LUCAS DEBATIN

DESENVOLVIMENTO E ANÁLISE DE DESEMPENHO DO RECONHECIMENTO OFF-LINE DE VOZ CONTÍNUO EM DISPOSITIVOS MÓVEIS

Itajaí (SC), fevereiro de 2019

UNIVERSIDADE DO VALE DO ITAJAÍ

CURSO DE MESTRADO ACADÊMICO EM COMPUTAÇÃO APLICADA

DESENVOLVIMENTO E ANÁLISE DE DESEMPENHO DO RECONHECIMENTO OFF-LINE DE VOZ CONTÍNUO EM DISPOSITIVOS MÓVEIS

por

Lucas Debatin

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada.
Orientador: Rudimar Luís Scaranto Dazzi, Dr.

Itajaí (SC), fevereiro de 2019

FOLHA DE APROVAÇÃO

Esta página é reservada para inclusão da folha de assinaturas, a ser disponibilizada pela Secretaria do Curso para coleta da assinatura no ato da defesa.

Dedico este trabalho aos meu pais Edesio e Olivia e a minha namorada Isabelle que, com muito amor e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

“So the problem is not so much to see what nobody has yet seen, as to think what nobody has yet thought concerning that which everybody sees”
Arthur Schopenhauer.

AGRADECIMENTOS

Primeiramente, agradeço a Deus que me deu forças e iluminou o meu caminho, ajudando-me a superar as dificuldades e os obstáculos nesta trajetória.

Ao meu orientador Rudimar Luís Scaranto Dazzi que incansavelmente tirou minhas dúvidas, direcionando meu foco e impondo limites para tarefas impossíveis.

Gostaria de agradecer também aos avaliadores Alejandro Rafael Garcia Ramirez e Aluizio Haendchen Filho, que contribuíram para a melhoria do trabalho por meio de sugestões e críticas. Além disso, agradeço de modo especial ao avaliador Aluizio pela sugestão de tema da pesquisa e todo o auxílio desde o início dos trabalhos.

Agradeço à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão da bolsa durante todo o período de realização deste mestrado.

Agradeço a esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Aos meus pais Edesio Debatin e Olivia Inês Hodecker Debatin por me ensinarem o valor e a importância do estudo e da instrução para a minha vida.

Agradeço a minha namorada Isabelle Angelica Erbs que em todos os momentos desta trajetória esteve ao meu lado, ajudando e me dando forças para continuar.

Aos amigos de pesquisa do LIA (Laboratório de Inteligência Aplicada): Alex Luciano Roesler Rese, Ivan Galvagno, João Victor Ribeiro, Rafael de Oliveira Schmitt, Rodrigo Lyra, Rudson Mendes, Thiago Felski Pereira, em especial ao Jonathan Nau e ao Fernando Concatto por sempre abdicar de seus afazeres para contribuir na resolução de problemas e dúvidas.

E, finalmente, de modo geral agradeço a todos que acreditaram na realização deste projeto e me incentivaram nesta longa trajetória.

DESENVOLVIMENTO E ANÁLISE DE DESEMPENHO DO RECONHECIMENTO OFF-LINE DE VOZ CONTÍNUO EM DISPOSITIVOS MÓVEIS

Lucas Debatin

Fevereiro / 2019

Orientador: Rudimar Luís Scaranto Dazzi, Dr.

Área de Concentração: Computação Aplicada

Linha de Pesquisa: Inteligência Aplicada

Palavras-chave: Reconhecimento de Voz Contínuo, Off-line, Dispositivos Móveis.

Número de páginas: 144

RESUMO

O reconhecimento de voz é uma forma de acessibilidade utilizada para executar tarefas com as mãos e os olhos livres em aparelhos eletrônicos, e isso é vantajoso independentemente do tipo de usuário. Atualmente, o reconhecimento de voz é realizado por meio de APIs que apresentam algumas limitações: (*i*) dependem de conexão com a internet; e (*ii*) muitas vezes são softwares proprietários, ou seja, há um custo para a aquisição de licenças de uso. Visando a solução desses problemas, o presente trabalho propôs o desenvolvimento do reconhecimento off-line de voz contínuo do português brasileiro para dispositivos móveis com Android. Inicialmente, realizou-se uma revisão sistemática da literatura, no qual identificou as técnicas mais utilizadas no reconhecimento de voz contínuo, obtendo o estado da arte da pesquisa. Durante a leitura completa dos artigos selecionados na revisão, percebeu-se o uso de bibliotecas para facilitar a implementação, tais como CMUSphinx, HTK e Kaldi. Para cada biblioteca foram criados 10 arquivos de configurações de treinamento, e as configurações que obtiveram as melhores métricas de avaliação (WER e SER) foram implementadas na versão desktop. Para o treinamento e testes, utilizou-se os corpora de voz do grupo FalaBrasil. A versão desktop foi responsável por realizar a análise de desempenho das bibliotecas em um computador desktop, isto é, foram verificados os percentuais de uso do processador e da memória. A biblioteca Kaldi obteve o melhor resultado da análise de desempenho, entretanto não foi possível implementá-la no aplicativo móvel devido a restrições de permissionamento. Por isso, utilizou-se a biblioteca CMUSphinx que apresentou resultados semelhantes ao Kaldi. Esse aplicativo móvel foi testado em 11 dispositivos, com diferentes versões do Android e configurações de hardware. Nos testes foram capturadas informações sobre o uso do processador, da memória e da bateria do dispositivo, realizando assim a análise de desempenho, que comprovou que o reconhecimento de voz contínuo pode ser executado com êxito em dispositivos móveis. O WER obtido pelo aplicativo foi de 9,6% no corpus de voz com vários locutores e 3,2% no corpus com apenas um locutor. Algumas sugestões de trabalhos futuros são apresentadas, entre elas a criação de um corpus de voz de vários locutores com maior duração, implementação da biblioteca Kaldi em dispositivos móveis, redução do custo computacional exigido pelas RNAs para serem utilizadas nos dispositivos móveis, testes em

dispositivos móveis com IOS, comparação dos resultados desse trabalho com o das APIs e testes em sistemas embarcados.

DEVELOPMENT AND PERFORMANCE ANALYSIS OF OFFLINE CONTINUOUS VOICE RECOGNITION IN MOBILE DEVICES

Lucas Debatin

February / 2019

Advisor: Rudimar Luís Scaranto Dazzi, Dr.

Area of Concentration: Applied Computer Science

Research Line: Applied Intelligence

Keywords: Continuous Speech Recognition, Offline, Mobile Devices.

Number of pages: 144

ABSTRACT

Voice recognition is a form of accessibility used to execute tasks leaving hands and eyes free in electronic devices, which is advantageous no matter the kind of user. Currently, voice recognition is performed through APIs that present some limitations: (*i*) depend on an Internet connection; and (*ii*) are often proprietary software, i.e. have a license that must be purchased to enable its usage. Aiming to solve these problems, this work proposed the development of an offline continuous voice recognition for the Brazilian Portuguese language for mobile devices running Android. Initially, a systematic review of the literature was performed, which identified the most frequently used techniques for offline voice recognition, obtaining the state of the art in the field. During the examination of the papers selected in the literature review, a frequent usage of libraries such as CMUSphinx, HTK and Kaldi was noticed. For each of these libraries, 10 training configuration files were created, and the settings that obtained the best values for the WER and SER evaluation metrics were implemented in a desktop version. For the training and validation steps, voice corpora from the FalaBrasil group were used. The desktop version was responsible for running a performance comparison between the libraries in a desktop computer, i.e. the usage ratios of the CPU and RAM were verified. The Kaldi library obtained the best result in the performance comparison; however, the library could not be implemented in the mobile application due to permission restrictions. For this reason, the CMUSphinx library was used, which presented results similar to Kaldi. The mobile application was tested in 11 devices, with differing versions of Android and hardware specifications. In the tests, information about CPU, RAM and battery usage were captured, thus achieving a performance analysis, which demonstrated that continuous voice recognition can be performed successfully in mobile devices. The mobile application obtained a WER of 9.6% in the voice corpus with multiple speakers and 3.2% in the corpus with a single speaker. A few suggestions for future research are presented, such as developing a voice corpus with multiple speakers with a longer duration, implementing the Kaldi library in mobile devices, reducing the computational resources demanded by the ANNs in mobile devices, testing in mobile devices running iOS, comparing the results of this work with proprietary APIs and testing with embedded systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
DCT	*Discrete Cosine Transform*
DNN	*Deep Neural Network*
FFT	*Fast Fourier Transform*
fMLLR	*feature space Maximum Likelihood Linear Regression*
G2P	*Grapheme to Phoneme*
GMM	Gaussian *Mixture Models*
HMM	*Hidden Markov Models*
HTK	*Hidden Markov Models Toolkit*
HZ	Hertz
IA	Inteligência Artificial
IHC	Interação Homem-Computador
LM	*Language Modeling*
LPCC	*Linear Predictive Cepstral Coefficients*
LSTM	*Long Short-Term Memory*
MFCC	*Mel Frequency Cepstral Coefficients*
ML	*Maximum Likelihood*
MLP	*Multilayer Perceptron*
NDK	Android *Native Development Kit*
OCSR	*Offline Continuous Speech Recognition*
PLN	Processamento de Linguagem Natural
PLP	*Perceptual Linear Prediction*
ReLU	*Rectified Linear Units*
RNA	Redes Neurais Artificiais
RNN	*Recurrent Neural Network*
SDK	*Software Development Kit*
SER	*Sentence Error Rate*
SRILM	SRI *Language Modeling Toolkit*
SCT	*Speech Corpus Treatment*
TanH	Tangente Hiperbólica
WER	*Word Error Rate*
xRT	*Real Time Factor*

LISTA DE SÍMBOLOS

α	Alfa, minúsculo (alfabeto grego)
Δ	Delta, maiúsculo (alfabeto grego)
Π	Pi, maiúsculo (alfabeto grego)
π	Pi, minúsculo (alfabeto grego)
Σ	Sigma, maiúsculo (alfabeto grego)
υ	Úpsilon, minúsculo (alfabeto grego)
φ	Fi, minúsculo (alfabeto grego)

SUMÁRIO

1 INTRODUÇÃO

Desde o surgimento dos computadores, pesquisadores buscam formas de tornar os sistemas computacionais mais inteligentes. Uma dessas formas é a compreensão da fala, que tem como objetivo fazer com que as máquinas sejam capazes de entender e se comunicar em linguagem natural. Para compreender a fala computacionalmente, é necessário converter a linguagem falada em texto; esse processo é chamado de reconhecimento de voz (ALENCAR, 2005; SILVA, 2010).

Para Yu e Deng, (2015), o reconhecimento de voz é uma importante tecnologia para melhorar a IHC (Interação Homem-Computador), visto que a voz é uma característica humana que a maioria das pessoas possui. De acordo com Benyon (2011), a IHC é responsável por garantir que os sistemas sejam fáceis de usar e de aprender, ou seja, utiliza métodos, diretrizes, princípios e padrões focados na melhora da usabilidade do sistema.

O reconhecimento de voz pode ser classificado em dois tipos: (*i*) palavras isoladas, que necessita que as sentenças sejam pronunciadas com pausas entre cada palavra, por isso são utilizados em sistemas simples e com vocabulário pequeno, tais como, sistemas de comando e controle por voz; e (*ii*) contínuo, tem como objetivo tornar a comunicação mais eficaz para os seres humanos, visto que reconhecem sentenças pronunciadas de forma natural, isto é, sem a necessidade de pausas entre as palavras (ALENCAR, 2005; HUANG; DENG, 2010; SILVA, 2010).

No contexto dos tipos de reconhecimento, este trabalho realiza o reconhecimento do tipo contínuo, que é mais complexo se comparado com o de palavras isoladas, pois deve ser capaz de lidar com todas as características e vícios de linguagem da fala. Além disso, muitas palavras podem ser substituídas ou não identificadas, pois nesse tipo de reconhecimento não há informação de onde começam e terminam determinadas palavras ou fonemas (ALENCAR, 2005; JURAFSKY; MARTIN, 2008; RUSSELL; NORVIG, 2004; SILVA, 2010; TEVAH, 2006).

Atualmente, as APIs (*Application Programming Interface*, em português: interface de programação de aplicação) Web Speech, Java Speech, Google Cloud Speech, Bing Speech, dentre outras, facilitam a implementação do reconhecimento de voz contínuo do português brasileiro em softwares e aplicações. Uma API permite que aplicações possam apenas requisitar serviços de programação, isto é, não é necessário se envolver com os detalhes de implementação (DEBATIN; HAENDCHEN FILHO; DAZZI, 2017; DEBATIN; HAENDCHEN FILHO; DAZZI, 2018; PERICO;

SHINOHARA; SARMENTO, 2014). Em outras palavras, as APIs para reconhecimento de voz recebem o áudio por meio de uma requisição, realizam o processamento do reconhecimento em servidores disponíveis na internet, e retornam o que foi falado no áudio em formato de texto. Entretanto, as APIs atualmente disponibilizadas não podem ser empregadas em qualquer tipo de aplicação, pois apresentam algumas limitações, que serão descritas na seção 1.1.

O termo off-line embora caracterize uma abordagem antiga, visto que atualmente a sociedade encontra-se na era da computação em nuvem, apresenta algumas vantagens: (*i*) não sofrem de problemas relacionados à latência e à largura de banda, pois os serviços em nuvem são disponibilizados por servidores remotos; (*ii*) não apresentam problemas relacionados ao compartilhamento do mesmo servidor, visto que os serviços de nuvem atendem a vários clientes, e se as requisições de um usuário comprometer o servidor, também poderá comprometer aplicativos de outros usuários; e (*iii*) não apresentam problemas de segurança, conformidade e regulamentares, pois os dados na nuvem podem ser acessíveis a terceiros (GROSSMAN, 2009).

Para implementar o reconhecimento de voz é necessário conhecer algumas de suas propriedades fundamentais, tais como: (*i*) as características do sinal da voz, que são as diversas informações sobre o locutor presentes no sinal de áudio; (*ii*) as formas para extração de características do áudio e para a decodificação, que são utilizados para gerar a melhor sequência textual a partir do sinal de áudio de entrada; e (*iii*) as métricas de avaliação, que são utilizadas para medir o desempenho do reconhecimento de voz. Além disso, com os avanços tecnológicos aumentou o interesse no desenvolvimento do reconhecimento de voz utilizando técnicas de aprendizagem profunda, tais como DNN (*Deep Neural Network*, em português: rede neural profunda) (DAHL et al., 2012; FERREIRA; SOUZA, 2017; SAMPAIO NETO, 2011; SILVA, 2010; VEIGA, 2013).

A etapa de decodificação manipula os modelos, acústico e de linguagem, para gerar a melhor sequência textual a partir das propriedades acústicas extraídas do áudio, independentemente do idioma a ser utilizado, entretanto é necessário possuir um corpus[1] de voz e de texto. Nesse trabalho, desenvolveu-se uma ferramenta para criação dos modelos acústicos do português brasileiro utilizando

[1] Corpus é um conjunto de documentos ou dados sobre determinado assunto (BAUER; AARTS, 2013).

os corpora de voz do grupo de pesquisa FalaBrasil[2] (FERREIRA; SOUZA, 2017; SILVA, 2010; VEIGA, 2013).

Dentro desse contexto, esse trabalho identificou e selecionou as principais técnicas que foram utilizadas na implementação do reconhecimento de voz contínuo. Na implementação utilizou-se as bibliotecas CMUSphinx, HTK (*Hidden Markov Models Toolkit*, em português: kit de ferramentas dos modelos ocultos de Markov) e Kaldi, porém cada biblioteca possui arquivos de configuração que podem ser editados, por isso realizou-se um estudo comparativo para encontrar a configuração com o melhor custo-benefício entre desempenho e precisão da taxa WER (*Word Error Rate*, em português: taxa de erro de palavras). Por fim, esse trabalho também comparou o processamento e uso de memória das bibliotecas em um computador desktop, e em seguida, implementou a que obteve os melhores resultados em um aplicativo Android para testar seu desempenho em diversos dispositivos móveis.

1.1 PROBLEMA DE PESQUISA

A primeira limitação que as APIs existentes apresentam é que nenhuma delas realiza o reconhecimento em modo off-line, ou seja, é necessário que o usuário esteja conectado à internet. Essa limitação é uma barreira no Brasil, pois aproximadamente 36% da população, com idade acima de 10 anos, não está conectada à internet. Isso afeta diversas pessoas que possuem capacidades limitadas e moram em localidades sem internet, uma vez que o reconhecimento de voz é um importante meio de acessibilidade. Além disso, também afeta empresas que possuem, em seus aplicativos móveis, o reconhecimento de voz via APIs, visto que em muitos casos não é possível distribuir o sinal *wireless* por toda a empresa, ou a empresa não oferece um dispositivo móvel com acesso à internet para o seu funcionário por motivos de confiança (DEBATIN; HAENDCHEN FILHO; DAZZI, 2018; IBGE, 2016).

Para que esse reconhecimento seja executado em modo off-line, é necessário realizar o processamento do reconhecimento de voz no próprio dispositivo móvel. Segundo Alencar (2005), o reconhecimento de voz apresenta uma alta complexidade computacional e requer uma grande

[2] Grupo do Laboratório de Processamento de Sinais (LaPS), da Universidade Federal do Pará, cujo o objetivo é a criação e disponibilização de ferramentas e recursos para reconhecimento de voz em português brasileiro.

quantidade de memória, e isso é uma limitação, dado que muitos *smartphones* e *tablets* possuem recursos de hardware limitados.

Outra limitação, é que as APIs são softwares proprietários, e em muitos casos o valor pago pela licença de uso se torna alto, visto que depende diretamente da quantidade de requisições que a API realiza. Essa limitação também afeta as empresas, pois é fundamental que o mesmo seja gratuito, devido ao grande número de requisições que é necessário (DEBATIN; HAENDCHEN FILHO; DAZZI, 2018).

Para solucionar os problemas dessa pesquisa foram levantadas as seguintes perguntas de pesquisa:

1. As técnicas atualmente utilizadas no reconhecimento de voz contínuo para extração de características do áudio e para implementação dos modelos, acústico e de linguagem, funcionam corretamente ao serem aplicadas em dispositivos móveis?

2. Além de funcionar corretamente, é possível que a solução desenvolvida também tenha um WER menor que 14% para o português brasileiro?

O valor máximo de 14% do WER é proveniente do Capítulo 3 , no qual o WER médio dos trabalhos selecionados na revisão sistemática da literatura foi de 14,01% para uma grande variedade de idiomas.

1.1.1 Solução Proposta

Nesse trabalho é apresentado uma solução para as limitações das APIs de reconhecimento de voz existentes no mercado, isto é, realizando o reconhecimento de voz contínuo de modo off-line e gratuito. Desta forma, foram adaptadas para o português brasileiro as técnicas de extração de características do áudio e de implementação dos modelos, acústico e de linguagem, selecionadas por meio de uma revisão sistemática da literatura. Visto que o reconhecimento desenvolvido é executado em dispositivos móveis, é necessário que o mesmo apresente um bom desempenho, por isso foram identificadas: (*i*) as configurações das bibliotecas que apresentam os melhores resultados; e (*ii*) a biblioteca que apresenta o melhor desempenho em um computador desktop. Essa solução procura comprovar as hipóteses para cada pergunta de pesquisa, apresentadas no Quadro 1.

Quadro 1 - Hipóteses de pesquisa para cada pergunta

Pergunta	**Hipótese nula (H_0)**	**Hipótese alternativa (H_1)**
Primeira	As técnicas utilizadas não podem ser aplicadas em nenhum modelo de dispositivo móvel, pois não há recursos computacionais suficientes.	Pelo menos um modelo de dispositivo móvel possui recursos computacionais suficientes.
Segunda	Com as técnicas que funcionam corretamente nos dispositivos móveis, não é possível ter um WER menor que 14% para o português brasileiro.	Com as técnicas que funcionam corretamente nos dispositivos móveis, é possível ter um WER menor que 14% para o português brasileiro.

1.1.2 Delimitação de Escopo

Para Gordillo (2013), uma das principais dificuldades do reconhecimento de voz é delimitar as características do reconhecimento de acordo com a necessidade da aplicação. Essas características aumentam ou diminuem a complexidade.

O reconhecimento de voz desenvolvido nesse trabalho possui as seguintes delimitações: (*i*) tipo contínuo; (*ii*) para o português brasileiro; e (*iii*) executado em modo off-line em dispositivos móveis com o sistema operacional Android.

O treinamento dos modelos acústico e de linguagem das bibliotecas foram realizados em computadores desktop, pois os mesmos possuem maiores recursos de hardware. Para cada configuração de biblioteca foi realizado um treinamento a fim de encontrar a que apresente o melhor custo-benefício entre o valor da taxa WER e o desempenho exigido.

Esse projeto possuirá duas versões para testes: (*i*) desktop, que foi utilizada para testar o desempenho das bibliotecas em um computador desktop, com o objetivo de selecionar a biblioteca que apresente os melhores resultados para ser implementada nos dispositivos móveis; e (*ii*) móvel, que foi utilizada para testar o desempenho da biblioteca selecionada em diversos dispositivos móveis com Android.

O treinamento e os testes foram realizados utilizando dois corpora de voz com: (*i*) apenas um locutor e com um vocabulário de 5.327 palavras; e (*ii*) vários locutores e com um vocabulário de 2.730 palavras. Ambos descobertos por meio do website do grupo FalaBrasil. Além disso, o reconhecimento desenvolvido é de vocabulário restrito, pois reconhece apenas as palavras que estão presentes nos corpora de voz.

1.1.3 Justificativa

O aumento do uso de interfaces adaptativas utilizando reconhecimento de voz, deve-se ao fato de que a fala é a forma mais natural de interação, pois torna-se mais rápido o uso e o acesso às informações nos softwares e aplicações (HEARST, 2011).

Os avanços nas técnicas de reconhecimento de voz viabilizaram o uso dessa tecnologia em diversas aplicações, sobretudo em dispositivos móveis. Embora muitas tarefas sejam melhor resolvidas com interfaces visuais (teclado, mouse, entre outros), a voz tem o potencial de ser uma interface mais natural, visto que pode proporcionar interação mesmo se o usuário estiver com as mãos e olhos ocupados ou se o usuário possuir capacidades limitadas, ou seja, é vantajosa independentemente do tipo de usuário, exceto para pessoas com afonia ou disfemia (JURAFSKY; MARTIN, 2008; SILVA, 2010; VEIGA, 2013; YU; DENG, 2015).

No Brasil, o decreto número 5.296 apresenta diretrizes de acessibilidade que, segundo a Casa Civil (2004), estabelece que todos os dispositivos devem oferecer recursos que exijam menor esforço físico e mental, minimizando sua desorientação ou sobrecarga cognitiva.

Em ambientes comerciais e industriais, além da dificuldade em distribuir o sinal *wireless* por toda a empresa, o uso da internet e da tecnologia móvel durante o horário de trabalho para fins pessoais é um grande problema. Além de perdas financeiras resultantes da redução da produtividade dos trabalhadores, isso também ameaça a segurança da rede e reduz a largura de banda organizacional. Por isso, se faz necessário desenvolver funções e aplicativos corporativos de modo off-line (VITAK; CROUSE; LAROSE, 2011).

Nesse contexto, e considerando as limitações das APIs de reconhecimento de voz, que essencialmente dependem da internet e, com sua tendência de manter-se indisponível de forma gratuita, é útil dispor desse recurso em modo off-line em dispositivos móveis.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver o reconhecimento off-line de voz contínuo do português brasileiro e analisar o seu processamento e uso de memória em dispositivos móveis.

1.2.2 Objetivos Específicos

1. Identificar e selecionar, por meio de uma revisão sistemática da literatura, as principais técnicas que são utilizadas na extração de características do áudio e na implementação dos modelos, acústico e de linguagem, do reconhecimento de voz contínuo;
2. Testar diversos valores dos parâmetros de configuração de treinamento das bibliotecas para obter o melhor custo-benefício entre desempenho e precisão;
3. Verificar o WER para o português brasileiro do reconhecimento off-line de voz contínuo;
4. Medir o processamento e uso de memória do reconhecimento off-line de voz contínuo em diversos dispositivos móveis com o sistema operacional Android.

1.3 METODOLOGIA

1.3.1 Metodologia da Pesquisa

Nesse projeto é aplicada a metodologia hipotético-dedutiva, visto que esse método procura evidências empíricas para rejeitar as hipóteses: (*i*) de que não é possível aplicar as técnicas mais utilizadas em nenhum dispositivo móvel; e (*ii*) de que não é possível ter um WER abaixo de 14% para o reconhecimento de voz do português brasileiro ao utilizar as técnicas que funcionam corretamente em dispositivos móveis.

Além disso, foram aplicados experimentos e análises para avaliar: (*i*) quais valores para os parâmetros de configuração de treinamento das bibliotecas possuem melhor desempenho, para serem aplicadas em dispositivos móveis; (*ii*) o WER do português brasileiro; e (*iii*) o desempenho do reconhecimento proposto em diferentes dispositivos móveis.

O método de pesquisa segue o princípio metodológico dos trabalhos científicos, visando contribuir com a comunidade científica. A seguir são exibidos os pontos de vista: de sua natureza, da forma de abordagem do problema e de seus objetivos.

Sob o ponto de vista de sua natureza

Sob o ponto de vista de sua natureza a pesquisa é aplicada, pois tem como objetivo gerar conhecimentos para aplicações práticas dirigidas à solução do problema proposto, isto é, busca

conhecimentos para desenvolver o reconhecimento de voz proposto com o menor processamento e uso de memória, e com uma boa precisão para o português brasileiro.

Sob o ponto de vista da forma de abordagem do problema

A pesquisa tem uma abordagem quantitativa para o problema, pois tem como objetivo: (*i*) verificar os melhores valores para os parâmetros de configuração de treinamento das bibliotecas; (*ii*) avaliar o WER no português brasileiro; e (*iii*) avaliar o desempenho do reconhecimento desenvolvido em diferentes dispositivos móveis.

Sob o ponto de vista de seus objetivos

Aplicou-se uma pesquisa exploratória, pois tem como objetivo proporcionar maior familiaridade com o problema, visto que envolve o levantamento bibliográfico, busca de autores que possuem experiências práticas e análise de exemplos do problema pesquisado. Além disso, foi utilizada a pesquisa explicativa, uma vez que essa procura documentar e analisar os resultados obtidos por meio dos testes de desempenho e dos valores da taxa WER.

1.3.2 Procedimentos Metodológicos

Os procedimentos metodológicos adotados nessa pesquisa são:

1. Revisão bibliográfica: essa etapa tem como objetivo proporcionar a fundamentação teórica necessária ao desenvolvimento da pesquisa;

2. Estado da arte: essa etapa tem como objetivo realizar uma revisão sistemática da literatura sobre o tema de pesquisa para identificar as técnicas que são utilizadas na extração de características do áudio e na implementação dos modelos acústico e de linguagem, do reconhecimento de voz contínuo;

3. Preparação dos corpora de voz: essa etapa tem como objetivo realizar a preparação dos dados dos corpora de voz para serem utilizados no treinamento dos modelos acústico e de linguagem de cada biblioteca;

4. Desenvolvimento: essa etapa tem como objetivo implementar, por meio do uso de bibliotecas, o reconhecimento off-line de voz contínuo;

5. Análise dos resultados: essa etapa tem como objetivo analisar os resultados obtidos, reconhecendo e explanando possíveis limitações, utilizando como base os valores do: (*i*) desempenho dos diversos parâmetros de configuração de treinamento das bibliotecas; (*ii*) WER do português brasileiro; (*iii*) desempenho das bibliotecas em um computador desktop; e (*iv*) processamento e uso de memória do reconhecimento desenvolvido em diferentes dispositivos móveis;

6. Conclusão: essa etapa tem como objetivo analisar as contribuições da pesquisa e apresentar sugestões relevantes de trabalhos futuros.

1.4 ESTRUTURA DA DISSERTAÇÃO

O trabalho está organizado em 6 capítulos correlacionados. O Capítulo 1 apresenta uma contextualização do tema proposto nesse trabalho, estabelecendo os resultados esperados, por meio da definição dos objetivos da pesquisa, e apresentando as delimitações do trabalho permitindo uma visão clara do escopo proposto.

No Capítulo 2 é demonstrada a fundamentação teórica sobre os assuntos relevantes ao tema, sendo eles: (*i*) reconhecimento de voz; (*ii*) extração de características do áudio; (*iii*) decodificador, focando nos assuntos relacionados aos modelos acústico e de linguagem; e (*iv*) métricas de avaliação.

O Capítulo 3 apresenta o resultado da revisão sistemática da literatura para obter o estado da arte do tema da pesquisa. Os resultados dessa revisão demostram as melhores abordagens atualmente utilizadas no reconhecimento de voz contínuo.

No Capítulo 4 descreve as etapas utilizadas no desenvolvimento do reconhecimento de voz proposto, sendo elas: (*i*) instalação das bibliotecas; (*ii*) preparação dos corpora linguísticos; (*iii*) implementação do treinamento; e (*iv*) implementação dos testes.

O Capítulo 5 é dedicado à discussão dos resultados obtidos no estudo comparativo dos valores: (*i*) do WER do português brasileiro dos diversos parâmetros de configuração de treinamento das bibliotecas; (*ii*) do desempenho das bibliotecas em um computador desktop; e (*ii*) do processamento e uso de memória do reconhecimento proposto em diferentes dispositivos móveis.

O Capítulo 6 apresenta as conclusões do trabalho, relacionando os objetivos do trabalho com os resultados obtidos. Além disso, são listadas as contribuições da pesquisa, as publicações realizadas durante o desenvolvimento da dissertação e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a base teórica dos temas abordados na dissertação. Além de servir como base para o desenvolvimento do trabalho, a fundamentação teórica esclarece os conceitos que auxilia no entendimento da pesquisa.

Os principais conceitos e elementos utilizados para auxiliar esse estudo são: reconhecimento de voz contínuo, extração de características do áudio, decodificador e métricas de avaliação, introduzidos, respectivamente, nas seções 2.1, 2.2, 2.3 e 2.4.

2.1 RECONHECIMENTO DE VOZ

O reconhecimento de voz é o processo de converter o sinal de voz analógico em sua representação textual, isto é, o texto gerado é composto pela sequência de palavras que foram identificadas a partir do sinal de entrada (RUSSELL; NORVIG, 2004; SILVA, 2010; VEIGA, 2013).

Segundo Gordillo (2013), esse reconhecimento processa a mensagem contida na onda acústica por meio do processo de classificação dos sinais em sequências de padrões, e requer conhecimentos de diferentes áreas, tais como fisiologia, acústica, processamento de sinal, linguística, computação, entre outros.

De acordo com Yu e Deng (2015), o reconhecimento de voz é uma área de pesquisa que está ativa por mais de cinco décadas. Antigamente, a voz era pouco utilizada na IHC, visto que a mesma não superava a eficiência e precisão do teclado e do mouse. Isso deve-se ao fato que a tecnologia da época não era boa o suficiente; porém, nos últimos anos, o poder computacional aumentou, possibilitando o treinamento de modelos maiores e mais complexos, reduzindo assim o WER (*Word Error Rate*, em português: taxa de erro de palavras).

Para Jurafsky e Martin (2008), a fala entre humano e computador é mais fácil de reconhecer do que a fala entre humanos, isto é, reconhecer a fala de humanos conversando com computadores, por exemplo, via comando de voz, tende a ser mais fácil do que reconhecer o discurso de dois seres humanos conversando entre si. Um dos motivos para essa facilidade é porque quando os humanos falam com computadores, eles tendem a simplificar bastante a fala, ou seja, falando mais devagar e com mais clareza.

Segundo Veiga (2013), o reconhecimento de voz simula o sistema de audição humana, e tem seu desempenho influenciado pelas características que afetam o sinal da fala, tais como o vocabulário utilizado, características dos locutores e ruídos de fundo. A subseção 2.1.2 irá aprofundar o conhecimento sobre esses fatores de complexidade para o reconhecimento de voz.

Além disso, para que o reconhecimento tenha um bom desempenho é importante conhecer as características do sinal da voz (ALENCAR, 2005); isso é base para a seção 2.2 na qual são descritos os métodos de extração dessas características. Além do mais, é fundamental descrever: os fatores que aumentam a complexidade do reconhecimento de voz e a sua estrutura básica.

2.1.1 Características da Voz

Nos seres humanos a voz é produzida da seguinte maneira: o ar proveniente dos pulmões provoca uma vibração ao passar pelas cordas vocais, porém esse som ainda é fraco e possui poucos harmônicos. Esse som então é amplificado ao passar pelas cavidades de ressonância, que são formadas pela laringe, faringe, fossas nasais e boca. Os movimentos da língua, lábios, mandíbula, dentes e palato também auxiliam na formação da voz (HUCHE; ALLALI, 1999; GORDILLO, 2013; MELO, 2011).

A taxa de amostragem[3] de um sinal de áudio é entre 8 e 16 kHz, e a precisão de cada medição é determinada pelo fator de quantização[4], que geralmente são de 8 a 12 bits. Um sistema de baixo custo, no qual a amostragem de 8 kHz com quantização de 8 bits exigiria quase a metade de um *megabyte* por minuto de fala. A quantidade de dados gerada durante a fala é grande, porém as características essenciais da voz mudam lentamente, isto é, requer uma menor quantidade de dados para representar as características mais importantes. Por isso, em sistemas de voz resumem-se as propriedades do sinal ao longo de intervalos chamados quadros, e cada quadro é representado por um vetor de características que será visto na seção 2.2 (PATRA, 2007; RUSSELL; NORVING, 2004).

[3] É o número de amostras de um sinal analógico, obtidas em um certo intervalo de tempo, para conversão em um sinal digital. O número de amostras dessa taxa deve ser o suficiente para reconstituir o sinal analógico original.
[4] É o número de bits utilizado na representação do sinal e determina a resolução do conversor.

Além disso, esse sinal de áudio possui diversas informações sobre o locutor, que são classificadas em: (*i*) baixo nível, que são os tons, intensidade, correlações espectrais, entre outros; e (*ii*) alto nível, que são variações na entonação, tais como dialeto, contexto, estilo de falar, estado emocional (por exemplo, dor e alegria), entre outros (GORDILLO, 2013; MÜLLER, 2006; PATRA, 2007; PLANNERER, 2005).

