduação na Universidade Federal do ABC.

"Quem quer passar além do Bojador
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu,
Mas nele é que espelhou o céu."

Fernando Pessoa.

RESUMO

Este trabalho de graduação apresenta como foi construído um sistema de reconhecimento automático de locutores baseado nos coeficientes mel cepstrais (MFCCs, do inglês Mel Frequency Cepstral Coefficients) e nas misturas de Gaussianas (GMMs, do inglês Gaussian Mixture Models). A implementação computacional utiliza apenas ferramentas e bibliotecas de código aberto da linguagem Python, de modo que o sistema possa ser um ponto de partida acessível para o estudo do ramo. Através de um ambiente de testes automatizados, desenvolvido durante o trabalho em conjunto com o sistema de reconhecimento, foi possível observar as taxas de reconhecimento obtidas em diferentes condições de treinamento e parametrização. Na melhor configuração, utilizando 64 componentes Gaussianas, matrizes diagonais de covariância e 20 MFCCs, o sistema alcançou taxas de reconhecimento próximas de 90%, diferenciando sentenças de diálogos entre 2 interlocutores, e de aproximadamente 70%, diferenciando os mesmos diálogos mas com um interlocutor externo a mais no treinamento.

Palavras-chave: Reconhecimento automático de interlocutores, Python, código aberto, MFCCs, GMMs, Transcrição rica

ABSTRACT

This bachelor's dissertation will present how an automatic speaker recognition (ASR) system was built using Mel Frequency Cepstrum Coefficients and Gaussian Mixture Models. The computational implementation utilizes only open-source tools and libraries for Python, in order to make it an accessible starting point for those interested in studying this subject. With an automated testing environment, developed in conjunction with the recognition system, it was possible to analyze the recognition rates obtained in different training conditions and parametrization. In the best configuration, using 64 Gaussian components, diagonal covariance matrices and 20 MFCCs, the ASR system obtained recognition rates of approximately 90%, differentiating sentences in dialogs between 2 speakers, and approximately 70%, differentiating the same dialogs but with an additional speaker in the training phase.

Keywords: Automatic speaker recognition (ASR), Python, open-source, MFCCs, GMMs, Rich Transcription

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de reconhecimento automático de interlocutor, ou ASR (do inglês *Automatic Speaker Recognition*) são implementações computacionais capazes de reconhecer a voz de um interlocutor e diferenciá-la da voz de outros interlocutores. As características na voz de cada ser humano são diferentes por conta das distinções na forma do trato vocal, tamanho da laringe e outras partes dos seus órgãos que produzem a voz (KENNUNEN & LI, 2010; LU & DANG, 2008). Captar e diferenciar essas características através de algoritmos de aprendizado de máquina tem sido um tema de estudo para aplicações de controle de acesso biométrico, transcrição rica, entre outros (REYNOLDS, 2002; KENNUNEN & LI, 2010; ANGUERA, 2010).

Reconhecer os interlocutores presentes em gravações de áudio é bastante importante para estudar o conteúdo de grandes bases de dados não estruturados, usando, por exemplo, métodos estatísticos (REYNOLDS, 2002; ANGUERA, 2010). Além disso, com o crescimento das interações humano-máquina, inclusive através de voz, saber qual interlocutor transmitiu uma determinada mensagem e, eventualmente, de qual forma (humor, ironia), pode ser importante para que um robô tome melhores decisões. Em ambos os casos, os sistemas de ASR geralmente são combinados com sistemas de transcrição para realizar um processo de transcrição rica.

A transcrição rica é um tipo de transformação da voz para texto que além de informar as palavras ditas pelo interlocutor, pode também mostrar, no mínimo, quem falou e quando. O interesse no aprimoramento de sistemas que realizam transcrição rica automaticamente fez o Instituto Nacional de Padrões e Tecnologias dos Estados Unidos (NIST) criar uma iniciativa para padronizar os métodos de avaliação desse tipo de sistema. O objetivo, segundo o próprio instituto, é incentivar a criação de tecnologias que irão produzir transcrições melhores de ler para os seres humanos e mais aproveitáveis para os computadores (NIST, 2016).

Nesse sentido, esse trabalho busca construir um sistema de reconhecimento de interlocutores que poderia ser usado em combinação com sistemas de transcrição simples, para obter uma transcrição rica. Utilizar-se-á dos métodos considerados de referência no ramo, como a extração de MFCCs (*Mel Frequency cepstrum coefficients*) e a modelagem de interlocutores por misturas de Gaussianas (KENNUNEN & LI, 2010; LU & DANG, 2008; VENTURINI, ZÃO & COELHO, 2014). De modo que esse trabalho e o sistema construído possam servir como uma porta de entrada para outros estudantes interessados no assunto.

Além disso, o sistema será avaliado usando um ambiente capaz de realizar testes automatizados, para que novas técnicas de pré-processamento, extração de características, modelagem de interlocutores, entre outros artifícios, possam ser comparadas com esse sistema clássico de reconhecimento. O ambiente conta com um banco de dados próprio de áudios anotados e foi todo construído usando ferramentas e bibliotecas open-source na linguagem Python, para que qualquer pessoa consiga reproduzi-lo.

Dessa forma, esse trabalho começa descrevendo os conceitos que serão necessários para compreensão das seções posteriores, as quais mostram a metodologia utilizada e os resultados obtidos. Estes conceitos envolvem desde uma revisão sobre transcrição rica e definições gerais de sistemas de reconhecimento de interlocutores, até o aprofundamento dos MFCCs e das misturas de Gaussianas. No fim os resultados revelaram não só, que o sistema funciona mas também que, utilizando os melhores parâmetros de projeto estudados, as taxas de acerto são consideravelmente maiores do que a *baseline*.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de reconhecimento automático de interlocutor ou sistemas de ASR (*Automatic Speaker Recognition*)

Durante o processo de comunicação vocal entre dois ou mais interlocutores, as mensagens codificadas através das palavras podem ser consideradas as partes mais importantes do conjunto de informações trocadas entre eles. Entretanto, além dessa fração principal, saber qual interlocutor transmitiu uma determinada mensagem e, eventualmente, de qual forma (humor, ironia), pode ser fundamental para que essa mensagem seja melhor aproveitada.

Dessa forma, ensinar o computador não só a transcrever o texto, mas também a extrair essas outras informações automaticamente pode ser muito importante para a organização e análise estatística de grande bases de dados contendo áudios. Além de permitir que a interação com computadores usando voz, que na grande maioria das vezes exige que o áudio seja transformado para texto, torne-se mais efetiva. Um avanço cada vez mais necessário já que a construção de interfaces de conversão, em assistentes virtuais por exemplo, está sendo cada vez mais comum.

Esse processo de transcrição que além de extrair o que foi dito em um trecho de áudio, procura também incluir outras informações possíveis, como determinar quem falou e quando, é chamado de transcrição rica ou completa. A iniciativa do NIST de padronizar os métodos de avaliação dos sistemas de transcrição rica inclui bases de dados anotadas para que pesquisadores possam avaliar e comparar seus sistemas com o restante da academia. As bases possuem em seu conteúdo principalmente gravações de jornais televisivos e reuniões de conferência. Elas são anotadas com o que foi dito, quem falou, o sexo do interlocutor, o idioma utilizado, entre outros.

Por esse motivo o reconhecimento de interlocutores é uma das tarefas essenciais do processo de transcrição rica, mas outras aplicações também são possíveis. Um outro uso comum para sistemas de ASR é o controle de acesso biométrico por voz, por exemplo. Para resolver cada uma dessas aplicações distintas existem tipos diferentes de sistemas de ASR, suas distinções e também os princípios básicos serão discutidos nesta seção.

De forma geral, os sistemas de reconhecimento automático de interlocutor são projetados para resolver duas tarefas distintas: Identificação de interlocutores e verificação de interlocutores (REYNOLDS; ROSE, 1995; REYNOLDS, 2002; DE LARA, 2005). Na tarefa de identificação o objetivo é determinar a qual interlocutor pertence uma determinada voz, dentre um conjunto pré-determinado de interlocutores possíveis. Já na tarefa de verificação, o objetivo é determinar se a voz de um interlocutor realmente se assemelha com a voz do interlocutor que ele se diz ser.

No primeiro caso, o computador devolve o nome (ou identificação qualquer) do interlocutor mais provável, no segundo caso o computador devolve se reconhece, ou não, aquele interlocutor. Aplicações para controle de acesso biométrico, citadas anteriormente, estão relacionadas à tarefa de verificação de interlocutor. No processo de transcrição rica, em contra partida, a tarefa é de identificação.

Os sistemas de ASR podem ser ainda dependentes ou independentes do texto que é falado pelo interlocutor (REYNOLDS, 2002; DE LARA, 2005). Quando o reconhecimento não depende da mensagem, avalia-se características da voz que aparecem em qualquer palavra, características que são um reflexo das diferenças no trato vocal dos seres humanos. Quando o reconhecimento depende da mensagem, avalia-se como o interlocutor pronuncia uma palavra específica, que pode ser uma senha por exemplo. Nesse último caso, as características da voz também influenciam, mas de maneira limitada para aquela palavra.

Entretanto, independente do tipo de sistema de reconhecimento, alguns fundamentos são comuns durante o processo de funcionamento. De modo geral, existe uma primeira etapa chamada de extração de características e uma segunda de modelagem dos interlocutores. Na primeira, o sinal sonoro será transformado em um conjunto de características que enfatizam as

diferenças entre os interlocutores e diminuem a importância de efeitos redundantes que não dependem do interlocutor. Na segunda, esse conjunto de características é comparado com modelos existentes para cada interlocutor, com o objetivo de encontrar qual é o modelo mais similar (KENNUNEN & LI, 2010). Nas seções seguintes será feito um detalhamento de cada uma dessas etapas.

2.2 Extração de características

O sinal de áudio não contém apenas características que são importantes para determinar o interlocutor. Portanto, extrair as características que são relevantes é uma necessidade antes de criar um modelo com elas. Por exemplo, ao ouvir uma gravação de uma pessoa conhecida falando uma outra língua, nós seres humanos conseguimos saber que ela está usando um idioma diferente por conta de uma característica do áudio que são os sons das palavras, mas ainda assim sabemos qual pessoa está falando porque essa classificação decorre de outras características.

No artigo de revisão de Kennunen e Li (2010) são descritos inúmeros tipos de características que podem ser extraídas de um sinal de áudio para identificar interlocutores, mas como informado no próprio artigo, os tipos mais comuns estão dentro de um grupo conhecido como características espectrais de curto prazo. Essas são as características relacionadas com o timbre natural de cada ser humano, que decorrem das distinções na forma do trato vocal, tamanho da laringe e outras partes dos seus órgãos que produzem a voz.

O nome do grupo está justificado no fato de que essas características são obtidas a partir do espectro de frequências de pequenas segmentações do sinal de voz chamadas de *frames*. Cada *frame* contém geralmente de 20 - 30 ms do sinal, pois acredita-se que durante esse intervalo as características do sinal de voz permanecem relativamente constantes. Em cada segmentação do sinal, foi percebido que a magnitude do espectro de frequências, também conhecido como envelope espectral, carrega informações sobre o timbre de voz (KENNUNEN & LI, 2010; REYNOLDS, 2002).

Os vários tipos de características possíveis dentro do grupo das espectrais de curto prazo irão se diferenciar pela técnica utilizada para extrair uma versão simplificada do envelope espectral capaz de diferenciar interlocutores. A técnica mais utilizada e que vem mostrando os melhores resultados (REYNOLDS & ROSE, 1995; REYNOLDS, 2002; LU & DANG, 2008; KENNUNEN & LI, 2010) é a extração dos coeficientes conhecidos como *Mel Frequency Cepstrum Coefficients* ou MFCCs, os quais serão descritos a seguir.

2.2.1 MFCCs (*Mel Frequency cepstrum coefficients*)

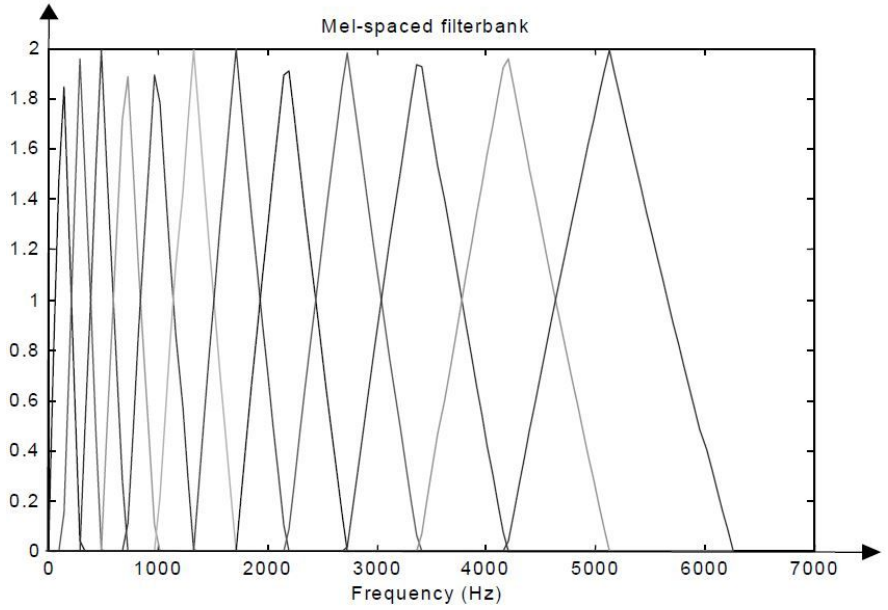
Os MFCCs são obtidos por um processo que começa com a separação do áudio em frames, seguida pela obtenção dos periodogramas de cada frame através da FFT (*Fast Fourier Transform*), por motivos descritos na seção anterior. Tendo em vista que, antes de aplicar a transformada, é preciso fazer um janelamento no frame, para que as descontinuidades presentes no início e no fim do frame não causem deformações no espectro.

O próximo passo é a simplificação do envelope espectral descrito no periodograma, o qual será multiplicado por um banco de filtros. Cada filtro calcula a quantidade de energia presente em uma determinada banda (faixa de frequências) do envelope. A definição de cada banda leva em conta estudos psicoacústicos do ouvido humano, o qual é capaz de diferenciar melhor as baixas frequências do que as altas frequências.