As palavras são a interpretação gráfica dos sons da voz, e a fonologia é o estudo dos fonemas (sons da linguagem) que compõem palavras, isto é, relaciona os sons da fala com as funções que eles exercem em um idioma. Os fonemas são as unidades sonoras mais simples da língua, e dividem-se em vogais, semivogais e consoantes. O fonema não apresenta significado próprio, entretanto é utilizado para diferenciar palavras (COPPIN, 2010; GORDILLO, 2013; MAIA, 1999; SILVEIRA, 1986). O Quadro 2 apresenta exemplos de fonemas de palavras do português.

Quadro 2 - Exemplo de fonema de palavras

Palavra	**Fonema**
Hoje = 4 letras	/o/ /j/ /e/ = 3 fonemas
Táxi = 4 letras	/t/ /a/ /k/ /s/ /i/ = 4 fonemas
Elevador = 8 letras	/e/ /l/ /e/ /v/ /a/ /d/ /o/ /r/ = 8 fonemas

O fonema é o elemento básico do som, que é caracterizado pelo fato de que duas palavras são diferentes se pelo menos um de seus elementos básicos diferirem. Por exemplo, as frases “ontem comi um pão no café da manhã” e “ontem comi um cão no café da manhã”, apresentam uma diferença sonora que faz mudar o sentido da palavra, apenas mudando um de seus elementos básicos (ALENCAR, 2005; GORDILLO, 2013).

Uma descoberta importante da fonologia é que todos os idiomas possuem um repertório limitado de aproximadamente 40 ou 50 sons, chamados de fones. Um fone é o som que corresponde a uma única vogal ou consoante, mas existem algumas complicações, pois “lh” e “rr” possuem fones singulares. Além do mais, algumas letras produzem fones diferentes, por exemplo o som da letra “c” da palavra “casa” é diferente se comparado com o da palavra “céu” (RUSSELL; NORVIG, 2004).

2.1.2 Fatores de Complexidade

O entendimento de todas as características da voz humana por parte do computador é uma tarefa difícil e complexa de se realizar, porém alguns problemas que afetam a precisão do

reconhecimento vêm sendo reduzidos e eliminados, com pesquisas na área e com o surgimento de novas tecnologias (SILVA, 2010).

Além do tipo de reconhecimento de voz (palavras isoladas ou contínuo) existem outros fatores que tornam o reconhecimento de voz uma tarefa difícil e complexa, como por exemplo o tamanho do vocabulário, variabilidade, tipo de locutor, presença de ruído e limitações do corpus.

Quanto maior for o tamanho do vocabulário, maior é a probabilidade de erro do reconhecimento de voz, por dois motivos: (*i*) sistemas com grandes vocabulários possuem muitas palavras que são homófonas, isto é, apresentam a mesma pronúncia, por exemplo “sessão/cessão”, “mais/mas”, “consertar/concertar”, entre outros; e (*ii*) o tempo de treinamento e a quantidade de memória utilizada aumenta linearmente com o aumento do vocabulário, ou seja, é proporcional ao número de palavras. Em comparação, os sistemas com vocabulários de tamanho reduzido apresentam ótimos resultados, pois são menos suscetíveis a erros. Vocabulários considerados pequenos possuem menos de 20 palavras, já os grandes possuem mais de 20 mil palavras (ALENCAR, 2005; GORDILLO, 2013; JURAFSKY; MARTIN, 2008; SILVA, 2010).

Outra complexidade está relacionada à variabilidade de fatores internos ou externos do áudio. Os fatores internos são as diferenças: (*i*) de uma pessoa para outra, que está relacionado a diversidade de gênero, idade e sotaques dos locutores, isto é, cada locutor apresenta características diferentes de fala; e (*ii*) em um mesmo indivíduo, que ocorrem devido a distintas situações físicas e psicológicas, tais como estado emocional, contexto da conversação, inclusão de ruídos do ambientes, e além disso, a mesma palavra, se pronunciada várias vezes, pode apresentar diferentes formas de onda. Já os fatores externos estão relacionados ao modo de transmissão do sinal acústico, pois pode haver diferenças entre caraterísticas de microfones, linhas de transmissão, entre outros. Na fala nenhum som é exatamente idêntico a outro (ALENCAR, 2005; GORDILLO, 2013; HUCHE; ALLALI, 1999; MAIA, 1999; SILVA, 2010).

Os sistemas de reconhecimento de voz podem ser classificados como dependentes ou independentes de locutor. No dependente o sistema é treinado somente para um locutor, sendo assim, reconhece apenas o locutor para o qual foi treinado, e por esse motivo apresenta uma boa taxa de acerto. Já os sistemas independentes de locutor são capazes de reconhecer a fala de qualquer locutor, mesmo aquele que não participou do treinamento do sistema (JURAFSKY; MARTIN, 2008; SILVA, 2010).

Treinar modelos acústicos que modelam toda a variabilidade do sinal de voz necessita de um corpus com uma grande variedade e quantidade de amostras, isto é, modelar corretamente a maioria das variações acústicas aumenta a confiabilidade dos resultados do sistema de reconhecimento de voz de amplo vocabulário e independente do locutor. Entretanto, para o português brasileiro a disponibilidade de corpora de voz de grande porte é uma das principais limitações, pois os que existem atualmente possuem poucas horas de duração. Além disso, para treinar um modelo que utiliza um enorme corpus é necessário possuir computadores com elevada capacidade de armazenamento e de processamento (VEIGA, 2013; GORDILLO, 2013; SILVA, 2010).

Uma das maiores dificuldades existentes no reconhecimento de voz é a presença de ruído no sinal da voz. Os ruídos do ambiente, tais como vozes de outros locutores, sons de equipamentos, e, até mesmo, os provocados pelo próprio locutor (tosses, espirros, suspiros, respiração forte, entre outros), são inevitáveis e influenciam no conteúdo do sinal, fazendo com que a mensagem que o emissor quer transmitir seja diferente à que o receptor vai ouvir (ALENCAR, 2005; GORDILLO, 2013; SILVA, 2010).

Durante o reconhecimento, quanto menor a presença de ruído no sinal de voz mais fácil é para o decodificador identificar o que foi dito, pois reconhecer a fala de um locutor em um escritório silencioso é muito mais fácil do que reconhecer a fala de um locutor em um carro na estrada com a janela aberta (JURAFSKY; MARTIN, 2008; SILVA, 2010).

Os ruídos podem ser do tipo: (*i*) estacionário, se possuir uma densidade espectral que não varia com o tempo, como o caso do ruído branco, o qual tem a sua potência distribuída uniformemente no espectro de frequência, gerando um espetro de potência plana, e (*ii*) não estacionário, se suas densidades espectrais mudam com o tempo, por exemplo, as vozes espontâneas, efeitos da respiração, entre outros. Além desses tipos de ruído, existe um ruído chamado de distorção, esse se une com o sinal de voz no domínio do tempo (GORDILLO, 2013).

Segundo Ferreira e Souza (2017), os sistemas de reconhecimento de voz atuais não possuem precisão absoluta, visto que é praticamente impossível que um sistema seja perfeito a ponto de abranger todos os fatores de complexidade citados nessa seção.

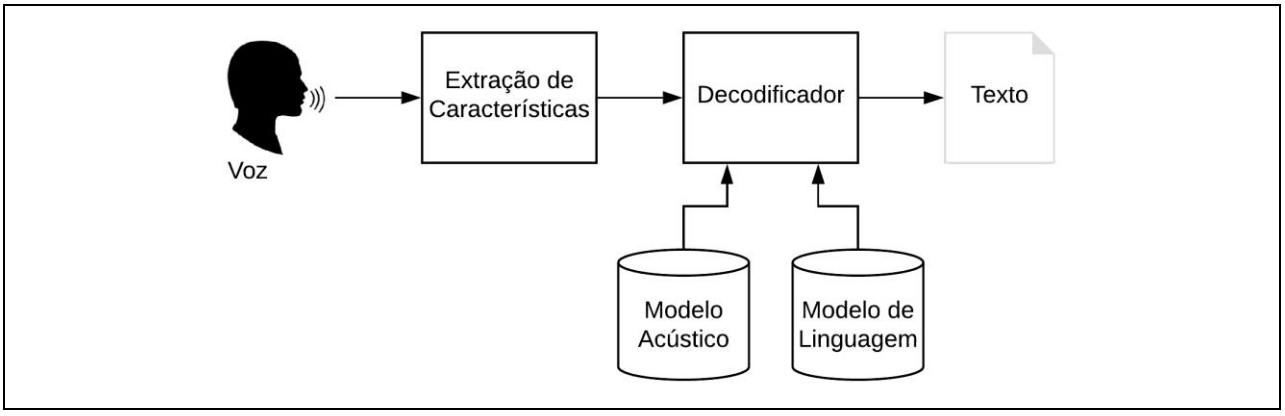
2.1.3 Estrutura do Reconhecimento de Voz Contínuo

A estrutura básica dos sistemas de reconhecimento de voz, representada na Figura 1, é dividida em duas etapas:

1. Extração de características: aplica algoritmos no sinal de voz da entrada a fim de representá-lo de uma forma mais compacta e robusta. Para isso, é necessário converter o sinal analógico em uma representação digital, e, além disso, é necessário determinar o que é silêncio/ruído e o que de fato é informação de voz útil, reduzindo assim a quantidade de informação de voz e diminuindo, consequentemente, o custo computacional (MELO, 2011; VEIGAS, 2013).

2. Decodificador: procura a melhor sequência de palavras num conjunto de hipóteses possíveis dada a representação de características do sinal de voz. Essa etapa utiliza os modelos: (*i*) acústico, que transforma o sinal que está sendo processado em palavras e sentenças; e (*ii*) de linguagem, que é responsável por caracterizar o idioma e condicionar a combinação de palavras descartando frases gramaticalmente incorretas (FERREIRA; SOUZA, 2017; SILVA, 2010; VEIGAS, 2013).

Figura 1 - Estrutura do reconhecimento de voz



Fonte: Adaptado de Sampaio Neto (2011); Silva (2010); Veiga (2013).

Nas seções 2.2 e 2.3 serão fundamentadas as etapas: (*i*) extração de características, no qual descreve o funcionamento do MFCC (*Mel Frequency Cepstral Coefficients*, em português: coeficientes *cepstrais* de frequência Mel); e (*ii*) decodificador, no qual apresenta os modelos acústico e de linguagem.

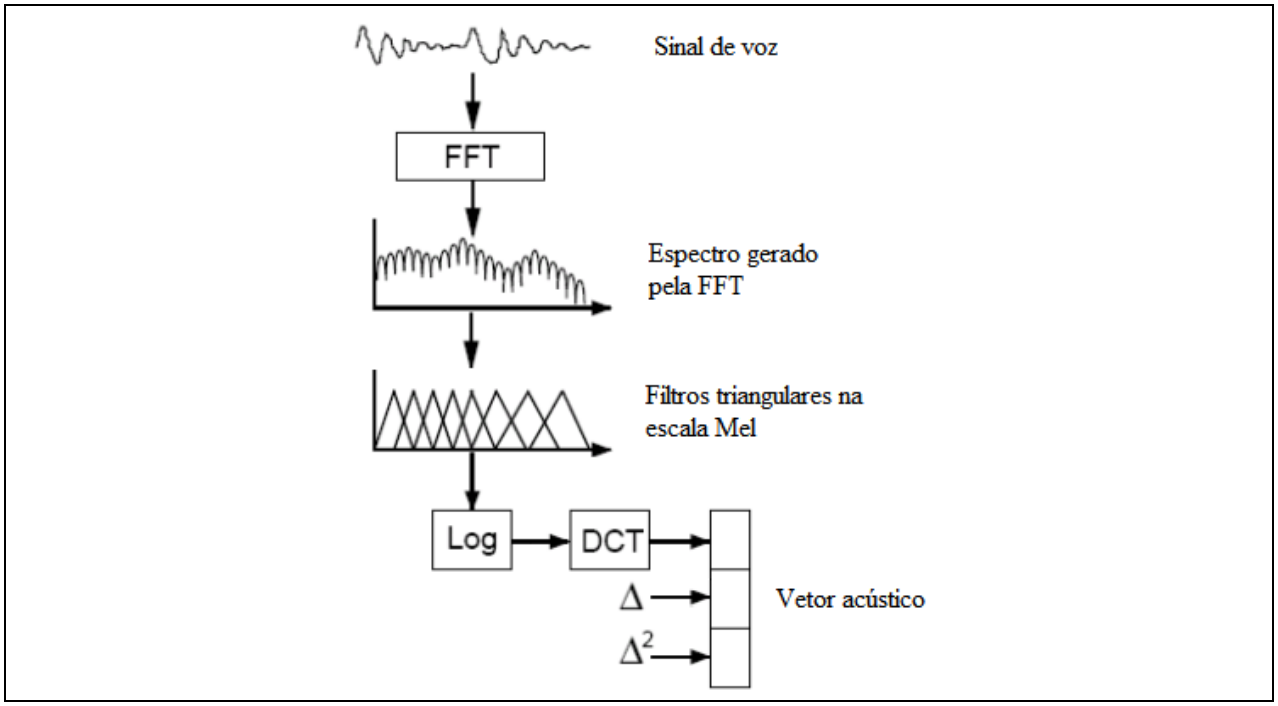
2.2 EXTRAÇÃO DE CARACTERÍSTICAS DO ÁUDIO

A extração e seleção da melhor representação paramétrica dos sinais acústicos é uma tarefa importante do sistema de reconhecimento de voz, visto que afeta significativamente no seu desempenho. Além disso, é importante focar na extração de características, pois um dos problemas do reconhecimento de voz é obter informações úteis do áudio. Pode-se citar como exemplo, os seguintes métodos de extração de características: MFCC, LPCC (*Linear Predictive Cepstral Coefficients*, em português: coeficientes *cepstrais* preditivos lineares), PLP (*Perceptual Linear Prediction*, em português: previsão linear perceptual), entre outros. Os coeficientes dinâmicos (energia, delta e delta-delta) também podem ser utilizados em conjunto com os anteriores (GORDILLO, 2013; TIWARI, 2010; VEIGA, 2013).

Apesar de existirem muitas representações de características, o MFCC é o mais utilizado para o reconhecimento de voz (JURAFSKY; MARTIN, 2008; TIWARI, 2010; VEIGA, 2013). Na revisão sistemática da literatura (Capítulo 3) a maioria dos trabalhos selecionados também utilizavam esse método de extração.

O MFCC faz uma análise de características espectrais de curto prazo, baseando-se no uso do espectro da voz convertido em uma escala de frequências denominada Mel. Essa escala visa transcrever as características perceptíveis pelo ouvido humano, ou seja, as de baixo nível, pois são foneticamente mais importantes para a percepção humana do que as de alto nível. Em outras palavras, a escala Mel tem como objetivo imitar o comportamento dos ouvidos humanos (GORDILLO, 2013; PATRA, 2007; SAUNDADE; KURLE, 2014). A Figura 2, representa as etapas da extração de características do MFCC.

Figura 2 - Etapas da extração de características do MFCC



Fonte: Tevah (2006).

Os sinais da voz são representados analogicamente, por isso é necessário realizar a conversão analógico-digital para que os computadores possam representar e processar esses sinais. Portanto, o reconhecimento de voz pode ser considerado um exemplo de processamento de sinais digitais (DINIZ; SILVA; NETTO, 2014; SAUNDADE; KURLE, 2014).

Primeiramente, é necessário aplicar o filtro pré-ênfase, que é representado pela equação (1), no sinal da voz para compensar a atenuação dos componentes de alta frequência. No qual *z* é o sinal de áudio e α é a frequência de corte com valores variando entre 0,95 e 0,98 (GORDILLO, 2013; VEIGA, 2013).

$$H(z) = 1 - \alpha z^{-1} \tag{1}$$

Em seguida, é necessário determinar com precisão os pontos de início e fim das palavras, isto é, remover o silêncio do sinal de áudio visando reduzir o tempo de cálculo, e após isso realizar a segmentação, dividindo o sinal de voz em quadros. Para suavizar cada quadro é aplicado a janela Hamming, pois a segmentação apresenta o problema de descontinuidade no início e no final devido ao fato de começar e terminar bruscamente, para isso são reduzidos os valores do sinal para zero nos

limites de cada quadro. A equação (2) representa o cálculo da janela Hamming, no qual *n* é o sinal de áudio de cada quadro (GORDILLO, 2013; JURAFSKY; MARTIN, 2008; VEIGA, 2013).

$$w(n) = \left\{ \begin{array}{rl} 0{,}54 - 0{,}46cos\left(\frac{2\pi n}{N-1}\right) & \text{para} \quad 0 \leq n \leq N-1, \\ 0 & \text{para} \quad \text{caso contrário.} \end{array} \right\} \tag{2}$$

Após a etapa de "janelamento" do sinal, é necessário converter o sinal da voz do seu domínio original para uma representação no domínio da frequência. Segundo Diniz, Silva e Netto (2014), para isso pode-se utilizar o DFT (*Discrete Fourier Transform*, em português: transformada discreta de Fourier), porém esse apresenta uma limitação que está associada ao grande número de operações aritméticas envolvidas no cálculo para longas amostras de sinal. Esse problema foi parcialmente resolvido com a criação de algoritmos eficientes para o DFT, conhecidos como FFT (*Fast Fourier Transform*, em português: transformação rápida de Fourier) representado pela equação (3). Além disso, o método FFT permite obter o mesmo resultado em menor tempo e complexidade, quando comparado com o DFT.

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{nk} \tag{3}$$

Em que, *x(n)* é o sinal da voz, *N* é o número de amostras na potência de dois, *k* é a frequência. A equação (3) também apresenta os elementos *x(n)* dos índices pares e ímpares do sinal.

Os resultados da FFT são informações sobre a quantidade de energia em cada banda de frequência. Como já visto, a audição humana é menos sensível em frequências mais altas, então nessa etapa ocorre apenas a extração dos recursos essenciais. Para isso, aplica-se um banco de filtros à potência espectral, que é formado por filtros triangulares, espaçados de acordo com a escala de frequência Mel, representada pela equação (4), no qual *f* é a frequência de corte (GORDILLO, 2013; JURAFSKY; MARTIN, 2008).

$$mel(f) = 1127ln\left(1 + \frac{f}{700}\right) \tag{4}$$

Após esse cálculo é realizado o *log* de cada um dos valores do espectro Mel, visto que a resposta humana ao nível do sinal é logarítmica. O uso do *log* faz com que o recurso seja menos

sensível a variações na entrada, como por exemplo as variações de energia devido à proximidade do microfone em relação a boca do locutor (JURAFSKY; MARTIN, 2008).

Na etapa final, os coeficientes do espectro do *log* Mel são convertidos novamente no domínio do tempo usando a DCT (*Discrete Cosine Transform*, em português: transformação discreta de cosseno). Utilizou-se o DCT porque a maior parte da energia é concentrada em poucos coeficientes, e isso é uma propriedade muito importante quando aplicada a sinais de voz. A equação (5) calcula o DCT, no qual *N* é o comprimento do sinal *x(n)* (DINIZ; SILVA; NETTO, 2014; SAUNDADE; KURLE, 2014).

$$C(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) cos \frac{\pi \left(n + \frac{1}{2}\right) k}{N}, \quad \text{para } 0 \leq k \leq N-1 \tag{5}$$

Para calcular o valor de α(*k*), utiliza-se a equação (6).

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{para } k = 0 \\ \sqrt{\frac{2}{N}} & \text{para } 1 \leq k \leq N-1 \end{cases} \tag{6}$$

Um vetor acústico de MFCC é computado para cada quadro. Geralmente esse vetor apresenta 39 elementos, que são: (*i*) estáticos: 12 coeficientes *cepstrais* extraídos do MFCC; (*ii*) dinâmicos: 1 coeficiente de energia, 13 coeficientes de velocidade (delta) e 13 coeficientes de aceleração (delta-delta) (GORDILLO, 2013; JURAFSKY; MARTIN, 2008; VEIGA, 2013).

Os coeficientes dinâmicos são utilizados para captar as mudanças temporais bruscas presentes no espectro. A energia de um quadro é calculada por meio da soma ao longo do tempo da capacidade das amostras no quadro. A equação (7) representa o cálculo da energia, em que *x* é um sinal em uma janela da amostra de tempo t_1 para a amostra de tempo t_2 (JURAFSKY; MARTIN, 2008).

$$Energia = \sum_{t=t_1}^{t_2} x^2[t] \tag{7}$$

Para o cálculo do coeficiente delta (Δ) é utilizado regressão linear sobre um quadro, ou seja, calculando a diferença entre os quadros anteriores e posteriores. O valor delta *d(t)* para um

determinado valor *cepstral c(t)* no tempo *t* pode ser estimado pela equação (8) (GORDILLO, 2013; JURAFSKY; MARTIN, 2008).

$$d(t) = \frac{c(t+1) - c(t-1)}{2} \tag{8}$$

Os parâmetros de segunda ordem, chamados delta-delta (Δ^2), são adquiridos replicando a derivada sobre os resultados obtidos na primeira derivação que foi calculada na equação (8) (GORDILLO, 2013).

2.3 DECODIFICADOR

No decodificador, a sequência textual é concebida pelo modelo acústico e corrigida pelo modelo de linguagem. Esses modelos trabalham em conjunto e um depende do outro, pois, como já visto, existem palavras homófonas e é praticamente impossível para o modelo acústico diferenciá-las, por isso utiliza-se o modelo de linguagem (FERREIRA; SOUZA, 2017; JURAFSKY; MARTIN, 2008; RUSSELL; NORVIG, 2004; SILVA, 2010; VEIGA, 2013).

Essa etapa necessita de treinamento, visto que é necessário gerar modelos acústicos e de linguagem que forneçam bons resultados e que sejam adequados ao contexto de aplicação, mas para isso é necessário possuir grandes corpora de áudio e de linguagem. No treinamento do modelo acústico, os vetores de características do sinal da voz são utilizados para determinar um padrão que melhor represente cada frase do corpus. Já o treinamento de linguagem é utilizado para modelar e compreender as regras gramaticais. Na etapa de teste, após o treinamento de todos os modelos, utiliza-se a extração de características para converter o sinal de entrada em parâmetros e o decodificador para encontrar a melhor sentença (MELO, 2011; SAMPAIO NETO, 2011; VEIGA, 2013).

Para solucionar os problemas complexos do reconhecimento de voz são utilizados, nos modelos acústicos e de linguagem, algumas técnicas de IA (Inteligência Artificial), tais como HMM (*Hidden Markov Models*, em português: modelos ocultos de Markov), RNA (Redes Neurais Artificiais) e PLN (Processamento de Linguagem Natural). Segundo Russell e Norvig (2004), a IA surgiu logo após a segunda guerra mundial (1956), e tenta não apenas compreender, mas também construir entidades inteligentes. Para Coppin (2010, p.4), “IA envolve utilizar métodos baseados no

comportamento inteligente de humanos e outros animais para solucionar problemas complexos". Já para Luger (2013, p.1) "IA pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente".

Essa seção está estruturada da seguinte forma: na subseção 2.3.1 será descrito o modelo acústico, juntamente com a descrição da RNA e do HMM; e na subseção 2.3.2 será descrito o modelo de linguagem com base no PLN e com foco no modelo n-grama.

2.3.1 Modelo Acústico

O modelo acústico é o componente do sistema de reconhecimento de fala responsável por definir, a partir das características extraídas do áudio de entrada, a sequência mais provável de: palavras, se o reconhecimento for do tipo isolado; ou fonemas, no caso de reconhecimento de voz contínuo (PERICO; SHINOHARA; SARMENTO, 2014; SILVA, 2010). Nessa dissertação o modelo acústico é baseado em fonemas, pois foi desenvolvido o reconhecimento de voz contínuo.

O dicionário fonético é indispensável no processo de treino de modelos acústicos. Esse dicionário é uma lista de palavras que são possíveis de reconhecer, com suas respectivas pronúncias expressas em uma sequência de fonemas. Isto é, tem como objetivo converter as palavras em fonemas, e ao contrário, convertendo de fonema para texto (FERREIRA; SOUZA, 2017; JURAFSKY; MARTIN, 2008; SILVA, 2010; TEVAH, 2006; VEIGA, 2013).

Nesse modelo são utilizados dois tipos de classificadores: (*i*) RNA, que resolve o problema do reconhecimento de padrões da sequência de características, que diferenciam o sinal de voz, por meio de um processo de aprendizagem; (*ii*) modelo estatístico HMM, que é utilizado para solucionar o problema do treinamento da RNA no reconhecimento de voz, pois pode combinar as probabilidades acústicas produzidas pela rede com as probabilidades de transição de estado do HMM, ou seja, esses geralmente são utilizados em conjunto, combinando suas potencialidades, gerando assim sistemas híbridos bastante robustos para a tarefa de reconhecimento de voz contínuo (ALENCAR, 2005; GRAVES; JAITLY; MOHAMED, 2013; HAYKIN, 2001; MÜLLER, 2006; SAMPAIO NETO, 2011; SILVA, 2010). Ambos os classificadores utilizados no modelo acústico para o reconhecimento dos padrões da fala serão apresentados nos tópicos a seguir.

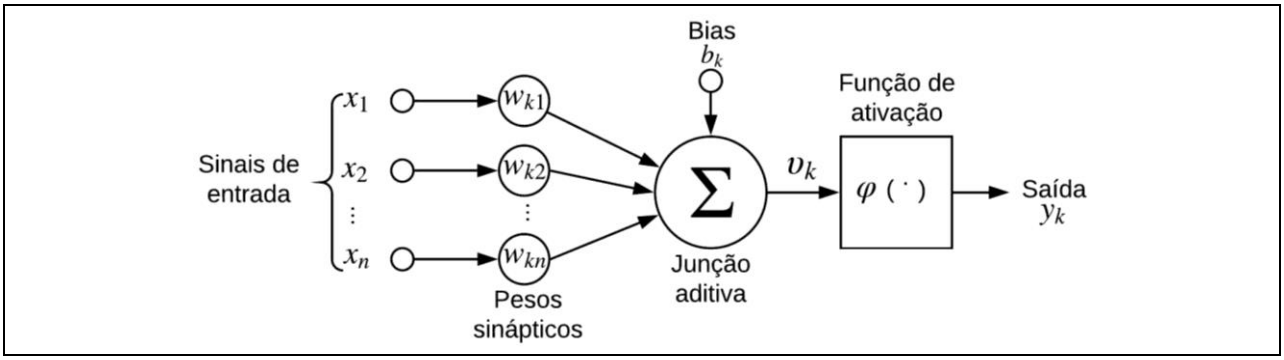
2.3.1.1 Redes Neurais Artificiais

A RNA tem como principal fonte de inspiração as redes neurais biológicas, visto que se assemelha ao cérebro humano em dois aspectos básicos: (*i*) o conhecimento é adquirido pela rede a partir de seu ambiente, por intermédio do processo de aprendizagem; e (*ii*) as forças de conexão entre neurônios são utilizadas para armazenar o conhecimento adquirido. As RNAs são utilizadas para solucionar problemas complexos, tais como: reconhecimento de padrões (diagnósticos médicos, previsões no mercado financeiro, entre outros), processamento de sinais e imagens, sistemas de controle, classificação, entre outros (HAYKIN, 2001; SPÖRL; CASTRO; LUCHIARI, 2011).

As redes neurais também são conhecidas como sistemas conexionistas, e isso faz com que a informação e o processamento da rede sejam distribuídos e paralelos, pois os neurônios processam as suas entradas simultaneamente e independentemente (LUGER, 2013; RUSSELL; NORVIG, 2004).

Essas redes são compostas por cinco elementos básicos, sendo eles: (*i*) sinais de entrada (x_n): são dados que podem vir do ambiente ou da ativação de outros neurônios; (*ii*) pesos sinápticos (w_{kn}): são conexões entre os neurônios da rede, que possuem um peso para representar a força da conexão, podendo ser negativo ou positivo; (*iii*) *bias* (b_k): valor aplicado externamente a cada neurônio e tem o efeito de aumentar ou diminuir a entrada da função de ativação; (*iv*) junção aditiva (υ_k): realiza as somas dos sinais de entrada, ponderados pelos pesos sinápticos; (*v*) função de ativação (φ): restringe a amplitude do valor de saída de um neurônio; e (*vi*) sinal de saída (y_k): são os valores dos neurônios da camada de saída e são constituídos por meio da resposta global da rede para o padrão de ativação fornecido pelos valores de entrada. O índice *n* representa os neurônios da rede e o índice *k* representa um neurônio em questão (COPPIN, 2010; HAYKIN, 2001; LUGER, 2013; RUSSELL; NORVIG, 2004). A Figura 3 representa visualmente todos os componentes do neurônio artificial descritos no parágrafo.

Figura 3 - Modelo de neurônio artificial



Fonte: Haykin (2001).

A RNA é representada por valores numéricos, com isso pode-se dizer que o processo de aprendizado é o resultado de operações numéricas (LUGER, 2013; RUSSELL; NORVIG, 2004).

Segundo Haykin (2001), a função de ativação é restritiva, pois limita o intervalo do valor da saída do neurônio em um valor finito, geralmente, esse intervalo é entre [0,1] ou [-1,1]. No reconhecimento de voz as funções mais utilizadas são: (*i*) sigmoide é definida como uma função estritamente crescente que exibe um balanceamento adequado entre comportamento linear e não-linear, além disso, essa função é contínua, o que permite uma medida mais precisa de erro, visto que mapeia a maioria dos valores para regiões próximas a 0 ou 1; (*ii*) TanH (Tangente Hiperbólica) é semelhante a função sigmoide, porém apresenta o intervalo entre -1 a 1, e é utilizada para acelerar o processo de treinamento/aprendizagem, necessitando de menos iterações para encontrar uma solução.; e (*iii*) ReLU (*Rectified Linear Units*, em português: unidades lineares retificadas) é uma função de ativação convencionalmente usada nas camadas ocultas para melhorar o treinamento do DNN, e funciona por valores de limiar em 0, isto é, gera 0 quando $x < 0$ e gera uma função linear com inclinação de 1 quando $x \geq 0$ (AGARAP, 2018; COPPIN, 2010; DIMITRIADIS; BOCCHIERI, 2015; HAYKIN, 2001; KIPYATKOVA; KARPOV, 2017; LUGER, 2013; MIKOLOV et al., 2010; VEIGA, 2013; YU; DENG, 2015).

Além disso, geralmente utiliza-se a função *softmax* na camada de saída, em conjunto com outras funções de ativação presentes nas camadas ocultas. Essa função auxilia na solução do aprendizado profundo de problemas de classificação, por isso é conhecido como uma função de classificação (AGARAP, 2018; HAYKIN, 2001).

Uma rede neural pode ser caracterizada por dois aspectos principais: (*i*) método de determinação dos pesos das conexões (algoritmo de treinamento ou aprendizado); e (*ii*) padrão de conexões entre as unidades (arquitetura) (HAYKIN, 2010). Nessa dissertação serão utilizadas a aprendizagem supervisionada e as arquiteturas MLP (*Multilayer Perceptron*, em português: *perceptron* multicamadas) e RNN (*Recurrent Neural Network*, em português: rede neural recorrente).

Algoritmo de aprendizagem é o procedimento utilizado para modificar os pesos sinápticos da rede. Esse peso associado a cada conexão pode ser alterado em resposta a conjuntos específicos de entradas e de eventos, visto que se a saída da rede estiver incorreta, os pesos serão ajustados para melhor classificar a entrada (COPPIN, 2010).

Na aprendizagem supervisionada o aprendizado ocorre da seguinte forma: a RNA é exposta a um corpus pré-classificado, e para cada amostra desse conjunto a rede oferece uma saída. Essa saída é então comparada com a saída desejada do corpus, a diferença entre a resposta desejada e a resposta real da rede gera um sinal de erro, e com isso são ajustados os parâmetros da rede de modo a reduzir o erro. Dessa forma, o conhecimento disponível é transferido para a rede neural por meio de treinamento, e quando o conhecimento for adquirido pode-se então dispensar o corpus. Esse procedimento tem o efeito de produzir um conjunto de pesos que pretende minimizar o erro sobre todo o corpus (HAYKIN, 2001; LUGER, 2013).

Após a execução de cada amostra do conjunto de treinamento, inicia-se novamente o mesmo processo, isso é chamado de época, e são repetidas até alcançar algum critério de parada. Para cada época é essencial randomizar o corpus de treinamento para evitar possíveis vícios da rede. Um dos critérios de parada consiste em definir uma taxa mínima de erro aceitável e forçar a parada, porém esse critério pode resultar em um encerramento prematuro do processo de aprendizagem, ou pode requerer um tempo longo para atingir o valor desejado ou, no pior caso, nunca atingir. Outro critério utilizado para encerrar o ajuste dos pesos é o mínimo local ou global da superfície de erro (HAYKIN, 2001; RUSSELL; NORVIG, 2004).

Um fator que impacta no aprendizado é a taxa de aprendizagem, pois quanto menor for a taxa, menor serão as variações dos pesos sinápticos da rede, entretanto o custo disso é uma aprendizagem lenta, isto é, requer mais épocas. Por outro lado, ao utilizar uma alta taxa o processo de aprendizagem é acelerado, visto que há grandes modificações nos pesos sinápticos, porém pode deixar a rede instável (HAYKIN, 2001).

Como já visto, a RNA consiste em vários neurônios organizados em camadas. Basicamente as redes possuem uma camada de entrada, que provocam a ativação de alguns neurônios e esses enviam sinais aos neurônios aos quais estão conectados. Desse modo, um padrão complexo de ativações é organizado pela rede, resultando na ativação dos neurônios da camada de saída. Isto é, quando uma entrada é dada, a saída não aparece imediatamente porque leva um tempo para os sinais passarem de um neurônio ao outro (COPPIN, 2010). Atualmente há uma família de arquiteturas, cada qual adequada para funcionalidades específicas; abaixo serão descritas as arquiteturas MLP e RNN.

A MLP é chamada de redes alimentadas adiante com múltiplas camadas. Alimentação adiante quer dizer que os dados são alimentados a partir dos nós de entrada em direção aos nós de saída, ou seja, os sinais de saída da primeira camada são utilizados como entradas para a segunda camada, e assim por diante; e múltiplas camadas quer dizer que a rede, além das camadas de entrada e saída, possui uma ou mais camadas ocultas. A adição de camadas ocultas aumenta o espaço de hipóteses que a rede pode representar, e com isso, resolvem problemas que não são linearmente separáveis, tais como a função OU exclusivo, também conhecida como XOR (COPPIN, 2010; HAYKIN, 2001; RUSSELL; NORVIG, 2004).

A função dos neurônios ocultos é intervir entre a entrada e a saída da rede de uma maneira útil, agindo como detectores de características, isto é, esses neurônios começam a encontrar características que individualizam os dados de treinamento conforme o processo de aprendizagem avança. Além disso, a quantidade de neurônios na camada de entrada e saída da rede e a existência e quantidade de neurônios nas camadas ocultas são características do problema a ser resolvido (COPPIN, 2010; HAYKIN, 2001).

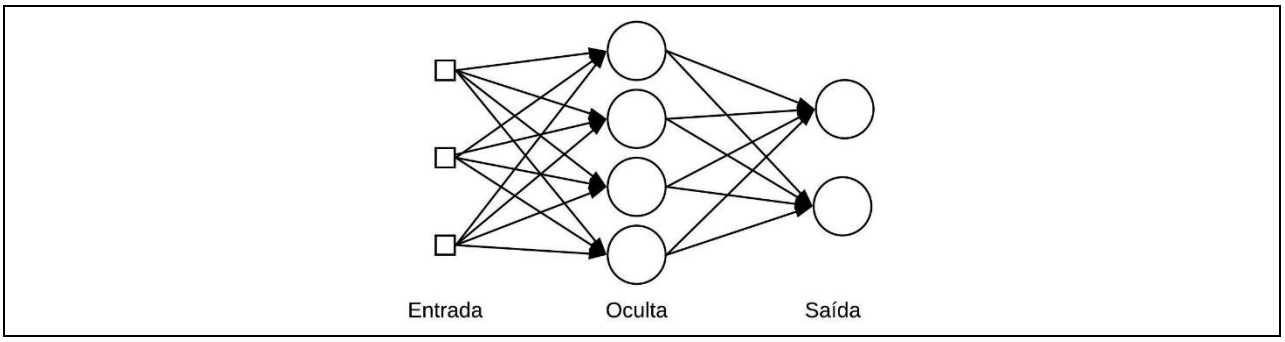
Os erros das camadas ocultas são emblemáticos uma vez que os dados de treinamento não informam quais valores os nós ocultos devem possuir, por isso a análise da fonte de erro na camada de saída é complexa, e consequentemente, dificulta os ajustes dos pesos. Para facilitar isso existe um algoritmo de retropropagação (em inglês: *backpropagation*) que consiste em iniciar a sua execução na camada de saída e propagar o erro retroativamente através das camadas ocultas (LUGER, 2013; RUSSELL; NORVIG, 2004).

De acordo com Haykin (2001), a retropropagação é baseada na aprendizagem supervisionada e consiste em dois passos, um para frente (propagação) e o outro para trás (retropropagação). No passo para frente, são inseridos os valores de entrada do conjunto de treinamento, para obter os valores

de saída, que é a resposta real da rede, e então é calculado o sinal de erro (diferença entre a resposta real e desejada). Durante o passo para trás, os pesos sinápticos são todos ajustados de acordo com o erro calculado, para fazer com que a resposta real da rede se mova para mais perto da resposta desejada, em um sentido estatístico.

A Figura 4 demonstra a arquitetura de uma rede MLP, com uma camada de entrada, uma camada oculta e uma camada de saída. Como pode-se perceber essa rede é totalmente conectada, ou seja, um neurônio de qualquer camada está conectado a todos os neurônios da camada posterior (quando possuir). Entretanto, se algumas das conexões sinápticas estiverem faltando, pode-se dizer que a rede é parcialmente conectada (HAYKIN, 2001).

Figura 4 - Arquitetura da MLP

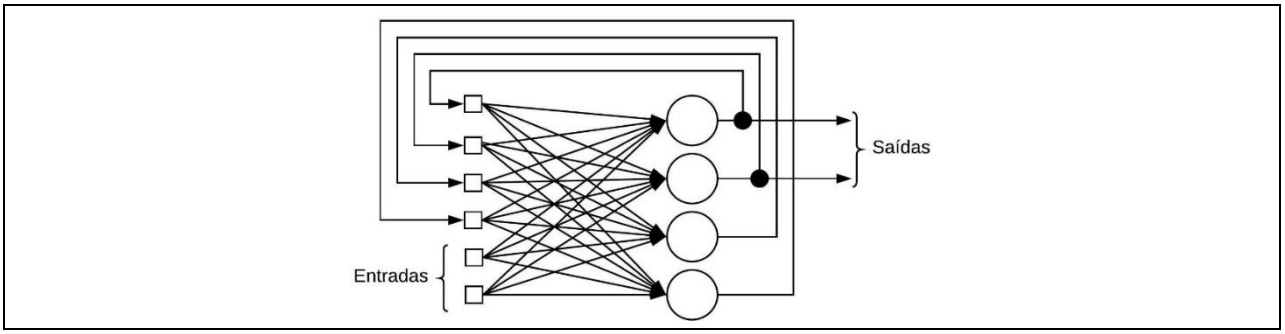


Fonte: Adaptado de Haykin (2001).

Uma arquitetura RNN se distingue de uma MLP quando apresenta um ou mais laços de realimentação, isto é, utiliza suas saídas para alimentar de volta suas entradas. Uma rede recorrente pode consistir: (*i*) em uma única camada de neurônios com cada neurônio alimentando seu sinal de saída de volta para as entradas de todos os outros; ou (*ii*) com a presença de uma camada oculta no qual as conexões de realimentação se originam dos neurônios ocultos ou de saída (COPPIN, 2010; HAYKIN, 2001).

A presença de realimentação tem um impacto profundo na capacidade de aprendizagem da rede e no seu desempenho, além disso, o uso de realimentação faz com que as RNN sejam aplicáveis em diversas áreas, tais como previsão não-linear e modelagem, equalização adaptativa de canais de comunicação, processamento de voz, entre outros. Qualquer que seja o uso, as redes recorrentes apresentam a característica de serem estáveis (HAYKIN, 2001; RUSSELL; NORVIG, 2004). A Figura 5 representa a arquitetura de uma RNN.

Figura 5 - Arquitetura da RNN



Fonte: Haykin (2001).

Resumindo, a resposta da rede a uma determinada entrada depende das entradas iniciais e das anteriores, e com isso pode-se dizer que este tipo de RNA possui uma memória de curto prazo. Entretanto, em algumas aplicações há a necessidade da RNN possuir LSTM (*Long Short-Term Memory*, em português: memória de longo prazo), que é projetada para modelar sequências temporais e suas dependências de longo alcance de maneira mais precisa do que as RNNs convencionais, pois contém unidades especiais chamadas blocos de memória na camada oculta recorrente. Esses blocos possuem células de memória que armazenam o estado temporal da rede, além de unidades multiplicativas especiais chamadas de portas, que controlam o fluxo de informações (RUSSELL; NORVIG, 2004; SAK; SENIOR; BEAUFAYS, 2014).

2.3.1.2 Modelos Ocultos de Markov

Uma cadeia de Markov é um autômato finito ponderado[5], pois a sequência de entrada determina de maneira única quais estados o autômato irá passar, porém não são apropriados para a marcação de fala. Isso ocorre porque, na marcação da fala, enquanto observa-se as palavras na entrada, não se observa os fonemas, assim não se pode condicionar qualquer probabilidade. Por isso utiliza-se o HMM, visto que permite ponderar os estados observados (palavras na entrada) e os ocultos (fonemas) no modelo probabilístico. Isto é, são capazes de modelar tanto as variabilidades acústicas

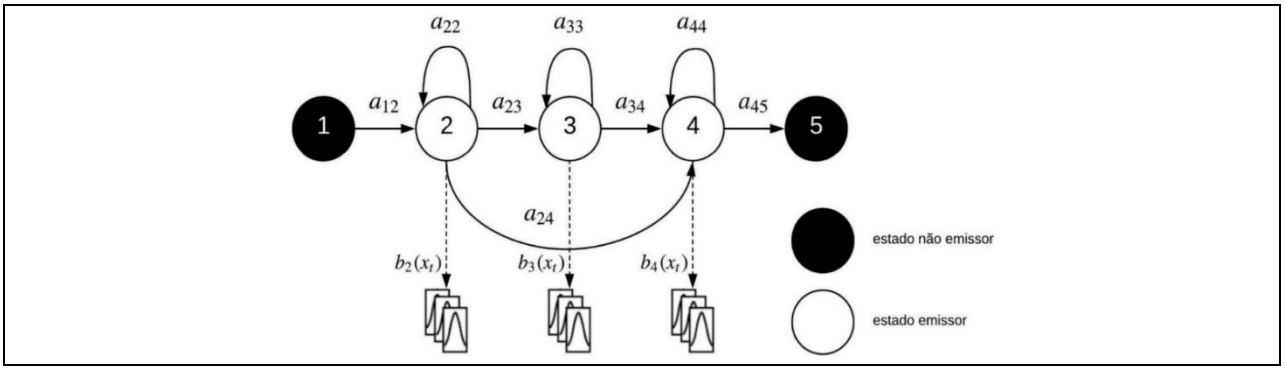
[5] Um autômato finito é definido por um conjunto de estados e transições entre estados que são baseadas nas observações de entrada. Já um autômato finito ponderado é um simples aumento do autômato finito no qual cada arco é associado a uma probabilidade, indicando o caminho a ser seguido (JURAFSKY; MARTIN, 2008).

como temporais do sinal de voz, e permitem a construção hierárquica dos modelos acústicos das sentenças (GORDILLO, 2013; JURAFSKY; MARTIN, 2008; SILVA, 2010).

Os HMMs são aplicados para a identificação temporal de fonemas, palavras ou frases, pois é uma técnica largamente utilizada para estimação dos símbolos indicados pelo sinal codificado, ou seja, a sua principal característica é a modelagem temporal de sequências de símbolos (MÜLLER, 2006; RUSSELL; NORVIG, 2004). Nessa dissertação serão utilizados os modelos HMM baseados em fonemas.

No reconhecimento de voz, é comum a utilização da estrutura esquerda-direita (do inglês: *letf-right*), isto é, o estado atual depende apenas do anterior e só pode transitar para ele próprio ou para o seguinte. Cada estado oculto está associado a um vetor de probabilidades para cada fonema de observação, por isso pode ser facilmente entendido como sendo um gerador de sequências de vetores. A Figura 6 mostra o modelo básico de um HMM com 3 estados emissores e 2 não emissores que são utilizados para concatenação de modelos, ou seja, o estado de saída do modelo de um fonema pode estar ligado ao de entrada de outro, criando assim um HMM composto, e isso permite a formação de palavras. A probabilidade de transição do estado *i* para o estado *j* que ocorre a cada tempo *t* é definida por a_{ij}, e o estado sofre transição para ele quando $i=j$. O $b_i(x_t)$ é a probabilidade da observação x no tempo t dado o estado i (GORDILLO, 2013; JURAFSKY; MARTIN, 2008; RUSSELL; NORVIG, 2004; SILVA, 2010; TEVAH, 2006; VEIGA, 2013).

Figura 6 - Modelo HMM com estrutura esquerda-direita



Fonte: Adaptado de Silva (2010) e Veiga (2013).

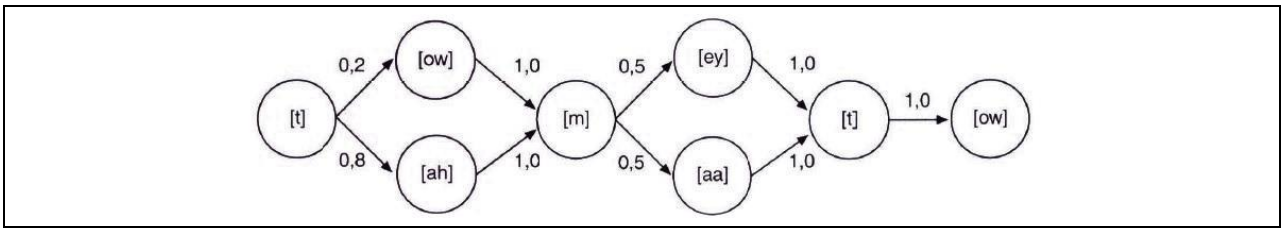
Em outras palavras, o HMM é definido pelos seguintes parâmetros: número de estados, matriz de probabilidades de transição entre estados e uma função de densidade de probabilidade, que

caracteriza os parâmetros acústicos observados nesse estado. A razão pela qual os HMMs são populares é que seus parâmetros podem ser estimados automaticamente a partir de uma grande quantidade de dados, e, além disso, são simples e computacionalmente viáveis (HUANG; DENG, 2010; VEIGA; 2013).

O número de estados e a topologia dos modelos HMM são definidos antes de iniciar o processo de construção dos modelos de transições de estados e treinamento das probabilidades dessas transições. O método de treino do ML (*Maximum Likelihood*, em português: máxima verosimilhança) é muito utilizado para estimar os parâmetros dos modelos recorrendo ao algoritmo de viterbi (MÜLLER, 2006; VEIGA, 2013).

O algoritmo viterbi cria *N* colunas de estado. Para cada coluna, e para cada estado na coluna 1, calcula-se a probabilidade de mover para cada estado na coluna 2 e assim por diante. Esse algoritmo de decodificação é o mais utilizado em HMMs, para o processamento de fala e linguagem. Essa técnica obtém de maneira rápida a sequência mais provável de estados para uma sequência emitida pelo HMM, pois em vez de considerar todas as combinações de transições de estado possíveis, considera somente a sequência com maior probabilidade de produzir a sequência de observações, ou seja, para o reconhecimento de voz interessa apenas uma classificação ordenada das probabilidades, visto que isso já permite determinar qual palavra foi reconhecida. Por exemplo, a palavra inglesa *tomato* pode ser pronunciada de diferentes formas devido aos efeitos do dialeto e de coarticulação, e a função do algoritmo é escolher qual o melhor caminho, conforme a Figura 7 (ALENCAR, 2005; GORDILLO, 2013; JURAFSKY; MARTIN, 2008; RUSSELL; NORVIG, 2004).

Figura 7 - Diagrama com as transições permitidas da palavra inglesa *tomato*



Fonte: Russell e Norvig (2004).

2.3.2 Modelo de Linguagem

No reconhecimento de voz contínuo de grande vocabulário é inviável desenvolver um corpus com todas as frases possíveis para lidar com a gramática do idioma, pois em um vocabulário de tamanho X com o reconhecimento de uma sequência de Y palavras, existem X^Y possibilidades. Para isso, faz-se o uso de modelos de linguagem, que buscam caracterizar a língua para capturar suas regularidades e assim condicionar as combinações de palavras, evitando frases gramaticalmente incorretas (SILVA, 2010; TEVAH, 2006; GORDILLO, 2013).

O modelo de linguagem define um caminho de maior probabilidade, em relação à conexão das palavras dentro de uma sentença levando em consideração as regras gramaticais, isto é, esses modelos são construídos a partir de regras gramaticais básicas que são otimizadas pelo sistema por meio de probabilidade. Em outras palavras, estima a probabilidade de uma palavra, em uma sentença, dadas as palavras anteriores (MÜLLER, 2006; SILVA et al., 2004).

Gordillo (2013) apresenta um exemplo de um problema que pode ser resolvido pelo modelo de linguagem: as palavras “norte” e “morte” apresentam ondas sonoras quase idênticas, porém sabe-se que quando a palavra anterior for “polo” a próxima palavra será “norte”. Isso melhora consideravelmente o desempenho e taxa de erro do reconhecedor, pois reduz significativamente o espaço de busca da frase correta.

Esse modelo utiliza o PLN para processar e manipular a linguagem falada em diversos níveis. Segundo Coppin (2010), os cinco níveis de processamento e manipulação da linguagem são fonologia, morfologia, sintaxe, semântica e pragmática. A fonologia esteve presente no modelo acústico, e a pragmática, que tem como objetivo compreender o contexto da frase, não é utilizada nessa dissertação. A morfologia, sintaxe e semântica são descritas abaixo:

1. Morfologia é o primeiro estágio da análise que é aplicado a palavras que foram identificadas pelo sistema. Esse estágio baseia-se em regras que analisam as palavras e as classificam segundo tabelas de afixos. Por exemplo, a entrada ‘zinho’ de uma tabela de sufixos está associada a um diminutivo de um substantivo, portanto, a palavra bonezinho é o diminutivo da palavra boné, que é seu radical. Com isso, antes do nível de sintaxe, são reconhecidas e corrigidas as palavras que não estão na sua forma padrão. A análise morfológica é importante para determinar o papel de uma palavra em uma sentença (LUGER, 2013; MÜLLER, 2006).

2. No nível da sintaxe são aplicadas as regras de gramática do idioma que está sendo utilizado, ou seja, a sintaxe determina o papel de cada palavra em uma sentença para verificar se as palavras geram uma frase válida e gramaticalmente correta (COPPIN, 2010; LUGER, 2013).

3. Semântica examina o significado das sentenças formadas. Apesar do extenso processamento realizado nas análises morfológica e sintática, apenas com essas não é possível distinguir certas categorias de palavras e muito menos prever o objetivo da frase, isto é, a sentença pode estar sintaticamente correta, mas ser incorreta semanticamente (LUGER, 2013; MÜLLER, 2006).

A maneira mais usada e simples de se obter as probabilidades do modelo de linguagem é com a utilização de *n*-grama, que dependem apenas das palavras anteriores da frase, ou seja, a probabilidade de cada palavra em uma sentença depende apenas das $n-1$ palavras anteriores a ela. O custo computacional para validar o modelo é proporcional ao valor de *n*, isto é, se o *n* for muito grande então o custo computacional pode ser muito alto (GORDILLO, 2013; SILVA, 2010; TEVAH, 2006).

Ao capturar a correlação existente entre palavras anteriores, os n-grama acabam absorvendo a sintaxe, semântica e pragmática existente nas frases observadas. Isso os faz extremamente efetivos no idioma português, visto que a ordem das palavras é importante, uma vez que os efeitos contextuais vêm dos vizinhos mais próximos, sem a necessidade de criação de regras e nem de uma gramática formal. As distribuições de probabilidade são computadas diretamente de frases prontas e a estimação pode ser executada simplesmente contando o número de ocorrências, porém para se obter uma boa estimação é necessário um conjunto de milhares de frases (SILVA, 2010; TEVAH, 2006).

Nos sistemas de reconhecimento de voz os modelos de linguagem tendem a ser 2-grama e 3-grama. Pode-se usar também 4-grama, entretanto isso exige uma enorme quantidade de memória, dificultando a sua utilização pelos aplicativos de linguagem em dispositivos móveis (JURAFSKY; NORVIG, 2004; SILVA, 2010). Segundo Russell e Norvig (2004), a forma de calcular 2-grama está representada na equação (9), no qual apenas é considerada a palavra anterior.

$$P(W) = P(w_n|w_{n-1}) \tag{9}$$

Uma grande vantagem do modelo 2-grama é a facilidade de treinar o modelo, contando apenas o número de vezes que cada par de palavras ocorre em um corpus para estimar as probabilidades. Por exemplo, na Figura 8, a probabilidade da palavra “eu” ser antes de “sou” é de 0,67, pois a palavra “eu” aparece três vezes no corpus, porém a palavra “sou” aparece apenas duas vezes após a palavra “eu”, ou seja, quando aparecer “eu sou” na frase tem 67% de estar correto. No corpus de linguagem utiliza-se marcadores de início e fim de sentenças, tais como <s> e </s>, respectivamente (LUGER, 2013; RUSSELL; NORVIG, 2004; SILVA, 2010).

Figura 8 - Cálculo de probabilidades

Corpus	Probabilidades	
<s> Eu sou Sam </s> <s> Sam eu sou </s> <s> Eu gosto de café </s>	$P(\text{eu}\|\text{<s>}) = \frac{2}{3} = 0,67$	$P(\text{Sam}\|\text{sou}) = \frac{1}{2} = 0,5$
	$P(\text{</s>}\|\text{Sam}) = \frac{1}{2} = 0,5$	$P(\text{sou}\|\text{eu}) = \frac{2}{3} = 0,67$
	$P(\text{Sam}\|\text{<s>}) = \frac{1}{3} = 0,33$	$P(\text{gosto}\|\text{eu}) = \frac{1}{3} = 0,33$

Fonte: Adaptado de Jurafsky e Norvig (2008).

Já no modelo 3-grama a interpretação da palavra atual depende das duas anteriores, e seu cálculo é representado pela equação (10) (LUGER, 2013; RUSSELL; NORVIG, 2004).

$$P(W) = P(w_n|w_{n-2}w_{n-1}) \tag{10}$$

O uso de 3-grama, se comparado com o 2-grama, apresenta melhor desempenho, pois a maioria das palavras possui uma forte dependência das duas palavras anteriores. Por exemplo, a frase “comeu uma bandana” está correta no modelo 2-grama, visto que o mesmo calcula a probabilidade da palavra “bandana” com a palavra anterior “uma”. Já com o modelo 3-grama é possível notar que a frase está semanticamente incorreta, uma vez que calcula a probabilidade da palavra “bandana” com as palavras anteriores “comeu uma”, com isso, o modelo de linguagem deverá localizar uma palavra para realizar a correção do erro, e a tendência será a palavra “banana” (RUSSELL; NORVIG, 2004; SILVA, 2010).

2.4 MÉTRICAS DE AVALIAÇÃO

O desempenho do reconhecimento de voz depende da precisão dos modelos acústicos, da complexidade da tarefa definida pelo modelo de linguagem e da qualidade do sinal de áudio adquirido. Para isso, existem na literatura diversas funções matemáticas as quais podem ser chamadas de métricas de avaliação. Para esse trabalho serão estudadas as seguintes: WER, SER (*Sentence Error Rate*, em português: taxa de erro de sentença) e xRT (*Real Time Factor*, em português: fator em tempo real) (FERREIRA; SOUZA, 2017; SAMPAIO NETO, 2011; VEIGA, 2013).

A WER é uma das métricas mais utilizadas em sistemas de reconhecimento de voz contínuo (TEVAH, 2006). Isso também pode ser observado na revisão sistemática da literatura (Capítulo 3), haja visto que todos os trabalhos selecionados utilizavam essa métrica para avaliar a qualidade. Para Jurafsky e Martin (2008), a WER é baseada na quantidade de palavras que foram inseridas incorretamente, que foram excluídas e que foram substituídas em comparação com a frase de referência. Essa taxa é calculada pela equação (11).

$$WER = \frac{S + I + E}{N} \tag{11}$$

No qual, *N* é o número total de palavras da frase de referência e *S*, *I* e *E* são, respectivamente, o número total de erros por substituição, inserção e exclusão da frase gerada em comparação com a frase de referência. A Figura 9 apresenta um exemplo de comparação entre as frases de referência e gerada. Como pode-se observar a frase gerada apresenta duas substituições, uma inclusão e uma exclusão, totalizando quatro erros, e a frase de referência possui cinco palavras, sendo assim, ao substituir os respectivos valores na equação (11), o valor do WER é de 0,8 (80%).

Figura 9 - Comparação entre as frases de referência e gerada para cálculo do WER

Referência =	Tem	um	preso	na	*	cela
Gerada =	Tem	*	preso	em	uma	sela
		E		S	I	S

Segundo Ferreira e Souza (2017), a SER representa quantas frases possuem pelo menos um erro, ou seja, quantas frases apresentaram um WER maior que 0%. A SER é calculada utilizando a equação (12).

$$SER = \frac{E}{T} \tag{12}$$

Em que *E* é a quantidade de sentenças com pelo menos um erro e *T* é a quantidade total de sentenças. Por exemplo, em um corpus com 100 frases, apenas 30 apresentaram um WER maior que 0%, sendo assim, a SER é de 0,3 (30%).

Já o fator xRT é utilizado para calcular a velocidade do processo de reconhecimento de voz, ou seja, é calculado dividindo o tempo que o sistema gasta para reconhecer uma sentença pela sua duração, de acordo com a equação (13). Assim, quanto menor for o fator xRT mais rápido será o reconhecimento (SAMPAIO NETO, 2011).

$$xRT = \frac{P}{D} \tag{13}$$

Em que *P* é o tempo de processamento gasto para realizar o reconhecimento da voz do arquivo de áudio e *D* é o tempo de duração do arquivo de áudio. Por exemplo, o computador leva 1,2 segundo para reconhecer um arquivo de 28 segundos, então o seu fator xRT é 0,04.

2.5 CONSIDERAÇÕES

Neste capítulo foram apresentados os conceitos básicos do reconhecimento de voz, descrevendo as características da voz e os fatores de complexidade. Além disso, foram apresentadas as etapas que são utilizadas pelo reconhecimento de voz contínuo, juntamente com a descrição de suas principais técnicas.

O uso das técnicas de IA neste trabalho tem o propósito de melhorar o desempenho e as métricas de avaliação do reconhecimento de voz contínuo. Dentre essas técnicas de IA pode-se citar o uso de RNA, HMM e PLN no desenvolvimento dos modelos acústico e de linguagem.

3 ESTADO DA ARTE

Este capítulo apresenta a revisão sistemática da literatura que tem como objetivo identificar e selecionar as principais técnicas que são utilizadas na extração de características do áudio e na implementação dos modelos, acústico e de linguagem, para o desenvolvimento do reconhecimento de voz contínuo. Para isso, realizou-se uma revisão contemplando artigos publicados nos últimos cinco anos em quatro repositórios diferentes, sendo esses: ACM, IEEE, ScienceDirect e Scopus. Além disso, o presente capítulo possui duas publicações, que são demonstradas na seção 6.1.

O capítulo está organizado da seguinte maneira: a seção 3.1 apresenta a revisão sistemática de literatura com o protocolo de busca e os trabalhos selecionados que oferecem uma proposta de solução relevante a esse problema; na seção 3.2 é realizada uma análise comparativa dos trabalhos relacionados; e na seção 3.3 são apresentadas algumas considerações sobre esse capítulo.

3.1 REVISÃO SISTEMÁTICA DA LITERATURA

De acordo com Kitchenham e Charters (2007), uma revisão sistemática da literatura é um meio de identificar, avaliar e interpretar todas as pesquisas disponíveis e relevantes para uma determinada questão de pesquisa ou uma área de interesse. Os motivos mais comuns para realizar essa revisão são para: (*i*) sumarizar as evidências existentes sobre uma tecnologia; (*ii*) identificar lacunas na pesquisa atual; (*iii*) criar uma base para posicionar adequadamente novas atividades de pesquisa; (*iv*) examinar até que ponto a evidência empírica sustenta ou contradiz as hipóteses; e (*v*) auxiliar na geração de novas hipóteses.

A metodologia aplicada nessa revisão consiste na delimitação: (*i*) das perguntas de pesquisa; (*ii*) dos repositórios e estratégia de pesquisa; e (*iii*) seleção dos artigos.

3.1.1 Perguntas de Pesquisa

Com base no objetivo dessa revisão sistemática da literatura foram desenvolvidas três perguntas de pesquisa que são apresentadas no Quadro 3.

Quadro 3 - Perguntas de pesquisa da revisão sistemática da literatura

ID	**Pergunta de pesquisa**
P1	Quais técnicas estão sendo utilizadas na implementação do modelo acústico do reconhecimento de voz contínuo?
P2	Quais técnicas estão sendo utilizadas na implementação do modelo de linguagem para aperfeiçoar o reconhecimento de voz contínuo?
P3	Quais soluções estão sendo estudadas para reduzir as taxas de erros do reconhecimento de voz contínuo?

Essas perguntas de pesquisa não abordaram o português brasileiro, pois as técnicas, identificadas e selecionadas na revisão, podem ser adaptadas para qualquer corpus de voz, de qualquer idioma. Além disso, não foi abordado o termo off-line, porque muitas técnicas do modo on-line também podem ser adaptadas.

3.1.2 Repositórios e Estratégia de Pesquisa

Para responder a essas perguntas, foram selecionados quatro repositórios eletrônicos. No Quadro 4, pode-se observar o nome e o endereço de acesso na web de cada um.

Quadro 4 - Repositórios eletrônicos

Repositório	**Endereço de acesso**
ACM	http://www.acm.org
IEEE	http://ieeexplore.ieee.org
ScienceDirect	http://www.sciencedirect.com
Scopus	https://www.scopus.com

Com base nos repositórios selecionados desenvolveu-se uma expressão de busca a partir de palavras-chave que não apresentassem redundância nos resultados, visando contemplar o maior número de artigos, e ao mesmo tempo servindo como um filtro para o retorno dos artigos mais relevantes ao tema.

Utilizou-se a seguinte expressão de busca, utilizada na pesquisa em cada repositório: "*continuous speech recognition*" E ("*acoustic models*" OU "*neural networks*" OU ann OU "*deep learning*") E ("*language models*" OU lm OU *n-gram* OU "*natural language processing*" OU nlp). Esses termos de busca foram adaptados para o formato de cada repositório de busca, ou seja, sem alterar as palavras ou o valor dos operadores lógicos.

3.1.3 Seleção dos Artigos

Os critérios de inclusão e exclusão para a escolha dos artigos são apresentados no Quadro 5.

Quadro 5 - Critérios de inclusão e exclusão

Inclusão	**Exclusão**
CI1: artigos publicados entre 01/01/2014 até 31/12/2018.	CE1: artigos que não possuem resultados relacionados ao desenvolvimento do reconhecimento de voz contínuo.
CI2: expressão de busca filtrando os artigos por meio do título, resumo e palavras-chave.	CE2: artigos que não apresentam o desenvolvimento do modelo acústico e do modelo de linguagem.
CI3: artigos em inglês e português.	CE3: ausência de especificação das técnicas utilizadas no desenvolvimento dos modelos acústico e de linguagem.
	CE4: artigos curtos (5 páginas ou menos).

Além dos critérios, realizou-se a seleção dos artigos por meio da leitura dos seguintes tópicos: (*i*) título e palavras-chave; (*ii*) resumo; e (*iii*) introdução e conclusão. Esses critérios para a leitura e seleção foram úteis para minimizar o esforço na leitura e seleção de trabalhos que realmente contribuem para responder às perguntas do tema de pesquisa.

3.1.4 Resultados

Ao aplicar a expressão de busca e considerar os critérios de inclusão e exclusão em cada repositório, foram localizados 80 artigos, porém 16 artigos apareceram em mais de um repositório, ou seja, apenas 64 artigos foram descobertos. Em seguida, realizou-se a seleção desses artigos, por meio da leitura do título, palavras-chave, resumo, introdução e conclusão. Após essa seleção, o número de artigos reduziu para 10, isto é, apenas 16% dos artigos descobertos apresentaram alguma relação direta com o tema proposto. A Tabela 1 apresenta o número de artigos descobertos e selecionados por repositório, já os artigos excluídos podem ser consultados no APÊNDICE A.

Tabela 1 - Número de artigos descobertos e selecionados

Repositório	**Descobertos**	**Selecionados**
ACM	7	0
IEEE	20	3
ScienceDirect	3	1
Scopus	34	6

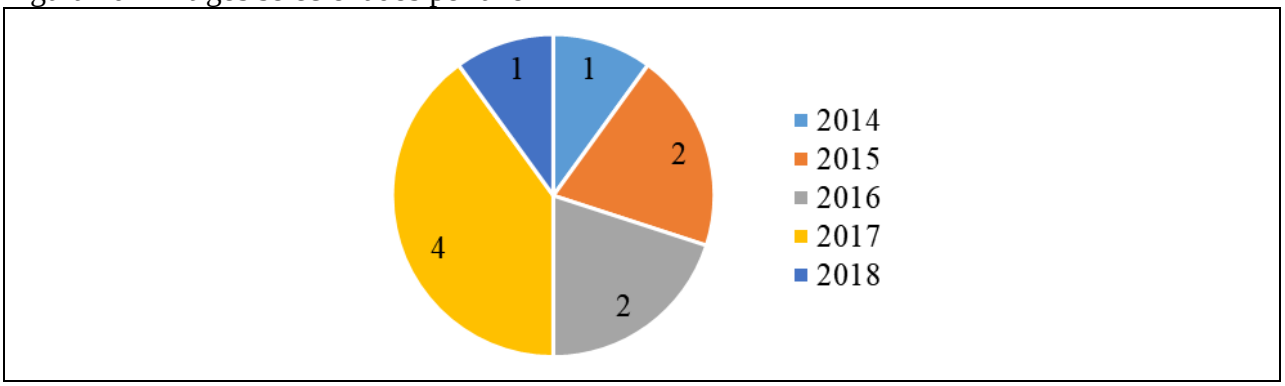
O Quadro 6 apresenta os artigos selecionados em ordem alfabética, cada qual com a sua identificação, título, repositório e ano de publicação.

Quadro 6 - Relação de artigos selecionados

Identificação	**Título**	**Repositório**	**Ano**
Abushariah (2017)	*TAMEEM V1.0: speakers and text independent Arabic automatic continuous speech recognizer*	Scopus	2017
Georgescu, Cucu e Burileanu (2017)	*SpeeD's DNN approach to Romanian speech recognition*	IEEE	2017
Kipyatkova e Karpov (2017)	*A study of neural network Russian language models for automatic continuous speech recognition systems*	Scopus	2017
LAleye et al. (2016)	*First automatic fongbe continuous speech recognition system: Development of acoustic models and language models*	IEEE	2016
Naing et al. (2015)	*A Myanmar large vocabulary continuous speech recognition system*	IEEE	2015
Pakoci, Popović e Pekar (2017)	*Language model optimization for a deep neural network based speech recognition system for Serbian*	Scopus	2017
Pakoci, Popović e Pekar (2018)	*Improvements in Serbian Speech Recognition Using Sequence-Trained Deep Neural Networks*	Scopus	2018
Phull e Kumar (2016)	*Investigation of Indian English speech recognition using CMU sphinx*	Scopus	2016
Tachbelie, Abate e Besacier (2014)	*Using different acoustic, lexical and language modeling units for ASR of an under-resourced language – Amharic*	ScienceDirect	2014
Zhang, Bao e Gao (2015)	*Mongolian speech recognition based on deep neural networks*	Scopus	2015

A Figura 10 representa graficamente os artigos selecionados por ano. A maioria desses artigos foram publicados no ano de 2017.