Dessa forma, um número maior de filtros é utilizado nas baixas frequências para aumentar a importância dessa região e representá-la com mais detalhe, já nas altas frequências o número de filtros é menor. A escala mel é uma escala psicoacústica que foi pensada para representar numericamente esse comportamento, na qual as baixas frequências (até 1kHz) seguem uma escala linear e as altas frequências (maiores que 1kHz) uma escala logarítmica. Portanto, o espaçamento dos filtros é definido nessa escala (DE LARA, 2005).

Geralmente o formato dos filtros é triangular, mas isso não é uma regra. Cada valor de energia obtido pela multiplicação de um filtro com o envelope é considerado um coeficiente. Na figura a seguir observa-se o exemplo de um banco de 12 filtros obtido para calcular MFCCs.

Figura 1: Resposta em frequência de um Mel Frequency Filter com 12 filtros



Fonte: (DE LARA, 2005)

Depois de calcular os coeficientes, eles ainda passam por um processo de compressão aplicando um logaritmo de base 10 e um processo de ortogonalização usando a transformada do cosseno (DCT). Dessa forma o processo inteiro é constituído pelos seguintes passos:

- ➔ Divisão do áudio em frames
- ➔ Janelamento do frame
- ➔ Cálculo do espectro de frequência e periodograma através da FFT
- ➔ Filtragem na escala mel
- ➔ Compressão e ortogonalização

O último passo pode ser descrito pela fórmula (1), na qual os coeficientes c_n são obtidos a partir das energias resultantes da multiplicação com os M filtros $Y(m)$, considerando $m = 1, ..., M$ (KENNUNEN & LI, 2010):

$$c_n = \sum_{m=1}^{M} [logY(m)] cos\left[\frac{\pi n}{M}\left(m - \frac{1}{2}\right)\right] \quad (1)$$

Depois de extrair as características dos áudios de treinamento, o conjunto dos vetores de características extraídos precisam ser utilizados para criar um modelo que represente cada interlocutor. Esse processo será discutido na seção seguinte.

2.3 Modelagem de interlocutores

Cada vetor de características individualmente não consegue definir as especificidades da voz de um interlocutor, ou seja, não basta extrair um único vetor de MFCCs do áudio de treinamento e do áudio de teste e compará-los. Até porque, mesmo com todos os artifícios utilizados para que os MFCCs carreguem o máximo de informações da voz, eles ainda são afetados por outros fatores redundantes.

Dessa forma, o que se faz é extrair padrões estatísticos de um grande conjunto de vetores de características, e assim existe uma maior chance de que esses padrões estatísticos estejam relacionados com os padrões de voz dos interlocutores. Para fazer a classificação é preciso verificar se os padrões estatísticos de um áudio de teste se assemelham com os padrões dos áudios de treinamento. Existem várias formas de calcular ou perceber esses padrões estatísticos, o método mais comum e utilizado como referência é modelar as características como misturas de Gaussianas (Gaussian Mixture Models) (REYNOLDS & ROSE, 1995; REYNOLDS, 2002; LU & DANG, 2008; KENNUNEN & LI, 2010).

2.3.1 Gaussian Mixtures Models (GMMs)

Através das GMMs torna-se possível representar o conjunto de vetores de características dos áudios de treinamento com uma soma ponderada de k componentes Gaussianas, assumindo que cada uma das componentes representam potenciais padrões de voz. Com essa função será

possível calcular a probabilidade de um vetor qualquer de teste x fazer parte dessa representação (KENNUNEN & LI, 2010):

$$p(x|\lambda) = \sum_{k=1}^{K} P_k \, N(x|\mu_k, \Sigma_k) \tag{2}$$

Na equação (2) P_k são os pesos de cada componente e respeitam a restrição de que a somatória dos pesos é igual a 1. As k componentes Gaussianas N são um padrão estatístico calculado a partir de um vetor μ de médias e uma matriz Σ de covariâncias, com base nas D dimensões dos vetores de características (D coeficientes MFCCs por exemplo) (KENNUNEN & LI, 2010):

$$N(x|\mu_k, \Sigma_k) = (2\pi)^{\frac{-d}{2}} |\Sigma_k|^{\frac{-1}{2}} exp\left\{-\tfrac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right\} \tag{3}$$

Dessa forma, a mistura Gaussiana completa pode ser resumida em um modelo λ com base nos pesos P_k, no vetor de médias μ_k, e na matriz de covariâncias Σ_k. Cada interlocutor terá o seu próprio modelo λ (REYNOLDS & ROSE, 1995):

$$\lambda = \{ P_k, \ \mu_k, \ \Sigma_k \}, \ k = 1, \ldots, M \tag{4}$$

A matriz de covariâncias pode ainda ter várias tipos, modificando a forma das GMMs (REYNOLDS, 1995). Nas fórmulas descritas anteriormente, cada componente possui sua própria matriz de covariâncias, envolvendo as D dimensões de características. Uma outra opção seria ter uma única matriz de covariâncias para todas as componentes, por exemplo. Além disso, a matriz pode ser completa nas D dimensões ou apenas uma matriz diagonal.

O mais comum é utilizar uma matriz diagonal diferente para cada componente Gaussiana, porque as matrizes completas exigem mais tempo e mais dados para serem obtidas (KENNUNEN & LI, 2010). Além disso, um conjunto grande de matrizes diagonais vai conseguir modelar as correlações existentes entre cada dimensão do vetor de características,

mesmo que as D dimensões de características não sejam estatisticamente independentes (REYNOLDS & ROSE, 1995).

A efetividade das matrizes diagonais já foi comprovada empiricamente com bons resultados em sistemas de reconhecimento (REYNOLDS & ROSE, 1995). Alguns artigos mostram a opção de usar uma única ou poucas componentes de matriz cheia, como um método mais eficiente computacionalmente. Porém, as taxas de reconhecimento diminuem em relação a sistemas que usam várias componentes de matrizes diagonais (KENNUNEN & LI, 2010).

Depois de escolher sua forma, os parâmetros da função densidade de probabilidade podem ser calculados usando vários métodos, o mais famoso deles é estimando a função que maximiza a verossimilhança (*Maximum Likelihood* - ML), ou o logaritmo da verossimilhança, para os dados de treinamento (REYNOLDS; ROSE, 1995; KENNUNEN & LI, 2010). Essa métrica pode ser calculada da seguinte forma (KENNUNEN & LI, 2010):

$$LL_{avg} = (X, \lambda) = \frac{1}{T} \sum_{t=1}^{T} log \sum_{k=1}^{K} P_k N(x|\mu_k, \Sigma_k) \quad (5)$$

Quanto maior esse valor, maior a probabilidade de que um conjunto de vetores de características tenha se originado de um modelo λ (KENNUNEN & LI, 2010). Como essa maximização não pode ser feita diretamente, são utilizados algoritmos iterativos que aumentam a verossimilhança a cada passo para os vetores de treinamento. O algoritmo mais popular é o de Maximização de Expectativas (*Expectation Maximization* - EM)

De maneira simplificada o EM funciona da seguinte forma: As misturas são inicialmente definidas de forma randômica e aos vetores de treinamento é atribuída uma probabilidade de pertencer a cada componente da mistura (verossimilhança). Cada vetor vai ter uma probabilidade maior de pertencer a uma determinada componente, dessa forma estima-se uma nova mistura considerando essa estimativa de pertencimento de cada vetor de treinamento. Repete-se o cálculo das probabilidades e a estimativa de novas misturas até um critério de parada. (DO & BATZOGLOU, 2008)

Ao obter cada modelo λ de todos os interlocutores, é possível determinar a qual modelo pertence um áudio de teste. Basta calcular a média do logaritmo da verossimilhança em relação a todos os modelos possíveis. O modelo que obtiver o maior valor será o modelo escolhido para representá-lo, classificando aquele áudio como pertencente ao interlocutor que gerou aquele modelo.

2.4 *Voice activity Detection* (VAD)

Antes de demonstrar, na seção de metodologia, como foram implementados os processos de extração de características e de modelagem de interlocutores, se faz necessário apresentar alguns conceitos sobre etapas de pré-processamento. Em sistemas que fazem processamento de voz é muito comum que exista uma etapa para detecção de atividade vocal, antes de dar início ao processo de extração de características. (RAMÍREZ et al., 2005; KENNUNEN & LI, 2010; VENTURINI, ZÃO & COELHO, 2014).

Essa etapa é essencial pois em uma gravação de discurso comum, vão existir momentos em que o interlocutor faz uma pausa, mesmo que só por alguns alguns instantes. Durante o processo de divisão da gravação em *frames*, o qual foi descrito no início da seção 2.3, alguns desses segmentos curtos do sinal vão ser gerados a partir de trechos de pausa. A inclusão desse tipo de frame sem sinal vocálico nas análises não faz sentido, pois eles não carregam informações que vão diferenciar os interlocutores e, portanto, diminuem a eficácia do reconhecimento. (RAMIREZ et al., 2005; KENNUNEN & LI, 2010).

Deste modo, técnicas de detecção de atividade vocal podem ser utilizadas para detectar *frames* que não carregam voz em seu conteúdo e descartá-los. A estratégia mais comum deste tipo de sistema é calcular a energia média dos *frames* e definir um limiar de energia para que cada segmento individual possa ser considerado parte de um sinal de voz (KENNUNEN & LI, 2010). Entretanto, nos casos em que um ruído de fundo possui uma magnitude expressiva, estratégias mais sofisticadas precisam ser utilizadas, incluindo outros tipos de características além da energia (RAMIREZ et al., 2005; KENNUNEN & LI, 2010).

Além disso, para melhor fazer a classificação os sistemas de VAD mais complexos fazem modelagem estatística, inclusive através de misturas de Gaussianas como foi descrito na seção anterior. Nesses casos existe também uma etapa de treinamento, utilizando frames previamente marcados como silêncio ou não silêncio.

Na seção de metodologia a seguir, existirá uma etapa em que os trechos de silêncio serão descartados das gravações, utilizando ferramentas que realizam a detecção de atividade vocal. Tanto as ferramentas, quanto a natureza das gravações utilizadas neste trabalho serão explicadas. Depois desse processo, dar-se-á início à etapa de extração de MFCCs e à modelagem dos interlocutores usando GMMs.

3 METODOLOGIA

Nesta seção serão apresentados todos os passos necessários para construção do sistema de reconhecimento de interlocutores utilizando os MFCCs como método de extração de caraterísticas e GMMs para modelar e identificar cada interlocutor nos áudios de teste.

3.1 Definição e gravação dos áudios

Para construir um sistema de reconhecimento de interlocutores, um conjunto de gravações é imprescindível. Neste trabalho optou-se por construir um banco de dados próprio com áudios de teste e treinamento, apesar da existência de outras bases para construção e avaliação de sistemas que realizam Rich Transcription - e por consequência ASR - como foi descrito na fundamentação teórica.

A principal justificativa para essa escolha foi diminuir a complexidade dos testes aos quais esse sistema inicial seria submetido. Pois, em algumas das bases de testes disponíveis publicamente, como a do NIST, as gravações podem ter altos níveis de ruído ao fundo, podem estar intercaladas com trechos de música, além de outros fatores de complexidade, para que seja possível realmente avaliar a robustez dos sistemas.

Entretanto, neste trabalho o objetivo é criar um sistema que funcione em condições consideradas ideais primeiro. Depois que isso for comprovado, o sistema pode ser submetido a novos testes com as bases públicas para avaliar a sua robustez nas mais diversas situações. Ademais, é sempre importante construir uma base própria, mesmo que pequena, pois as bases públicas podem ter suas permissões de uso modificadas ou seu conteúdo perdido.

Dessa forma, nessa primeira etapa serão descritos os processos de gravação que deram origem a esse banco de dados próprio, para que haja compreensão da natureza dos áudios de

treinamento e teste que serão utilizados. No total foram obtidas 6 gravações, organizadas da seguinte forma:

Primeiro, 3 interlocutores (I1, I2 e I3) gravaram um áudio individual, os quais serão utilizados na etapa de treinamento para gerar o modelo de misturas de Gaussianas de cada interlocutor. As 3 gravações foram feitas com base na leitura da crônica *Cérebro eletrônico: o que sei é tão pouco* escrita por Clarice Lispector no Jornal do Brasil (conforme Anexo 1).

Depois, os 3 interlocutores se juntaram em duplas (I1, I2), (I2, I3) e (I1, I3) e cada dupla gravou um diálogo, os quais serão utilizados na etapa de teste do sistema. Os 3 diálogos foram feitos com base no diálogo presente na crônica *O Lixo* de Luís Fernando Veríssimo (conforme Anexo 2). Nessa crônica os interlocutores sempre falam de maneira intercalada, quase sem nenhuma sobreposição.

Destes 3 interlocutores, duas das vozes são masculinas (I2, I3) e uma das vozes é feminina (I1). Nenhum deles será identificado nesse projeto e todos forneceram os dados voluntariamente. Apenas certas características de voz serão descritas para que alguns dos resultados obtidos possam ser melhor compreendidos.

As gravações foram feitas utilizando o microfone de um aparelho celular em um ambiente de baixo ruído. Inicialmente os áudios estavam codificados na extensão .M4A a uma taxa de 44100 Hz e 128 kbps. Mas foram convertidos utilizando o reprodutor de mídia VLC[1] para a extensão .WAV a uma taxa de 8000 Hz (também em 128 kbps) para contornar limitações dos métodos utilizados, que serão demonstradas mais a frente.

3.2 Ferramentas de processamento computacional

Com as gravações salvas na extensão .WAV a 8000 Hz e 128kbps, já é possível dar início aos processos automatizados de treinamento e teste que serão descritos na seção seguinte, mas

[1] Disponível para download em: https://www.videolan.org/vlc/index.pt-BR.html

antes se faz necessário descrever em quais ferramentas computacionais esses processos estão baseados.