Figura 10 - Artigos selecionados por ano



A próxima seção apresenta uma análise comparativa, que tem como objetivo responder as três perguntas de pesquisa de maneira sintetizada por meio da leitura de cada artigo do Quadro 6, visando facilitar a interpretação e a tabulação dos dados.

3.2 ANÁLISE COMPARATIVA

O Quadro 7 apresenta os artigos selecionados e as técnicas utilizadas na implementação dos modelos, acústico e de linguagem, para cada artigo.

Quadro 7 - Técnicas utilizadas na implementação dos modelos

Identificação	**Modelo Acústico**	**Modelo de Linguagem**
Abushariah (2017)	HMM	1-grama, 2-grama e 3-grama
Georgescu, Cucu e Burileanu (2017)	RNN e HMM	1-grama, 2-grama e 3-grama
Kipyatkova e Karpov (2017)	HMM	RNN-LM e 3-grama
LAleye et al. (2016)	Métodos "monofone" e "trifone" da biblioteca Kaldi	3-grama, utilizando a ferramenta SRILM
Naing et al. (2015)	MLP e HMM	*Word-base*
Pakoci, Popović e Pekar (2017)	MLP e HMM	1-grama, 2-grama e 3-grama
Pakoci, Popović e Pekar (2018)	MLP, RNN e HMM	1-grama, 2-grama e 3-grama
Phull e Kumar (2016)	HMM e HMM	2-grama e 3-grama
Tachbelie, Abate e Besacier (2014)	HMM	3-grama, utilizando a ferramenta SRILM
Zhang, Bao e Gao (2015)	MLP e HMM	2-grama e 3-grama

As principais técnicas utilizadas no desenvolvimento do modelo acústico foram o HMM (*Hidden Markov Models*, em português: modelos ocultos de Markov) e as RNA (Redes Neurais Artificiais): MLP (*Multilayer Perceptron*, em português: *perceptron* multicamadas) e RNN (*Recurrent Neural Network*, em português: rede neural recorrente). Esses tipos de redes neurais artificias também são conhecidos como sendo DNN (*Deep Neural Network*, em português: rede neural profunda), e são aplicados com sucesso em modelos acústicos de sistemas de reconhecimento de voz de última geração, pois permitem que dados complexos sejam bem modelados (DAHL et al., 2012). O modelo 3-grama foi a principal técnica utilizada no desenvolvimento do modelo de linguagem. Esses assuntos foram abordados no Capítulo 2 .

No Quadro 8 são apresentadas as soluções mais utilizadas para reduzir a taxa de erro do reconhecimento de voz contínuo.

Quadro 8 - Soluções para reduzir a taxa de erro

Identificação	**Solução para reduzir a taxa de erro**
Abushariah (2017)	-
Georgescu, Cucu e Burileanu (2017)	Utilizando modelos acústicos baseados em DNN
Kipyatkova e Karpov (2017)	Com um modelo de linguagem baseado em RNN e 3-grama
LAleye et al. (2016)	Removendo diacríticos de tons do modelo de linguagem
Naing et al. (2015)	Utilizando DNN
Pakoci, Popović e Pekar (2017)	Combinando o uso de DNN, HMM e modelo de linguagem
Pakoci, Popović e Pekar (2018)	Utilizando uma DNN de 8 camadas com 625 neurônios cada
Phull e Kumar (2016)	-
Tachbelie, Abate e Besacier (2014)	Utilizando unidades acústicas de sílabas baseado em morfema
Zhang, Bao e Gao (2015)	Utilizando redes neurais profundas em conjunto com HMM

A grande maioria dos artigos selecionados apresentam alguma solução para reduzir as taxas de erros do reconhecimento de voz, e pode-se notar que, em muitos casos, essa solução está associada ao uso de DNN.

Com a leitura dos artigos foi possível extrair diversas particularidades essenciais para o reconhecimento de voz. A partir disso, elaborou-se o Quadro 9 para destacar e comparar os métodos de extração de características de áudio, se usou ou não bibliotecas para facilitar a implementação e os idiomas de cada artigo selecionado.

Quadro 9 - Extração de características, bibliotecas e idioma dos artigos selecionados

Identificação	**Extração de características**	**Bibliotecas**	**Idioma**
Abushariah (2017)	MFCC	CMUSphinx	Árabe (11 países)
Georgescu, Cucu e Burileanu (2017)	MFCC	Kaldi	Romeno
Kipyatkova e Karpov (2017)	MFCC	HTK	Russo
LAleye et al. (2016)	MFCC	Kaldi	Fongbe
Naing et al. (2015)	MFCC	-	Myanmar
Pakoci, Popović e Pekar (2017)	-	Kaldi	Sérvio
Pakoci, Popović e Pekar (2018)	MFCC	Kaldi	Sérvio
Phull e Kumar (2016)	MFCC	CMUSphinx	Inglês Indiano
Tachbelie, Abate e Besacier (2014)	MFCC	CMUSphinx	Amárica
Zhang, Bao e Gao (2015)	MFCC	-	Mongol

Pode-se observar que as bibliotecas CMUSphinx, HTK (*Hidden Markov Models Toolkit*, em português: kit de ferramentas dos modelos ocultos de Markov) e Kaldi foram utilizadas para facilitar a implementação dos modelos, acústico e de linguagem. Além disso, o principal método de extração

de características do áudio é por meio do MFCC (*Mel Frequency Cepstral Coefficients*, em português: coeficientes *cepstrais* de frequência Mel), que foi abordado no Capítulo 2 .

Ainda de acordo com o Quadro 9, pode-se notar que praticamente todos os artigos selecionados desenvolveram o reconhecimento de voz para algum idioma específico. Para isso, é necessário possuir um corpus de voz e de texto. O Quadro 10 realiza uma comparação do tamanho dos corpora de texto e voz dos artigos selecionados.

Quadro 10 - Corpora de texto e voz utilizados nos trabalhos relacionados

Identificação	**Corpora de voz**	**Corpora de texto**
Abushariah (2017)	Foram utilizadas 41.005 sentenças, resultando em cerca de 45 horas de dados de fala coletados de 36 falantes nativos de 11 países árabes diferentes	41.005 sentenças do corpus de voz
Georgescu, Cucu e Burileanu (2017)	Três corpora: (*i*) 100 horas em ambiente silencioso; (*ii*) 28 horas de transmissões de programas de entrevistas (afetados por ruídos) e noticiários (discurso limpo); e (*iii*) 103 horas de duração de conversação	Dois corpora: 315 milhões de palavras coletadas de *sites* de notícias; e 40 milhões de palavras de transcrições de reuniões
Kipyatkova e Karpov (2017)	Áudio de 327 frases, onde cada frase foi pronunciada por 50 oradores com língua russa nativa (25 de ambos os sexos). A gravação foi realizada em uma sala à prova de ruído, com duração total de 21 horas	Corpus de textos russos construídos a partir de notícias eletrônicas, com mais de 350 milhões de palavras
LAleye et al. (2016)	10 horas de fala, contendo 1.500 frases, gravados em um ambiente silencioso, com 28 falantes nativos, 8 mulheres e 20 homens. Os áudios foram gravados em 16 kHz	Corpus de 34.653 frases, com quase 10.130 palavras
Naing et al. (2015)	4 mil frases com 40 horas de duração, gravadas em um ambiente aberto. Participaram da gravação 52 oradores do sexo masculino e 48 do sexo feminino	86 mil frases
Pakoci, Popović e Pekar (2017) e Pakoci, Popović e Pekar (2018)	Dois corpora: (*i*) 154 horas de duração, com 87 mil enunciados, com uma média de 15 palavras cada, com 21 locutores do sexo masculino e 27 locutores do sexo feminino; e (*ii*) 61 horas de duração, com 170 locutores do sexo masculino e 181 do sexo feminino. Ambos com boa qualidade de gravação	Cerca de 1,5 milhões de palavras, das quais apenas 121.000 são diferentes, extraídas do corpus textual jornalístico sérvio
Phull e Kumar (2016)	75 locutores com 15 minutos cada, correspondendo o total de 23 horas. Os áudios foram amostrados no formato de 16 kHz e 16 bit	64.000 palavras, que ocorrem mais de 100 vezes no corpus
Tachbelie, Abate e Besacier (2014)	20 horas de treinamento de discurso coletadas de 100 falantes, com um total de 10.850 sentenças	120.262 frases

Zhang, Bao e Gao (2015)	78 horas de duração, de diálogos, notícias e artigos em mongol, e com 193 oradores diferentes, sendo 110 homens e 83 mulheres	85 milhões de tokens de páginas da web em mongol

Os resultados alcançados nos trabalhos relacionados são apresentados na Tabela 2 considerando os percentuais WER (*Word Error Rate*, em português: taxa de erro de palavras), que foi abordado no Capítulo 2 . Quanto menor essa taxa, maior será a precisão da saída do reconhecimento de voz.

Tabela 2 - Melhores resultados obtidos nos trabalhos relacionados

Identificação	**Melhor WER obtido**
Abushariah (2017)	2,68%
Georgescu, Cucu e Burileanu (2017)	4,5%[6]
Kipyatkova e Karpov (2017)	22,87%
LAleye et al. (2016)	14,83%
Naing et al. (2015)	15,63%
Pakoci, Popović e Pekar (2017)	12,01%
Pakoci, Popović e Pekar (2018)	7,23%
Phull e Kumar (2016)	19%
Tachbelie, Abate e Besacier (2014)	13,3%
Zhang, Bao e Gao (2015)	12,37%

Na Tabela 2, pode-se observar que o WER médio foi de 14,01% com vários idiomas, e esse trabalho tem como objetivo alcançar essa porcentagem para o português brasileiro. Além disso, apenas os trabalhos de Pakoci, Popović e Pekar (2017) e de Pakoci, Popović e Pekar (2018), apresentaram resultados da taxa de erro das palavras em modo off-line e para mobile.

3.3 CONSIDERAÇÕES

Nesse capítulo foi apresentado o estado da arte sobre o desenvolvimento do reconhecimento de voz contínuo. O uso dos critérios de inclusão e exclusão ajudou na seleção dos artigos mais relevantes, pois todos os artigos selecionados apresentaram diferentes respostas e soluções para as questões de pesquisa.

[6] Em base de dados sem ruídos, porém em uma base com ruído o WER foi de 20,2%.

Os trabalhos selecionados foram úteis para descobrir: (*i*) as técnicas utilizadas na implementação dos modelos, acústico e de linguagem; (*ii*) as soluções existentes para reduzir a taxa de erro; (*iii*) as técnicas utilizadas na extração das características do áudio; (*iv*) as bibliotecas utilizadas para a implementação; (*v*) o tamanho dos corpora de texto e voz; e (*vi*) os melhores resultados obtidos.

Essa revisão auxiliou na aquisição de conhecimento sobre as vantagens de cada abordagem, e as melhores foram utilizadas no desenvolvimento do reconhecimento off-line de voz contínuo do português brasileiro para dispositivos móveis, com o objetivo de obter um bom processamento e uso da memória e um baixo valor da métrica WER. O capítulo a seguir descreverá a implementação do reconhecimento de voz contínuo proposto.

4 DESENVOLVIMENTO

Este capítulo descreve os procedimentos que foram utilizados para o desenvolvimento do reconhecimento off-line de voz contínuo, que são: (*i*) instalação das bibliotecas CMUSphinx, HTK (*Hidden Markov Models Toolkit*, em português: kit de ferramentas dos modelos ocultos de Markov) e Kaldi; (*ii*) preparação dos corpora linguísticos utilizados; (*iii*) implementação do treinamento; e (*iv*) implementação dos testes em computadores desktops e em dispositivos móveis. Além disso, o presente capítulo possui uma publicação, que pode ser encontrada na seção 6.1.

4.1 INSTALAÇÃO DAS BIBLIOTECAS

A rapidez no desenvolvimento do reconhecimento de voz contínuo foi um dos principais motivos para a utilização de bibliotecas, porém é necessário que elas estejam em constante atualização e que tenham uma boa documentação. As bibliotecas CMUSphinx, HTK e Kaldi são as mais utilizadas nos artigos selecionados no Capítulo 3 e atendem aos requisitos previamente mencionados.

Essas bibliotecas são gratuitas e foram instaladas em um computador com o sistema operacional Antergos (distribuição Linux baseada em Arch Linux) versão 64 bits. A escolha pelo Linux se deu devido a fácil instalação das bibliotecas, pois com algumas linhas de comandos é possível instalar as bibliotecas e suas dependências.

Nas subseções abaixo serão descritos os detalhes da instalação dessas bibliotecas. Além disso, será realizado um comparativo das técnicas utilizadas nos artigos do Capítulo 3 com as técnicas utilizadas pelas bibliotecas.

4.1.1 CMUSphinx

Segundo Lee, Hon e Reddy (1990), o Sphinx é uma biblioteca para reconhecimento de voz contínuo independente de locutor. Essa biblioteca utiliza a técnica HMM (*Hidden Markov Models*, em português: modelos ocultos de Markov) no modelo acústico e a técnica n-grama no modelo de linguagem.

As ferramentas CMUSphinx são projetadas especificamente para plataformas de baixo recurso e a sua licença é semelhante à BSD (Berkeley *Software Distribution*), que permite a

distribuição comercial. Além disso, essa biblioteca pode ser utilizada para diversas finalidades relacionadas ao reconhecimento de voz, tais como identificação de palavras-chave, alinhamento, avaliação de pronúncia, entre outros (CMUSPHINX, 2019).

O kit de ferramentas CMUSphinx possui diversos pacotes de bibliotecas para diferentes tarefas e aplicações. Nesse trabalho utilizou-se o pacote PocketSphinx que é ideal para ser utilizada em sistemas embarcados. O Pocketsphinx é escrito na linguagem de programação C e pode ser utilizado com Linux, Microsoft Windows, MacOS, iPhone e Android (CMUSPHINX, 2019).

A biblioteca PocketSphinx depende da instalação de outras bibliotecas: (*i*) SphinxBase, que é a base para todos os projetos CMUSphinx; e (*ii*) SphinxTrain, que fornece ferramentas de treinamento de modelo acústico. O download das bibliotecas é realizado pelo website[7], e a instalação é por meio da compilação dos códigos-fonte das bibliotecas usando o programa "make"[8]. Nesse trabalho utilizou as bibliotecas "pocketsphinx", "sphinxbase" e o "sphinxtrain" da versão "5prealpha" (CMUSPHINX, 2019).

4.1.2 HTK

O HTK, disponível na linguagem de programação C, é um kit de ferramentas para construir e manipular HMMs. O HTK é usado principalmente para a pesquisa de reconhecimento de voz, porém pode ser utilizado em inúmeras aplicações, tais como pesquisa em síntese de voz, reconhecimento de caracteres e sequenciamento de DNA. Além disso, as ferramentas do HTK fornecem recursos sofisticados para análise de fala, treinamento de HMM, testes e análise de resultados (HTK, 2019).

Para realizar o download da biblioteca é necessário se cadastrar no website e estar de acordo com a sua licença de uso[9]. Embora a Microsoft mantenha os direitos autorais do código-fonte do HTK, os desenvolvedores podem fazer alterações e contribuí-los para inclusão nas futuras versões. Após baixar o código-fonte da biblioteca é necessário compilá-lo utilizando o programa "make". Esse

[7] Link para download da biblioteca CMUSphinx: https://cmusphinx.github.io/wiki/download/
[8] É responsável por construir automaticamente programas e bibliotecas executáveis a partir do código-fonte.
[9] Link para download da biblioteca HTK: http://htk.eng.cam.ac.uk/download.shtml

projeto inclui o HTKBook[10] que é uma documentação detalhada sobre cada funcionalidade. Nesse trabalho foi utilizada a versão estável 3.4.1 da biblioteca (HTK, 2019).

4.1.3 Kaldi

Segundo Povey et al. (2011), Kaldi é um kit de ferramentas de código aberto para reconhecimento de voz desenvolvido na linguagem de programação C++ e licenciado sob a Apache *License* v2.0. As suas ferramentas compilam em sistemas do tipo Unix e Microsoft Windows, e a biblioteca está disponível para download em seu website[11]. A biblioteca está hospedada na plataforma GitHub e neste trabalho utilizou-se a versão 5.5.

As bibliotecas CMUSphinx e HTK são as principais concorrentes do Kaldi, porém essas bibliotecas não possuem uma estrutura baseada em transdutor de estado finito, amplo suporte à álgebra linear e uma licença não restritiva (POVEY et al., 2011). Além disso, o Kaldi é a única biblioteca entre as três que possui suporte a DNN (*Deep Neural Network*, em português: rede neural profunda), tais como, MLP (*Multilayer Perceptron*, em português: *perceptron* multicamadas) e RNN (*Recurrent Neural Network*, em português: rede neural recorrente) (KALDI, 2019).

A instalação do Kaldi também é por meio do programa "make". Essa biblioteca possui dependências de ferramentas externas, e nesse trabalho foram utilizadas as seguintes: (*i*) OpenFST, é a ferramenta mais importante para o Kaldi, pois como visto a estrutura da biblioteca é baseada em transdutor de estado finito; (*ii*) OpenBLAS, é responsável pelo suporte à álgebra linear; e (*iii*) SRILM (SRI *Language Modeling Toolkit*, em português: kit de ferramentas do modelo de linguagem SRI), é responsável por executar o modelo de linguagem (KALDI, 2019). A biblioteca possui uma documentação detalhada sobre cada funcionalidade[12].

[10] Link para download do HTKBook: http://htk.eng.cam.ac.uk/docs/docs.shtml
[11] Link para download da biblioteca Kaldi: http://kaldi-asr.org
[12] Documentação da biblioteca Kaldi: http://kaldi-asr.org/doc/

4.1.4 Comparativo

O Quadro 11 apresenta um comparativo das técnicas utilizadas nos artigos do Capítulo 3 com as técnicas utilizadas pelas bibliotecas, para a implementação dos modelos acústico e de linguagem.

Quadro 11 - Comparativo entre o estado da arte do trabalho e as bibliotecas

	Estado da arte	**CMUSphinx**	**HTK**	**Kaldi**
Extração de características	MFCC	MFCC	MFCC	MFCC e PLP
Modelo acústico	HMM, MLP e RNN	HMM	HMM	fMLLR, GMM, HMM, MLLR, MLP, RNN, MLP-HMM e RNN-HMM
Modelo de linguagem	1-grama, 2-grama e 3-grama	1-grama, 2-grama e 3-grama	2-grama e 3-grama	Kit de ferramentas IRSTLM ou SRILM
Métricas de avaliação	WER	SER e WER	SER e WER	SER e WER

Com base no Quadro 11, pode-se observar que as técnicas utilizadas pelas bibliotecas CMUSphinx e Kaldi apresentam uma grande compatibilidade com as técnicas utilizadas nos artigos do estado da arte.

4.2 PREPARAÇÃO DOS CORPORA DE VOZ

Segundo Bauer e Aarts (2013), corpora linguísticos são coleções de dados de linguagem, escritos ou falados, que servem para vários tipos de pesquisa e que podem ser usados em todos os ramos da linguística. Em outras palavras, pode-se dizer que os corpora têm como objetivo representar uma linguagem particular como um todo. Por exemplo, um pesquisador pode estudar um corpus de conversas via telefone para comprovar que as pessoas falam no telefone de maneira diferente do que quando conversando pessoalmente.

Para treinar o modelo acústico do reconhecimento de voz contínuo, é necessário possuir um corpus de voz que é um conjunto de arquivos de áudio com suas respectivas transcrições, e dependendo do idioma desejado é difícil obtê-lo com arquivos de alta qualidade. O áudio do corpus de voz pode ser proveniente de textos que são lidos ou de falas espontâneas do locutor (GORDILLO, 2013; SILVA, 2010).

De acordo com Ferreira e Souza (2017), para os idiomas que são diferentes do inglês, como por exemplo o português brasileiro, a obtenção de um corpus de grande porte e gratuito é um dos principais problemas encontrados pelos pesquisadores da área.

Nesse trabalho, foram utilizados os corpora disponíveis no website do grupo FalaBrasil, porém esses corpora possuem poucas horas de duração. Segundo Silva et al. (2005), a maioria dos trabalhos publicados para o português brasileiro restringe-se a um vocabulário reduzido. Por isso, o modelo de linguagem é constituído apenas pelas frases que estão presentes em cada corpus de voz, isto é, o reconhecimento desenvolvido é de vocabulário restrito, pois reconhece apenas as palavras que estão nos corpora de voz.

O corpus LaPS Benchmark[13] é composto por 700 frases e possui 35 locutores com 20 frases cada, sendo 25 homens e 10 mulheres, o que corresponde a aproximadamente 54 minutos de áudio. Todas as gravações foram realizadas em computadores utilizando microfones comuns, e o ambiente não é controlado (presença de ruído). A taxa de amostragem utilizada foi de 22050 Hz e cada amostra foi representada com 16 bits.

Para reconhecer diversas variações e então tornar-se mais abrangente, é necessário que o reconhecimento de voz seja independente de locutor. As frases devem ser capturadas por muitos locutores, preferencialmente de diferentes idades, sotaques e gêneros, para que haja uma grande variação no ritmo, timbre e intensidade nos áudios de treinamento. Entretanto nesse caso deve haver uma grande variedade de locutores no corpus de treino, o que dificulta a construção de tais sistemas (MELO, 2011; SILVA, 2010).

Já o corpus de voz Constituição Federal[14] é composto por 1.255 frases com aproximadamente 30 segundos de duração cada, totalizando aproximadamente 9 horas de áudio com apenas um locutor do sexo masculino. Os arquivos de áudio foram amostrados em 22050 Hz com 16 bits. Além disso, utilizou-se um ambiente de gravação controlado, isto é, com pouca presença de ruído.

[13] Link para download do corpus LaPS Benchmark: http://bit.ly/2SMWxoR
[14] Link para download do corpus Constituição Federal: http://bit.ly/2Flp8yt

Em cada corpus de voz foi verificado se a escrita estava: (*i*) normalizada, ou seja, números e siglas traduzidos para suas respectivas representações textuais; (*ii*) com a ortografia correta; (*iii*) sem caracteres especiais e de pontuação; e (*iv*) em minúsculo, sem a presença de letras em caixa alta. Esses itens são importantes para padronizar a escrita dos corpora utilizados. Além disso, foi verificado se os arquivos de áudio estavam com a mesma configuração, isto é, mesma taxa de amostragem, fator de quantização, entre outros (FERREIRA; SOUZA, 2017; TEVAH, 2006).

Foi necessário dividir os corpora de voz em arquivos de áudio de treinamento e testes. Dividiu-se o corpus LaPS Benchmark em: (*i*) treinamento com 30 locutores (640 arquivos), sendo 23 do sexo masculino e 9 do sexo feminino; e (*ii*) testes com 3 locutores (60 arquivos), sendo 2 do sexo masculino e 1 do sexo feminino. Já o corpus Constituição Federal foi dividido em 1.129 arquivos (90%) para treinamento e 126 arquivos (10%) para testes.

Além disso, também foi necessário desenvolver um dicionário fonético para cada corpus utilizado. Para isso foi convertido cada palavra em uma sequência de fonemas utilizando a ferramenta de conversão de G2P (*Grapheme to Phoneme*, em português: grafema para fonema) disponibilizada de forma gratuita pelo grupo FalaBrasil[15].

Para utilizar estes corpora nas bibliotecas foi necessário realizar a preparação dos dados. Para isso, desenvolveu-se a ferramenta SCT (*Speech Corpus Treatment*, em português: tratamento do corpus de voz) que está disponibilizada de forma gratuita no GitLab[16]. Essa ferramenta foi desenvolvida na linguagem de programação Java, e tem como objetivo realizar a preparação dos dados dos dois corpora de voz, para as bibliotecas CMUSphinx, HTK e Kaldi. A preparação dos dados dos corpora foi baseada na documentação de cada biblioteca. Além disso, essa ferramenta poderá ser facilmente adaptada para um novo corpus de voz.

Essa ferramenta, também possibilita a redução da taxa de amostragem dos arquivos de áudio das bibliotecas. Nesse trabalho todos os arquivos de áudio dos dois corpora de voz foram alterados para 16000 Hz de taxa de amostragem.

---

[15] Link para download da ferramenta G2P: http://labvis.ufpa.br/falabrasil/downloads/
[16] Link do GitLab da ferramenta: https://gitlab.com/lucasdebatin/speech-corpus-treatment-kaldi/

4.3 IMPLEMENTAÇÃO DO TREINAMENTO

Os modelos acústico e de linguagem do reconhecimento de voz necessitam de treinamento. Nessas bibliotecas são utilizadas a técnica de treinamento supervisionado, pois aprendem a classificar os dados de treinamento que já foram classificados manualmente pelos humanos (COPPIN, 2010; FERREIRA; SOUZA, 2017).

Para cada biblioteca e para cada corpus foi criado um diretório de treinamento que possui as configurações e os arquivos do modelo acústico do corpus, ambos criados de acordo com a documentação das bibliotecas. Esses diretórios estão disponibilizados no GitLab[17]. A Tabela 3 apresenta o número de treinamentos realizados em cada diretório.

Tabela 3 - Número de treinamentos realizados

	CMUSphinx	**HTK**	**Kaldi**
LaPS Benchmark	10	10	10
Constituição Federal	10	10	10

No total foram executados 30 treinamentos em cada corpus, isto é, 10 arquivos com diferentes configurações para cada biblioteca. Esses 10 arquivos foram divididos em dois conjuntos, modificando a lógica de alteração das configurações, conforme destacados abaixo:

- Biblioteca CMUSphinx: foram alterados os valores da configuração padrão da biblioteca de modo uniforme: (*i*) reduziu-se os valores em 80%; (*ii*) reduziu-se os valores em 40%; (*iii*) manteve a configuração padrão; (*iv*) aumentou-se os valores em 40%; e (*v*) aumentou-se os valores em 80%. Além disso, gerou-se mais cinco arquivos de configuração alterando os valores padrão previamente mencionados e alterando as opções textuais dos parâmetros de configuração, por exemplo, onde era “no” passou a ser “yes”. As configurações dessa biblioteca estão no APÊNDICE B.

- Biblioteca HTK: os valores da configuração padrão da biblioteca foram alterados em -80%, -40%, 0%, 40% e 80%. Além disso, gerou-se mais cinco arquivos de configuração alterando as configurações padrão de extração de características do áudio, usando a mesma

[17] Link do GitLab: https://gitlab.com/lucasdebatin/train-examples

lógica de alteração uniforme, -80%, -40%, 0%, 40% e 80%. As configurações dessa biblioteca estão no APÊNDICE C.

- Biblioteca Kaldi: a configuração padrão da biblioteca foi alterada para os valores de -80%, -40%, 0%, 40% e 80%. Entretanto, cinco configurações utilizaram DNN e as outras cinco não. As configurações dessa biblioteca estão no APÊNDICE D.

O treinamento das bibliotecas foram executados utilizando arquivos Shell Script, que geraram em cada diretório os seguintes arquivos de resultados para cada configuração: (*i*) a data e hora de início e fim; e (*ii*) a saída da biblioteca com o valor das métricas de avaliação SER (*Sentence Error Rate*, em português: taxa de erro de sentença) e WER (*Word Error Rate*, em português: taxa de erro de palavras).

4.4 IMPLEMENTAÇÃO DOS TESTES

Esse projeto possui duas versões (desktop e móvel) que são utilizadas para testar o desempenho, uso de processador e memória das bibliotecas de reconhecimento de voz contínuo que foram implementadas para o português brasileiro. O nome escolhido para as versões de teste é OCSR (*Offline Continuous Speech Recognition*, em português: reconhecimento off-line de voz contínuo). Para os testes das bibliotecas foram: (*i*) implementados apenas os códigos-fonte que são responsáveis por gerar a saída dos modelos já treinados; e (*ii*) utilizados apenas os arquivos de testes dos corpora.

4.4.1 Desktop

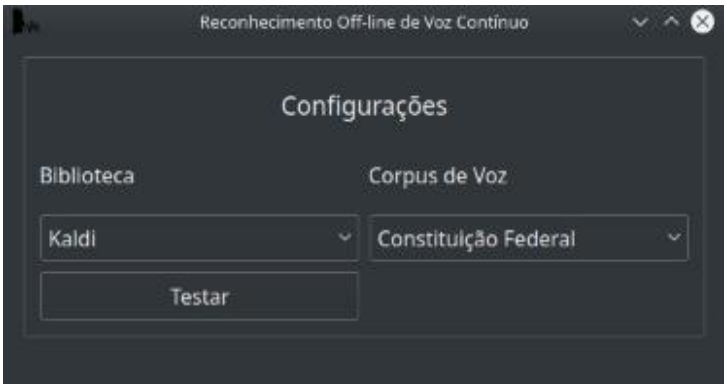
A versão para desktop foi utilizada para testar o desempenho das bibliotecas em um computador desktop. O resultado desse teste foi utilizado para encontrar a biblioteca que apresentou o melhor desempenho, que foi implementada na versão mobile.

Na implementação, utilizou-se a linguagem de programação C++ em conjunto com o Qt SDK (*Software Development Kit*, em português: kit de desenvolvimento de software). Nessa versão foram utilizadas as quatro melhores configurações de treinamento obtidas para cada biblioteca,

CMUSphinx, HTK e Kaldi. A implementação da versão desktop está disponível de forma gratuita no GitLab[18].

A Figura 11 apresenta a tela da versão para desktop. Esta tela possui as seguintes opções de configuração: (*i*) biblioteca, que é responsável por selecionar a biblioteca (CMUSphinx, HTK e Kaldi) que será utilizada nos testes; e (*ii*) corpus de voz, que é responsável por selecionar o corpus de voz (LaPS Benchmark e Constituição Federal) que será utilizado nos testes. O botão "Testar" gera os arquivos de resultados para a biblioteca e corpus selecionados.

Figura 11 - Tela da versão desktop



Para cada biblioteca e corpus são gerados três arquivos de resultados: (*i*) data e hora de início; (*ii*) uso de processador e memória, isto é, o desempenho exigido; e (*iii*) data e hora de encerramento. O desempenho exigido é capturado utilizando o comando Linux "top"[19].

4.4.2 Móvel

A versão móvel foi utilizada para testar o desempenho, em dispositivos móveis, da biblioteca que apresentou o melhor desempenho na versão desktop. De acordo com o Capítulo 5 , os modelos da biblioteca Kaldi apresentaram os melhores resultados para os corpora. Entretanto, a implementação dessa biblioteca no aplicativo não obteve sucesso, pois os executáveis funcionam apenas em dispositivos que estão liberadas as permissões de "superusuário" (root). As tentativas de

[18] Link do GitLab da versão desktop: https://gitlab.com/lucasdebatin/ocsr-desktop
[19] Exibe os dados sobre os processos em execução no dispositivo.

implementar o Kaldi foram baseadas no tutorial de compilação da biblioteca no Android[20]. Por esse motivo, utilizou-se apenas os modelos da biblioteca CMUSphinx, que também apresentou excelentes resultados em comparação com os resultados da biblioteca Kaldi.

Os testes foram realizados apenas em dispositivos móveis com o sistema operacional Android, sendo a versão 4.1 a mínima suportada. Para a implementação, utilizou-se a linguagem de programação Java e NDK[21] (Android *Native Development Kit*, em português: kit de desenvolvimento nativo Android). O uso do NDK facilitou a compilação do código-fonte em C da biblioteca. O código-fonte do aplicativo está disponível no GitLab[22]

A Figura 12 apresenta a tela do aplicativo, e como pode-se observar apenas uma mensagem avisando que o aplicativo está: (*i*) inicializando o aplicativo: copiando os modelos da biblioteca para a memória interna do dispositivo; e (*ii*) gerando o arquivo de testes: executando a biblioteca com os arquivos de testes dos corpora de voz e gerando os arquivos de resultados.

Figura 12 - Tela da versão móvel em Android

[20] Link para o tutorial: http://jcsilva.github.io/2017/03/18/compile-kaldi-android/
[21] NDK é um conjunto de ferramentas que permitem usar código C e C++ em aplicativos Android.
[22] Link do GitLab da versão móvel: https://gitlab.com/lucasdebatin/ocsr-android-native

Ao finalizar os testes é gerado um texto com os seguintes dados de resultados para cada corpora: (*i*) percentual de uso do processador inicial, para verificar se o processador não está sobrecarregado com outro processo; (*ii*) a quantidade de memória disponível, indicador de memória cheia e a quantidade de memória total do dispositivo; (*iii*) percentual de bateria no início e fim dos testes; (*iv*) a data e hora de início e fim; e (*v*) o uso do processador (máximo, mínimo e médio) e de memória (máxima, mínima e média) durante a execução dos testes.

O percentual de uso do processador é obtido utilizando: (*i*) o comando "top" em versões menores que a versão 8 do Android; e (*ii*) o comando "ps"[23] nas versões 8 e 9 do Android. A quantidade de memória e bateria do dispositivo são capturadas utilizando classes do Java.

Para os testes em dispositivos móveis foi necessário reduzir a quantidade de arquivos de testes do corpus Constituição Federal de 126 para 30, pois em alguns dispositivos o tempo gasto para o processamento ultrapassava 30 minutos. Essa redução foi necessária para tornar os testes mais atrativos e rápidos para os voluntários da pesquisa. Além disso, o aplicativo foi disponibilizado na Play Store[24] para facilitar a sua instalação nos dispositivos de teste.

4.5 CONSIDERAÇÕES

Esse capítulo apresentou as etapas que foram necessárias para o desenvolvimento do aplicativo Android que realiza o reconhecimento off-line de voz contínuo do português brasileiro. As bibliotecas foram instaladas e os modelos acústicos criados conforme a documentação de cada biblioteca. Os códigos-fonte desse trabalho foram disponibilizados ao público para servir de suporte para pesquisas futuras na área ou para a reprodução desse estudo.

A dificuldade de implementar a biblioteca Kaldi no Android foi um dos principais contratempos do trabalho, visto que isso atrasou o início da etapa de testes do aplicativo. Os motivos para isso foram levantados, destacando-se a necessidade de permissão de "superusuário" nos dispositivos móveis.

[23] Exibe os dados sobre os processos em execução no dispositivo.
[24] Link de download Play Store: https://play.google.com/store/apps/details?id=com.debatin.ocsr_android_native

Os corpora utilizados nesse estudo diferem em tamanho e conteúdo dos empregados nos artigos selecionadas na revisão sistemática da literatura, porém isso não comprometeu os resultados. Os cálculos de desempenho foram obtidos por meio de comandos e classes já existentes. No capítulo seguinte são apresentados e discutidos os resultados obtidos.

5 RESULTADOS

Esse capítulo apresenta e discute os resultados do projeto, permitindo avaliar a sua contribuição, o alcance dos seus objetivos e as respostas para as hipóteses de pesquisa. Na seção 5.1 são apresentadas as melhores configurações de treinamento para cada biblioteca. A seção 5.2 apresenta o desempenho da melhor configuração de cada biblioteca em um computador desktop. Na seção 5.3 apresenta o desempenho da biblioteca selecionada em diversos dispositivos móveis. Por fim, na seção 5.4 são apresentadas algumas considerações sobre os resultados obtidos. O Quadro 12 apresenta a configuração do computador desktop utilizado para o treinamento e para o teste da versão desktop.

Quadro 12 - Configuração do computador desktop utilizado

Processador	Intel Core i5-7200U 2.50GHz
Memória	16 GB DDR4 2400MHz
Sistema operacional	Linux Antergos de 64 bits
Versão do Java	1.8.0_192
Versão do compilador GCC	8.2.1

5.1 MELHORES CONFIGURAÇÕES DE TREINAMENTO

Esta seção apresenta as métricas de avaliação obtidas para cada arquivo de configuração elaborado. Com base no percentual das métricas foram escolhidas as melhores para realizar testes de desempenho em um computador desktop. As subseções abaixo apresentam os resultados obtidos para cada biblioteca.

Para o cálculo das métricas de avaliação SER (*Sentence Error Rate*, em português: taxa de erro de sentença) e WER (*Word Error Rate*, em português: taxa de erro de palavras) utilizou-se os arquivos de testes dos corpora: (*i*) LaPS Benchmark, que contém 60 frases e 614 palavras; e (*ii*) Constituição Federal, que contém 126 frases e 7073 palavras. Os cálculos dessas métricas são realizados por cada biblioteca.

Cada biblioteca possuía 10 configurações com dois conjuntos diferentes de modificações, por isso foram selecionadas as duas melhores para cada um. O principal motivo para isso é analisar o desempenho de cada conjunto de configuração, evitando selecionar apenas os melhores resultados no geral, pois eles podem ser os piores em desempenho.

5.1.1 CMUSphinx

O Quadro 13 apresenta os valores obtidos por cada configuração no corpus LaPS Benchmark. A coluna ID é um identificador da configuração e servirá como referência para os resultados da seção 5.2. Nesse trabalho, a unidade dos valores do tempo de duração foi hora, minuto e segundo (hh:mm:ss). As especificações de configurações de cada ID estão no APÊNDICE B.

Quadro 13 - Configurações da biblioteca CMUSphinx e do corpus LaPS Benchmark

ID	**WER**	**SER**	**Duração**
1	49,4%	93,3%	00:03:08
2	**9,6%**	**55%**	**00:03:01**
3	**9,1%**	**56,7%**	**00:03:17**
4	11,4%	61,7%	00:05:29
5	22,1%	68,3%	00:08:21
6	38%	81,7%	00:07:53
7	8,8%	41,7%	00:04:48
8	**6,7%**	**45%**	**00:05:37**
9	**6,2%**	**35%**	**00:09:13**
10	11,9%	50%	00:11:39

De acordo com o Quadro 13 pode-se notar que o as melhores configurações obtidas possuem um WER inferior a 14%, isto é, atendem a um dos requisitos desse trabalho. Além disso, pode-se observar que as melhores métricas de avaliação são provenientes de configurações cujo os valores de configuração alterados são próximos dos valores de configuração padrão da biblioteca. A duração total do treinamento de todas as configurações foi em torno de 1 hora. Já o Quadro 14 apresenta os valores obtidos no corpus Constituição Federal.

Quadro 14 - Configurações da biblioteca CMUSphinx e do corpus Constituição Federal

ID	**WER**	**SER**	**Duração**
1	14,5%	96,8%	00:22:56
2	5%	77,8%	00:22:42
3	**3,2%**	**75,4%**	**00:27:17**
4	**4,6%**	**81%**	**00:30:29**
5	5,6%	81,7%	00:32:18
6	10,8%	90,5%	00:40:43
7	3%	70,6%	00:43:36
8	**2,2%**	**59,5%**	**00:56:03**
9	**2%**	**62,7%**	**01:17:39**
10	3,4%	67,5%	01:24:58

Os resultados obtidos no Quadro 14 demonstram que os testes com apenas um locutor possuem maior precisão (WER) do que com vários locutores, do Quadro 13. Quase todas as configurações possuem um WER inferior a 14%, entretanto apresentaram um alto valor na métrica SER, isto é, muitas frases geradas com um ou mais erros. Isso deve-se ao fato de que cada áudio de teste do corpus tem duração média de 30 segundos, ou seja, são frases extensas. A duração total do treinamento de todas as configurações da biblioteca ultrapassou 6 horas, e um dos motivos para isso também é a duração média dos arquivos de teste do corpus.

5.1.2 HTK

A biblioteca HTK (*Hidden Markov Models Toolkit*, em português: kit de ferramentas dos modelos ocultos de Markov) apresentou os piores resultados, pois com qualquer configuração e em qualquer corpus não foi possível ter um WER abaixo de 80%. O Quadro 15 apresenta os valores das métricas de avaliação obtidos pela biblioteca no corpus LaPS Benchmark. As especificações de configurações de cada ID estão no APÊNDICE C. A coluna Tipo representa o nome da classe utilizada para gerar os valores de saída. O APÊNDICE E apresenta os valores obtidos pelos demais tipos utilizados.

Quadro 15 - Configurações da biblioteca HTK e do corpus LaPS Benchmark

ID	**Tipo**	**WER**	**SER**	**Duração**
1	HVite (2-grama)	94,95%	100%	00:05:19
2	**HVite (2-grama)**	**92,18%**	**100%**	**00:05:33**
3	HVite (2-grama)	93,32%	100%	00:05:25
4	**HVite (2-grama)**	**92,35%**	**100%**	**00:05:21**
5	HVite (2-grama)	93,49%	100%	00:05:25
6	HVite (2-grama)	95,44%	100%	00:05:21
7	HVite (2-grama)	93,00%	100%	00:05:25
8	HVite (2-grama)	92,83%	100%	00:05:19
9	**HVite (2-grama)**	**92,83%**	**100%**	**00:05:18**
10	**HVite (2-grama)**	**92,18%**	**100%**	**00:05:17**

Ao realizar uma comparação com os resultados obtidos pela biblioteca CMUSphinx no mesmo corpus (Quadro 13) pode-se observar que os resultados do Quadro 15 foram insignificantes. Além disso, ao comparar as frases dos arquivos de teste com as frases geradas pela biblioteca pode-se notar que todas possuíram um ou mais erros. A duração média de treinamento foi de 5 minutos e 22 segundos para cada configuração. O Quadro 16 apresenta os resultados obtidos no corpus Constituição Federal.

Quadro 16 - Configurações da biblioteca HTK e do corpus Constituição Federal

ID	Tipo	WER	SER	Duração
1	HVite (2-grama)	91,98%	100%	02:08:11
2	HVite (2-grama)	89,64%	100%	02:12:03
3	**HVite (2-grama)**	**86,17%**	**100%**	**02:09:21**
4	HVite (2-grama)	86,47%	100%	02:06:09
5	**HVite (2-grama)**	**84,97%**	**100%**	**02:06:32**
6	HVite (2-grama)	93,85%	100%	01:58:51
7	HVite (2-grama)	88,60%	100%	02:03:42
8	**HVite (2-grama)**	**83,84%**	**100%**	**02:08:38**
9	**HVite (2-grama)**	**82,47%**	**100%**	**02:11:46**
10	HVite (2-grama)	84,21%	100%	02:19:31

Os valores obtidos no Quadro 16 apresentaram melhores resultados, porém longe do WER de 14%, que é um dos requisitos desse trabalho. Além disso, todas as frases geradas também possuíram um ou mais erros. O tempo de treinamento médio foi de 2 horas e 8 minutos para cada configuração.

5.1.3 Kaldi

O Quadro 17 apresenta os valores obtidos por cada configuração da biblioteca Kaldi no corpus LaPS Benchmark. As especificações de configurações de cada ID estão no APÊNDICE D. Essa biblioteca também possui diversas classes que são utilizadas para gerar os valores de saída, e as melhores foram representadas na coluna Tipo. O APÊNDICE F apresenta os valores obtidos pelos demais tipos utilizados.

Quadro 17 - Configurações da biblioteca Kaldi e do corpus LaPS Benchmark

ID	Tipo	WER	SER	Duração
1	**MLP**	**5,05%**	**41,67%**	**00:49:17**
2	**RNN**	**2,61%**	**21,67%**	**02:24:06**
3	RNN	6,19%	41,67%	14:18:37
4	RNN	8,14%	51,67%	92:54:37
5	N/A	N/A	N/A	N/A
6	**tri3b**	**5,05%**	**40%**	**00:23:41**
7	mono0a	7,17%	38,33%	00:45:13
8	tri1	6,68%	43,33%	00:42:41
9	**tri1**	**5,37%**	**31,67%**	**00:38:46**
10	tri1	6,51%	41,67%	00:40:57

De acordo com o Quadro 17 pode-se perceber que o uso de RNAs (Redes Neurais Artificiais) apresentaram os melhores resultados, entretanto o único problema é o elevado tempo gasto para o treinamento. Por isso, a configuração de ID 5 não foi treinada, pois levaria mais de 100 horas. Os

melhores resultados obtidos foram utilizando RNN (*Recurrent Neural Network*, em português: rede neural recorrente) e MLP (*Multilayer Perceptron*, em português: *perceptron* multicamadas). Além disso, as classes tri3b e tri1 também apresentaram bons resultados. O Quadro 18 apresenta os resultados da biblioteca no corpus Constituição Federal.

Quadro 18 - Configurações da biblioteca Kaldi e do corpus Constituição Federal

ID	**Tipo**	**WER**	**SER**	**Duração**
1	MLP	1,44%	46,03%	07:45:17
2	RNN	0,98%	41,27%	26:38:27
3	**RNN**	**0,93%**	**38,89%**	**166:47:14**
4	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A
6	**tri3b**	**1,48%**	**46,03%**	**01:50:37**
7	tri1	1,61%	49,21%	01:54:36
8	**tri1**	**1,34%**	**38,89%**	**02:02:11**
9	tri1	1,75%	46,03%	05:32:35
10	tri1	1,80%	48,41%	07:09:14

No Quadro 18 pode-se observar que foram gerados apenas os três treinamentos para as RNAs, pois o tempo gasto de treinamento para o ID 3 ultrapassou 150 horas. Por isso, foram selecionadas apenas três configurações, e não quatro como nas outras bibliotecas. O uso de RNN e as classes tri3b e tri1 apresentaram os melhores resultados das métricas de avaliação.

5.2 ANÁLISE DO DESEMPENHO EM DESKTOPS

Os testes foram realizados em apenas um computador desktop, conforme configuração já destacada no Quadro 12. Esses testes foram realizados sem conexão com a internet, sem nenhum software sendo executado em paralelo e utilizando apenas os arquivos de testes dos corpora.

O Quadro 19 apresenta os resultados obtidos no corpus LaPS Benchmark. Para o cálculo do fator xRT (*Real Time Factor*, em português: fator em tempo real) utilizou-se a duração total dos arquivos de teste do corpus, que é 00:04:36 (276 segundos). Além disso, foram colocados os valores da métrica de avaliação WER para facilitar a comparação da análise de desempenho.

Quadro 19 - Desempenho das bibliotecas no corpus LaPS Benchmark

Biblioteca	**ID**	**WER**	**Duração**	**xRT**	**Processador**	**Memória**
CMUSphinx	**2**	**9,6%**	**00:00:11**	**0,040**	**Máx.: 100,00%** **Méd.: 97,50%** **Mín.: 86,70%**	**Máx.: 0,50%** **Méd.: 0,50%** **Mín.: 0,50%**
CMUSphinx	3	9,1%	00:00:19	0,069	Máx.: 106,70%[25] Méd.: 98,09% Mín.: 87,50%	Máx.: 0,50% Méd.: 0,50% Mín.: 0,50%
CMUSphinx	8	6,7%	00:00:36	0,130	Máx.: 106,70% Méd.: 98,69% Mín.: 87,50%	Máx.: 0,50% Méd.: 0,50% Mín.: 0,50%
CMUSphinx	9	6,2%	00:01:01	0,221	Máx.: 106,70% Méd.: 99,20% Mín.: 93,30%	Máx.: 0,50% Méd.: 0,50% Mín.: 0,50%
HTK	2	92,18%	00:02:44	0,594	Máx.: 106,70% Méd.: 99,17% Mín.: 87,50%	Máx.: 0,60% Méd.: 0,60% Mín.: 0,50%
HTK	4	92,35%	00:02:43	0,590	Máx.: 106,70% Méd.: 98,68% Mín.: 87,50%	Máx.: 0,60% Méd.: 0,60% Mín.: 0,50%
HTK	9	92,83%	00:02:44	0,594	Máx.: 106,70% Méd.: 99,00% Mín.: 87,50%	Máx.: 0,60% Méd.: 0,60% Mín.: 0,50%
HTK	**10**	**92,18%**	**00:02:41**	**0,583**	**Máx.: 106,70%** **Méd.: 98,79%** **Mín.: 87,50%**	**Máx.: 0,60%** **Méd.: 0,60%** **Mín.: 0,50%**
Kaldi	1	5,05%	00:02:18	0,500	Máx.: 100,00% Méd.: 98,26% Mín.: 80,00%	Máx.: 0,60% Méd.: 0,43% Mín.: 0,30%
Kaldi	2	2,61%	00:03:31	0,764	Máx.: 100,00% Méd.: 97,79% Mín.: 66,70%	Máx.: 0,40% Méd.: 0,39% Mín.: 0,20%
Kaldi	**6**	**5,05%**	**00:00:14**	**0,050**	**Máx.: 100,00%** **Méd.: 91,92%** **Mín.: 80,00%**	**Máx.: 0,20%** **Méd.: 0,20%** **Mín.: 0,20%**
Kaldi	9	5,37%	00:00:19	0,069	Máx.: 100,00% Méd.: 96,73% Mín.: 80,00%	Máx.: 0,30% Méd.: 0,21% Mín.: 0,20%

Para cada biblioteca foi selecionada a melhor configuração. No Quadro 19 pode-se observar que as bibliotecas CMUSphinx e Kaldi possuíram os melhores resultados, porém a biblioteca Kaldi

[25] O comando “top”, por padrão, exibe a porcentagem de uma única CPU, isto é, em computadores com vários núcleos pode-se ter porcentagens maiores que 100%.

se destacou, pois obteve: (*i*) o menor percentual WER; (*ii*) a menor média de uso do processador; e (*iii*) a menor média de uso de memória.

O Quadro 20 apresenta os resultados de desempenho obtidos utilizando o corpus Constituição Federal. Para o cálculo do fator xRT utilizou-se a duração total dos arquivos de teste do corpus, que é 00:53:06 (3.186 segundos).

Quadro 20 - Desempenho das bibliotecas no corpus Constituição Federal

Biblioteca	**ID**	**WER**	**Duração**	**xRT**	**Processador**	**Memória**
CMUSphinx	**3**	**3,2%**	**00:02:29**	**0,047**	**Máx.: 106,70%** **Méd.: 99,27%** **Mín.: 86,70%**	**Máx.: 0,50%** **Méd.: 0,50%** **Mín.: 0,50%**
CMUSphinx	4	4,6%	00:04:06	0,077	Máx.: 106,70% Méd.: 98,35% Mín.: 87,50%	Máx.: 0,50% Méd.: 0,50% Mín.: 0,50%
CMUSphinx	8	2,2%	00:04:29	0,084	Máx.: 106,70% Méd.: 99,07% Mín.: 81,20%	Máx.: 0,50% Méd.: 0,50% Mín.: 0,50%
CMUSphinx	9	2%	00:07:45	0,146	Máx.: 106,70% Méd.: 98,92% Mín.: 81,20%	Máx.: 0,50% Méd.: 0,50% Mín.: 0,50%
HTK	3	86,17%	01:26:02	1,620	Máx.: 106,70% Méd.: 99,14% Mín.: 87,50%	Máx.: 0,70% Méd.: 0,70% Mín.: 0,50%
HTK	5	84,97%	01:25:52	1,617	Máx.: 106,70% Méd.: 99,20% Mín.: 87,50%	Máx.: 0,70% Méd.: 0,70% Mín.: 0,50%
HTK	8	83,84%	01:25:58	1,619	Máx.: 106,70% Méd.: 99,10% Mín.: 87,50%	Máx.: 0,70% Méd.: 0,70% Mín.: 0,50%
HTK	**9**	**82,47%**	**01:26:07**	**1,622**	**Máx.: 113,30%** **Méd.: 98,52%** **Mín.: 75,00%**	**Máx.: 0,70%** **Méd.: 0,70%** **Mín.: 0,50%**
Kaldi	3	0,93%	02:39:45	3,008	Máx.: 100,00% Méd.: 98,11% Mín.: 81,20%	Máx.: 1,00% Méd.: 0,95% Mín.: 0,70%
Kaldi	**6**	**1,48%**	**00:01:30**	**0,028**	**Máx.: 100,00%** **Méd.: 94,76%** **Mín.: 40,00%**	**Máx.: 0,30%** **Méd.: 0,20%** **Mín.: 0,20%**
Kaldi	8	1,34%	00:01:54	0,036	Máx.: 100,00% Méd.: 97,50% Mín.: 87,50%	Máx.: 0,40% Méd.: 0,37% Mín.: 0,30%

Para esse corpus também foi selecionada a melhor configuração de cada biblioteca. No Quadro 20 pode-se observar que as bibliotecas CMUSphinx e Kaldi possuíram os melhores

resultados. Nesse corpus a biblioteca Kaldi também se destacou, pois obteve: (*i*) o menor percentual WER; (*ii*) o menor valor xRT; (*iii*) a menor média de uso do processador; e (*iv*) a menor média de uso de memória.

De acordo com o Quadro 19 e o Quadro 20, as bibliotecas utilizaram mais recursos de processamento do que de memória RAM. Além disso, o uso de RNAs apresentou os melhores resultados (2,61% corpus LaPS Benchmark e 0,93% corpus Constituição Brasileira), entretanto, requer um grande custo computacional, em alguns casos ultrapassando duas horas de processamento, comprometendo assim o seu uso em dispositivos móveis.

A biblioteca Kaldi se destacou nos dois corpora, entretanto, a implementação dessa biblioteca em dispositivos móveis não obteve sucesso, como já visto na seção 4.4.2 Por isso, optou-se por utilizar a biblioteca CMUSphinx que, de acordo com o Quadro 19 e o Quadro 20, obteve resultados bem semelhantes aos da biblioteca Kaldi e um WER inferior a 14%.

5.3 ANÁLISE DO DESEMPENHO EM DISPOSITIVOS MÓVEIS

Para os testes em dispositivos móveis utilizou-se a biblioteca CMUSphinx. Os testes foram realizados em 11 dispositivos, com diferentes versões de Android (superiores a 4.1) e com diferentes configurações de hardware. Em cada dispositivo foi solicitado para: (*i*) desativar qualquer forma de conexão com a internet (*wi-fi*, dados móveis); (*ii*) retirar o celular da fonte de alimentação, se o mesmo estivesse; (*iii*) verificar se o dispositivo tinha no mínimo 30% de bateria; e (*iv*) fechar todos os aplicativos abertos.

Todos os dispositivos móveis testados possuíram um baixo uso de processador inicial e memória suficiente para realizar os testes. O Quadro 21 apresenta as configurações dos dispositivos móveis utilizados nos testes. A coluna ID é um identificador do dispositivo móvel e servirá como referência para os resultados do Quadro 22 e do Quadro 23.

Quadro 21 - Configuração dos dispositivos móveis utilizados nos testes

ID	**Dispositivo**	**Android**	**Configuração de hardware**
1	Samsung SM-T110	4.4.2	Processador: Dual-Core de 1.2 GHz Memória RAM: 1 GB
2	Samsung SM-J200BT	5.1.1	Processador: Quad-Core de 1.3 GHz Memória RAM: 1 GB
3	Quantum Quantum Fly	6.0	Processador: Deca-Core de 2.1 GHz

			Memória RAM: 3 GB
4	Samsung SM-J500M	6.0.1	Processador: Quad-Core de 1.2 GHz Memória RAM: 1,5 GB
5	LGE LG-M700	7.1.1	Processador: Octa-Core de 1.4 GHz Memória RAM: 2 GB
6	Xiaomi Redmi 4X	7.1.2	Processador: Quad-Core de 1.4 GHz Memória RAM: 3 GB
7	Motorola Moto Z (2)	8.0.0	Processador: Octa-Core de 2.35 GHz Memória RAM: 6 GB
8	Motorola XT1635-02	8.0.0	Processador: Octa-Core de 2 GHz Memória RAM: 3 GB
9	Motorola Moto G (5S)	8.1.0	Processador: Octa-Core de 1.4 GHz Memória RAM: 2 GB
10	Samsung SM-J610G	8.1.0	Processador: Quad-Core de 1.4 GHz Memória RAM: 3 GB
11	Xiaomi Mi A2	9.0.0	Processador: Octa-Core de 2.2 GHz Memória RAM: 4 GB

O Quadro 22 apresenta os resultados obtidos durante a execução do corpus LaPS Benchmark nos dispositivos móveis. Foram executados todos os arquivos de testes do corpus, com duração de 00:04:36 (276 segundos).

Quadro 22 - Desempenho do corpus LaPS Benchmark em dispositivos móveis

ID	**Duração**	**xRT**	**Bateria**	**Processador**	**Memória**
1	01:53:27	24,66	0%	Máx.: 46% Méd.: 32% Mín.: 25%	Máx.: 18,05 MB Méd.: 17,49 MB Mín.: 17,07 MB
2	00:01:11	0,257	0%	Máx.: 35% Méd.: 25% Mín.: 22%	Máx.: 27,41 MB Méd.: 27,22 MB Mín.: 27,04 MB
3	00:00:41	0,149	0%	Máx.: 49% Méd.: 45% Mín.: 47%	Máx.: 81,31 MB Méd.: 67,32 MB Mín.: 64,63 MB
4	00:00:46	0,167	1%	Máx.: 25% Méd.: 25% Mín.: 25%	Máx.: 42,13 MB Méd.: 41,45 MB Mín.: 40,93 MB
5	00:00:36	0,130	0%	Máx.: 25% Méd.: 21% Mín.: 20%	Máx.: 63,17 MB Méd.: 63,05 MB Mín.: 62,93 MB
6	00:00:34	0,123	0%	Máx.: 30% Méd.: 25% Mín.: 22%	Máx.: 50,62 MB Méd.: 50,43 MB Mín.: 50,27 MB
7	00:00:28	0,101	0%	Máx.: 11% Méd.: 6% Mín.: 3%	Máx.: 22,17 MB Méd.: 21,98 MB Mín.: 21,82 MB
8	00:00:27	0,978	0%	Máx.: 2%	Máx.: 53,57 MB

				Méd.: 1% Mín.: 1%	Méd.: 53,39 MB Mín.: 53,29 MB
9	00:00:43	0,156	0%	Máx.: 33% Méd.: 27% Mín.: 22%	Máx.: 49,34 MB Méd.: 43,81 MB Mín.: 41,90 MB
10	00:00:43	0,156	0%	Máx.: 4% Méd.: 3% Mín.: 3%	Máx.: 31,22 MB Méd.: 30,96 MB Mín.: 30,71 MB
11	00:00:10	0,036	0%	Máx.: 12% Méd.: 12% Mín.: 12%	Máx.: 27,40 MB Méd.: 27,30 MB Mín.: 27,21 MB

Com base no Quadro 22 pode-se perceber que em todos os dispositivos o uso médio de memória foi abaixo de 70 MB e o uso médio de processador foi abaixo de 50%. Pode-se observar também que o processamento dos arquivos de teste pela biblioteca não consome recursos da bateria do dispositivo.

Já o Quadro 23 apresenta os resultados obtidos durante a execução do corpus Constituição Federal. Na seção 4.4.2 foi descrito que para esse corpus foram utilizados apenas 30 arquivos de testes, com duração de 00:12:32 (752 segundos), alterando o cálculo do fator xRT.

Quadro 23 - Desempenho do corpus Constituição Federal em dispositivos móveis

ID	**Duração**	**xRT**	**Bateria**	**Processador**	**Memória**
1	02:44:02	13,08	1%	Máx.: 49% Méd.: 33% Mín.: 24%	Máx.: 22,46 MB Méd.: 21,52 MB Mín.: 20,47 MB
2	00:04:51	0,387	1%	Máx.: 25% Méd.: 20% Mín.: 23%	Máx.: 36,86 MB Méd.: 32,97 MB Mín.: 28,13 MB
3	00:02:55	0,233	1%	Máx.: 47% Méd.: 46% Mín.: 44%	Máx.: 81,13 MB Méd.: 74,97 MB Mín.: 68,83 MB
4	00:05:05	0,406	0%	Máx.: 25% Méd.: 24% Mín.: 24%	Máx.: 42,53 MB Méd.: 38,31 MB Mín.: 35,06 MB
5	00:02:11	0,174	1%	Máx.: 39% Méd.: 36% Mín.: 31%	Máx.: 65,42 MB Méd.: 59,80 MB Mín.: 56,02 MB
6	00:04:00	0,319	1%	Máx.: 35% Méd.: 25% Mín.: 16%	Máx.: 53,69 MB Méd.: 51,55 MB Mín.: 48,61 MB
7	00:00:28	0,037	0%	Máx.: 22% Méd.: 22% Mín.: 22%	Máx.: 26,67 MB Méd.: 26,67 MB Mín.: 26,67 MB

8	00:01:36	0,128	1%	Máx.: 11% Méd.: 9% Mín.: 5%	Máx.: 55,52 MB Méd.: 55,47 MB Mín.: 55,45 MB
9	00:01:58	0,157	1%	Máx.: 28% Méd.: 28% Mín.: 28%	Máx.: 53,10 MB Méd.: 53,04 MB Mín.: 52,95 MB
10	00:02:17	0,182	0%	Máx.: 20% Méd.: 20% Mín.: 20%	Máx.: 33,36 MB Méd.: 33,28 MB Mín.: 33,21 MB
11	00:00:32	0,043	0%	Máx.: 4% Méd.: 4% Mín.: 4%	Máx.: 30,51 MB Méd.: 30,51 MB Mín.: 30,51 MB

De acordo com o Quadro 23 também pode-se perceber que em todos os dispositivos: (*i*) o uso médio de memória foi abaixo de 80 MB; (*ii*) o uso médio de processador foi abaixo de 50%; e (*iii*) o processamento dos arquivos de teste não consumiu recursos da bateria dos dispositivos.

Haja visto que o valor do xRT é composto pelo tempo de processamento, desenvolveu-se o Quadro 24 que apresenta um comparativo entre a relação do processador utilizado com o valor xRT obtido para cada corpus. Esse quadro foi desenvolvido com base nas configurações de hardware dos dispositivos (Quadro 21) e com base nos valores do xRT do Quadro 22 e do Quadro 23.

Quadro 24 - Comparativo entre processador e valor xRT

Processador	**xRT LaPS Benchmak**	**xRT Constituição Federal**
Dual-Core de 1.2 GHz	24,66	13,08
Quad-Core de 1.2 GHz	0,167	0,406
Quad-Core de 1.3 GHz	0,257	0,387
Quad-Core de 1.4 GHz	0,123	0,319
Quad-Core de 1.4 GHz	0,156	0,182
Octa-Core de 1.4 GHz	0,130	0,174
Octa-Core de 1.4 GHz	0,156	0,157
Octa-Core de 2 GHz	0,978	0,128
Octa-Core de 2.2 GHz	0,036	0,043
Octa-Core de 2.35 GHz	0,101	0,037
Deca-Core de 2.1 GHz	0,149	0,233

Com base no Quadro 24, pode-se observar que os valores do xRT são baixos em quase todos os processadores, isto é, praticamente todos possuíram valores abaixo de 1. Entretando, o tempo para processamento de cada áudio dos corpora de testes em um processador “Dual-Core de 1.2 GHz” foi alto, pois esse processador é antigo e possui somente dois núcleos com apenas 1.2 GHz cada.

5.4 CONSIDERAÇÕES

Nesse capítulo foram apresentados e discutidos os resultados da análise de desempenho do reconhecimento de voz proposto em dispositivos móveis. Para isso foram selecionadas as configurações de treinamento que apresentaram as melhores métricas de avaliação, porém algumas configurações não foram treinadas. Os motivos para isso foram discutidos, destacando-se o tempo elevado para gerar o treinamento nessas configurações.

A versão para desktop foi útil para encontrar a biblioteca que apresentou o melhor desempenho, porém a mesma não pode ser implementada em dispositivos móveis. A solução para esse problema foi a substituição da biblioteca Kaldi pela CMUSphinx, visto que as duas apresentaram resultados muito próximos.

Os resultados obtidos na análise de desempenho da versão desktop demonstraram que o uso de RNAs, no reconhecimento de voz conínuo, requer um grande custo computacional, em contrapartida apresentaram os melhores resultados. O melhor custo-beneficio entre precisão e desempenho foram obtidos por meio do uso de HMMs.

Os testes foram realizados em diversos dispositivos móveis, e com isso pode-se observar que em todas as versões do Android testadas a biblioteca realizou com êxito o reconhecimento off-line de voz contínuo. Vale destacar que as versões mais recentes do Android realizaram o processamento dos arquivos de testes em tempo real, devido as modernas configurações de hardware dos dispositivos móveis.

Foram realizados experimentos com dois corpora de voz, um com vários locutores e o outro com apenas um locutor. O corpus com apenas um locutor apresentou os melhores resultados para a taxa WER. Sendo assim, pode-se confirmar que realmente a presença de diversos locutores é um fator de complexidade do reconhecimento de voz contínuo.

6 CONCLUSÕES

Essa pesquisa teve como objetivo principal o desenvolvimento do reconhecimento off-line de voz contínuo do português brasileiro e a análise do seu desempenho em diversos dispositivos móveis com o sistema operacional Android. Visando atender esse objetivo, foram definidos quatro objetivos específicos.

Por meio de uma revisão sistemática da literatura, que compôs o estado da arte, foram levantadas as principais técnicas de extração de características do sinal do áudio e de implementação dos modelos acústico e de linguagem, que são utilizadas no reconhecimento de voz contínuo. Desse modo atendeu-se totalmente o objetivo específico 1. Realizou-se um estudo teórico sobre essas técnicas para identificar características importantes para o desenvolvimento do reconhecimento de voz proposto. Além disso, por meio da leitura dos artigos também foram selecionadas as bibliotecas mais utilizadas para auxiliar na implementação do reconhecimento de voz contínuo, tais como CMUSphinx, Kaldi e HTK.

O objetivo específico 2 tratou de encontrar as melhores configurações de treinamento das bibliotecas para obter o melhor custo-benefício entre desempenho e precisão. Para atender esse objetivo foram criados 30 arquivos de configurações diferentes. Esses arquivos foram executados para os dois corpora utilizados no trabalho, e para cada um foram obtidas as métricas de avaliação. A biblioteca Kaldi apresentou três configurações que demandaram um tempo maior de treinamento, e por esse motivo foram omitidas dos resultados. Por isso, esse objetivo foi parcialmente atendido.

As melhores configurações foram selecionadas com base nos valores das métricas de avaliação WER (*Word Error Rate*, em português: taxa de erro de palavras) obtidos nos treinamentos, atendendo assim ao objetivo específico 4. Essas configurações foram utilizadas na versão desktop visando encontrar a biblioteca que possuía os melhores resultados de desempenho nos testes realizados em um computador desktop. A biblioteca Kaldi apresentou os melhores resultados, porém não foi possível implementá-la nos dispositivos móveis devido a restrições de permissionamento. Por esse motivo utilizou-se a biblioteca CMUSphinx que obteve resultados semelhantes ao Kaldi. A biblioteca HTK apresentou os piores resultados, pois o menor valor WER obtido foi maior que 80%, sendo muito maior que o WER máximo desejdo de 14% deste trabalho.

A biblioteca CMUSphinx foi executada nos dispositivos móveis por meio de um aplicativo. Buscando atender ao objetivo específico 3 foram adicionados, nesse aplicativo, métodos responsáveis por capturar o uso do processador, da memória e da bateria enquanto os arquivos de testes dos corpora eram executados. Os testes de desempenho foram realizados em diversos dispositivos móveis, com diferentes versões de Android e de hardware. Esses testes demonstraram que em configurações de hardware com maiores recursos computacionais é possível aplicar essa biblioteca em aplicativos para reconhecimento off-line de voz contínuo em tempo real. Já nas configurações de hardware com recursos computacionais limitados somente é possível aplicar essa biblioteca em aplicativos que realizam transcrições de áudio de entrevistas, por exemplo. Todos os testes foram realizados sem conexão com a internet, buscando comprovar que este reconhecimento de voz contínuo desenvolvido funciona com êxito em ambientes que não possuem internet.

Os resultados desses testes foram responsáveis por atender de maneira positiva a hipótese da pesquisa de que "é possível aplicar as técnicas mais utilizadas nos dispositivos móveis", pois a biblioteca CMUSphinx executou com êxito as técnicas HMM (*Hidden Markov Models*, em português: modelos ocultos de Markov) e n-grama nos dispositivos móveis. Além disso, o estado da arte também destacou o uso das RNA (Redes Neurais Artificiais), que possuíram os melhores resultados das métricas de avaliação, destacando os resultados obtidos pelas RNN (*Recurrent Neural Network*, em português: rede neural recorrente). Entretanto, resultou num elevado custo computacional para processar os áudios de teste dos corpora.