As implementações computacionais desse projeto foram feitas na linguagem Python, utilizando Jupyter Notebooks (PÉREZ & GRANGER, 2007) como um plataforma de desenvolvimento. A escolha da linguagem, do ambiente e de outros componentes (bibliotecas) foi embasada em dois fatores principais: a possibilidade de replicação, tendo em vista que tudo está disponível de forma aberta (seguindo as licenças de utilização), e a popularidade dentro da comunidade de pesquisadores e desenvolvedores. As principais bibliotecas serão referenciadas durante o texto, mas algumas outras utilizadas para tarefas não tão relevantes do processo de implantação estão descritas no Apêndice A.

A plataforma Jupyter Notebooks é uma forma de executar o código Python em células, cada célula pode ter sua saída analisada antes de continuar com a execução da próxima, ideal para fazer análises detalhadas dos códigos e acompanhar a progressão dos dados de entrada a cada passo de transformação. Além disso, na linguagem Python a implementação do algoritmo poderia ser hospedada em um servidor, para que o sistema como um todo possa ser utilizado de maneira distribuída na rede. Um serviço como esse poderia ter aplicações comerciais.

3.3 Eliminação de silêncio e segmentação de sentenças.

A primeira etapa dentro do processo de automatização é um pré-processamento para eliminação de silêncio e segmentação de sentenças, que é realizado tanto para os áudios de teste quanto de treinamento. O objetivo é obter dentro do áudio original quais são os trechos em que realmente existe atividade vocal, eliminando trechos de silêncio.

No caso dos áudios individuais, os períodos de silêncio são causados por pausas durante o discurso. No caso dos diálogos, não só existem pausas durante o discurso, mas também pausas para que possa haver a troca de interlocutores. Dessa forma, ao encontrar um período

de silêncio, conseguimos separar as sentenças que estão antes ou depois dessa pausa, gerando a segmentação.

A identificação dos períodos de silêncio foi feita utilizando uma biblioteca que realiza VAD do projeto WebRTC[2], um *framework* aberto e gratuito, apoiado por grandes empresas como Google, Mozilla e Opera, para permitir comunicação em tempo real nos navegadores web, incluindo uma série de componentes como o VAD. Um sistema próprio para esse fim não foi construído durante o trabalho pois essa não é uma tarefa trivial e desviaria o foco de estudar os algoritmos para reconhecimento de interlocutores. Assim como foi explicado na seção 2.5, esse VAD já foi pré-treinado pelo projeto que o desenvolve e utiliza um grande conjunto de características para construir um modelo estatístico de classificação.

Assim como foi discutido na seção de fundamentação teórica, a maioria dos VADs fazem análise do espectro de frequência a curto prazo, portanto o áudio precisa ser dividido em frames. No caso dessa biblioteca, podem ser usados frames de 10 ms, 20 ms ou 30 ms e optou-se por usar frames de 20ms. Quando um arquivo de áudio não tem o número de amostras necessárias para ser dividido em um número inteiro de frames, foram adicionadas amostras de valor zero ao final para que a divisão fosse possível.

Também só são permitidos alguns valores de taxa de amostragem: 8000, 16000, 32000 ou 48000 Hz, esse foi o principal motivo que exigiu a mudança das taxas de amostragem no áudio original. A única opção compatível com as taxas de amostragem, que o VLC permite na conversão para .WAV, é a taxa de 8000 Hz. Outro programa de conversão poderia ter sido utilizado, mas como uma amostragem de 8000 Hz já respeita o critério de Nyquist para os sinais de voz, essa metodologia foi mantida.

Mais um parâmetro que pode ser definido é a "agressividade" do VAD (termo utilizado pelos próprios desenvolvedores), o qual pode ser um número inteiro de 1 a 3. Quanto maior o número, maior a "agressividade", o que significa que a classificação de cada frame como um

[2] Mais informações em https://webrtc.org, último acesso em abril de 2019.

frame de silêncio será mais provável. Como o objetivo era retirar o máximo de silêncio, foi utilizado o nível de maior "agressividade".

Depois de escolher a configuração desejada, a biblioteca permite o uso de um método que retorna falso, caso o frame seja considerado silêncio e verdadeiro caso contrário. Aplicando esse método a todos os frames é possível obter um vetor booleano com o tamanho do número de frames. Entretanto, só com as classificações de cada frame como silêncio ou não-silêncio, não foi possível fazer a segmentação, porque esse VAD é passível de erros.

Dessa forma mesmo em uma janela de silêncio, pode ser que alguns poucos frames sejam classificados como um frame de voz. Para aumentar a confiança da segmentação foi preciso fazer uso de uma janela deslizante que busca por 3 frames de silêncio antes de assumir que um trecho de silêncio começou, e 3 frames de voz para assumir que esse trecho de silêncio terminou. Com esse artifício, que também é recomendado pelos desenvolvedores em um script de exemplo, foi possível obter um vetor de segmentos de áudio.

3.4 Obtenção das classes reais para cada segmento.

Depois de segmentados, os áudios precisam ser organizados e anotados com as classes que serão consideradas corretas no momento de avaliar a eficácia do reconhecedor na etapa de testes. No caso dos áudios de treinamento, que serão utilizados para a construção do modelo Gaussiano de cada interlocutor, os segmentos são todos concatenados, gerando novamente um único arquivo .WAV, mas agora sem os trechos de silêncio. A segmentação não é necessária porque todos os segmentos possuem a mesma classe que é o identificador de cada interlocutor (I1, I2 ou I3).

No caso dos áudios de teste, cada segmento é salvo como um único arquivo .WAV. Cada um desses arquivos precisa ser escutado por um ser humano para que sejam definidos quais interlocutores estão falando em cada áudio. Para auxiliar nesse processo, o sistema ajuda o avaliador com 2 artifícios, o primeiro é uma tabela que contém o nome do arquivo, a sua

duração e um campo vazio para que o avaliador coloque a classe correta. O segundo é um áudio em que todos os segmentos estão separados por um tom, de modo que o avaliador possa realizar o processo reproduzindo um único arquivo.

A tabela será utilizada no processo de teste para avaliar o sistema e o áudio separado por tons será descartado. Na tabela o avaliador deve indicar com o número identificador (1, 2 ou 3) qual interlocutor está falando em cada áudio. Se existir algum áudio irreconhecível ou um arquivo em que é possível identificar dois interlocutores, o avaliador deve preencher o campo com o número 0.

Áudios irreconhecíveis podem acontecer porque, mesmo utilizando o método de janela deslizante, algum trecho de silêncio ou ruído pode ter sido incluído por engano. E arquivos em que estão presentes dois interlocutores podem surgir porque em algumas situações o trecho de silêncio durante a troca de interlocutores não foi suficiente para que o processo de segmentação os considerasse distintos, ou foi preenchido por algum ruído que o VAD considerou como um sinal de voz.

O sistema de reconhecimento de interlocutores proposto neste trabalho só consegue indicar uma classe para cada áudio, então esse áudios contendo dois interlocutores nunca poderiam ser classificados corretamente. Dessa forma, no momento de avaliar a assertividade do reconhecedor, as linhas da tabela que possuem uma classe 0 serão desconsideradas.

3.5 Extração de características

Na etapa de extração de características cada um dos 3 áudios de treinamento e dos n áudios de teste, obtidos na etapa de segmentação, serão transformados em um vetor de MFCCs, por motivos já descritos na seção de fundamentação teórica. A implementação em Python do processo de extração tomou como base a implementação de Orchisama Das (2016), a qual estava coerente com a descrita em de Lara (2005).

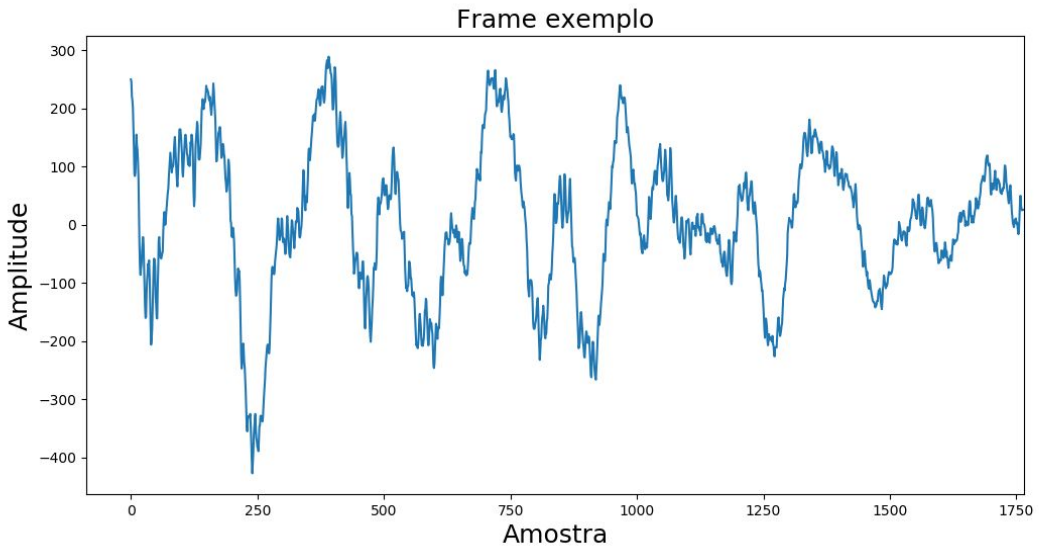
Existem bibliotecas que possuem métodos para abstrair o processo de obtenção desse vetor de MFCCs, entretanto, optou-se por não fazer uso delas pois, apesar de terem código aberto, as implementações estão otimizadas para realizar o processo da maneira mais rápida e econômica possível, e não para serem facilmente compreendidas. A implementação usada como base permitiu um controle maior do processo, melhor compreensão e facilidade para implementar as modificações necessárias. A seguir serão descritas todas as etapas até a geração do vetor de MFCCs.

3.5.1 Divisão do áudio em frames.

O processo começa com a divisão do áudio em frames novamente, mas dessa vez em frames de 30ms e com 10ms de sobreposição. Os intervalos escolhidos são estratégicos pois na etapa anterior os audios sofreram uma segmentação em frames de 20ms sem sobreposição, portanto na nova segmentação, com sobreposição, o número de frames será o mesmo.

A sobreposição é necessária pois cada frame será submetido a um janelamento no passo seguinte que vai atenuar as descontinuidades geradas nas extremidades do frame. Dessa forma, a sobreposição com o frame anterior permite que todas as amostras da onda original ocupem em algum frame a parte central. Caso contrário, todas as amostras nas extremidades seriam perdidas pela atenuação.

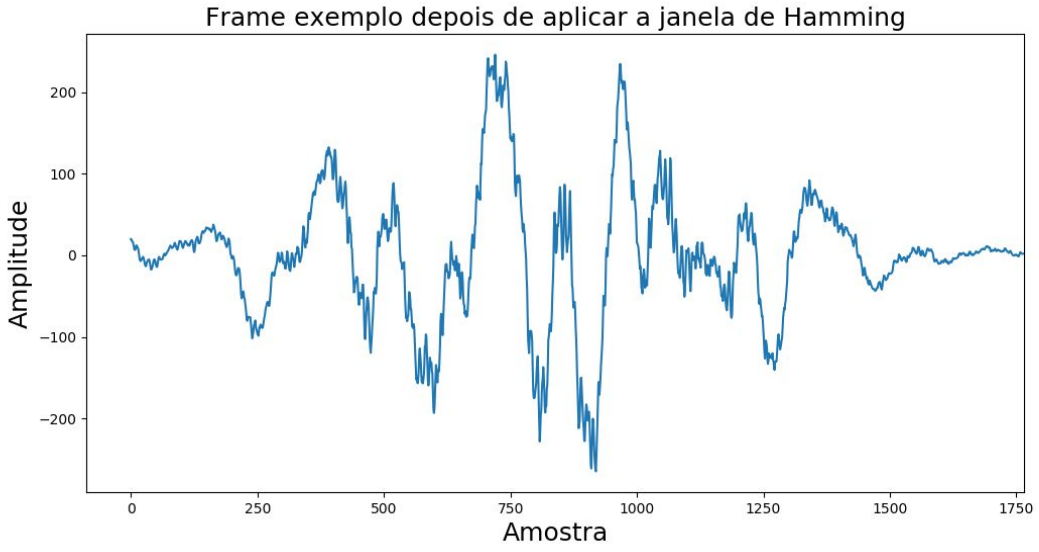
Figura 2: Exemplo de um dos Frames extraídos



3.5.2 Janelamento do frame

Depois de extrair os frames, é preciso aplicar uma janela de Hamming a cada um deles para atenuar as descontinuidades presentes na extremidade. Essas descontinuidades provocariam distorções indesejadas no espectro de frequência.

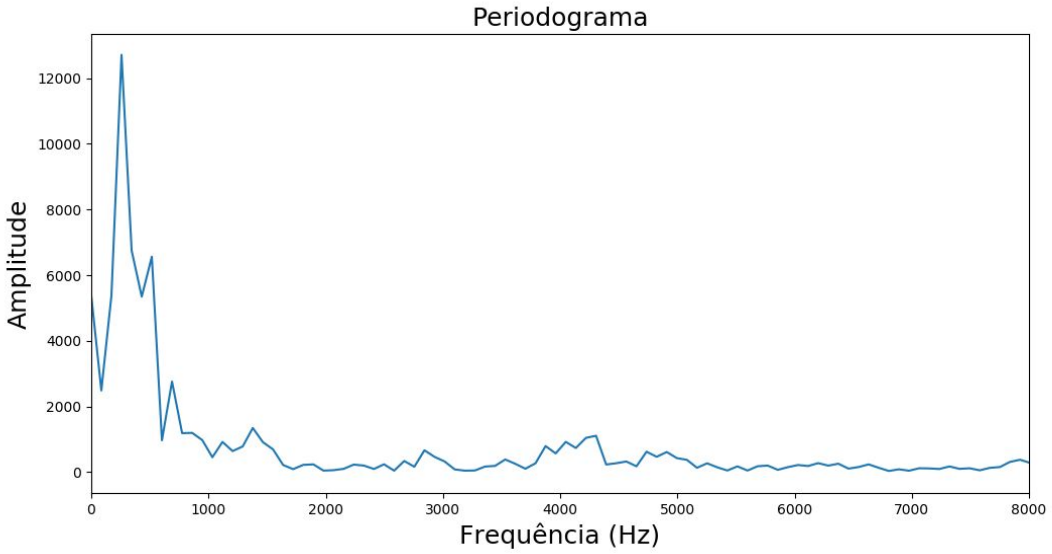
Figura 3: Frame exemplo depois de aplicada a janela de Hamming.