A hipótese de que "é possível ter um WER abaixo de 14% para o reconhecimento de voz do português brasileiro ao utilizar as técnicas que funcionam corretamente em dispositivos móveis" também foi atendida de maneira positiva, visto que o valor da taxa WER das melhores configurações da biblioteca CMUSphinx utilizadas no aplicativo foi de: (*i*) 9,6% para o corpus LaPS Benchmark; e (*ii*) 3,2% para o corpus Constituição Federal.

6.1 CONTRIBUIÇÕES

A principal contribuição dessa pesquisa foi o desenvolvimento do reconhecimento off-line de voz contínuo para o português brasileiro e sua implementação em computadores desktop e em dispositivos móveis. Esse reconhecimento de voz desenvolvido poderá ser utilizado em softwares e aplicativos: (*i*) que auxiliam na comunicação de pessoas com deficiência; (*ii*) empresariais que

agilizam o trabalho dos funcionários; e (*iii*) que necessitam desta função em áreas sem conexão com a internet.

Outra contribuição relevante dessa pesquisa foi o desenvolvimento de uma ferramenta para a preparação dos modelos acústicos do português brasileiro para as bibliotecas CMUSphinx, HTK (*Hidden Markov Models Toolkit*, em português: kit de ferramentas dos modelos ocultos de Markov) e Kaldi. Inicialmente foram utilizados os corpora LaPS Benchmark e Constituição Federal, porém é possível preparar os modelos acústicos para qualquer corpus de voz.

Além disso, pode-se destacar outras contribuições do trabalho: (*i*) levantamento, por meio de uma revisão sistemática da literatura, das principais técnicas e bibliotecas utilizadas no reconhecimento de voz contínuo; (*ii*) comparativo entre as configurações de treinamento para cada biblioteca visando a melhor métrica de avaliação; (*iii*) análise de desempenho das bibliotecas em um computador desktop; e (*iii*) análise de desempenho da biblioteca CMUSphinx em diversos dispositivos móveis.

Durante o desenvolvimento dessa pesquisa foram publicados artigos que estão relacionados ao tema do projeto. O Quadro 25 apresenta os três artigos que foram aceitos para publicação.

Quadro 25 - Artigos aceitos para publicação

Tipo	**Qualis**	**Ano**	**Autores**	**Título**	**Evento**
Resumo Expandido	B2	2017	Lucas Debatin; Aluizio Haendchen Filho; Rudimar L. S. Dazzi	O Problema do Reconhecimento de Voz Offline em Dispositivos Móveis: em Busca de uma Abordagem Racional	XXIII Simpósio Brasileiro de Sistemas Multimídia e Web
Artigo Completo	B2	2018	Lucas Debatin; Aluizio Haendchen Filho; Rudimar L. S. Dazzi	*Offline Speech Recognition Development: A Systematic Review of the Literature*	XX *International Conference on Enterprise Information Systems*
Resumo Expandido	B4	2019	Lucas Debatin; Aluizio Haendchen Filho; Rudimar L. S. Dazzi	Reconhecimento Off-line de Voz Contínuo para Dispositivos Móveis: Uma Análise Comparativa de Métricas de Avaliação	X *Computer on the Beach*

6.2 TRABALHOS FUTUROS

Ao longo do desenvolvimento deste trabalho, puderam ser identificadas algumas possibilidades de melhoria e de continuação a partir de futuras pesquisas, as quais incluem:

- A criação de um novo corpus de voz com vários locutores, para o português brasileiro e com no mínimo dez horas de duração, a fim de se obter resultados de desempenho fidedignos a um cenário real de aplicação, isto é, sem um vocabulário restrito;

- A implementação da biblioteca Kaldi em dispositivos móveis sem a necessidade de permissão de “superusuário”, para isso serão necessários compilar os arquivos da biblioteca de outra maneira;

- A redução do custo computacional (processamento) exigido pelas RNAs que realizam o reconhecimento de voz contínuo para serem aplicadas nos dispositivos móveis. Para isso será necessário encontrar e implementar RNAs que possuam um bom desempenho;

- A análise de desempenho do reconhecimento off-line de voz contínuo em dispositivos móveis com o sistema operacional IOS. Para isso, será possível utilizar a mesma biblioteca utilizada no Android;

- A comparação das métricas de avaliação do reconhecimento off-line de voz contínuo desenvolvido com as APIs existentes no mercado. Para isso será necessário instalar as APIs de reconhecimento de voz contínuo e executar os arquivos de teste dos corpora;

- O teste do reconhecimento off-line de voz contínuo em sistemas embarcados, a fim de obter o desempenho em situações com recursos de processamento e memória limitados.

REFERÊNCIAS

ABUSHARIAH, M. A. TAMEEM V1.0: speakers and text independent Arabic automatic continuous speech recognizer. **International Journal of Speech Technology**, Nova Iorque, v. 20, n. 2, p. 261-280, jun. 2018.

AGARAP, A. F. M. **Deep Learning using Rectified Linear Units (ReLU)**. 2018. Disponível em: <https://arxiv.org/abs/1803.08375>. Acesso em: 28 jun. 2018.

ALENCAR, V. F. S. **Atributos e Domínios de Interpolação Eficientes em Reconhecimento de Voz Distribuído**. 2005. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

BAUER, M. W.; AARTS, B. A construção do corpus: um princípio para a coleta de dados qualitativos. In: BAUER, M. W., GASKELL, G. (Orgs.). **Pesquisa qualitativa com texto, imagem e som**. 11.ed. Petrópolis: Vozes, 2013. p. 39-63.

BENYON, D. **Interação humano-computador**. 2. ed. São Paulo: Pearson Prentice Hall, 2011.

CASA CIVIL. **Decreto nº 5.296 de 2 de dezembro de 2004**. 2004. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm>. Acesso em: 11 mai. 2018.

CMUSPHINX. **Open source speech recognition toolkit**. 2019. Disponível em: <https://cmusphinx.github.io>. Acesso em: 06 jan. 2019.

COPPIN, B. **Inteligência artificial**. Rio de Janeiro: LTC, 2010.

DAHL, G. E.; YU, D.; DENG, L.; ACERO, A. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. **IEEE Transactions on Audio, Speech, and Language Processing**, Piscataway, v. 20, n. 1, p. 30-42, jan. 2012.

DEBATIN, L.; HAENDCHEN FILHO, A.; DAZZI, R. L. S. O Problema do Reconhecimento de Voz Offline em Dispositivos Móveis: em Busca de uma Abordagem Racional. In: SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E WEB – WEBMEDIA, 23., 2017, Gramado. **Anais dos Workshops e Pôsteres do Webmedia**. Porto Alegre: Sociedade Brasileira de Computação, 2017. p. 229-230.

DEBATIN, L.; HAENDCHEN FILHO, A.; DAZZI, R. L. S. Offline Speech Recognition Development: A Systematic Review of the Literature. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS – ICEIS, 20., 2018, Funchal. **Proceedings…** Setúbal: SciTePress, 2018. p. 551-558.

DIMITRIADIS, D.; BOCCHIERI, E. Use of micro-modulation features in large vocabulary continuous speech recognition tasks. **IEEE/ACM Transactions on Audio, Speech and Language Processing**, Piscataway, v. 23, n. 8, p. 1348-1357, ago. 2015.

DINIZ, P. S. R.; SILVA, E. A. B.; NETTO, S. L. **Processamento Digital de Sinais:** Projeto e Análise de Sistemas. 2. ed. Porto Alegre: Bookman, 2014.

FERREIRA, M. V. G.; SOUZA, J. F. Use of Automatic Speech Recognition Systems for Multimedia Applications. In: SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E WEB – WEBMEDIA, 23., 2017, Gramado. **Anais dos Workshops e Pôsteres do Webmedia**. Porto Alegre: Sociedade Brasileira de Computação, 2017. p. 139-176.

GEORGESCU, A.; CUCU, H.; BURILEANU, C. SpeeD's DNN approach to Romanian speech recognition. In: INTERNATIONAL CONFERENCE ON SPEECH TECHNOLOGY AND HUMAN-COMPUTER DIALOGUE – SPED, 9., 2017, Bucharest. **Proceedings…** Piscataway: IEEE, 2017. p. 1-8.

GORDILLO, C. D. A**. Reconhecimento de Voz Contínua Combinando os Atributos MFCC e PNCC com Métodos de Robustez SS, WD, MAP e FRN.** 2013. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

GRAVES, A.; JAITLY, N.; MOHAMED, A. Hybrid speech recognition with Deep Bidirectional LSTM. In: IEEE WORKSHOP ON AUTOMATIC SPEECH RECOGNITION AND UNDERSTANDING, 8., 2013, Olomouc. **Proceedings…** Piscataway: IEEE, 2013. p. 273-278.

GROSSMAN, R. L. The Case for Cloud Computing. **IT Professional**, Piscataway, v. 11, n. 2, p. 23-27, mar. 2009.

HAYKIN, S. **Redes Neurais:** Princípios e prática. 2.ed. São Paulo: Bookman, 2001.

HEARST, M. A. 'Natural' search user interfaces. **Communications of the ACM**, New York, v. 54, n. 11, p. 60-67, nov. 2011.

HTK. **HTK Speech Recognition Toolkit**. 2019. Disponível em: <http://htk.eng.cam.ac.uk>. Acesso em: 10 jan. 2019.

HUANG, X.; DENG, L. An overview of modern speech recognition. In: INDURKHYA, N., DAMERAU, F. J. (Ed.). **Handbook of Natural Language Processing**. 2.ed. Londres: Chapman and Hall/CRC, 2010. p. 339-366.

HUCHE, F. L.; ALLALI, A. **A voz**: anatomia e fisiologia dos órgãos da voz e da fala. 2. ed. Porto Alegre: Artmed, 1999.

IBGE. **Pesquisa nacional por amostra de domicílios**. 2016. Disponível em: <http://www.ibge.gov.br/>. Acesso em: 11 mai. 2018.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing:** An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 2. ed. Upper Saddle River: Prentice-Hall, 2008.

KALDI. **Kaldi ASR**. 2019. Disponível em: <http://kaldi-asr.org>. Acesso em: 08 jan. 2019.

KIPYATKOVA, I.S.; KARPOV, A.A. A study of neural network Russian language models for automatic continuous speech recognition systems. **Automation and Remote Control**, Nova Iorque, v. 78, n. 5, p. 858-867, mai. 2017.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering.** 2007. Disponível em: <https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf>. Acesso em: 15 mai. 2018.

LALEYE, F. A. A; BESACIER, L.; EZIN, E. C.; MOTAMED, C. First automatic fongbe continuous speech recognition system: Development of acoustic models and language models. In: FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS - FEDCSIS, 5., 2016, Gdansk. **Proceedings...** Piscataway: IEEE, 2016. p. 477-482.

LEE, K.; HON, H.; REDDY, R. An Overview of the SPHINX Speech Recognition System. **IEEE Transactions on Acoustic Speech, and Signal Processing**, Piscataway, v. 38, n. 1, p. 35-45, jan. 1990.

LUGER, G. F. **Inteligência artificial**. 6. ed. São Paulo: Pearson, 2013.

MAIA, E. M. **No reino da fala:** a linguagem e seus sons. 4. ed. São Paulo: Ática, 1999.

MELO, D. B. **Um Sistema de Reconhecimento de Comandos de Voz Utilizando a Rede Neural ELM**. 2011. Monografia (Bacharelado em Engenharia de Teleinformática) – Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará, Fortaleza.

MIKOLOV, T.; KARAFIAT, M.; BURGET, L.; CERNOCKY, J.; KHU-DANPUR, S. Recurrent neural network based language model. In: ANNUAL CONFERENCE OF THE INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION – INTERSPEECH, 11., 2010, Makuhari. **Proceedings…** Baixas: ISCA, 2010. p. 1045-1048.

MÜLLER, D. N. **COMFALA - Modelo Computacional do Processo de Compreensão da Fala**. 2006. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

NAING, H. M. S.; HLAING, A. M.; PA, W. P.; HU, X.; THU, Y. K.; HORI, C.; KAWAI, H. A Myanmar large vocabulary continuous speech recognition system. In: ASIA-PACIFIC SIGNAL AND INFORMATION PROCESSING ASSOCIATION ANNUAL SUMMIT AND CONFERENCE – APSIPA, 3., 2015, Hong Kong. **Proceedings...** Piscataway: IEEE, 2015. p. 320-327.

PAKOCI, E.; POPOVIĆ, B.; PEKAR, D. Improvements in Serbian Speech Recognition Using Sequence-Trained Deep neural Networks. In: ARTIFICIAL INTELLIGENCE, KNOWEDGE AND DATA ENGINEERING, 2018, St. Petersburg. **SPIIRAS Proceedings.** St. Petersburg: SPIIRAS, 2018. p. 53-76.

PAKOCI, E.; POPOVIĆ, B.; PEKAR, D. Language model optimization for a deep neural network based speech recognition system for Serbian. In: INTERNATIONAL CONFERENCE ON SPEECH AND COMPUTER, 19., 2017, Hatfield. **Proceedings…** Nova Iorque: Springer, 2017. p. 483-492.

PATRA, S. **Robust Speaker Identification System**. 2007. Disponível em: <http://www.serc.iisc.ernet.in/graduation-theses/spatra_dec07.pdf >. Acesso em: 26 mai. 2018.

PERICO, A.; SHINOHARA, C. S.; SARMENTO, C. D. **Sistema de Reconhecimento de Voz para Automatização de uma Plataforma Elevatória**. 2014. Monografia (Bacharelado em Engenharia Industrial Elétrica) – Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná, Curitiba.

PHULL, D. K.; KUMAR, G. B. Investigation of Indian English Speech Recognition using CMU Sphinx. **International Journal of Applied Engineering Research**, Delhi, v. 11, n. 6, p. 4167-4174, 2016.

PLANNERER, B. **An Introduction to Speech Recognition**. 2005. Disponível em: <https://www.scribd.com/document/52115407/An-Introduction-to-Speech-Recognition-B-Plannere>. Acesso em: 05 jun. 2018.

POVEY, D.; GHOSHAL, A.; BOULIANNE, G.; BURGET, L.; GLEMBEK, O.; GOEL, N.; HANNEMANN, M.; MOTLICEK, P.; QIAN, Y.; SCHWARZ, P.; SILOVSKY, J.; STEMMER, G.; VESELY, K. The Kaldi Speech Recognition Toolkit. In: WORKSHOP ON AUTOMATIC SPEECH RECOGNITION AND UNDERSTANDING, 12., 2011, Hawaii. **Proceedings…** Piscataway: IEEE, 2011. p. 1-4.

RUSSELL, S.; NORVIG, P. **Inteligência artificial**. 2. ed. Rio de Janeiro: Elsevier, 2004.

SAK, H.; SENIOR, A.; BEAUFAYS, F. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In: ANNUAL CONFERENCE OF THE INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION – INTERSPEECH, 15., 2014, Singapore. **Proceedings…** Baixas: ISCA, 2014. p. 338-342.

SAMPAIO NETO, N. C. **Ferramentas e recursos livres para reconhecimento e síntese de voz em português brasileiro**. 2011. Tese (Doutorado em Engenharia Elétrica) – Instituto de Tecnologia, Universidade Federal do Pará, Belém.

SAUNDADE, M.; KURLE, P. Speech Recognition using Digital Signal Processing. **International Journal of Electronics, Communication & Soft Computing Science and Engineering**, Nova Deli, v. 2, n. 6, p. 31-34, 2014.

SILVA, C. P. A. **Um software de reconhecimento de voz para português brasileiro**. 2010. Dissertação (Mestrado em Engenharia Elétrica) – Instituto de Tecnologia, Universidade Federal do Pará, Pará.

SILVA, E.; BAPTISTA, L.; FERNANDES, H.; KLAUTAU, A. Desenvolvimento de um Sistema de Reconhecimento Automático de Voz Contínua com Grande Vocabulário para o Português Brasileiro. In: WORKSHOP TIL, 2005, São Leopoldo. **Anais do XXV Congresso da Sociedade Brasileira de Computação**. Porto Alegre: Sociedade Brasileira de Computação, 2005. p. 2258-2267.

SILVA, E.; PANTOJA, M.; CELIDÔNIO, J.; KLAUTAU, A. **Modelos de linguagem n-grama para reconhecimento de voz com grande vocabulário**. 2004. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/til/2004/009.pdf>. Acesso em: 03 fev. 2018.

SILVEIRA, R. C. P. **Estudos de fonologia portuguesa.** São Paulo: Cortez, 1986.

SPÖRL, C.; CASTRO, E. G.; LUCHIARI, A. Aplicação de Redes Neurais Artificiais na Construção de Modelos de Fragilidade Ambiental. **Revista do Departamento de Geografia**, v. 21, n. 1, p. 113-135, 2011.

TACHBELIE, M. Y.; ABATE, S. T.; BESACIER, L. Using different acoustic, lexical and language modeling units for ASR of an under-resourced language - Amharic. **Speech Communication**, Amsterdam, v. 56, n. 1 p. 181-194, jan. 2014.

TACHIOKA, Y.; WATANABE, S. Discriminative method for recurrent neural network language models. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING – ICASSP, 39., 2015, Brisbane. **Proceedings…** Piscataway: IEEE, 2015. p. 5386-5390.

TEVAH, R. T. **Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro**. 2006. Dissertação (Mestrado em Ciências em Engenharia Elétrica) – Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

TIWARI, V. MFCC and its applications in speaker recognition. **International Journal on Emerging Technologies**, Viena, v. 1, n. 1 p. 19-22, fev. 2010.

VEIGA, A. O. **Treino não supervisionado de modelos acústicos para reconhecimento de fala**. 2013. Tese (Doutorado em Engenharia Eletrotécnica e de Computadores) – Departamento de Engenharia Eletrotécnica e de Computadores, Universidade de Coimbra, Coimbra.

VITAK, J.; CROUSE, J.; LAROSE, R. Personal Internet use at work: Understanding cyberslacking. **Computers in Human Behavior**, Amsterdam, v. 27, n. 5, p. 1751-1759, set. 2011.

YU, D.; DENG, L. **Automatic Speech Recognition:** A Deep Learning Approach. Londres: Springer, 2015.

ZHANG, H.; BAO, F.; GAO, G. Mongolian speech recognition based on deep neural networks. In: CHINESE COMPUTATIONAL LINGUISTICS AND NATURAL LANGUAGE PROCESSING BASED ON NATURALLY ANNOTATED BIG DATA, 14., 2015, Guangzhou. **Proceedings…** Nova Iorque: Springer, 2015. p. 180-188.

GLOSSÁRIO

Corpora	Plural de corpus.
Corpus	Um conjunto de documentos ou dados sobre determinado assunto.
Decodificador	Localiza a melhor sequência de palavras num conjunto de hipóteses possíveis dada a representação de características do sinal de voz.
Dicionário fonético	Lista de palavras possíveis de reconhecer com suas respectivas pronúncias expressas em uma sequência de fonemas.
Fonema	Unidades sonoras mais simples da língua, e divide-se em vogais, semivogais e consoantes.
Modelo acústico	Componente do sistema de reconhecimento de fala responsável por definir, a partir das características extraídas do áudio de entrada, a sequência mais provável de palavras ou fonemas.
Modelo de linguagem	Responsável por caracterizar o idioma e condicionar a combinação de palavras descartando frases que são gramaticalmente incorretas.
Retropropagação	Algoritmo de aprendizado da RNA que consiste em iniciar na camada de saída e propagar o erro retroativamente através das camadas ocultas.
Supervisionado	Esse tipo de treinamento tenta aprender a classificar os dados de treinamento que já foram classificados manualmente pelos humanos.
Voz	Som ou conjunto de sons produzidos pelas cordas vocais, e são responsáveis pela comunicação entre os seres humanos.

APÊNDICE A – ARTIGOS EXCLUÍDOS NA REVISÃO SISTEMÁTICA DA LITERATURA

O quadro abaixo apresenta os 54 artigos excluídos na revisão sistemática da literatura por não pertinência ao escopo da pesquisa.

Identificação	**Título**	**Repositório**	**Ano**
(Sem Autor)	*2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015 - Proceedings*	Scopus	2016
Al-Anzi, F.; AbuZeina, D.	*Literature Survey of Arabic Speech Recognition*	IEEE	2018
Arisoy, E.; Chen, S. F.; Ramabhadran, B.; Sethy, A.	*Converting Neural Network Language Models into Back-off Language Models for Efficient Decoding in Automatic Speech Recognition*	ACM	2014
Bahdanau, D.; Chorowski, J.; Serdyuk, D.; Brakel, P.; Bengio, Y.	*End-to-end attention-based large vocabulary speech recognition*	IEEE	2016
Ban, S. M.; Kim, H. S.	*Instantaneous model adaptation method for reverberant speech recognition*	IEEE	2015
Biagetti, G.; Crippa, P.; Falaschetti, L.; Orcioni, S.; Turchetti, C.	*An algorithm for automatic words extraction from a stream of phones in dictionary-based large vocabulary continuous speech recognition systems*	Scopus	2016
Brocki, Ł.; Koržinek, D.; Marasek, K.	*Improved factorization of a connectionist language model for single-pass real-time speech recognition*	Scopus	2014
Chaloupka, J.	*Digits to words converter for slavic languages in systems of automatic speech recognition*	Scopus	2017
Chen, L.; Dai, Y.; La, B.; Sun, M.; Xiong, Z.	*The Key Technology of Speech Interaction Based on Deep Learning*	Scopus	2018
Chen, N. F.; Sivadas, S.; Lim, B. P.; Ngo, H. G.; Xu, H.; Pham, V. T.; Ma, B.; Li, H.	*Strategies for Vietnamese keyword search*	IEEE	2014
Chien, J.; Ku, Y.	*Bayesian Recurrent Neural Network for Language Modeling*	IEEE	2016
Cucu, H.; Buzo, A.; Besacier, L.; Burileanu, C.	*SMT-based ASR domain adaptation methods for under-resourced languages: Application to Romanian*	Scopus	2014
Dimitriadis, D.; Bocchieri, E.	*Use of micro-modulation features in large vocabulary continuous speech recognition tasks*	ACM	2015
Gajecki, L.	*Architectures of neural networks applied for LVCSR language modeling*	ScienceDirect	2014

Gales, M. J. F.; Knill, K. M.; Ragni, A.	*Unicode-based graphemic systems for limited resource languages*	IEEE	2015
Gündoğdu, B.; Yusuf, B.; Saraçlar, M.	*Joint Learning of Distance Metric and Query Model for Posteriorgram-Based Keyword Search*	IEEE	2017
Hu, C. C.; Liu, B.; Shen, J. P.; Lane, I.	*Online Incremental Learning for Speaker-Adaptive Language Models*	Scopus	2018
Kanda, N.; Lu, X.; Kawai, H.	*Minimum Bayes risk training of CTC acoustic models in maximum a posteriori based decoding framework*	Scopus	2017
Kipyatkova, I.; Karpov, A.	*Language models with RNNs for rescoring hypotheses of Russian ASR*	Scopus	2016
Kitaoka, N.; Enami, D.; Nakagawa, S.	*Effect of acoustic and linguistic contexts on human and machine speech recognition*	ScienceDirect	2014
Kuamr, A.; Dua, M.; Choudhary, A.	*Implementation and performance evaluation of continuous Hindi speech recognition*	IEEE	2014
Kurimo, M.; Enarvi, S.; Tilk, O.; Varjokallio, M.; Mansikkaniemi, A.; Alumäe, T.	*Modeling under-resourced languages for speech recognition*	Scopus	2017
Lee, K.; Park, C.; Kim, I.; Kim, N.; Lee, J.	*Applying GPGPU to recurrent neural network language model based fast network search in the real-time LVCSR*	Scopus	2015
Liu, Q.; Qian, Y.; Yu, K.	*Future vector enhanced LSTM language model for LVCSR*	Scopus	2018
Londhe, N.D.; Kshirsagar, G.B.	*Continuous speech recognition system for chhattisgarhi*	Scopus	2018
Madhavaraj, A.; Ramakrishnan, A. G.	*Design and development of a large vocabulary, continuous speech recognition system for Tamil*	IEEE	2017
Mitra, V.; Sivaraman, G.; Nam, H.; Espy-Wilson, C.; Saltzman, E.	*Articulatory features from deep neural networks and their role in speech recognition*	IEEE	2014
Nguyen, Q. B.; Mai, V. T.; Le, Q. T.; Dam, B. Q.; Do, V. H.	*Development of a Vietnamese Large Vocabulary Continuous Speech Recognition System under Noisy Conditions*	ACM	2018
Nguyen, T. C.; Chaloupka, J.; Nouza, J.	*Study on incorporating tone into speech recognition of Vietnamese*	Scopus	2015
Popović, B.; Pakoci, E.; Jakovljević, N.; Kočiš, G.; Pekar, D.	*Voice assistant application for the Serbian language*	IEEE	2015
Popović, B.; Pakoci, E.; Pekar, D.	*A Comparison of Language Model Training Techniques in a Continuous Speech Recognition System for Serbian*	Scopus	2018
Sailor, H. B.; Patil, H. A.	*Novel Unsupervised Auditory Filterbank Learning Using Convolutional RBM for Speech Recognition*	ACM	2016

Sajjan, S.C.; Vijaya, C.	*Continuous Speech Recognition of Kannada language using triphone modeling*	Scopus	2016
Sangeetha, J.; Jothilakshmi, S.	*Speech translation system for english to dravidian languages*	Scopus	2017
Sangeetha, J.; Jothilakshmi, S.; Devendrakumar, R.N.	*Efficient continuous speech recognition approaches for dravidian languages*	Scopus	2014
Schlüter, R.; Doetsch, P.; Golik, P.; Kitza, M.; Menne, T.; Irie, K.; Tüske, Z.; Zeyer, A.	*Automatic speech recognition based on neural networks*	Scopus	2016
Sercu, T.; Puhrsch, C.; Kingsbury, B.; LeCun, Y.	*Very deep multilingual convolutional neural networks for LVCSR*	IEEE	2016
Shaik, M. A. B.; Tüske, Z.; Tahir, M. A.; Nußbaum-Thom, M.; Schlüter, R.; Ney, H.	*RWTH LVCSR systems for Quaero and EU-bridge: German, Polish, Spanish and Portuguese*	Scopus	2014
Smit, P.; Gangireddy, S. R.; Enarvi, S.; Virpioja, S.; Kurimo, M.	*Character-based units for unlimited vocabulary continuous speech recognition*	Scopus	2018
Soltau, H.; Liao, H.; Sak, H.	*Reducing the computational complexity for whole word models*	Scopus	2018
Sterpu, G.	*Large Vocabulary Continuous Audio-Visual Speech Recognition*	ACM	2018
Sterpu, G.; Saam, C.; Harte, N.	*Attention-based Audio-Visual Fusion for Robust Automatic Speech Recognition*	ACM	2018
Szabó, L.; Mihajlik, P.; Balog, A.; Fegyó, T.	*Unified simplified grapheme acoustic modeling for medieval Latin LVCSR*	Scopus	2017
Tachioka, Y.; Watanabe, S.	*Discriminative method for recurrent neural network language models*	IEEE	2015
Takagi, A.; Konno, K.; Kato, M.; Kosaka, T.	*Unsupervised cross-adaptation using language model and deep learning based acoustic model adaptations*	IEEE	2014
Triefenbach, F.; Demuynck, K.; Martens, J.-P.	*Large vocabulary continuous speech recognition with reservoir-based acoustic models*	Scopus	2014
Tüske, Z.; Irie, K.; Schlüter, R.; Ney, H.	*Investigation on log-linear interpolation of multi-domain neural network language model*	IEEE	2016
Van Hout, J.; Mitra, V.; Lei, Y.; Vergyri, D.; Graciarena, M.; Mandal, A.; Franco, H.	*Recent improvements in SRI's keyword detection system for noisy audio*	Scopus	2014
Vanhainen, N.; Salvi, G.	*Free acoustic and language models for large vocabulary continuous speech recognition in swedish*	Scopus	2014
Wang, H.; Khyuru, K.; Li, J.; Li, G.; Dang, J.; Huang, L.	*Investigation on acoustic modeling with different phoneme set for continuous Lhasa Tibetan recognition based on DNN method*	Scopus	2017

Yusuf, B.; Gundogdu, B.; Saraclar, M.	*Beyond Posteriorgram: Bottleneck Features for Keyword Search*	IEEE	2018
Zhang, S.; Bao, Y.; Zhou, P.; Jiang, H.; Dai, L.	*Improving deep neural networks for LVCSR using dropout and shrinking structure*	IEEE	2014
Zhang, S.; Jiang, H.; Xiong, S.; Wei, S.; Dai, L.	*Compact feedforward sequential memory networks for large vocabulary continuous speech recognition*	Scopus	2016
Zhang, W.; Fung, P.	*Discriminatively trained sparse inverse covariance matrices for speech recognition*	ACM	2014

APÊNDICE B – CONFIGURAÇÕES DA BIBLIOTECA CMUSPHINX

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 1 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 1;
$CFG_NUM_FILT = 8;
$CFG_LO_FILT = 26;
$CFG_HI_FILT = 1360;
$CFG_LIFTER = "4";
$CFG_AGC = 'none';
$CFG_DIAGFULL = 'no';
$CFG_VTLN_START = 0.16;
$CFG_VTLN_END = 0.28;
$CFG_VTLN_STEP = 0.01;
$CFG_LANGUAGEWEIGHT = "2.3";
$CFG_WORDPENALTY = "0.04";
$CFG_MMIE_MAX_ITERATIONS = 1;
$CFG_MMIE_TYPE = "rand";
$CFG_MMIE_CONSTE = "0.6";
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'no';
$CFG_FINAL_NUM_DENSITIES = 2;
$CFG_FALIGN_CI_MGAU = 'no';
$CFG_CI_MGAU = 'no';
$CFG_N_TIED_STATES = 40;
$CFG_LDA_DIMENSION = 5;
$CFG_CONVERGENCE_RATIO = 0.02;
$CFG_G2P_MODEL= 'no';
$DEC_CFG_VERBOSE = 1;
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_WORDPENALTY = "0.04";
$DEC_CFG_NPART = 1;
$CFG_DONE = 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 2 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 1;
$CFG_NUM_FILT = 24;
$CFG_LO_FILT = 78;
$CFG_HI_FILT = 4080;
$CFG_LIFTER = "13";
$CFG_AGC = 'none';
$CFG_DIAGFULL = 'no';
$CFG_VTLN_START = 0.48;
$CFG_VTLN_END = 0.84;
$CFG_VTLN_STEP = 0.03;
$CFG_LANGUAGEWEIGHT = "6.9";
$CFG_WORDPENALTY = "0.12";
$CFG_MMIE_MAX_ITERATIONS = 3;
```

```
$CFG_MMIE_TYPE = "rand";
$CFG_MMIE_CONSTE = "1.8";
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'no';
$CFG_FINAL_NUM_DENSITIES = 4;
$CFG_FALIGN_CI_MGAU = 'no';
$CFG_CI_MGAU = 'no';
$CFG_N_TIED_STATES = 120;
$CFG_LDA_DIMENSION = 17;
$CFG_CONVERGENCE_RATIO = 0.06;
$CFG_G2P_MODEL= 'no';
$DEC_CFG_VERBOSE = 1;
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_WORDPENALTY = "0.12";
$DEC_CFG_NPART = 1;
$CFG_DONE = 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 3 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 1;
$CFG_NUM_FILT = 40;
$CFG_LO_FILT = 130;
$CFG_HI_FILT = 6800;
$CFG_LIFTER = "22";
$CFG_AGC = 'none';
$CFG_DIAGFULL = 'no';
$CFG_VTLN_START = 0.80;
$CFG_VTLN_END = 1.40;
$CFG_VTLN_STEP = 0.05;
$CFG_LANGUAGEWEIGHT = "11.5";
$CFG_WORDPENALTY = "0.2";
$CFG_MMIE_MAX_ITERATIONS = 5;
$CFG_MMIE_TYPE = "rand";
$CFG_MMIE_CONSTE = "3.0";
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'no';
$CFG_FINAL_NUM_DENSITIES = 8;
$CFG_FALIGN_CI_MGAU = 'no';
$CFG_CI_MGAU = 'no';
$CFG_N_TIED_STATES = 200;
$CFG_LDA_DIMENSION = 29;
$CFG_CONVERGENCE_RATIO = 0.1;
$CFG_G2P_MODEL= 'no';
$DEC_CFG_VERBOSE = 1;
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_WORDPENALTY = "0.2";
$DEC_CFG_NPART = 1;
$CFG_DONE = 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 4 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 2;
$CFG_NUM_FILT = 56;
```

```
$CFG_LO_FILT = 182;
$CFG_HI_FILT = 8000;
$CFG_LIFTER = "30";
$CFG_AGC = 'none';
$CFG_DIAGFULL = 'no';
$CFG_VTLN_START = 1.12;
$CFG_VTLN_END = 1.96;
$CFG_VTLN_STEP = 0.07;
$CFG_LANGUAGEWEIGHT = "16.1";
$CFG_WORDPENALTY = "0.28";
$CFG_MMIE_MAX_ITERATIONS = 7;
$CFG_MMIE_TYPE = "rand";
$CFG_MMIE_CONSTE = "4.2";
$CFG_STATESPERHMM = 4;
$CFG_SKIPSTATE = 'no';
$CFG_FINAL_NUM_DENSITIES = 12;
$CFG_FALIGN_CI_MGAU = 'no';
$CFG_CI_MGAU = 'no';
$CFG_N_TIED_STATES = 280;
$CFG_LDA_DIMENSION = 40;
$CFG_CONVERGENCE_RATIO = 0.14;
$CFG_G2P_MODEL= 'no';
$DEC_CFG_VERBOSE = 2;
$DEC_CFG_LANGUAGEWEIGHT = "14";
$DEC_CFG_WORDPENALTY = "0.28";
$DEC_CFG_NPART = 2;
$CFG_DONE = 2;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 5 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 3;
$CFG_NUM_FILT = 72;
$CFG_LO_FILT = 234;
$CFG_HI_FILT = 8000;
$CFG_LIFTER = "39";
$CFG_AGC = 'none';
$CFG_DIAGFULL = 'no';
$CFG_VTLN_START = 1.44;
$CFG_VTLN_END = 2.52;
$CFG_VTLN_STEP = 0.09;
$CFG_LANGUAGEWEIGHT = "20.7";
$CFG_WORDPENALTY = "0.36";
$CFG_MMIE_MAX_ITERATIONS = 9;
$CFG_MMIE_TYPE = "rand";
$CFG_MMIE_CONSTE = "5.4";
$CFG_STATESPERHMM = 5;
$CFG_SKIPSTATE = 'no';
$CFG_FINAL_NUM_DENSITIES = 14;
$CFG_FALIGN_CI_MGAU = 'no';
$CFG_CI_MGAU = 'no';
$CFG_N_TIED_STATES = 360;
$CFG_LDA_DIMENSION = 52;
$CFG_CONVERGENCE_RATIO = 0.18;
$CFG_G2P_MODEL= 'no';
$DEC_CFG_VERBOSE = 3;
$DEC_CFG_LANGUAGEWEIGHT = "18";
```

```
$DEC_CFG_WORDPENALTY = "0.36";
$DEC_CFG_NPART = 3;
$CFG_DONE = 3;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 6 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 1;
$CFG_NUM_FILT = 8;
$CFG_LO_FILT = 26;
$CFG_HI_FILT = 1360;
$CFG_LIFTER = "4";
$CFG_AGC = 'max';
$CFG_DIAGFULL = 'yes';
$CFG_VTLN_START = 0.16;
$CFG_VTLN_END = 0.28;
$CFG_VTLN_STEP = 0.01;
$CFG_LANGUAGEWEIGHT = "2.3";
$CFG_WORDPENALTY = "0.04";
$CFG_MMIE_MAX_ITERATIONS = 1;
$CFG_MMIE_TYPE = "best";
$CFG_MMIE_CONSTE = "0.6";
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'yes';
$CFG_FINAL_NUM_DENSITIES = 2;
$CFG_FALIGN_CI_MGAU = 'yes';
$CFG_CI_MGAU = 'yes';
$CFG_N_TIED_STATES = 40;
$CFG_LDA_DIMENSION = 5;
$CFG_CONVERGENCE_RATIO = 0.02;
$CFG_G2P_MODEL= 'yes';
$DEC_CFG_VERBOSE = 1;
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_WORDPENALTY = "0.04";
$DEC_CFG_NPART = 1;
$CFG_DONE = 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 7 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 1;
$CFG_NUM_FILT = 24;
$CFG_LO_FILT = 78;
$CFG_HI_FILT = 4080;
$CFG_LIFTER = "13";
$CFG_AGC = 'max';
$CFG_DIAGFULL = 'yes';
$CFG_VTLN_START = 0.48;
$CFG_VTLN_END = 0.84;
$CFG_VTLN_STEP = 0.03;
$CFG_LANGUAGEWEIGHT = "6.9";
$CFG_WORDPENALTY = "0.12";
$CFG_MMIE_MAX_ITERATIONS = 3;
$CFG_MMIE_TYPE = "best";
$CFG_MMIE_CONSTE = "1.8";
```

```
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'yes';
$CFG_FINAL_NUM_DENSITIES = 4;
$CFG_FALIGN_CI_MGAU = 'yes';
$CFG_CI_MGAU = 'yes';
$CFG_N_TIED_STATES = 120;
$CFG_LDA_DIMENSION = 17;
$CFG_CONVERGENCE_RATIO = 0.06;
$CFG_G2P_MODEL= 'yes';
$DEC_CFG_VERBOSE = 1;
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_WORDPENALTY = "0.12";
$DEC_CFG_NPART = 1;
$CFG_DONE = 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 8 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 1;
$CFG_NUM_FILT = 40;
$CFG_LO_FILT = 130;
$CFG_HI_FILT = 6800;
$CFG_LIFTER = "22";
$CFG_AGC = 'max';
$CFG_DIAGFULL = 'yes';
$CFG_VTLN_START = 0.80;
$CFG_VTLN_END = 1.40;
$CFG_VTLN_STEP = 0.05;
$CFG_LANGUAGEWEIGHT = "11.5";
$CFG_WORDPENALTY = "0.2";
$CFG_MMIE_MAX_ITERATIONS = 5;
$CFG_MMIE_TYPE = "best";
$CFG_MMIE_CONSTE = "3.0";
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'yes';
$CFG_FINAL_NUM_DENSITIES = 8;
$CFG_FALIGN_CI_MGAU = 'yes';
$CFG_CI_MGAU = 'yes';
$CFG_N_TIED_STATES = 200;
$CFG_LDA_DIMENSION = 29;
$CFG_CONVERGENCE_RATIO = 0.1;
$CFG_G2P_MODEL= 'yes';
$DEC_CFG_VERBOSE = 1;
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_WORDPENALTY = "0.2";
$DEC_CFG_NPART = 1;
$CFG_DONE = 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 9 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 2;
$CFG_NUM_FILT = 56;
$CFG_LO_FILT = 182;
$CFG_HI_FILT = 8000;
```

```
$CFG_LIFTER = "30";
$CFG_AGC = 'max';
$CFG_DIAGFULL = 'yes';
$CFG_VTLN_START = 1.12;
$CFG_VTLN_END = 1.96;
$CFG_VTLN_STEP = 0.07;
$CFG_LANGUAGEWEIGHT = "16.1";
$CFG_WORDPENALTY = "0.28";
$CFG_MMIE_MAX_ITERATIONS = 7;
$CFG_MMIE_TYPE = "best";
$CFG_MMIE_CONSTE = "4.2";
$CFG_STATESPERHMM = 4;
$CFG_SKIPSTATE = 'yes';
$CFG_FINAL_NUM_DENSITIES = 12;
$CFG_FALIGN_CI_MGAU = 'yes';
$CFG_CI_MGAU = 'yes';
$CFG_N_TIED_STATES = 280;
$CFG_LDA_DIMENSION = 40;
$CFG_CONVERGENCE_RATIO = 0.14;
$CFG_G2P_MODEL= 'yes';
$DEC_CFG_VERBOSE = 2;
$DEC_CFG_LANGUAGEWEIGHT = "14";
$DEC_CFG_WORDPENALTY = "0.28";
$DEC_CFG_NPART = 2;
$CFG_DONE = 2;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 10 para a biblioteca CMUSphinx.

```
$CFG_VERBOSE = 3;
$CFG_NUM_FILT = 72;
$CFG_LO_FILT = 234;
$CFG_HI_FILT = 8000;
$CFG_LIFTER = "39";
$CFG_AGC = 'max';
$CFG_DIAGFULL = 'yes';
$CFG_VTLN_START = 1.44;
$CFG_VTLN_END = 2.52;
$CFG_VTLN_STEP = 0.09;
$CFG_LANGUAGEWEIGHT = "20.7";
$CFG_WORDPENALTY = "0.36";
$CFG_MMIE_MAX_ITERATIONS = 9;
$CFG_MMIE_TYPE = "best";
$CFG_MMIE_CONSTE = "5.4";
$CFG_STATESPERHMM = 5;
$CFG_SKIPSTATE = 'yes';
$CFG_FINAL_NUM_DENSITIES = 14;
$CFG_FALIGN_CI_MGAU = 'yes';
$CFG_CI_MGAU = 'yes';
$CFG_N_TIED_STATES = 360;
$CFG_LDA_DIMENSION = 52;
$CFG_CONVERGENCE_RATIO = 0.18;
$CFG_G2P_MODEL= 'yes';
$DEC_CFG_VERBOSE = 3;
$DEC_CFG_LANGUAGEWEIGHT = "18";
$DEC_CFG_WORDPENALTY = "0.36";
$DEC_CFG_NPART = 3;
```

```
$CFG_DONE = 3;
```

APÊNDICE C – CONFIGURAÇÕES DA BIBLIOTECA HTK

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 1 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.002 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms10/macros -H
     hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms11/macros -H
     hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms12/macros -H
     hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms13/macros -H
     hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H $i"G"/macros -H
     $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M $i"G" tiedlist
```

```
# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
CEPLIFTER = 22
ENORMALISE = T
SOURCERATE = 227
ZMEANSOURCE = T
HPARM: CMNTCONST = 0.995
HPARM: CMNRESETONSTOP = F
HPARM: CMNMINFRAMES = 12
HREC:FORCEOUT = T
HLM:UPPERCASELM=T
HLM:RAWMITFORMAT=F
AREC: NTOKS=3
AREC: NGSCALE=15.0
AREC: WORDPEN=-20.0
AREC: GENBEAM=250.0
AREC: WORDBEAM=230.0
AREC: NBEAM=250.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = T
FORCECXTEXP = T
ALLOWXWRDEXP = T
CFWORDBOUNDARY = FALSE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 2 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.006 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms10/macros -H
     hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms11/macros -H
     hmms11/hmmdefs -M hmms12 hmmlist.txt
```

```
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms12/macros -H
     hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms13/macros -H
     hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H $i"G"/macros -
     H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
CEPLIFTER = 22
ENORMALISE = T
SOURCERATE = 227
ZMEANSOURCE = T
HPARM: CMNTCONST = 0.995
HPARM: CMNRESETONSTOP = F
HPARM: CMNMINFRAMES = 12
HREC:FORCEOUT = T
HLM:UPPERCASELM=T
HLM:RAWMITFORMAT=F
AREC: NTOKS=3
AREC: NGSCALE=15.0
AREC: WORDPEN=-20.0
AREC: GENBEAM=250.0
AREC: WORDBEAM=230.0
AREC: NBEAM=250.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = T
FORCECXTEXP = T
ALLOWXWRDEXP = T
CFWORDBOUNDARY = FALSE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 3 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.01 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms10/macros -
     H hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms11/macros -
     H hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms12/macros -
     H hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms13/macros -
     H hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H $i"G"/macros
     -H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 100000.0
SAVECOMPRESSED = T
```

```
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
CEPLIFTER = 22
ENORMALISE = T
SOURCERATE = 227
ZMEANSOURCE = T
HPARM: CMNTCONST = 0.995
HPARM: CMNRESETONSTOP = F
HPARM: CMNMINFRAMES = 12
HREC:FORCEOUT = T
HLM:UPPERCASELM=T
HLM:RAWMITFORMAT=F
AREC: NTOKS=3
AREC: NGSCALE=15.0
AREC: WORDPEN=-20.0
AREC: GENBEAM=250.0
AREC: WORDBEAM=230.0
AREC: NBEAM=250.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = T
FORCECXTEXP = T
ALLOWXWRDEXP = T
CFWORDBOUNDARY = FALSE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 4 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.014 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms10/macros -
     H hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms11/macros -
     H hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms12/macros -
     H hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms13/macros -
     H hmms13/hmmdefs -M hmms14 hmmlist.txt
```

```
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H $i"G"/macros
     -H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
CEPLIFTER = 22
ENORMALISE = T
SOURCERATE = 227
ZMEANSOURCE = T
HPARM: CMNTCONST = 0.995
HPARM: CMNRESETONSTOP = F
HPARM: CMNMINFRAMES = 12
HREC:FORCEOUT = T
HLM:UPPERCASELM=T
HLM:RAWMITFORMAT=F
AREC: NTOKS=3
AREC: NGSCALE=15.0
AREC: WORDPEN=-20.0
AREC: GENBEAM=250.0
AREC: WORDBEAM=230.0
AREC: NBEAM=250.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = T
FORCECXTEXP = T
ALLOWXWRDEXP = T
CFWORDBOUNDARY = FALSE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 5 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.018 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
echo "HERest - 2a reestimação ..."
HERest -I phones0.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms10/macros -
     H hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms11/macros -
     H hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms12/macros -
     H hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms13/macros -
     H hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H $i"G"/macros
     -H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
```

```
CEPLIFTER = 22
ENORMALISE = T
SOURCERATE = 227
ZMEANSOURCE = T
HPARM: CMNTCONST = 0.995
HPARM: CMNRESETONSTOP = F
HPARM: CMNMINFRAMES = 12
HREC:FORCEOUT = T
HLM:UPPERCASELM=T
HLM:RAWMITFORMAT=F
AREC: NTOKS=3
AREC: NGSCALE=15.0
AREC: WORDPEN=-20.0
AREC: GENBEAM=250.0
AREC: WORDBEAM=230.0
AREC: NBEAM=250.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = T
FORCECXTEXP = T
ALLOWXWRDEXP = T
CFWORDBOUNDARY = FALSE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 6 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.002 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms10/macros -H
     hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms11/macros -H
     hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms12/macros -H
     hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 50.0 30.0 200 -S mfc_train.list -H hmms13/macros -H
     hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
```

```
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 50.0 30.0 200.0 -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H $i"G"/macros -H
     $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 50 30 200 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 20000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
WINDOWSIZE = 50000.0
USEHAMMING = F
PREEMCOEF = 0.1
CEPLIFTER = 2
ENORMALISE = F
SOURCERATE  = 45
ZMEANSOURCE = F
HPARM: CMNTCONST = 0.199
HPARM: CMNRESETONSTOP = T
HPARM: CMNMINFRAMES = 2
HREC:FORCEOUT = F
HLM:UPPERCASELM=F
HLM:RAWMITFORMAT=T
AREC: NTOKS=1
AREC: NGSCALE=3.0
AREC: WORDPEN=-4.0
AREC: GENBEAM=50.0
AREC: WORDBEAM=46.0
AREC: NBEAM=50.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = F
FORCECXTEXP = F
ALLOWXWRDEXP = F
CFWORDBOUNDARY = TRUE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 7 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.006 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms10/macros -H
     hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms11/macros -H
     hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms12/macros -H
     hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 150.0 90.0 600 -S mfc_train.list -H hmms13/macros -H
     hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 150.0 90.0 600.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H $i"G"/macros -
     H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 150 90 600 -S mfc_train.list -H hmms/macros -H
     hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 60000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
WINDOWSIZE = 150000.0
USEHAMMING = F
PREEMCOEF = 0.3
CEPLIFTER = 12
```

```
ENORMALISE = F
SOURCERATE = 136
ZMEANSOURCE = F
HPARM: CMNTCONST = 0.597
HPARM: CMNRESETONSTOP = T
HPARM: CMNMINFRAMES = 7
HREC:FORCEOUT = F
HLM:UPPERCASELM=F
HLM:RAWMITFORMAT=T
AREC: NTOKS=2
AREC: NGSCALE=9.0
AREC: WORDPEN=-12.0
AREC: GENBEAM=150.0
AREC: WORDBEAM=138.0
AREC: NBEAM=150.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = F
FORCECXTEXP = F
ALLOWXWRDEXP = F
CFWORDBOUNDARY = TRUE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 8 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.01 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms10/macros -
     H hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms11/macros -
     H hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms12/macros -
     H hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 250.0 150.0 1000 -S mfc_train.list -H hmms13/macros -
     H hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
```

```
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 250.0 150.0 1000.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H $i"G"/macros
     -H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 250 150 1000 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 100000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
WINDOWSIZE = 250000.0
USEHAMMING = F
PREEMCOEF = 0.5
CEPLIFTER = 22
ENORMALISE = F
SOURCERATE = 227
ZMEANSOURCE = F
HPARM: CMNTCONST = 0.995
HPARM: CMNRESETONSTOP = T
HPARM: CMNMINFRAMES = 12
HREC:FORCEOUT = F
HLM:UPPERCASELM=F
HLM:RAWMITFORMAT=T
AREC: NTOKS=3
AREC: NGSCALE=15.0
AREC: WORDPEN=-20.0
AREC: GENBEAM=250.0
AREC: WORDBEAM=230.0
AREC: NBEAM=250.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = F
FORCECXTEXP = F
ALLOWXWRDEXP = F
CFWORDBOUNDARY = TRUE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 9 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.014 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
```

```
HERest -I phones0.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms10/macros -
     H hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms11/macros -
     H hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms12/macros -
     H hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 350.0 210.0 1400 -S mfc_train.list -H hmms13/macros -
     H hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 350.0 210.0 1400.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H $i"G"/macros
     -H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 350 210 1400 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 140000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
WINDOWSIZE = 350000.0
USEHAMMING = F
PREEMCOEF = 0.7
CEPLIFTER = 32
ENORMALISE = F
SOURCERATE = 317
ZMEANSOURCE = F
HPARM: CMNTCONST = 1.393
```

```
HPARM:  CMNRESETONSTOP = T
HPARM:  CMNMINFRAMES = 16
HREC:FORCEOUT = F
HLM:UPPERCASELM=F
HLM:RAWMITFORMAT=T
AREC: NTOKS=4
AREC: NGSCALE=21.0
AREC: WORDPEN=-28.0
AREC: GENBEAM=350.0
AREC: WORDBEAM=322.0
AREC: NBEAM=350.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = F
FORCECXTEXP = F
ALLOWXWRDEXP = F
CFWORDBOUNDARY = TRUE
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 10 para a biblioteca HTK.

```
HCompV -C confs/hcomp.conf -f 0.018 -m -S mfc_train.list -M hmms0
     hmms0/proto.hmm
HERest -I phones0.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms0/macros -H
     hmms0/hmmdefs -M hmms1 hmmlist.txt
echo "HERest - 2a reestimação ..."
HERest -I phones0.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms1/macros -H
     hmms1/hmmdefs -M hmms2 hmmlist.txt
HERest -I phones0.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms2/macros -H
     hmms2/hmmdefs -M hmms3 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms5/macros -H
     hmms5/hmmdefs -M hmms6 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms6/macros -H
     hmms6/hmmdefs -M hmms7 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms7/macros -H
     hmms7/hmmdefs -M hmms8 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms8/macros -H
     hmms8/hmmdefs -M hmms9 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms9/macros -H
     hmms9/hmmdefs -M hmms10 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms10/macros -
     H hmms10/hmmdefs -M hmms11 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms11/macros -
     H hmms11/hmmdefs -M hmms12 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms12/macros -
     H hmms12/hmmdefs -M hmms13 hmmlist.txt
HERest -I phonesp.mlf -t 450.0 270.0 1800 -S mfc_train.list -H hmms13/macros -
     H hmms13/hmmdefs -M hmms14 hmmlist.txt
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms18/macros -H hmms18/hmmdefs -M hmms19 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms19/macros -H hmms19/hmmdefs -M hmms20 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms20/macros -H hmms20/hmmdefs -M hmms21 trifone
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0 -s stats -S mfc_train.list -H
     hmms21/macros -H hmms21/hmmdefs -M hmms22 trifone
```

```
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -B -I wintri.mlf -t 450.0 270.0 1800.0  -S mfc_train.list -H
     hmmsTree/macros -H hmmsTree/hmmdefs -M hmmsTree tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H 2G/macros -H
     2G/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M 2G tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H $i"G"/macros
     -H $i"G"/hmmdefs -M hmms tiedlist
HERest -u tmvw -I wintri.mlf -t 450 270 1800 -S mfc_train.list -H hmms/macros
     -H hmms/hmmdefs -M $i"G" tiedlist

# ----> Arquivo: hcopy-wav.conf
TARGETRATE = 180000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
WINDOWSIZE = 450000.0
USEHAMMING = F
PREEMCOEF = 0.97
CEPLIFTER = 42
ENORMALISE = F
SOURCERATE = 408
ZMEANSOURCE = F
HPARM: CMNTCONST = 1.791
HPARM:  CMNRESETONSTOP = T
HPARM:  CMNMINFRAMES = 21
HREC:FORCEOUT = F
HLM:UPPERCASELM=F
HLM:RAWMITFORMAT=T
AREC: NTOKS=5
AREC: NGSCALE=27.0
AREC: WORDPEN=-36.0
AREC: GENBEAM=450.0
AREC: WORDBEAM=414.0
AREC: NBEAM=450.0

# ----> Arquivo: hvite.conf
ALLOWCXTEXP = F
FORCECXTEXP = F
ALLOWXWRDEXP = F
CFWORDBOUNDARY = TRUE
```

APÊNDICE D – CONFIGURAÇÕES DA BIBLIOTECA KALDI

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 1 para a biblioteca Kaldi.

```
lm_order=2
steps/train_deltas.sh --cmd "$train_cmd" 600 8000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 1000 12000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=2 --
     right-context=2" 1300 15000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 1500 20000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 1
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 2 para a biblioteca Kaldi.

```
lm_order=2
steps/train_deltas.sh --cmd "$train_cmd" 1800 24000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 3000 36000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=3 --
     right-context=3" 3900 45000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 4500 60000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 2
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 3 para a biblioteca Kaldi.

```
lm_order=3
steps/train_deltas.sh --cmd "$train_cmd" 3000 40000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 5000 60000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=5 --
     right-context=5" 6500 75000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 7500 100000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 3
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 4 para a biblioteca Kaldi.

```
lm_order=3
steps/train_deltas.sh --cmd "$train_cmd" 4200 56000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 7000 84000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=7 --
     right-context=7" 9100 105000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 10500 140000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 4
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 5 para a biblioteca Kaldi.

```
lm_order=4
steps/train_deltas.sh --cmd "$train_cmd" 5400 72000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 9000 108000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=9 --
     right-context=9" 11700 135000 data/train data/lang exp/tri2a_ali
     exp/tri2b || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 13500 180000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 5
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 6 para a biblioteca Kaldi.

```
lm_order=2
steps/train_deltas.sh --cmd "$train_cmd" 600 8000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 1000 12000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=2 --
     right-context=2" 1300 15000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 1500 20000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 1
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;

# REDES NEURAIS
# ---> Ivector
min_seg_len=0.31
steps/train_lda_mllt.sh --cmd "$train_cmd" --num-iters 1 --mllt-iters "1 3 5"
     --splice-opts "--left-context=2 --right-context=2" 600 2000
```

```
     $temp_data_root/${train_set}_hires data/lang $gmm_dir
     exp/nnet3${nnet3_affix}/tri5
steps/online/nnet2/train_diag_ubm.sh --cmd "$train_cmd" --nj $nj --num-frames
     44537 --num-threads $num_threads_ubm
     ${temp_data_root}/${train_set}_sp_hires_subset 32
     exp/nnet3${nnet3_affix}/tri5 exp/nnet3${nnet3_affix}/diag_ubm
utils/data/modify_speaker_info.sh --utts-per-spk-max 1
     data/${train_set}_sp_hires_comb
     ${temp_data_root}/${train_set}_sp_hires_comb_max2

# ---> RNN
cell_dim=256
hidden_dim=256
recurrent_projection_dim=32
non_recurrent_projection_dim=32
chunk_width=4
chunk_left_context=8
chunk_right_context=8
num_epochs=1
initial_effective_lrate=0.0003
final_effective_lrate=0.00003
momentum=0.1
num_chunk_per_minibatch=20
samples_per_iter=4000
extra_left_context=10
extra_right_context=10
lstm_opts="decay-time=4 cell-dim=$cell_dim"
fast-lstmp-layer name=lstm1-forward input=lda delay=-1 $lstm_opts
fast-lstmp-layer name=lstm1-backward input=lda delay=1 $lstm_opts
output-layer name=output output-delay=$label_delay dim=$num_targets max-
     change=0.3
steps/nnet3/train_rnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir=$train_ivector_dir \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.srand=$srand \
    --trainer.num-epochs=$num_epochs \
    --trainer.samples-per-iter=$samples_per_iter \
    --trainer.optimization.num-jobs-final=1 \
    --trainer.optimization.initial-effective-lrate=$initial_effective_lrate \
    --trainer.optimization.final-effective-lrate=$final_effective_lrate \
    --trainer.optimization.shrink-value 0.198 \
    --trainer.rnn.num-chunk-per-minibatch=$num_chunk_per_minibatch \
    --trainer.optimization.momentum=$momentum \
    --egs.opts " --nj 3 " \
    --egs.chunk-width=$chunk_width \
    --egs.chunk-left-context=$chunk_left_context \
    --egs.chunk-right-context=$chunk_right_context \
    --egs.chunk-left-context-initial=0 \
    --egs.chunk-right-context-final=0 \
    --egs.dir="$common_egs_dir" \
    --cleanup.remove-egs=$remove_egs \
    --cleanup.preserve-model-interval=20 \
    --use-gpu=no \
    --feat-dir=$train_data_dir \
    --ali-dir=$ali_dir \
    --lang=data/lang \
    --dir=$dir || exit 1;
```

```
# ---> DNN
relu_dim=150
relu-renorm-layer name=tdnn1 dim=250
relu-renorm-layer name=tdnn2 dim=250 input=Append(-1,2)
relu-renorm-layer name=tdnn3 dim=250
output-layer name=output dim=$num_targets max-change=0.3
steps/nnet3/train_dnn.py --stage $train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir ${train_ivector_dir} \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.num-epochs 1 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.0017 \
    --trainer.optimization.final-effective-lrate 0.00017 \
    --trainer.samples-per-iter 80000 \
    --egs.dir "$common_egs_dir" \
    --egs.stage "$egs_stage" \
    --cleanup.remove-egs $remove_egs \
    --cleanup.preserve-model-interval 10 \
    --feat-dir=$train_data_dir \
    --ali-dir $ali_dir \
    --lang data/lang \
    --use-gpu=no \
    --dir=$dir || exit 1;
steps/nnet3/decode.sh --nj $nj --cmd "$decode_cmd" --acwt 0.2 --post-decode-
     acwt 2.0 --scoring-opts "--min-lmwt 1 " --num-threads $num_threads --
     online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_test_hires
     ${graph_dir} data/test_hires ${dir}/decode_test || exit 1

# ---> RNN+HMM
min_seg_len=0.31
chunk_left_context=8
label_delay=1
xent_regularize=0.02
extra_left_context=10
extra_right_context=0
frames_per_chunk=30
fast-lstmp-layer name=lstm1 cell-dim=128 recurrent-projection-dim=32 non-
     recurrent-projection-dim=32 delay=-1
output-layer name=output input=lstm1 output-delay=$label_delay include-log-
     softmax=false dim=$num_targets max-change=0.3
output-layer name=output-xent input=lstm1 output-delay=$label_delay
     dim=$num_targets learning-rate-factor=$learning_rate_factor max-
     change=0.3
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.02 \
    --chain.l2-regularize 0.00005 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=400" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width "$frames_per_chunk" \
    --egs.chunk-left-context "$chunk_left_context" \
    --egs.chunk-right-context "$chunk_right_context" \
    --trainer.num-chunk-per-minibatch 32 \
```

```
    --trainer.frames-per-iter 300000 \
    --trainer.max-param-change 0.4 \
    --trainer.num-epochs 1 \
    --trainer.deriv-truncate-margin 2 \
    --trainer.optimization.shrink-value 0.198 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.001 \
    --trainer.optimization.final-effective-lrate 0.0001 \
    --trainer.optimization.momentum 0.0 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 0.2 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 0.2 --post-decode-acwt 2.0 \
    --extra-left-context $extra_left_context  \
    --extra-right-context $extra_right_context  \
    --frames-per-chunk "$frames_per_chunk" \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 1 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;

# ---> DNN+HMM
min_seg_len=0.31
xent_regularize=0.02
relu-renorm-layer name=tdnn1 dim=90
relu-renorm-layer name=tdnn2 input=Append(-1,0,1) dim=90
relu-renorm-layer name=prefinal-chain input=tdnn2 dim=90 target-rms=0.1
output-layer name=output include-log-softmax=false dim=$num_targets max-
     change=0.3
relu-renorm-layer name=prefinal-xent input=tdnn2 dim=90 target-rms=0.1
output-layer name=output-xent dim=$num_targets learning-rate-
     factor=$learning_rate_factor max-change=0.3
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.02 \
    --chain.l2-regularize 0.00005 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=400" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width 30 \
    --trainer.num-chunk-per-minibatch 32 \
    --trainer.frames-per-iter 300000 \
    --trainer.num-epochs 1 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.001 \
    --trainer.optimization.final-effective-lrate 0.0001 \
    --trainer.max-param-change 0.4 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
```

```
utils/mkgraph.sh --self-loop-scale 0.2 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
     --acwt 0.2 --post-decode-acwt 2.0 \
     --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
     --scoring-opts "--min-lmwt 1 " \
     $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 7 para a biblioteca Kaldi.

```
lm_order=2
steps/train_deltas.sh --cmd "$train_cmd" 1800 24000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 3000 36000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=3 --
     right-context=3" 3900 45000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 4500 60000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 2
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;

# REDES NEURAIS
# ---> Ivector
min_seg_len=0.93
steps/train_lda_mllt.sh --cmd "$train_cmd" --num-iters 4 --mllt-iters "2 4 6"
     --splice-opts "--left-context=3 --right-context=3" 1800 6000
     $temp_data_root/${train_set}_hires data/lang $gmm_dir
     exp/nnet3${nnet3_affix}/tri5
steps/online/nnet2/train_diag_ubm.sh --cmd "$train_cmd" --nj $nj --num-frames
     133613 --num-threads $num_threads_ubm
     ${temp_data_root}/${train_set}_sp_hires_subset 98
     exp/nnet3${nnet3_affix}/tri5 exp/nnet3${nnet3_affix}/diag_ubm
utils/data/modify_speaker_info.sh --utts-per-spk-max 2
     data/${train_set}_sp_hires_comb
     ${temp_data_root}/${train_set}_sp_hires_comb_max2

# ---> RNN
cell_dim=512
hidden_dim=512
recurrent_projection_dim=64
non_recurrent_projection_dim=64
chunk_width=12
chunk_left_context=24
chunk_right_context=24
num_epochs=3
initial_effective_lrate=0.0123
final_effective_lrate=0.00123
momentum=0.3
num_chunk_per_minibatch=60
samples_per_iter=12000
extra_left_context=30
extra_right_context=30
lstm_opts="decay-time=12 cell-dim=$cell_dim"
fast-lstmp-layer name=lstm1-forward input=lda delay=-1 $lstm_opts
fast-lstmp-layer name=lstm1-backward input=lda delay=1 $lstm_opts
```

```
fast-lstmp-layer name=lstm2-forward input=Append(lstm1-forward, lstm1-
     backward) delay=-2 $lstm_opts
fast-lstmp-layer name=lstm2-backward input=Append(lstm1-forward, lstm1-
     backward) delay=2 $lstm_opts
output-layer name=output output-delay=$label_delay dim=$num_targets max-
     change=0.9
steps/nnet3/train_rnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir=$train_ivector_dir \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.srand=$srand \
    --trainer.num-epochs=$num_epochs \
    --trainer.samples-per-iter=$samples_per_iter \
    --trainer.optimization.num-jobs-final=1 \
    --trainer.optimization.initial-effective-lrate=$initial_effective_lrate \
    --trainer.optimization.final-effective-lrate=$final_effective_lrate \
    --trainer.optimization.shrink-value 0.594 \
    --trainer.rnn.num-chunk-per-minibatch=$num_chunk_per_minibatch \
    --trainer.optimization.momentum=$momentum \
    --egs.opts " --nj 3 " \
    --egs.chunk-width=$chunk_width \
    --egs.chunk-left-context=$chunk_left_context \
    --egs.chunk-right-context=$chunk_right_context \
    --egs.chunk-left-context-initial=0 \
    --egs.chunk-right-context-final=0 \
    --egs.dir="$common_egs_dir" \
    --cleanup.remove-egs=$remove_egs \
    --cleanup.preserve-model-interval=60 \
    --use-gpu=no \
    --feat-dir=$train_data_dir \
    --ali-dir=$ali_dir \
    --lang=data/lang \
    --dir=$dir || exit 1;

# ---> DNN
relu_dim=450
relu-renorm-layer name=tdnn1 dim=748
relu-renorm-layer name=tdnn2 dim=748 input=Append(-1,2)
relu-renorm-layer name=tdnn3 dim=748 input=Append(-3,3)
relu-renorm-layer name=tdnn4 dim=748
output-layer name=output dim=$num_targets max-change=0.9
steps/nnet3/train_dnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir ${train_ivector_dir} \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.num-epochs 2 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.0697 \
    --trainer.optimization.final-effective-lrate 0.00697 \
    --trainer.samples-per-iter 240000 \
    --egs.dir "$common_egs_dir" \
    --egs.stage "$egs_stage" \
    --cleanup.remove-egs $remove_egs \
    --cleanup.preserve-model-interval 30 \
    --feat-dir=$train_data_dir \
    --ali-dir $ali_dir \
    --lang data/lang \
    --use-gpu=no \
    --dir=$dir || exit 1;
```

```
steps/nnet3/decode.sh --nj $nj --cmd "$decode_cmd" --acwt 0.6 --post-decode-
     acwt 6.0 --scoring-opts "--min-lmwt 3 " --num-threads $num_threads --
     online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_test_hires
     ${graph_dir} data/test_hires ${dir}/decode_test || exit 1

# ---> RNN+HMM
min_seg_len=0.93
chunk_left_context=24
label_delay=3
xent_regularize=0.06
extra_left_context=30
extra_right_context=10
frames_per_chunk=90
fast-lstmp-layer name=lstm1 cell-dim=256 recurrent-projection-dim=64 non-
     recurrent-projection-dim=64 delay=-3
fast-lstmp-layer name=lstm2 cell-dim=256 recurrent-projection-dim=64 non-
     recurrent-projection-dim=64 delay=-3
output-layer name=output input=lstm2 output-delay=$label_delay include-log-
     softmax=false dim=$num_targets max-change=0.9
output-layer name=output-xent input=lstm2 output-delay=$label_delay
     dim=$num_targets learning-rate-factor=$learning_rate_factor max-
     change=0.9
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.06 \
    --chain.l2-regularize 0.00205 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=1200" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width "$frames_per_chunk" \
    --egs.chunk-left-context "$chunk_left_context" \
    --egs.chunk-right-context "$chunk_right_context" \
    --trainer.num-chunk-per-minibatch 64 \
    --trainer.frames-per-iter 900000 \
    --trainer.max-param-change 1.2 \
    --trainer.num-epochs 2 \
    --trainer.deriv-truncate-margin 6 \
    --trainer.optimization.shrink-value 0.594 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.041 \
    --trainer.optimization.final-effective-lrate 0.0041 \
    --trainer.optimization.momentum 0.0 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 0.6 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 0.6 --post-decode-acwt 6.0 \
    --extra-left-context $extra_left_context  \
    --extra-right-context $extra_right_context  \
    --frames-per-chunk "$frames_per_chunk" \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 3 " \
```

```
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;

# ---> DNN+HMM
min_seg_len=0.93
xent_regularize=0.06
relu-renorm-layer name=tdnn1 dim=270
relu-renorm-layer name=tdnn2 input=Append(-1,0,1) dim=270
relu-renorm-layer name=tdnn3 input=Append(-1,0,1,2) dim=270
relu-renorm-layer name=tdnn4 input=Append(-3,0,3) dim=270
relu-renorm-layer name=prefinal-chain input=tdnn4 dim=270 target-rms=0.3
output-layer name=output include-log-softmax=false dim=$num_targets max-
     change=0.9
relu-renorm-layer name=prefinal-xent input=tdnn4 dim=270 target-rms=0.3
output-layer name=output-xent dim=$num_targets learning-rate-
     factor=$learning_rate_factor max-change=0.9
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.06 \
    --chain.l2-regularize 0.00205 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=1200" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width 90 \
    --trainer.num-chunk-per-minibatch 64 \
    --trainer.frames-per-iter 900000 \
    --trainer.num-epochs 2 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.041 \
    --trainer.optimization.final-effective-lrate 0.0041 \
    --trainer.max-param-change 1.2 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 0.6 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 0.6 --post-decode-acwt 6.0 \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 3 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 8 para a biblioteca Kaldi.