3.5.3 Cálculo do espectro de frequência e periodograma.

O espectro de frequência é obtido usando a FFT no frame janelado. A partir da FFT pode ser obtido o periodograma.

Figura 4: Periodograma do Frame de exemplo.



3.5.4 Filtragem na escala mel.

O banco de filtros de escala mel é aplicado ao periodograma. Foram utilizados 12 filtros nesse exemplo, em um intervalo de frequências de 300 a 8000 Hz. Esse procedimento reduz o periodograma a apenas 12 coeficientes, com mais resolução para as baixas frequências, que são mais percebidas pelo ouvido humano.

Figura 5: Doze filtros de escala Mel calculados.

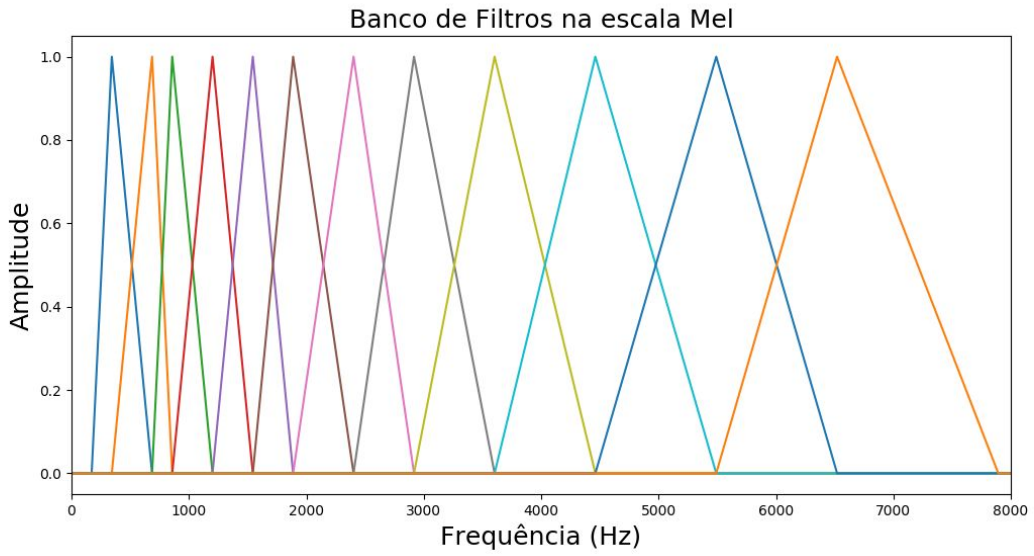
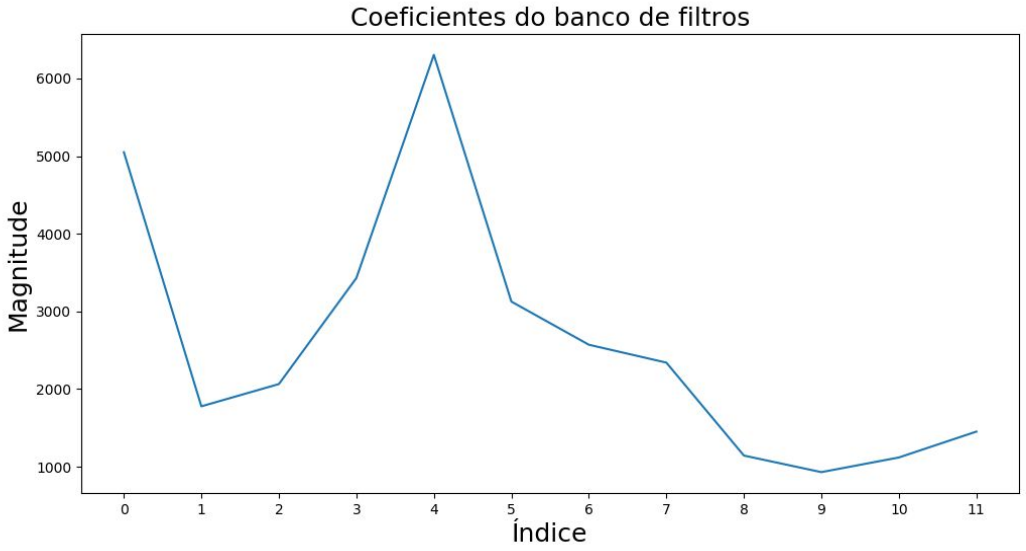


Figura 6: Doze coeficientes obtidos para o Frame exemplo.

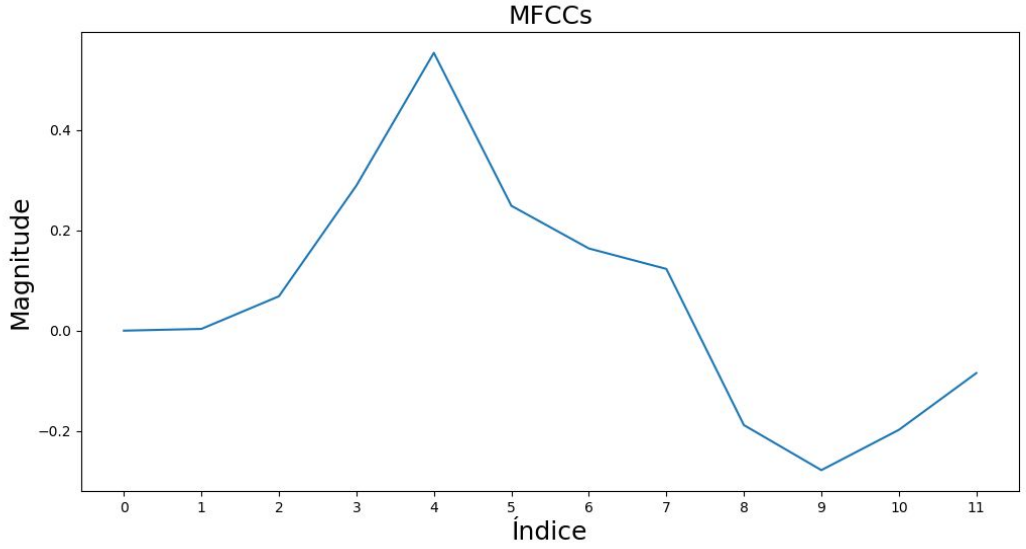


É possível observar como os detalhes do periodograma para as maiores frequências são simplificados, enquanto os picos nas baixas frequências, que eram estreitos, foram pronunciados.

3.5.5 Compressão e ortogonalização

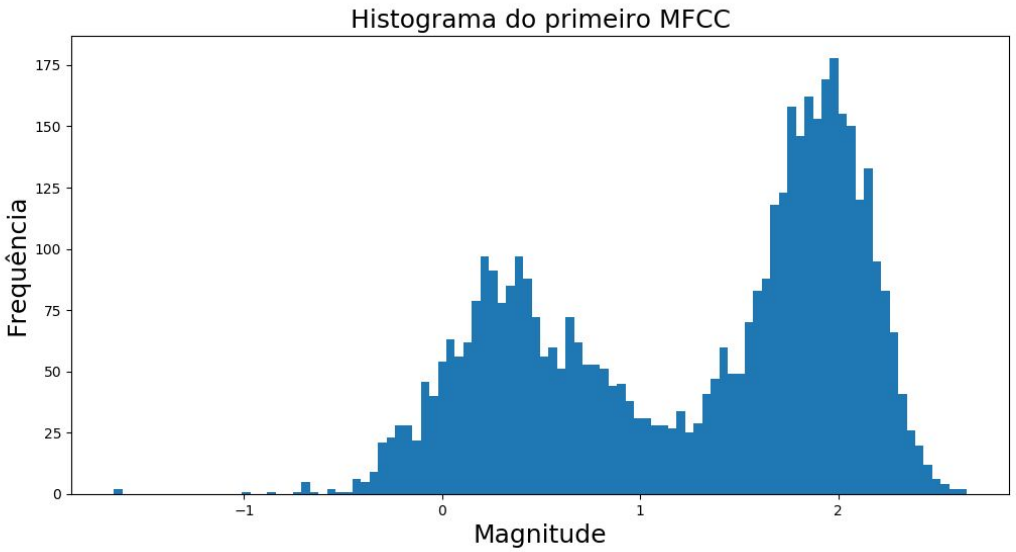
A aplica-se a transformada do cosseno ao logaritmo dos coeficientes do passo anterior. Geralmente a DCT-II é utilizada como uma forma equivalente da IDFT, mas que traz ganhos de ortogonalização. O primeiro coeficiente da DCT é posteriormente atribuído como zero para ser descartado pois esse primeiro coeficiente representa o valor médio, ou seja, não realiza nenhuma diferenciação por si só.

Figura 7: Doze coeficientes Mel-Cepstrais obtidos para o Frame exemplo.



Esse mesmo procedimento é repetido para todos os frames extraídos de cada áudio, gerando um vetor de dimensão M por N, sendo M o número de MFCCs (12 nesse caso) e N o número de frames. Podemos pegar como exemplo o áudio de treinamento do interlocutor 1, que tem 2 minutos e 19 segundos de duração. Esse arquivo foi dividido em 9190 frames de 1102 amostras. O histograma que representa a variação do primeiro coeficiente em todos os frames é mostrado abaixo.

Figura 8: Histograma da variação por segmento do primeiro coeficiente Mel-Cepstral



Pelo histograma podemos perceber como a variação do coeficiente seria descrita em uma mistura de pelo menos duas Gaussianas muito evidentes, uma de média aproximadamente 0,8 e maior variância, e outra de média aproximadamente 2,3 e menor variância. Como foi descrito em Reynolds e Rose (1995), esse comportamento ajuda a justificar o uso de modelos de misturas Gaussianas para esse tipo de problema, em detrimento de outros métodos de modelagem.

3.5.6 Escolha do número de MFCCs

Nas seções anteriores foram apresentados os passos para obtenção de 12 MFCCs a partir de cada frame do áudio original. Entretanto, o número de MFCCs pode ser maior ou menor, dependendo do número de filtros utilizado no banco de filtros. 12 filtros foi um número descrito na maioria das referências (DE LARA, 2005; REYNOLDS & ROSE, 1995; KENNUNEN & LI, 2010), entretanto números maiores também são válidos (LU & DANG, 2008).

Com base em testes preliminares das taxas de reconhecimento optou-se por utilizar um número fixo de 20 MFCCs. O desempenho do sistema completo foi melhor do que quando utilizando 12 MFCCs e o tempo de execução dos testes foi praticamente o mesmo. Com um

número maior que 20, notou-se um aumento expressivo do tempo de execução e as taxas de reconhecimento não melhoraram.

Esses resultados não serão exibidos em detalhes nesse trabalho pois a variação do número de MFCCs não se mostrou tão determinante quanto outros parâmetros na etapa de modelagem de interlocutores, como será descrito a seguir. Dessa forma, optou-se por fixar em 20 o número de MFCCs e dar foco na análise dos modelos. Análises envolvendo esse parâmetro podem ser feitas no futuro, também com testes automatizados.

3.6 Treinamento dos modelos de cada interlocutor

Depois de obtidos os vetores de MFCCs na etapa anterior, aqueles que foram originados de áudios de treinamento, serão utilizados para a obtenção de um modelo de misturas de Gaussianas que melhor represente cada interlocutor. Como foi discutido na seção de fundamentação teórica, os algoritmos utilizados para gerar esse tipo de modelo não são nada triviais, como o algoritmo EM (Expectation Maximization). Portanto, nesse caso, optou-se por utilizar uma biblioteca para gerar os modelos.

A biblioteca escolhida foi a Scikit Learn (PEDREGOSA, 2011), que além de ser uma iniciativa de código aberto, já possui mais de 1200 contribuidores no GitHub. Grandes empresas como a JPMorgan e o Spotify também fazem uso dela, comprovando a confiabilidade das implementação dos mais diversos modelos disponíveis.

O processo de geração de GMMs é feito utilizando o algoritmo EM, mas toda a complexidade fica abstraída para o desenvolvedor. Um detalhe importante de ser observado é que a inicialização do algoritmo EM é feita aleatoriamente. Como consequência, ainda que os mesmo métodos e dados de treinamento sejam utilizados, pode ser que os modelos finais sejam diferentes.

Para obter um modelo de misturas de Gaussianas através da biblioteca, o desenvolvedor precisa escolher alguns parâmetros de configuração, com destaque para o número de componentes desejado e o tipo da matriz de covariâncias, além de alguns outros. A escolha de cada um dos parâmetros utilizados será descrita nas etapas a seguir e tomaram como base o artigo de Reynolds e Rose (1995), que faz uma profunda discussão sobre esse tema.

3.6.1 Tipo da matriz de covariâncias

Na biblioteca os tipos disponíveis para a matriz de covariâncias são: *full* (cada componente possui sua matriz completa de covariâncias), *tied* (todas as componentes compartilham uma mesma matriz completa de covariâncias), *diag* (cada componente possui uma matriz diagonal de covariâncias) e *spherical* (cada componente tem sua única variância).

O artigo de Reynolds e Rose (1995) argumenta que o modelo de misturas de Gaussianas que obteve os melhores resultados foi o modelo em que cada componente possui uma matriz diagonal de covariâncias, que seria equivalente ao modelo *diag* da biblioteca. Tanto o modelo do tipo de *diag* quanto o modelo de tipo *full* foram utilizados para que fosse possível confirmar essas hipóteses na seção de resultados.

3.6.2 Número de componentes

O número de componentes é um tema bastante discutido no artigo, pois não existe uma forma teórica de calcular o melhor número *a priori*. Testes precisam ser feitos para saber qual o número mínimo de componentes que precisam ser utilizadas para obter as taxas de reconhecimento desejadas (REYNOLDS & ROSE, 1995). Esse número deve ser o mínimo possível pois quanto maior o número de componentes, maior o custo computacional para ele ser gerado.

Dessa forma, assim como foi feito no artigo, foram feitas simulações com diferentes números de componentes. Para que, depois de obtidos os resultados, fosse escolhida a quantidade ideal a ser utilizada em um ambiente definitivo.

3.6.3 Limitação das variâncias

Um dos cuidados recomendados por Reynolds e Rose (1995) é a utilização de um limite para a magnitude mínima que as variâncias podem alcançar. O artigo descreve que, para Gaussianas com um grande número de componentes, algumas variâncias podem alcançar magnitudes muito pequenas, por conta de ruído ou ainda uma quantidade insuficiente de dados. Quando esse fenômeno acontece, podem ocorrer distorções durante a classificação do interlocutor mais provável.

Dessa forma, se faz necessário utilizar um artifício que limite o valor das variâncias durante as iterações do algoritmo de geração dos modelos. A biblioteca do Scikit Learn possui um parâmetro que define a variância mínima permitida, então ele foi atribuído em 0.01, como recomendado no artigo. Em testes preliminares foi de fato percebida a existência das distorções.

3.6.4 Treinamento

Depois de escolher os parâmetros, basta usar o método *fit* em um vetor de D MFCCs (dimensões) e N frames para obter um modelo de misturas de Gaussianas. Isso é feito para cada um dos áudios de treinamento, gerando um modelo para cada um dos 3 interlocutores. Ao final do tempo de execução, dependendo do número K de componentes escolhido, a biblioteca irá calcular os K pesos de cada componente, as K médias para cada componente levando em contas as D dimensões de MFCCs (matriz de dimensão K por D) e a matriz de covariâncias, cuja dimensionalidade depende do tipo escolhido.

Se o tipo de matriz for *diag* o tamanho da matriz de covariâncias também será K por D, assim como a matriz de médias. Entretanto, se o tipo de matriz for *full* a matriz de covariâncias será de dimensão K x D x K. Todas essas informações ficam abstraídas em um objeto que representa o modelo como um todo, mas podem ser extraídas se for preciso.

3.6.5 Classificação dos áudios de teste

Com os modelos criados para cada interlocutor, inicia-se o processo de classificação dos áudios de teste. O objetivo é saber a qual dos modelos o vetor de MFCCs dos áudios de teste tem a maior probabilidade *a posteriori* de pertencer. A biblioteca calcula essa probabilidade através do método *score*, que realiza as somatórias descritas na equação 5 para obter o logaritmo da verossimilhança.

O melhor modelo é aquele que apresenta o maior valor da função *score* para um mesmo áudio de teste. O identificador do melhor modelo é adicionado a cada linha da tabela gerada na etapa de segmentação, dessa forma temos as classificações reais e as classificações geradas pelo reconhecedor.

3.7 Avaliação dos modelos

Para avaliar os modelos e a metodologia como um todo foram feitos alguns cenários de teste variando parâmetros para encontrar a melhor configuração que poderia ser usada em uma aplicação real. Em todos os cenários a taxa de acerto dos áudios de teste foi a principal métrica utilizada para comparação, calculada a partir da tabela de resultados citada na seção anterior. Espera-se, no mínimo, que o sistema apresente uma taxa de acerto maior do que a proporção da classe majoritária, também conhecida como *baseline* ou *ground truth*.

Entretanto, existem muitos parâmetros para serem avaliados, de modo que não é possível testar neste trabalho o efeito de todos eles nas taxas de acerto. Os cenários de teste escolhidos tomaram como base alguns dos testes feitos por Reynolds e Rose (1995), variando principalmente o número de componentes Gaussianas de cada modelo. Dessa forma, será possível comparar os resultados com testes anteriores da literatura.

Portanto, a seção de resultados apresentará a avaliação dos modelos de misturas de Gaussianas com matriz de covariâncias diagonal, um número fixo de MFCCs, variando o número de componentes, como foi feita na primeira parte do artigo de Reynolds e Rose (1995). Depois repete-se o mesmo teste, mas para matriz cheia, um teste que foi citado no artigo de referência, mas apenas como um teste preliminar, cujos resultados não foram demonstrados.

Além disso, existem em cada cenário duas opções possíveis de avaliação. Como o diálogo acontece sempre em duplas, podemos treinar o sistema com apenas os áudios dos dois interlocutores que estão presentes no diálogo, ou podemos treinar sempre com os 3 interlocutores possíveis. No primeiro caso, o sistema só tem uma possibilidade para qual ele pode errar, no segundo caso são duas possibilidades de erro.

4 RESULTADOS E DISCUSSÕES

Antes que cada cenário seja apresentado, se faz necessário uma pequena introdução com os resultados obtidos nas etapas de segmentação e eliminação de silêncio, de modo que a natureza das gravações de teste e treinamento fiquem expostas.

4.1 Resultados dos processos de segmentação e eliminação de silêncio.

Pela Tabela 1 é possível observar os efeitos produzidos na duração dos arquivos de áudio após a eliminação de silêncio utilizando o VAD.

Tabela 1: Comparação da duração de cada áudio de treinamento, antes e depois da eliminação de silêncio.

	Antes (m:s)	Depois (m:s)	Delta (m:s)
I1	02:44	2:19	0:25
I2	02:36	2:03	0:33
I3	02:13	1:54	0:19

Observa-se que a quantidade de silêncio retirada dos áudios está na ordem de dezenas de segundos. Provavelmente, isso já seria o suficiente para gerar, dentro de um modelo estatístico, padrões redundantes e não dependentes do interlocutor. As próximas tabelas irão mostrar os resultados da segmentação feita nos áudios de teste.

Tabela 2: Número de segmentos gerados a partir de cada áudio de teste.

	Duração (m:s)	Número de segmentos	Número de segmentos válidos
I1 e I2	04:04	123	115
I1 e I3	04:05	110	99
I2 e I3	04:10	131	126

De cada um dos áudios de diálogo são retirados segmentos, como foi detalhado nas seções 3.3 e 3.4, e para segmentos em que não foi possível detectar o interlocutor ou existem dois interlocutores falando ao mesmo tempo foi atribuída uma classe inválida. Na Tabela 2 é possível observar o número de áudios válidos que restaram. As métricas obtidas nos resultados para cada cenário de teste levam em conta pelo menos 99 áudios. Na Tabela 3 é possível observar também a distribuição deles do ponto de vista de duração.

Tabela 3: Número de segmentos extraídos para cada faixa de duração, em segundos.

Duração (s)	I1 e I2	I1 e I3	I2 e I3
0.0 > d > 0.5	7	5	7
0.5 > d > 1.0	33	23	44
1.0 > d > 1.5	20	22	27
1.5 > d > 2.0	15	13	20
2.0 > d > 2.5	23	18	9
2.5 > d > 3.0	6	10	6
3.0 > d > 3.5	7	2	8
3.5 > d > 4.0	2	3	2
4.0 > d > 4.5	0	1	1
4.5 > d > 5.0	2	2	2
Total	115	99	126

Nota-se que a maioria dos segmentos possui entre 0.5 e 1.0 segundos de duração e que a distribuição total é bastante diversa. Todos esses áudios serão avaliados em cenários de teste e a acurácia dependerá de quantos deles foram corretamente classificados ao seu interlocutor de origem. Em cada cenário é realizada alguma mudança de parâmetros para que possam ser avaliadas as influências deles no processo de reconhecimento.

4.2 Cenários de teste

Nesta seção serão evidenciados os resultados de cada cenário de testes, com o objetivo de analisar o efeito da variação de alguns parâmetros nas taxas de acurácia do sistema. Como explicado na seção 3.7, foi dado foco a dois parâmetros de projeto: a quantidade de componentes Gaussianas na GMM e o tipo da matriz de covariâncias utilizada. Além disso, outro fator considerado essencial para a avaliação é o número de interlocutores possíveis no treinamento.

Dessa forma, foram feitos 4 cenários de teste, um para cada tipo de matriz: cheia ou diagonal, com possibilidades de 2 ou 3 interlocutores no treinamento. Nos 4, vários números de componentes Gaussianas foram analisados e utiliza-se sempre 20 MFCCs.

- Primeiro cenário: Matriz de covariância diagonal e 2 interlocutores no treinamento;
- Segundo cenário: Matriz de covariância diagonal e 3 interlocutores no treinamento;
- Terceiro cenário: Matriz de covariância cheia e 2 interlocutores no treinamento;
- Quarto cenário: Matriz de covariância cheia e 3 interlocutores no treinamento;

4.2.1 Primeiro cenário de teste

O primeiro cenário simula um dos testes de Reynolds e Rose (1995), mostrando as taxas de acerto obtidas através de modelos de misturas de Gaussianas com matrizes diagonais. O número de componentes é o parâmetro principal de avaliação, portanto nas Figuras 9, 10 e 11 serão mostradas as taxas de acerto para n componentes, variando n entre 1, 8, 16, 32 e 64.

Além disso, nesse primeiro cenário, o sistema foi treinado apenas com os dois interlocutores presentes no diálogo. Isso fica evidenciado nas tabelas 4, 5 e 6, as quais representam as matrizes de confusão do sistema. Essas matrizes mostram exatamente quantos erros aconteceram no teste de maior acurácia e quais foram as classes trocadas em cada erro.

Figura 9: Taxa de acerto do diálogo entre I1 e I2, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

Tabela 4: Matriz de confusão do diálogo entre I1 e I2 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

		Valor previsto	
		I1	I2
Valor verdadeiro	I1	48	0
	I2	0	67

Figura 10: Taxa de acerto do diálogo entre I1 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

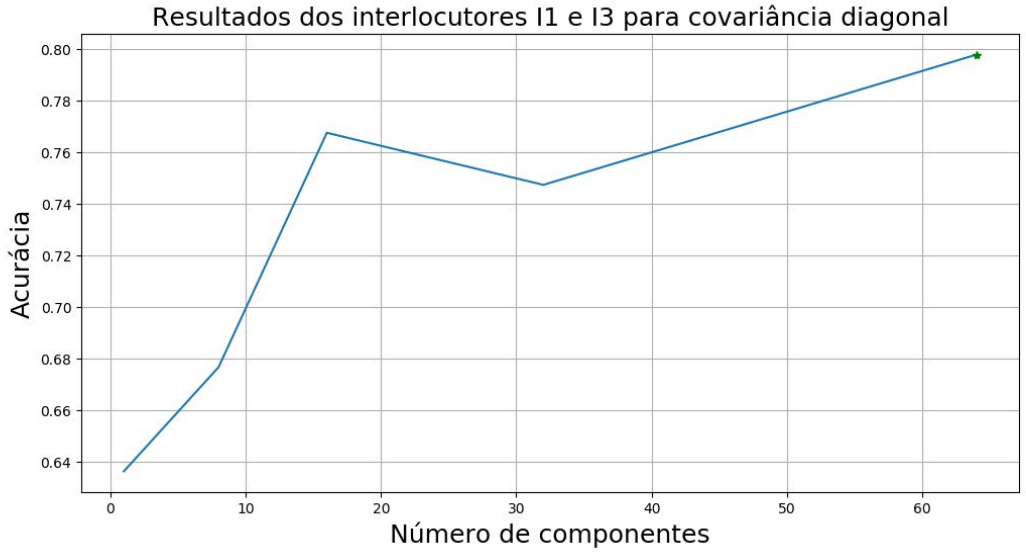


Tabela 5: Matriz de confusão do diálogo entre I1 e I3 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

		Valor previsto	
		I1	I2
Valor verdadeiro	I1	47	0
	I2	20	32

Figura 11: Taxa de acerto do diálogo entre I2 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

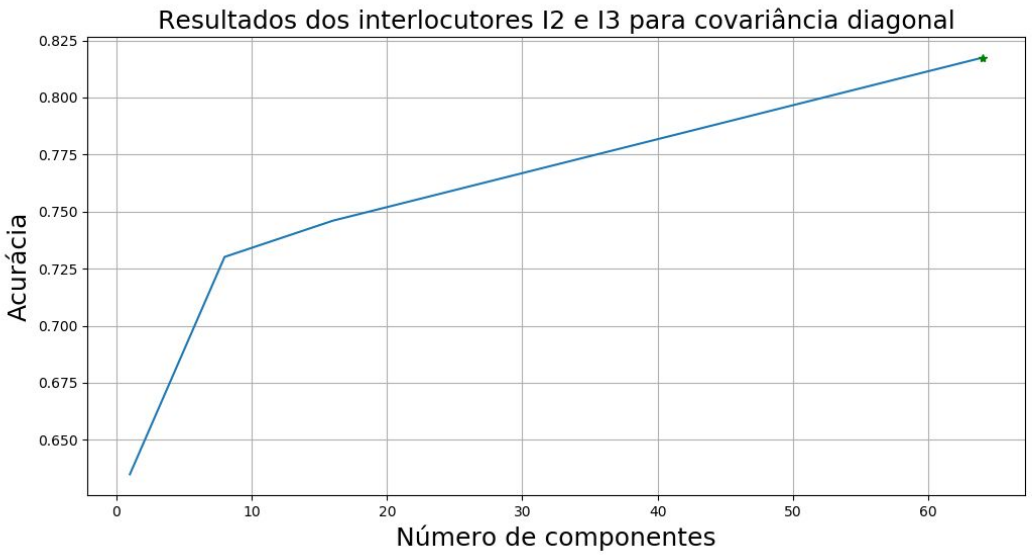


Tabela 6: Matriz de confusão do diálogo entre I2 e I3 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

		Valor previsto	
		I2	I3
Valor verdadeiro	I2	68	5
	I3	18	35

Tabela 7: Resumo das taxas e médias finais de acerto variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e apenas 2 interlocutores no treinamento.

	1	8	16	32	64
I1, I2	0.626	0.948	0.991	0.991	1.000
I1, I3	0.636	0.677	0.767	0.747	0.798
I2, I3	0.635	0.730	0.746	0.770	0.817
Média	0,632	0,785	0,835	0,836	0,872

Os resultados obtidos nesse primeiro cenário com as matrizes diagonais são semelhantes aos obtidos por Reynolds e Rose (1995), no qual observa-se o aumento da acurácia à medida que o número de componentes cresce. Dessa forma, o teste com 64 componentes é sempre igual ou superior aos testes anteriores. Isso é esperado do ponto de vista teórico, pois com um maior número de componentes é possível criar um modelo cada vez mais detalhado da distribuição estatística dos vetores de MFCCs.

Entretanto, também percebe-se que a curva de melhoria da acurácia pode seguir um padrão logarítmico, ou seja, o incremento de melhoria vai diminuindo até encontrar o que parece ser um limite assintótico horizontalmente. Portanto, apenas aumentando o número de componentes talvez nunca seja alcançado o valor de acurácia perfeito. Apenas em um dos diálogos esse limite foi alcançado, entre I1 e I2, o que significa que nesse áudio de teste todos os segmentos foram classificados corretamente.

De fato, do ponto de vista de um avaliador humano, as vozes I2 e I1 são as mais distintas entre si, pois representam uma voz masculina, mais grave, e outra feminina, mais aguda. Além disso, dentre as vozes masculinas, I2 é a voz mais grave delas. Observando as matrizes de confusão vemos na Tabela 5 que no diálogo entre I1 e I3 existe uma viés bastante grande em favor de I1, enquanto que, na Tabela 6, o sistema se mostra confuso para as duas classes.

4.2.2 Segundo cenário de testes

O segundo cenário é praticamente o mesmo que o anterior, porém, o sistema foi treinado com os três interlocutores, ou seja, um deles não está presente no diálogo mas tem chance de ser reconhecido pelo sistema. Como foi explicado na seção 3.7, isso significa uma complexidade maior porque será incluída uma outra possibilidade de erro. Os resultados de cada teste desse cenário e uma tabela contendo as médias serão apresentados abaixo:

Figura 12: Taxa de acerto do diálogo entre I1 e I2, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

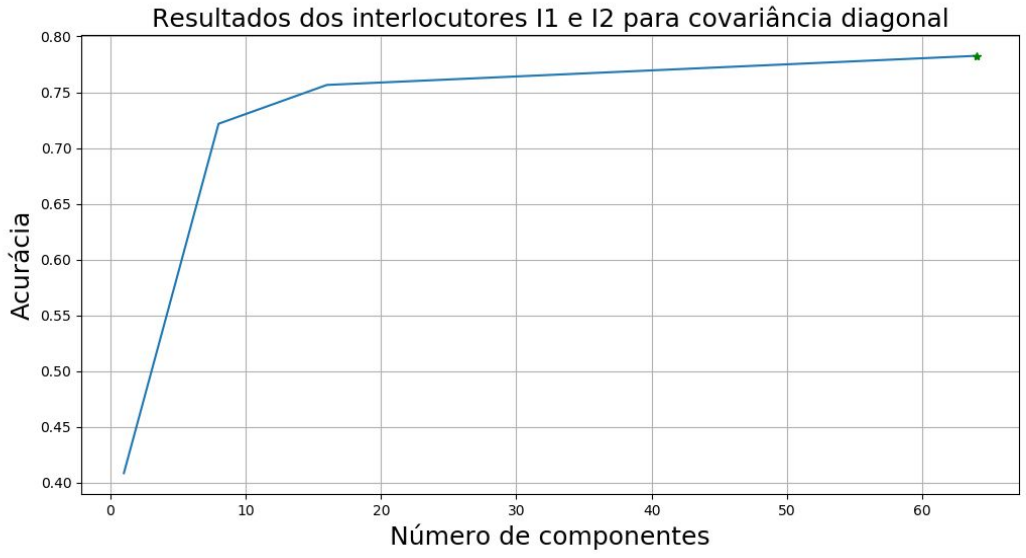


Tabela 8: Matriz de confusão do diálogo entre I1 e I2 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

		Valor previsto		
		I1	I2	I3
Valor verdadeiro	I1	47	0	1
	I2	0	43	24
	I3	0	0	0

Figura 13: Taxa de acerto do diálogo entre I1 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

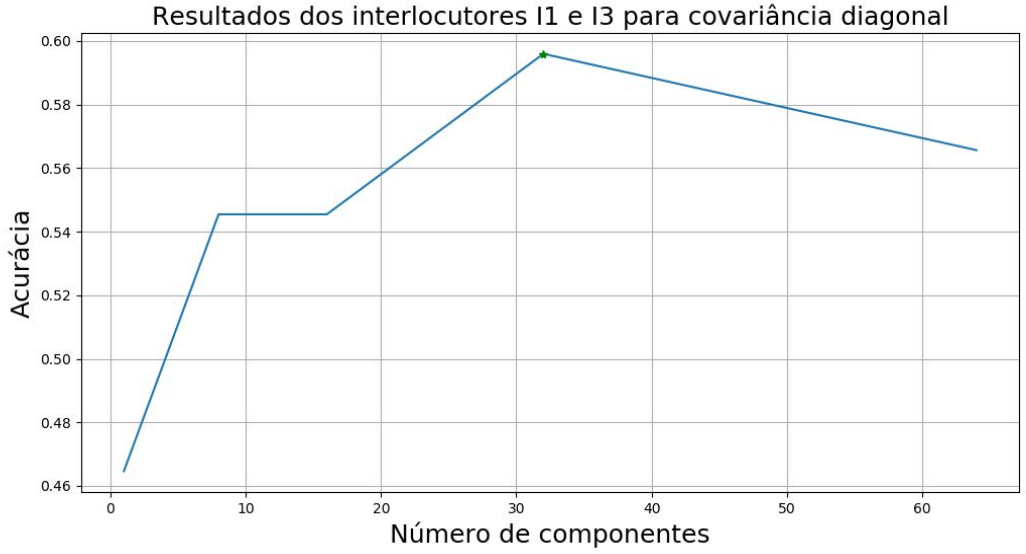


Tabela 9: Matriz de confusão do diálogo entre I1 e I3 para o melhor resultado do teste anterior, o qual utiliza 32 componentes, matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

		Valor previsto		
		I1	I2	I3
Valor verdadeiro	I1	47	0	0
	I2	0	0	0
	I3	11	29	12

Figura 14: Taxa de acerto do diálogo entre I2 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

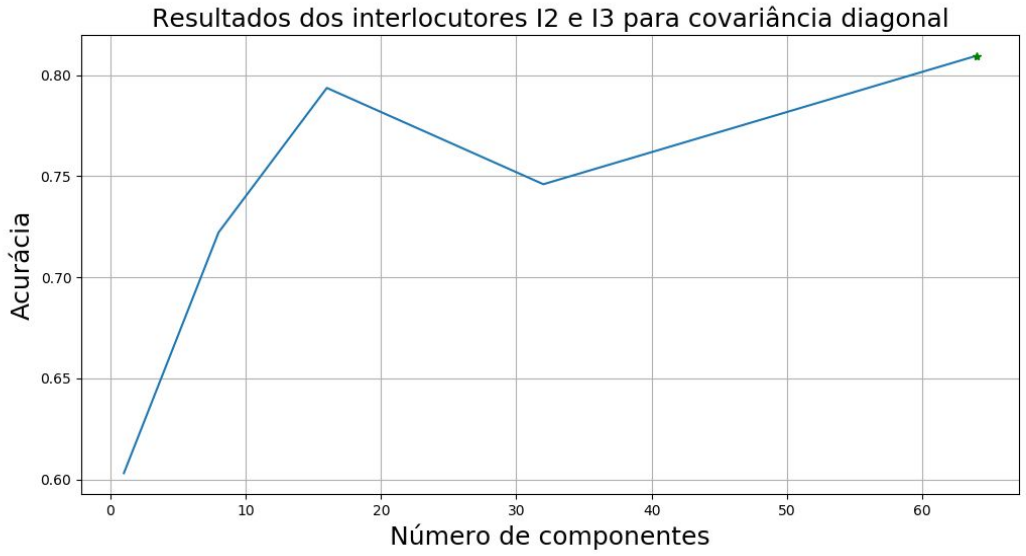


Tabela 10: Matriz de confusão do diálogo entre I2 e I3 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

		Valor previsto		
		I1	I2	I3
Valor verdadeiro	I1	0	0	0
	I2	0	67	6
	I3	1	17	35

Tabela 11: Resumo das taxas e médias finais de acerto variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes diagonais, 20 MFCCs e 3 interlocutores no treinamento.

	1	8	16	32	64
I1, I2	0.409	0.722	0.756	0.765	0.783
I1, I3	0.465	0.545	0.545	0.596	0.566
I2, I3	0.603	0.722	0.794	0.746	0.809
Média	0,492	0,663	0,698	0,702	0,719

Comparando com resultados anteriores, percebe-se que a inclusão de um terceiro interlocutor possível no treinamento faz com que a acurácia média do sistema diminua cerca de 15%. Vemos na matriz de confusão do diálogo entre I1 e I3, que foi o pior caso, uma grande quantidade de áudios que foram reconhecidos como sendo de I2, mesmo que ele nem esteja presente no áudio. De fato, o número de casos em que I2 é classificado, chega a ser maior do que I1 e I3.

A acurácia em todos os testes ainda fica maior do que a *baseline*, que é a taxa de acerto de um reconhecedor que chuta sempre a classe majoritária. Entretanto, podemos observar uma limitação do sistema, que precisaria contar com outras técnicas para diminuir esse efeito

degradativo quando mais interlocutores são incluídos. Existem discussões na literatura sobre esse efeito e algumas das técnicas que podem ser utilizadas para evitá-lo, envolvendo principalmente métodos de normalização.

Nota-se que a maior quantidade de erros ocorre entre as classificações de I2 e I3, que são as duas vozes masculinas. Sendo que, no geral, nota-se que I3 é a classe menos indicada pelo sistema. Isso pode ser devido ao fato da voz I3 estar, como discutido anteriormente, no meio termo entre os dois outros interlocutores quando se considera um gradiente indo da voz mais aguda até a voz mais grave.

4.2.3 Terceiro cenário de testes

O terceiro cenário também é similar ao primeiro, porém, os modelos de misturas de Gaussianas foram obtidos utilizando matrizes de covariância cheia.

Figura 15: Taxa de acerto do diálogo entre I1 e I2, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

Tabela 12: Matriz de confusão do diálogo entre I1 e I2 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

		Valor previsto	
		I2	I3
Valor verdadeiro	I2	48	0
	I3	0	67

Figura 16: Taxa de acerto do diálogo entre I1 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

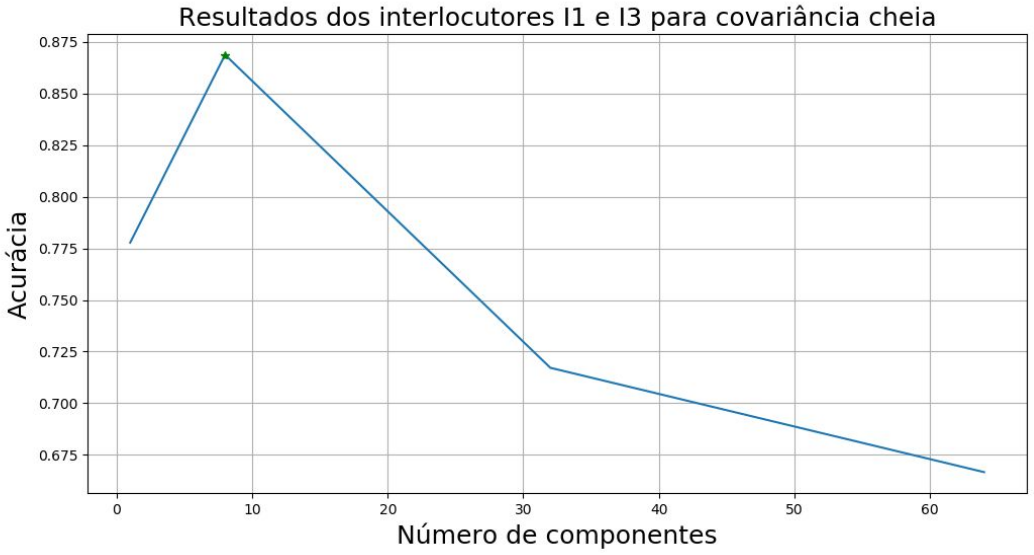


Tabela 13: Matriz de confusão do diálogo entre I1 e I3 para o melhor resultado do teste anterior, o qual utiliza 8 componentes, matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

		Valor previsto	
		I1	I3
Valor verdadeiro	I1	47	0
	I3	13	39

Figura 17: Taxa de acerto do diálogo entre I2 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

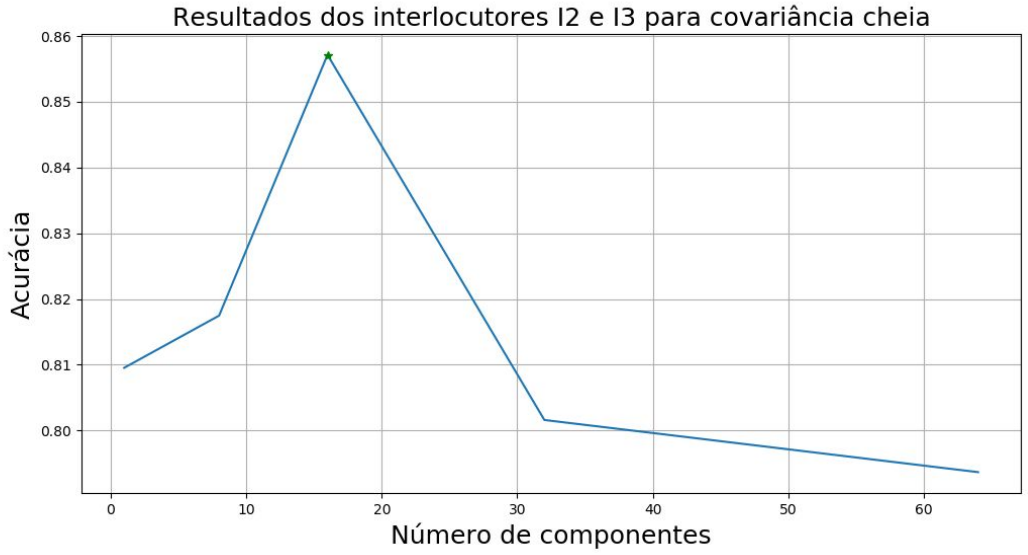


Tabela 14: Matriz de confusão do diálogo entre I2 e I3 para o melhor resultado do teste anterior, o qual utiliza 16 componentes, matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

		Valor previsto	
		I2	I3
Valor verdadeiro	I2	66	7
	I3	11	42

Tabela 15: Resumo das taxas e médias finais de acerto variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e apenas 2 interlocutores no treinamento.

	1	8	16	32	64
I1, I2	0.974	1.0	1.0	1.0	1.0
I1, I3	0.778	0.869	0.818	0.717	0.667
I2, I3	0.809	0.817	0.857	0.801	0.794
Média	0,854	0,895	0,892	0,839	0,820

Comparando esses resultados da Tabela 15 com os da Tabela 7, podemos observar as diferenças entre o uso de matrizes diagonais e matrizes cheias. Para o diálogo entre I1 e I2 quase não notou-se diferença. Entretanto, nos outros dois diálogos nota-se que para as matrizes de covariância cheia, a acurácia tende a crescer até um certo número de componentes e depois começa a cair. Diferente do padrão logarítmico observado nas matrizes diagonais.

Na média geral, observa-se que o uso de 64 componentes é a pior opção em contraste com o resultado das matrizes diagonais, no qual representava a melhor alternativa. Nota-se ainda que para o diálogo entre I1 e I3 o melhor número de componentes é apenas 8. Essa perda de acurácia com o aumento do número de componentes também faz sentido do ponto de vista

teórico. Como foi apresentado na seção 2.5, ao se utilizar matrizes cheias exige-se uma quantidade mínima de dados de treinamento maior do que exigido pela matrizes diagonais para obter as correlações entre todas as componentes.

Dessa forma, quando o número dessas componentes aumenta, a quantidade de dados necessária também cresce. Quando os dados de treinamento se tornam insuficientes para gerar uma matriz representativa das correlações, as taxas de reconhecimento começam a diminuir. Esse comportamento é citado tanto em Reynolds e Rose (1995), justificando o uso de matrizes diagonais, quanto em Lu e Dang (2008), justificando a escolha dos autores por matrizes cheias de apenas 4 componentes.

Entretanto, mesmo com essa queda nos modelos de muitas componentes, o uso das matrizes cheias mostrou-se competitivo o que vai, de certa forma, em contraste com o que foi descrito em Reynolds e Rose (1995). Comparando as melhores médias, vemos que a média de acerto para 8 componentes usando matrizes cheias foi superior ao uso de 64 componentes usando matrizes diagonais. Isso vai de acordo com o observado em Lu e Dang (2008).

4.2.3 Quarto cenário de testes

O quarto cenário também utiliza os modelos de misturas de Gaussianas com matrizes de covariância cheia. Porém, foi feita a mesma análise do segundo cenário, no qual o sistema foi treinado com 3 interlocutores.

Figura 18: Taxa de acerto do diálogo entre I1 e I2, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

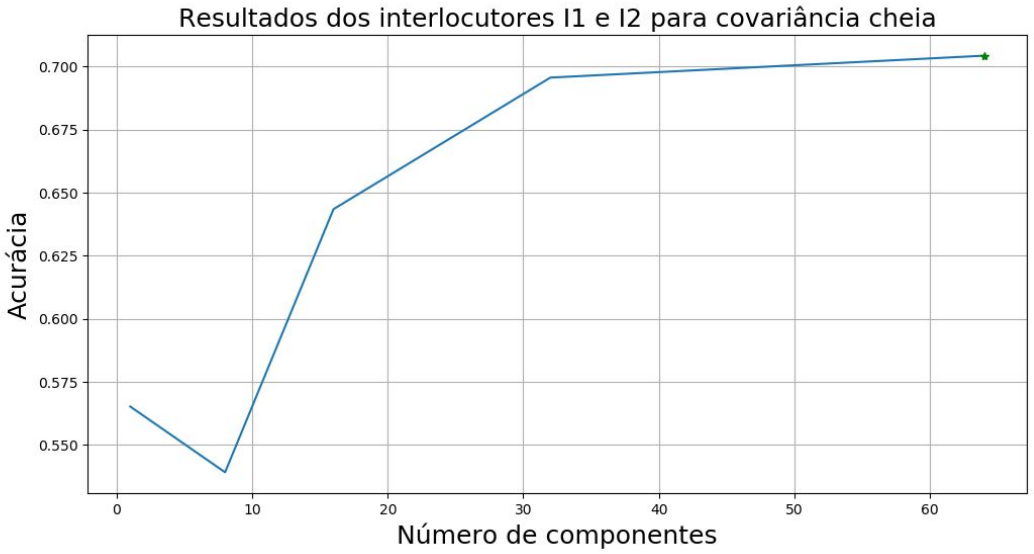


Tabela 16: Matriz de confusão do diálogo entre I1 e I2 para o melhor resultado do teste anterior, o qual utiliza 64 componentes, matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

		Valor previsto		
		I1	I2	I3
Valor verdadeiro	I1	48	0	0
	I2	0	33	34
	I3	0	0	0

Figura 19: Taxa de acerto do diálogo entre I1 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

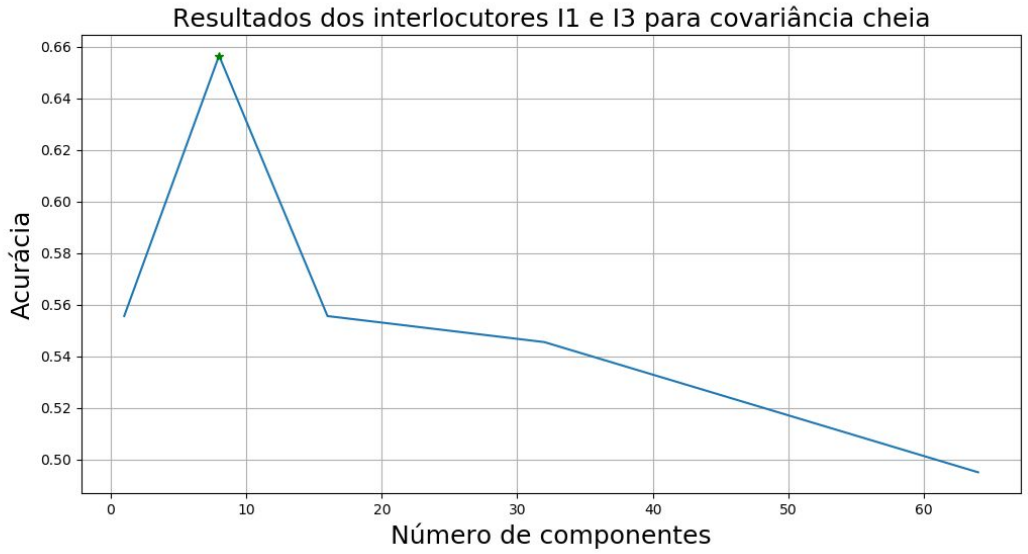


Tabela 17: Matriz de confusão do diálogo entre I1 e I3 para o melhor resultado do teste anterior, o qual utiliza 8 componentes, matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

		Valor previsto		
		I1	I2	I3
Valor verdadeiro	I1	47	0	0
	I2	0	0	0
	I3	10	24	18

Figura 20: Taxa de acerto do diálogo entre I2 e I3, variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

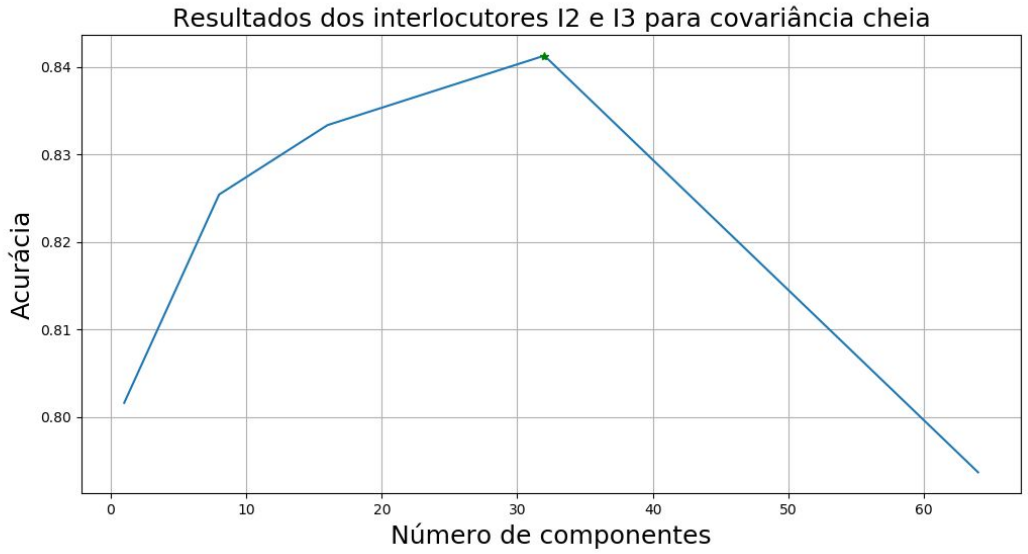


Tabela 18: Matriz de confusão do diálogo entre I2 e I3 para o melhor resultado do teste anterior, o qual utiliza 32 componentes, matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

		Valor previsto		
		I1	I2	I3
Valor verdadeiro	I1	0	0	0
	I2	0	67	0
	I3	2	12	39

Tabela 19: Resumo das taxas e médias finais de acerto variando o número de componentes Gaussianas entre 1, 8, 16, 32 e 64. Usando matrizes cheias, 20 MFCCs e 3 interlocutores no treinamento.

	1	8	16	32	64
I1, I2	0.565	0.539	0.643	0.696	0.704
I1, I3	0.556	0.656	0.555	0.545	0.495
I2, I3	0.801	0.825	0.833	0.841	0.794
Média	0,641	0,673	0,677	0,694	0,664

Através da comparação desses resultados da Tabela 19 com os Tabela 15, observa-se que os modelos de matrizes cheias também apresentam uma queda significativa da acurácia quando mais interlocutores são adicionados no conjunto de interlocutores possíveis. Ainda nesse tópico, observa-se através de uma comparação com a Tabela 11 que as matrizes diagonais apresentam no geral um resultado superior ao das matrizes cheias nesse cenário incluindo 3 interlocutores.

Além disso, observou-se na combinação dos resultados das Tabelas 15 e 19 que o número de componentes ideal para obter a melhor acurácia usando matrizes cheias varia bastante em cada diálogo. O resultado de 8 componentes que tinha a melhor média na Tabela 15 virou o terceiro melhor na tabela 19, por exemplo. Utilizando matrizes diagonais os resultados são mais consistentes, tendo em vista que o resultado de 64 componentes que era o de melhor média na Tabela 7, continuou sendo o melhor na tabela 11.

Essa consistência justificaria a escolha dos modelos de matrizes diagonais de 64 componentes como o mais adequado em uma aplicação real. Mas mais importante do que escolher uma configuração otimizada, esses resultados mostram a importância de se construir testes automatizados para testar parâmetros em um sistema como esse. Cada configuração pode mudar drasticamente a acurácia e a teoria só nos permite prever alguns comportamentos

5 CONCLUSÃO

O primeiro a concluir sobre esse trabalho é que foi possível construir um sistema automático de reconhecimento de interlocutores utilizando apenas ferramentas *open-source* e fontes de código abertas. Sua efetividade fica evidenciada porque foram obtidas, em vários testes, taxas de reconhecimento maiores do que a *baseline*, ou seja, a taxa de acerto de um sistema que vota sempre na classe majoritária.

A implementação descrita foi baseada nos métodos considerados por vários autores como a base de referência do ramo, utilizando os MFCCs na etapa de extração de características e GMMs para modelagem dos interlocutores. Dessa forma, fica também comprovada a capacidade dos MFCCs de evidenciar as diferenças nas vozes dos seres humanos, utilizando princípios psicoacústicos na análise do envelope espectral. Assim como a capacidade das GMMs para modelar os padrões estatísticos presentes nas características e comparar a similaridade desses padrões em áudios desconhecidos para fazer a classificação.

Ao utilizar esses métodos existem uma série de parâmetros de projeto que precisam ser estudados e avaliados, como: O número de componentes Gaussianas nas GMMs, o tipo de matriz de covariâncias utilizada, o limite mínimo de magnitude para as variâncias, o número de filtros no banco utilizado para gerar os MFCCs, entre outros. Por conta do grande número de possibilidades e combinações não foi possível criar cenários de teste para todos eles, portanto, o foco foi dado ao número de componentes Gaussianas e ao tipo da matriz de covariâncias.

Dessa forma, também foi preciso desenvolver um ambiente de testes automatizados que pudesse repetir os cenários diversas vezes modificando os parâmetros utilizados e recalculando as métricas de avaliação. Esse ambiente utiliza sempre as mesmas gravações que foram obtidas para a execução desse trabalho, padronizando a execução dos cenários. Isso permitiu a observação dos efeitos de cada parâmetro no processo completo de reconhecimento e a análise de quais deveriam ser os valores escolhidos em um sistema final. A importância de

automatizar os testes ficou muito clara observando a grande variação das taxas de acurácia em cada configuração, sendo que o conhecimento teórico só nos permite prever em partes esses comportamentos.

Comparando as métricas obtidas neste trabalho com os resultados da literatura, nota-se que ainda existe uma grande margem de evolução para o sistema desenvolvido. Principalmente nos cenários em que o sistema é treinado com vários interlocutores possíveis. Para alcançar melhores taxas, são necessários não só mais estudos sobre os parâmetros disponíveis até o momento, mas também o estudo de outras técnicas que poderiam ser incorporadas na metodologia.

Existem inúmeras possibilidades já descritas na literatura, envolvendo principalmente técnicas de pré-processamento do sinal de áudio e técnicas de normalização, tanto de características quanto de modelos (*Universal Background Model* - UBM) (KENNUNEN & LI, 2010; VENTURINI, ZÃO & COELHO, 2014). Existem também aprimoramentos que poderiam ser feitos aos métodos de extração de características - como a inclusão de delta-MFCCs (REYNOLDS; ROSE, 1995; KENNUNEN & LI, 2010) - e modelagem de interlocutores. Através do ambiente de testes construído também é possível avaliar, no futuro, a inclusão dessas novas técnicas e aprimoramentos, em busca de um sistema mais robusto de reconhecimento de interlocutores.

Entretanto, a finalidade deste trabalho não era realmente a construção de um sistema necessariamente robusto. O objetivo principal era construir um sistema de reconhecimento de interlocutores funcional que possa ser uma porta de entrada para o estudo do ramo. Nesse sentido, a relativa simplicidade da metodologia é algo benéfico. Também por conta desse objetivo que foram utilizadas ferramentas de livre acesso, de modo que qualquer pessoa possa observar as implementações e executar os testes apresentados, sem nenhum custo.

Do ponto de vista de aplicação em problemas reais, neste trabalho, além de construir um sistema capaz de reconhecer qual interlocutor está presente em uma gravação, foi feita a segmentação, mesmo que de forma bastante simples, das sentenças em um diálogo, tentando definir qual interlocutor está falando em cada instante de tempo. A motivação desde o início

foi o processo de transcrição rica, que aparenta ser um próximo desafio a ser vencido pelos sistemas voltados ao processamento de voz.

Também nesse sentido existem outros pontos de evolução para o sistema construído. O processo de segmentação utilizado além de ser insuficiente para separar o momento exato em que os interlocutores se intercalam, acaba separando sentenças de um mesmo interlocutor sem necessidade. Isso é um problema pois pequenos áudios são muito mais difíceis de se avaliar do que áudios maiores, por conta disso sistemas mais avançados utilizam o processo de *Speaker Diarization* antes do reconhecimento (ANGUERA et al, 2010).

Todavia, mesmo que de forma imprecisa, o sistema construído, se integrado a um sistema de transcrição, poderia fazer um processo simples de transcrição rica. Como foi construído em Python, ele poderia ainda ser implementado em servidores na nuvem e ser acessado via REST APIs por outras aplicações. Esse é o modelo principal de utilização atual para sistemas comerciais de transcrição e processamento de linguagem natural, desenvolvidos por grandes empresas como IBM, Microsoft e Google, por exemplo.

6 REFERÊNCIAS

P. K. AJMERA, D. V. JADHAV, R. S. HOLAMBE, “Text-independent speaker identification using Radon and discrete cosine transforms based features from speech spectrogram”, *Pattern Recognition*, vol. 44, no. 10–11, 2011, pp. 2749-2759.

J.R.C. DE LARA, “A Method of Automatic Speaker Recognition Using Cepstral Features and Vectorial Quantization”. In: A. Sanfeliu, M.L. Cortés (eds) “Progress in Pattern Recognition, Image Analysis and Applications”. CIARP 2005. *Lecture Notes in Computer Science*, vol. 3773. Springer, Berlin, Heidelberg.

K. DAQROUQ, T. A. TUTUNJI, “Speaker identification using vowels features through a combined method of formants, wavelets, and neural network classifiers”, *Applied Soft Computing*, vol. 27, 2015, pp. 231-239.

P. MELL, T. GRANCE, “The NIST Definition of Cloud Computing”. National Institute of Standards and Technology (NIST), Gaithersburg, 2011. Disponível em: <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>. Acesso em: 9 de Abril de 2018.

D. A. REYNOLDS, "An overview of automatic speaker recognition technology," *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, FL, USA, 2002, pp. IV-4072-IV-4075.

D. A. REYNOLDS, R. C. ROSE, "Robust text-independent speaker identification using Gaussian mixture speaker models," in *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72-83, Jan 1995.

PEDREGOSA, *et al.* “Scikit-learn: Machine Learning in Python” , JMLR 12, pp. 2825-2830, 2011.

O. DAS, “Speaker Recognition” (ASR_report.pdf), 2016. Disponível em:<https://github.com/orchidas/Speaker-Recognition/blob/master/ASR_report.pdf>, Acessado em Março de 2018.

X. ANGUERA et al. “Speaker diarization: A review of recent research”. IEEE transactions on acoustics, speech, and signal processing, Institute of Electrical and Electronics Engineers (IEEE), 2010, pp.1.

X. LU, J. DANG. “An investigation of dependencies between frequency components and speaker characteristics for text-independent speaker identification”. Speech Communication, 50(4), pp 312–322, 2018.

J. RAMÍREZ, J. M. GÓRRIZ, J. C. SEGURA. “Voice Activity Detection. Fundamentals and Speech Recognition System Robustness”. Robust Speech Recognition and Understanding, InTech, 2007. Disponível em: <http://www.intechopen.com/books/robust_speech_recognition_and_understanding/voice_activity_detection__fundamentals_and_speech_recognition_system_robustness>

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). “Rich Transcription Evaluation”, 2016. Disponível em: <https://www.nist.gov/itl/iad/mig/rich-transcription-evaluation>. Acesso em: 03 abril 2019

E. JONES, E. OLIPHANT, P. PETERSON, *et al.* “SciPy: Open Source Scientific Tools for Python”, 2001.

S. VAN DER WALT, S. C. COLBERT, G. VAROQUAUX. “The NumPy Array: A Structure for Efficient Numerical Computation”, Computing in Science & Engineering, 13, 22-30, 2011.

F. PÉREZ, B. E. GRANGER. “IPython: A System for Interactive Scientific Computing”, Computing in Science & Engineering, 9, 21-29, 2007.

J. D. HUNTER. “Matplotlib: A 2D Graphics Environment”, Computing in Science & Engineering, 9, 90-95, 2007.

W. MCKINNEY. “Data Structures for Statistical Computing in Python”, Proceedings of the 9th Python in Science Conference, 51-56, 2010.

C. B. DO, S. BATZOGLOU. “What is the expectation maximization algorithm?. Nature biotechnology, v. 26, n. 8, p. 897, 2008.

A. VENTURINI, L. ZÃO, R. COELHO, "On Speech Features Fusion, α-Integration Gaussian Modeling and Multi-Style Training for Noise Robust Speaker Classification", IEEE/ACM Transactions on Audio, Speech and Language Processing, v. 22, n. 12, pp. 1951-1964, 2014.

APÊNDICE A

Lista de bibliotecas utilizadas na implementação computacional do sistema automático de interlocutores:

1. SciPy (E. JONES, E. OLIPHANT & P. PETERSON, 2001)
    - Utilização:
        - Cálculo de FFT
        - Cálculo de DCT
        - Janelamento Hamming
        - Leitura de .WAV
    - Disponível em: http://www.scipy.org/. Acesso em: 03 abril 2019.
2. NumPy (S. VAN DER WALT, S. C. COLBERT, G. VAROQUAUX, 2011)
    - Utilização:
        - Manipulação de vetores
    - Disponível em: https://www.numpy.org. Acesso em: 03 abril 2019.
3. MatPlotLib (J. D. HUNTER, 2007)
    - Utilização:
        - Geração de Gráficos
    - Disponível em: https://www.matplotlib.org. Acesso em: 03 abril 2019.
4. Pandas (W. MCKINNEY, 2010)
    - Utilização:
        - Manipulação de tabelas (Dataframes)
    - Disponível em: https://www.pandas.pydata.org. Acesso em: 03 abril 2019.
5. Scikit Learn (PEDREGOSA, *et al*, 2011)
    - Utilização:
        - Geração de GMMs
    - Disponível em: https://scikit-learn.org. Acesso em: 03 abril 2019.
6. IPython (F. PÉREZ, B. E. GRANGER, 2007)
    - Utilização:
        - Ambiente computacional interativo (Jupyter Notebooks)
    - Disponível em: https://ipython.org/notebook.html. Acesso em: 03 abril 2019.

ANEXO 1

Crônica *Cérebro eletrônico: o que sei é tão pouco* de Clarice Lispector

Decididamente estou precisando ir ao médico e pedir um remédio contra falta de memória. Ou melhor, uma amiga já me deu dois vidros de umas pílulas vermelhas contra falta de memória mas é exatamente minha falta de memória que me faz esquecer de tomá-las. Isso parece velha anedota, mas é verdade.

Tudo isso vem a propósito de eu simplesmente não me lembrar quem me explicou sobre o cérebro eletrônico. E mais: tenho em mãos agora mesmo uma fita de papel cheia de buraquinhos retangulares e essa fita é exatamente a da memória do cérebro eletrônico. Cérebro eletrônico: a máquina computadora poupa gente. Os dados da pessoa ou do fato estão registrados na linguagem do computador (furos em cartões ou fitas). Daí vão para a memória: que é outro órgão computador (outra máquina) onde os dados ficam guardados até serem pedidos.

Partindo deste princípio, chegamos ao definidor eletrônico: a partir de um desenho feito em um papel magnético a máquina (ou o cérebro) pode reproduzir em matéria de desenho. Isto é: entra o desenho e sai o objeto (cibernética, etc.) Há a experiência plástica, visual e também literária da reprodução (número e quantidade). A sensação é de apoio para o homem. Compensação do erro. Há possibilidade de você lidar com uma máquina e seus sensores como a gente gostaria de lidar com nosso cérebro (e nossos sensores), fora da gente mesmo e numa função perfeita.

Bem. Acabo de dizer tudo, mas mesmo tudo, o que sei a respeito do cérebro eletrônico. Devo inclusive ter cometido vários erros, sem falar nas lacunas que, se fossem preenchidas, esclareceriam melhor o problema todo.

Peço a quem de direito, que me escreva explicando melhor o cérebro eletrônico em funcionamento. Mas peço que use termos tão leigos quanto possível, não só para que eu entenda como para que eu possa transmiti-los com relativo sucesso aos meus leitores.

Quando penso que cheguei a falar no mistério, que continua mistério, do cérebro eletrônico, só posso dizer como a gente dizia lá em Recife: Virgem Maria!...

Mas o amor é mais misterioso do que o cérebro eletrônico e no entanto já ousei falar de amor. É timidamente, é audaciosamente, que ouso falar sobre o mundo.

ANEXO 2

Crônica *O Lixo* de Luís Fernando Veríssimo:

- Bom dia…
- Bom dia.
- A senhora é do 610.
- E o senhor do 612
- É.
- Eu ainda não lhe conhecia pessoalmente…
- Pois é…
- Desculpe a minha indiscrição, mas tenho visto o seu lixo…
- O meu quê?
- O seu lixo.
- Ah…
- Reparei que nunca é muito. Sua família deve ser pequena…
- Na verdade sou só eu.
- Mmmm. Notei também que o senhor usa muito comida em lata.
- É que eu tenho que fazer minha própria comida. E como não sei cozinhar…
- Entendo.
- A senhora também…
- Me chame de você.
- Você também perdoe a minha indiscrição, mas tenho visto alguns restos de comida em seu lixo. Champignons, coisas assim…
- É que eu gosto muito de cozinhar. Fazer pratos diferentes. Mas, como moro sozinha, às vezes sobra…
- A senhora... Você não tem família?
- Tenho, mas não aqui.
- No Espírito Santo.
- Como é que você sabe?
- Vejo uns envelopes no seu lixo. Do Espírito Santo.
- É. Mamãe escreve todas as semanas.
- Ela é professora?
- Isso é incrível! Como foi que você adivinhou?
- Pela letra no envelope. Achei que era letra de professora.
- O senhor não recebe muitas cartas. A julgar pelo seu lixo.
- Pois é…
- No outro dia tinha um envelope de telegrama amassado.
- É.
- Más notícias?

- Meu pai. Morreu.
- Sinto muito.
- Ele já estava bem velhinho. Lá no Sul. Há tempos não nos víamos.
- Foi por isso que você recomeçou a fumar?
- Como é que você sabe?
- De um dia para o outro começaram a aparecer carteiras de cigarro amassadas no seu lixo.
- É verdade. Mas consegui parar outra vez.
- Eu, graças a Deus, nunca fumei.
- Eu sei. Mas tenho visto uns vidrinhos de comprimido no seu lixo…
- Tranqüilizantes. Foi uma fase. Já passou.
- Você brigou com o namorado, certo?
- Isso você também descobriu no lixo?
- Primeiro o buquê de flores, com o cartãozinho, jogado fora. Depois, muito lenço de papel.
- É, chorei bastante, mas já passou.
- Mas hoje ainda tem uns lencinhos…
- É que eu estou com um pouco de coriza.
- Ah.
- Vejo muita revista de palavras cruzadas no seu lixo.
- É. Sim. Bem. Eu fico muito em casa. Não saio muito. Sabe como é.
- Namorada?
- Não.
- Mas há uns dias tinha uma fotografia de mulher no seu lixo. Até bonitinha.
- Eu estava limpando umas gavetas. Coisa antiga.
- Você não rasgou a fotografia. Isso significa que, no fundo, você quer que ela volte.
- Você já está analisando o meu lixo!
- Não posso negar que o seu lixo me interessou.
- Engraçado. Quando examinei o seu lixo, decidi que gostaria de conhecê-la. Acho que foi a poesia.
- Não! Você viu meus poemas?
- Vi e gostei muito.
- Mas são muito ruins!
- Se você achasse eles ruins mesmo, teria rasgado. Eles só estavam dobrados.
- Se eu soubesse que você ia ler…
- Só não fiquei com eles porque, afinal, estaria roubando. Se bem que, não sei: o lixo da pessoa ainda é propriedade dela?
- Acho que não. Lixo é domínio público.
- Você tem razão. Através do lixo, o particular se torna público. O que sobra da nossa vida privada se integra com a sobra dos outros. O lixo é comunitário. É a nossa parte mais social. Será isso?
- Bom, aí você já está indo fundo demais no lixo. Acho que…
- Ontem, no seu lixo…
- O quê?

- Me enganei, ou eram cascas de camarão?
- Acertou. Comprei uns camarões graúdos e descasquei.
- Eu adoro camarão.
- Descasquei, mas ainda não comi. Quem sabe a gente pode…
- Jantar juntos?
- É.
- Não quero dar trabalho.
- Trabalho nenhum.
- Vai sujar a sua cozinha?
- Nada. Num instante se limpa tudo e põe os restos fora.
- No seu lixo ou no meu?