```
lm_order=3
steps/train_deltas.sh --cmd "$train_cmd" 3000 40000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 5000 60000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=5 --
     right-context=5" 6500 75000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
```

```
steps/train_sat.sh --cmd "$train_cmd" 7500 100000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 3
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;

# REDES NEURAIS
# ---> Ivector
min_seg_len=1.55
steps/train_lda_mllt.sh --cmd "$train_cmd" --num-iters 7 --mllt-iters "3 5 7"
     --splice-opts "--left-context=4 --right-context=4" 3000 10000
     $temp_data_root/${train_set}_hires data/lang $gmm_dir
     exp/nnet3${nnet3_affix}/tri5
steps/online/nnet2/train_diag_ubm.sh --cmd "$train_cmd" --nj $nj --num-frames
     222689 --num-threads $num_threads_ubm
     ${temp_data_root}/${train_set}_sp_hires_subset 162
     exp/nnet3${nnet3_affix}/tri5 exp/nnet3${nnet3_affix}/diag_ubm
utils/data/modify_speaker_info.sh --utts-per-spk-max 3
     data/${train_set}_sp_hires_comb
     ${temp_data_root}/${train_set}_sp_hires_comb_max2

# ---> RNN
cell_dim=1024
hidden_dim=1024
recurrent_projection_dim=128
non_recurrent_projection_dim=128
chunk_width=20
chunk_left_context=40
chunk_right_context=40
num_epochs=6
initial_effective_lrate=0.0243
final_effective_lrate=0.00243
momentum=0.5
num_chunk_per_minibatch=100
samples_per_iter=20000
extra_left_context=50
extra_right_context=50
lstm_opts="decay-time=20 cell-dim=$cell_dim"
fast-lstmp-layer name=lstm1-forward input=lda delay=-1 $lstm_opts
fast-lstmp-layer name=lstm1-backward input=lda delay=1 $lstm_opts
fast-lstmp-layer name=lstm2-forward input=Append(lstm1-forward, lstm1-
     backward) delay=-2 $lstm_opts
fast-lstmp-layer name=lstm2-backward input=Append(lstm1-forward, lstm1-
     backward) delay=2 $lstm_opts
fast-lstmp-layer name=lstm3-forward input=Append(lstm2-forward, lstm2-
     backward) delay=-3 $lstm_opts
fast-lstmp-layer name=lstm3-backward input=Append(lstm2-forward, lstm2-
     backward) delay=3 $lstm_opts
output-layer name=output output-delay=$label_delay dim=$num_targets max-
     change=1.5
steps/nnet3/train_rnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir=$train_ivector_dir \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.srand=$srand \
    --trainer.num-epochs=$num_epochs \
    --trainer.samples-per-iter=$samples_per_iter \
    --trainer.optimization.num-jobs-final=1 \
    --trainer.optimization.initial-effective-lrate=$initial_effective_lrate \
    --trainer.optimization.final-effective-lrate=$final_effective_lrate \
```

```
    --trainer.optimization.shrink-value 0.99 \
    --trainer.rnn.num-chunk-per-minibatch=$num_chunk_per_minibatch \
    --trainer.optimization.momentum=$momentum \
    --egs.opts " --nj 3 " \
    --egs.chunk-width=$chunk_width \
    --egs.chunk-left-context=$chunk_left_context \
    --egs.chunk-right-context=$chunk_right_context \
    --egs.chunk-left-context-initial=0 \
    --egs.chunk-right-context-final=0 \
    --egs.dir="$common_egs_dir" \
    --cleanup.remove-egs=$remove_egs \
    --cleanup.preserve-model-interval=100 \
    --use-gpu=no \
    --feat-dir=$train_data_dir \
    --ali-dir=$ali_dir \
    --lang=data/lang \
    --dir=$dir  || exit 1;

# ---> DNN
relu_dim=750
relu-renorm-layer name=tdnn1 dim=1248
relu-renorm-layer name=tdnn2 dim=1248 input=Append(-1,2)
relu-renorm-layer name=tdnn3 dim=1248 input=Append(-3,3)
relu-renorm-layer name=tdnn4 dim=1248 input=Append(-3,3)
relu-renorm-layer name=tdnn5 dim=1248 input=Append(-7,2)
relu-renorm-layer name=tdnn6 dim=1248
output-layer name=output dim=$num_targets max-change=1.5
steps/nnet3/train_dnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir ${train_ivector_dir} \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.num-epochs 3 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.1317 \
    --trainer.optimization.final-effective-lrate 0.01317 \
    --trainer.samples-per-iter 400000 \
    --egs.dir "$common_egs_dir" \
    --egs.stage "$egs_stage" \
    --cleanup.remove-egs $remove_egs \
    --cleanup.preserve-model-interval 50 \
    --feat-dir=$train_data_dir \
    --ali-dir $ali_dir \
    --lang data/lang \
    --use-gpu=no \
    --dir=$dir || exit 1;
steps/nnet3/decode.sh --nj $nj --cmd "$decode_cmd" --acwt 1.0 --post-decode-
     acwt 10.0 --scoring-opts "--min-lmwt 5 " --num-threads $num_threads --
     online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_test_hires
     ${graph_dir} data/test_hires ${dir}/decode_test || exit 1

# ---> RNN+HMM
min_seg_len=1.55
chunk_left_context=40
label_delay=5
xent_regularize=0.1
extra_left_context=50
extra_right_context=20
frames_per_chunk=150
```

```
fast-lstmp-layer name=lstm1 cell-dim=512 recurrent-projection-dim=128 non-
     recurrent-projection-dim=128 delay=-5
fast-lstmp-layer name=lstm2 cell-dim=512 recurrent-projection-dim=128 non-
     recurrent-projection-dim=128 delay=-5
fast-lstmp-layer name=lstm3 cell-dim=512 recurrent-projection-dim=128 non-
     recurrent-projection-dim=128 delay=-5
output-layer name=output input=lstm3 output-delay=$label_delay include-log-
     softmax=false dim=$num_targets max-change=1.5
output-layer name=output-xent input=lstm3 output-delay=$label_delay
     dim=$num_targets learning-rate-factor=$learning_rate_factor max-
     change=1.5
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.1 \
    --chain.l2-regularize 0.00405 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=2000" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width "$frames_per_chunk" \
    --egs.chunk-left-context "$chunk_left_context" \
    --egs.chunk-right-context "$chunk_right_context" \
    --trainer.num-chunk-per-minibatch 128 \
    --trainer.frames-per-iter 1500000 \
    --trainer.max-param-change 2.0 \
    --trainer.num-epochs 4 \
    --trainer.deriv-truncate-margin 10 \
    --trainer.optimization.shrink-value 0.99 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.081 \
    --trainer.optimization.final-effective-lrate 0.0081 \
    --trainer.optimization.momentum 0.0 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 1.0 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 1.0 --post-decode-acwt 10.0 \
    --extra-left-context $extra_left_context  \
    --extra-right-context $extra_right_context  \
    --frames-per-chunk "$frames_per_chunk" \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 5 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;

# ---> DNN+HMM
min_seg_len=1.55
xent_regularize=0.1
relu-renorm-layer name=tdnn1 dim=450
relu-renorm-layer name=tdnn2 input=Append(-1,0,1) dim=450
relu-renorm-layer name=tdnn3 input=Append(-1,0,1,2) dim=450
relu-renorm-layer name=tdnn4 input=Append(-3,0,3) dim=450
relu-renorm-layer name=tdnn5 input=Append(-3,0,3) dim=450
relu-renorm-layer name=tdnn6 input=Append(-6,-3,0) dim=450
```

```
relu-renorm-layer name=prefinal-chain input=tdnn6 dim=450 target-rms=0.5
output-layer name=output include-log-softmax=false dim=$num_targets max-
     change=1.5
relu-renorm-layer name=prefinal-xent input=tdnn6 dim=450 target-rms=0.5
output-layer name=output-xent dim=$num_targets learning-rate-
     factor=$learning_rate_factor max-change=1.5
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.1 \
    --chain.l2-regularize 0.00405 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=2000" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width 150 \
    --trainer.num-chunk-per-minibatch 128 \
    --trainer.frames-per-iter 1500000 \
    --trainer.num-epochs 4 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.081 \
    --trainer.optimization.final-effective-lrate 0.0081 \
    --trainer.max-param-change 2.0 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 1.0 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 1.0 --post-decode-acwt 10.0 \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 5 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 9 para a biblioteca Kaldi.

```
lm_order=3
steps/train_deltas.sh --cmd "$train_cmd" 4200 56000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 7000 84000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=7 --
     right-context=7" 9100 105000 data/train data/lang exp/tri2a_ali exp/tri2b
     || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 10500 140000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 4
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;

# REDES NEURAIS
# ---> Ivector
min_seg_len=2.17
```

```
steps/train_lda_mllt.sh --cmd "$train_cmd" --num-iters 9 --mllt-iters "4 6 8"
     --splice-opts "--left-context=5 --right-context=5" 4200 14000
     $temp_data_root/${train_set}_hires data/lang $gmm_dir
     exp/nnet3${nnet3_affix}/tri5
steps/online/nnet2/train_diag_ubm.sh --cmd "$train_cmd" --nj $nj --num-frames
     311764 --num-threads $num_threads_ubm
     ${temp_data_root}/${train_set}_sp_hires_subset 226
     exp/nnet3${nnet3_affix}/tri5 exp/nnet3${nnet3_affix}/diag_ubm
utils/data/modify_speaker_info.sh --utts-per-spk-max 4
     data/${train_set}_sp_hires_comb
     ${temp_data_root}/${train_set}_sp_hires_comb_max2

# ---> RNN
cell_dim=2048
hidden_dim=2048
recurrent_projection_dim=256
non_recurrent_projection_dim=256
chunk_width=28
chunk_left_context=56
chunk_right_context=56
num_epochs=8
initial_effective_lrate=0.0363
final_effective_lrate=0.00363
momentum=0.7
num_chunk_per_minibatch=140
samples_per_iter=28000
extra_left_context=70
extra_right_context=70
lstm_opts="decay-time=28 cell-dim=$cell_dim"
fast-lstmp-layer name=lstm1-forward input=lda delay=-1 $lstm_opts
fast-lstmp-layer name=lstm1-backward input=lda delay=1 $lstm_opts
fast-lstmp-layer name=lstm2-forward input=Append(lstm1-forward, lstm1-
     backward) delay=-2 $lstm_opts
fast-lstmp-layer name=lstm2-backward input=Append(lstm1-forward, lstm1-
     backward) delay=2 $lstm_opts
fast-lstmp-layer name=lstm3-forward input=Append(lstm2-forward, lstm2-
     backward) delay=-3 $lstm_opts
fast-lstmp-layer name=lstm3-backward input=Append(lstm2-forward, lstm2-
     backward) delay=3 $lstm_opts
fast-lstmp-layer name=lstm4-forward input=Append(lstm3-forward, lstm3-
     backward) delay=-4 $lstm_opts
fast-lstmp-layer name=lstm4-backward input=Append(lstm3-forward, lstm3-
     backward) delay=4 $lstm_opts
output-layer name=output output-delay=$label_delay dim=$num_targets max-
     change=2.1
steps/nnet3/train_rnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir=$train_ivector_dir \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.srand=$srand \
    --trainer.num-epochs=$num_epochs \
    --trainer.samples-per-iter=$samples_per_iter \
    --trainer.optimization.num-jobs-final=1 \
    --trainer.optimization.initial-effective-lrate=$initial_effective_lrate \
    --trainer.optimization.final-effective-lrate=$final_effective_lrate \
    --trainer.optimization.shrink-value 1.386 \
    --trainer.rnn.num-chunk-per-minibatch=$num_chunk_per_minibatch \
    --trainer.optimization.momentum=$momentum \
    --egs.opts " --nj 3 " \
```

```
    --egs.chunk-width=$chunk_width \
    --egs.chunk-left-context=$chunk_left_context \
    --egs.chunk-right-context=$chunk_right_context \
    --egs.chunk-left-context-initial=0 \
    --egs.chunk-right-context-final=0 \
    --egs.dir="$common_egs_dir" \
    --cleanup.remove-egs=$remove_egs \
    --cleanup.preserve-model-interval=140 \
    --use-gpu=no \
    --feat-dir=$train_data_dir \
    --ali-dir=$ali_dir \
    --lang=data/lang \
    --dir=$dir || exit 1;

# ---> DNN
relu_dim=1050
relu-renorm-layer name=tdnn1 dim=1748
relu-renorm-layer name=tdnn2 dim=1748 input=Append(-1,2)
relu-renorm-layer name=tdnn3 dim=1748 input=Append(-3,3)
relu-renorm-layer name=tdnn4 dim=1748 input=Append(-3,3)
relu-renorm-layer name=tdnn5 dim=1748 input=Append(-3,3)
relu-renorm-layer name=tdnn6 dim=1748 input=Append(-3,3)
relu-renorm-layer name=tdnn7 dim=1748 input=Append(-7,2)
relu-renorm-layer name=tdnn8 dim=1748
output-layer name=output dim=$num_targets max-change=2.1
steps/nnet3/train_dnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir ${train_ivector_dir} \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.num-epochs 4 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.2057 \
    --trainer.optimization.final-effective-lrate 0.02057 \
    --egs.dir "$common_egs_dir" \
    --egs.stage "$egs_stage" \
    --cleanup.remove-egs $remove_egs \
    --cleanup.preserve-model-interval 70 \
    --feat-dir=$train_data_dir \
    --ali-dir $ali_dir \
    --lang data/lang \
    --use-gpu=no \
    --dir=$dir || exit 1;
steps/nnet3/decode.sh --nj $nj --cmd "$decode_cmd" --acwt 1.4 --post-decode-
    acwt 14.0 --scoring-opts "--min-lmwt 7 " --num-threads $num_threads --
    online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_test_hires
    ${graph_dir} data/test_hires ${dir}/decode_test || exit 1

# ---> RNN+HMM
min_seg_len=2.17
chunk_left_context=56
chunk_right_context=10
label_delay=7
xent_regularize=0.14
extra_left_context=70
extra_right_context=30
frames_per_chunk=210
fast-lstmp-layer name=lstm1 cell-dim=1024 recurrent-projection-dim=256 non-
    recurrent-projection-dim=256 delay=-7
```

```
fast-lstmp-layer name=lstm2 cell-dim=1024 recurrent-projection-dim=256 non-
     recurrent-projection-dim=256 delay=-7
fast-lstmp-layer name=lstm3 cell-dim=1024 recurrent-projection-dim=256 non-
     recurrent-projection-dim=256 delay=-7
fast-lstmp-layer name=lstm4 cell-dim=1024 recurrent-projection-dim=256 non-
     recurrent-projection-dim=256 delay=-7
output-layer name=output input=lstm4 output-delay=$label_delay include-log-
     softmax=false dim=$num_targets max-change=2.1
output-layer name=output-xent input=lstm4 output-delay=$label_delay
     dim=$num_targets learning-rate-factor=$learning_rate_factor max-
     change=2.1
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.14 \
    --chain.l2-regularize 0.00605 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=2800" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width "$frames_per_chunk" \
    --egs.chunk-left-context "$chunk_left_context" \
    --egs.chunk-right-context "$chunk_right_context" \
    --trainer.num-chunk-per-minibatch 256 \
    --trainer.frames-per-iter 2100000 \
    --trainer.max-param-change 2.8 \
    --trainer.num-epochs 5 \
    --trainer.deriv-truncate-margin 14 \
    --trainer.optimization.shrink-value 1.386 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.121 \
    --trainer.optimization.final-effective-lrate 0.0121 \
    --trainer.optimization.momentum 0.0 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 1.4 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 1.4 --post-decode-acwt 14.0 \
    --extra-left-context $extra_left_context  \
    --extra-right-context $extra_right_context  \
    --frames-per-chunk "$frames_per_chunk" \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 7 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;

# ---> DNN+HMM
min_seg_len=2.17
xent_regularize=0.14
relu-renorm-layer name=tdnn1 dim=630
relu-renorm-layer name=tdnn2 input=Append(-1,0,1) dim=630
relu-renorm-layer name=tdnn3 input=Append(-1,0,1,2) dim=630
relu-renorm-layer name=tdnn4 input=Append(-3,0,3) dim=630
relu-renorm-layer name=tdnn5 input=Append(-3,0,3) dim=630
relu-renorm-layer name=tdnn6 input=Append(-3,0,3) dim=630
```

```
relu-renorm-layer name=tdnn7 input=Append(-3,0,3) dim=630
relu-renorm-layer name=tdnn8 input=Append(-6,-3,0) dim=630
relu-renorm-layer name=prefinal-chain input=tdnn8 dim=630 target-rms=0.7
output-layer name=output include-log-softmax=false dim=$num_targets max-
     change=2.1
relu-renorm-layer name=prefinal-xent input=tdnn8 dim=630 target-rms=0.7
output-layer name=output-xent dim=$num_targets learning-rate-
     factor=$learning_rate_factor max-change=2.1
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.14 \
    --chain.l2-regularize 0.00605 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=2800" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width 210 \
    --trainer.num-chunk-per-minibatch 256 \
    --trainer.frames-per-iter 2100000 \
    --trainer.num-epochs 5 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.121 \
    --trainer.optimization.final-effective-lrate 0.0121 \
    --trainer.max-param-change 2.8 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 1.4 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 1.4 --post-decode-acwt 14.0 \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 7 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;
```

O quadro abaixo apresenta as alterações realizadas na configuração padrão, gerando a configuração de ID 10 para a biblioteca Kaldi.

```
lm_order=4
steps/train_deltas.sh --cmd "$train_cmd" 5400 72000 data/train data/lang
     exp/mono0a_ali exp/tri1 || exit 1;
steps/train_deltas.sh --cmd "$train_cmd" 9000 108000 data/train data/lang
     exp/tri1_ali exp/tri2a || exit 1;
steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts "--left-context=9 --
     right-context=9" 11700 135000 data/train data/lang exp/tri2a_ali
     exp/tri2b || exit 1;
steps/train_sat.sh --cmd "$train_cmd" 13500 180000 data/train data/lang
     exp/tri2b_ali exp/tri3b || exit 1;
steps/decode_fmllr.sh --cmd "$decode_cmd" --nj $nj --num-threads 5
     exp/tri3b/graph data/test exp/tri3b/decode || exit 1;

# REDES NEURAIS
# ---> Ivector
```

```
min_seg_len=2.79
steps/train_lda_mllt.sh --cmd "$train_cmd" --num-iters 12 --mllt-iters "5 7 9"
     --splice-opts "--left-context=6 --right-context=6" 5400 18000
     $temp_data_root/${train_set}_hires data/lang $gmm_dir
     exp/nnet3${nnet3_affix}/tri5
steps/online/nnet2/train_diag_ubm.sh --cmd "$train_cmd" --nj $nj --num-frames
     400840 --num-threads $num_threads_ubm
     ${temp_data_root}/${train_set}_sp_hires_subset 292
     exp/nnet3${nnet3_affix}/tri5 exp/nnet3${nnet3_affix}/diag_ubm
utils/data/modify_speaker_info.sh --utts-per-spk-max 5
     data/${train_set}_sp_hires_comb
     ${temp_data_root}/${train_set}_sp_hires_comb_max2

# ---> RNN
cell_dim=4096
hidden_dim=4096
recurrent_projection_dim=512
non_recurrent_projection_dim=512
chunk_width=36
chunk_left_context=72
chunk_right_context=72
num_epochs=10
initial_effective_lrate=0.0543
final_effective_lrate=0.00543
momentum=0.9
num_chunk_per_minibatch=180
samples_per_iter=36000
extra_left_context=90
extra_right_context=90
lstm_opts="decay-time=36 cell-dim=$cell_dim"
fast-lstmp-layer name=lstm1-forward input=lda delay=-1 $lstm_opts
fast-lstmp-layer name=lstm1-backward input=lda delay=1 $lstm_opts
fast-lstmp-layer name=lstm2-forward input=Append(lstm1-forward, lstm1-
     backward) delay=-2 $lstm_opts
fast-lstmp-layer name=lstm2-backward input=Append(lstm1-forward, lstm1-
     backward) delay=2 $lstm_opts
fast-lstmp-layer name=lstm3-forward input=Append(lstm2-forward, lstm2-
     backward) delay=-3 $lstm_opts
fast-lstmp-layer name=lstm3-backward input=Append(lstm2-forward, lstm2-
     backward) delay=3 $lstm_opts
fast-lstmp-layer name=lstm4-forward input=Append(lstm3-forward, lstm3-
     backward) delay=-4 $lstm_opts
fast-lstmp-layer name=lstm4-backward input=Append(lstm3-forward, lstm3-
     backward) delay=4 $lstm_opts
fast-lstmp-layer name=lstm5-forward input=Append(lstm4-forward, lstm4-
     backward) delay=-5 $lstm_opts
fast-lstmp-layer name=lstm5-backward input=Append(lstm4-forward, lstm4-
     backward) delay=5 $lstm_opts
output-layer name=output output-delay=$label_delay dim=$num_targets max-
     change=2.7
steps/nnet3/train_rnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir=$train_ivector_dir \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.srand=$srand \
    --trainer.num-epochs=$num_epochs \
    --trainer.samples-per-iter=$samples_per_iter \
    --trainer.optimization.num-jobs-final=1 \
    --trainer.optimization.initial-effective-lrate=$initial_effective_lrate \
```

```
    --trainer.optimization.final-effective-lrate=$final_effective_lrate \
    --trainer.optimization.shrink-value 1.782 \
    --trainer.rnn.num-chunk-per-minibatch=$num_chunk_per_minibatch \
    --trainer.optimization.momentum=$momentum \
    --egs.opts " --nj 3 " \
    --egs.chunk-width=$chunk_width \
    --egs.chunk-left-context=$chunk_left_context \
    --egs.chunk-right-context=$chunk_right_context \
    --egs.chunk-left-context-initial=0 \
    --egs.chunk-right-context-final=0 \
    --egs.dir="$common_egs_dir" \
    --cleanup.remove-egs=$remove_egs \
    --cleanup.preserve-model-interval=180 \
    --use-gpu=no \
    --feat-dir=$train_data_dir \
    --ali-dir=$ali_dir \
    --lang=data/lang \
    --dir=$dir  || exit 1;

# ---> DNN
relu_dim=1350
relu-renorm-layer name=tdnn1 dim=2246
relu-renorm-layer name=tdnn2 dim=2246 input=Append(-1,2)
relu-renorm-layer name=tdnn3 dim=2246 input=Append(-3,3)
relu-renorm-layer name=tdnn4 dim=2246 input=Append(-3,3)
relu-renorm-layer name=tdnn5 dim=2746 input=Append(-3,3)
relu-renorm-layer name=tdnn6 dim=2746 input=Append(-3,3)
relu-renorm-layer name=tdnn7 dim=2746 input=Append(-3,3)
relu-renorm-layer name=tdnn8 dim=2246 input=Append(-7,2)
relu-renorm-layer name=tdnn9 dim=2246
output-layer name=output dim=$num_targets max-change=2.7
steps/nnet3/train_dnn.py --stage=$train_stage \
    --cmd="$decode_cmd" \
    --feat.online-ivector-dir ${train_ivector_dir} \
    --feat.cmvn-opts="--norm-means=false --norm-vars=false" \
    --trainer.num-epochs 5 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.3077 \
    --trainer.optimization.final-effective-lrate 0.03077 \
    --egs.dir "$common_egs_dir" \
    --egs.stage "$egs_stage" \
    --cleanup.remove-egs $remove_egs \
    --cleanup.preserve-model-interval 90 \
    --feat-dir=$train_data_dir \
    --ali-dir $ali_dir \
    --lang data/lang \
    --use-gpu=no \
    --dir=$dir || exit 1;
steps/nnet3/decode.sh --nj $nj --cmd "$decode_cmd" --acwt 1.8 --post-decode-
     acwt 18.0 --scoring-opts "--min-lmwt 9 " --num-threads $num_threads --
     online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_test_hires
     ${graph_dir} data/test_hires ${dir}/decode_test || exit 1

# ---> RNN+HMM
min_seg_len=2.79
chunk_left_context=72
chunk_right_context=20
label_delay=9
xent_regularize=0.18
```

```
extra_left_context=90
extra_right_context=40
frames_per_chunk=270
fast-lstmp-layer name=lstm1 cell-dim=2048 recurrent-projection-dim=512 non-
     recurrent-projection-dim=512 delay=-9
fast-lstmp-layer name=lstm2 cell-dim=2048 recurrent-projection-dim=512 non-
     recurrent-projection-dim=512 delay=-9
fast-lstmp-layer name=lstm3 cell-dim=2048 recurrent-projection-dim=512 non-
     recurrent-projection-dim=512 delay=-9
fast-lstmp-layer name=lstm4 cell-dim=2048 recurrent-projection-dim=512 non-
     recurrent-projection-dim=512 delay=-9
fast-lstmp-layer name=lstm5 cell-dim=2048 recurrent-projection-dim=512 non-
     recurrent-projection-dim=512 delay=-9
output-layer name=output input=lstm5 output-delay=$label_delay include-log-
     softmax=false dim=$num_targets max-change=2.7
output-layer name=output-xent input=lstm5 output-delay=$label_delay
     dim=$num_targets learning-rate-factor=$learning_rate_factor max-
     change=2.7
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.18 \
    --chain.l2-regularize 0.00905 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=3600" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width "$frames_per_chunk" \
    --egs.chunk-left-context "$chunk_left_context" \
    --egs.chunk-right-context "$chunk_right_context" \
    --trainer.num-chunk-per-minibatch 512 \
    --trainer.frames-per-iter 2700000 \
    --trainer.max-param-change 3.6 \
    --trainer.num-epochs 7 \
    --trainer.deriv-truncate-margin 18 \
    --trainer.optimization.shrink-value 1.782 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.181 \
    --trainer.optimization.final-effective-lrate 0.0181 \
    --trainer.optimization.momentum 0.0 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 1.8 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 1.8 --post-decode-acwt 18.0 \
    --extra-left-context $extra_left_context  \
    --extra-right-context $extra_right_context  \
    --frames-per-chunk "$frames_per_chunk" \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 9 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;

# ---> DNN+HMM
min_seg_len=2.79
```

```
xent_regularize=0.18
relu-renorm-layer name=tdnn1 dim=810
relu-renorm-layer name=tdnn2 input=Append(-1,0,1) dim=810
relu-renorm-layer name=tdnn3 input=Append(-1,0,1,2) dim=810
relu-renorm-layer name=tdnn4 input=Append(-3,0,3) dim=810
relu-renorm-layer name=tdnn5 input=Append(-3,0,3) dim=810
relu-renorm-layer name=tdnn6 input=Append(-3,0,3) dim=810
relu-renorm-layer name=tdnn7 input=Append(-3,0,3) dim=810
relu-renorm-layer name=tdnn8 input=Append(-3,0,3) dim=810
relu-renorm-layer name=tdnn9 input=Append(-3,0,3) dim=810
relu-renorm-layer name=tdnn10 input=Append(-6,-3,0) dim=810
relu-renorm-layer name=prefinal-chain input=tdnn10 dim=810 target-rms=0.9
output-layer name=output include-log-softmax=false dim=$num_targets max-
     change=2.7
relu-renorm-layer name=prefinal-xent input=tdnn10 dim=810 target-rms=0.9
output-layer name=output-xent dim=$num_targets learning-rate-
     factor=$learning_rate_factor max-change=2.7
steps/nnet3/chain/train.py --stage $train_stage \
    --cmd "$decode_cmd" \
    --feat.online-ivector-dir $train_ivector_dir \
    --feat.cmvn-opts "--norm-means=false --norm-vars=false" \
    --chain.xent-regularize $xent_regularize \
    --chain.leaky-hmm-coefficient 0.18 \
    --chain.l2-regularize 0.00905 \
    --chain.apply-deriv-weights false \
    --chain.lm-opts="--num-extra-lm-states=3600" \
    --egs.dir "$common_egs_dir" \
    --egs.opts "--frames-overlap-per-eg 0" \
    --egs.chunk-width 270 \
    --trainer.num-chunk-per-minibatch 512 \
    --trainer.frames-per-iter 2700000 \
    --trainer.num-epochs 7 \
    --trainer.optimization.num-jobs-final 1 \
    --trainer.optimization.initial-effective-lrate 0.181 \
    --trainer.optimization.final-effective-lrate 0.0181 \
    --trainer.max-param-change 3.6 \
    --cleanup.remove-egs true \
    --feat-dir $train_data_dir \
    --tree-dir $tree_dir \
    --lat-dir $lat_dir \
    --dir $dir
utils/mkgraph.sh --self-loop-scale 1.8 data/lang $dir $dir/graph
steps/nnet3/decode.sh --num-threads 1 --nj $decode_nj --cmd "$decode_cmd" \
    --acwt 1.8 --post-decode-acwt 18.0 \
    --online-ivector-dir exp/nnet3${nnet3_affix}/ivectors_${dset}_hires \
    --scoring-opts "--min-lmwt 9 " \
    $dir/graph data/${dset}_hires $dir/decode_${dset} || exit 1;
```

APÊNDICE E – DEMAIS TIPOS DA BIBLIOTECA HTK

O quadro abaixo apresenta os valores obtidos pelos demais tipos da biblioteca HTK para o corpus LaPS Benchmark.

ID	Tipo	WER	SER
1	HVite (trifone)	99,02%	100%
1	HLRescore (3-grama)	94,95%	100%
2	HVite (trifone)	98,37%	100%
2	HLRescore (3-grama)	92,02%	100%
3	HVite (trifone)	97,88%	100%
3	HLRescore (3-grama)	93,65%	100%
4	HVite (trifone)	98,05%	100%
4	HLRescore (3-grama)	92,51%	100%
5	HVite (trifone)	98,05%	100%
5	HLRescore (3-grama)	93,49%	100%
6	HVite (trifone)	98,37%	100%
6	HLRescore (3-grama)	95,60%	100%
7	HVite (trifone)	98,53%	100%
7	HLRescore (3-grama)	93,316%	100%
8	HVite (trifone)	98,53%	100%
8	HLRescore (3-grama)	92,83%	100%
9	HVite (trifone)	98,37%	100%
9	HLRescore (3-grama)	92,83%	100%
10	HVite (trifone)	97,56%	100%
10	HLRescore (3-grama)	92,18%	100%

O quadro abaixo apresenta os valores obtidos pelos demais tipos da biblioteca HTK para o corpus Constituição Federal.

ID	Tipo	WER	SER
1	HVite (trifone)	97,02%	100%
1	HLRescore (3-grama)	92,08%	100%
2	HVite (trifone)	96,96%	100%
2	HLRescore (3-grama)	89,98%	100%
3	HVite (trifone)	96,38%	100%
3	HLRescore (3-grama)	86,51%	100%
4	HVite (trifone)	96,34%	100%
4	HLRescore (3-grama)	86,74%	100%
5	HVite (trifone)	96,04%	100%
5	HLRescore (3-grama)	85,42%	100%
6	HVite (trifone)	98,43%	100%
6	HLRescore (3-grama)	93,93%	100%
7	HVite (trifone)	97,13%	100%

7	HLRescore (3-grama)	89,20%	100%
8	HVite (trifone)	96,13%	100%
8	HLRescore (3-grama)	84,45%	100%
9	HVite (trifone)	95,97%	100%
9	HLRescore (3-grama)	82,86%	100%
10	HVite (trifone)	97,16%	100%
10	HLRescore (3-grama)	84,45%	100%

APÊNDICE F – DEMAIS TIPOS DA BIBLIOTECA KALDI

O quadro abaixo apresenta os valores obtidos pelos demais tipos da biblioteca Kaldi para o corpus LaPS Benchmark.

ID	Tipo	WER	SER
1	RNN	96,42%	100%
1	MLP-HMM	99,67%	100%
1	RNN-HMM	73,62%	98,33%
2	MLP	46,09%	96,67%
2	MLP-HMM	4,89%	28,33%
2	RNN-HMM	6,68%	36,67%
3	MLP	58,63%	98,33%
3	MLP-HMM	7,49%	36,67%
3	RNN-HMM	8,96%	41,67%
4	MLP	64,66%	96,67%
4	MLP-HMM	12,70%	53,33%
4	RNN-HMM	100%	100%
5	N/A	N/A	N/A
5	N/A	N/A	N/A
5	N/A	N/A	N/A
6	mono0a	7,17%	38,33%
6	tri1	6,03%	38,33%
6	tri2a	8,31%	46,67%
6	tri2b	7%	46,67%
7	tri1	8,31%	40%
7	tri2a	7,98%	38,33%
7	tri2b	8,14%	50%
7	tri3b	8,14%	46,67%
8	mono0a	6,84%	36,67%
8	tri2a	7,17%	40%
8	tri2b	11,89%	61,67%
8	tri3b	7,82%	45%
9	mono0a	6,84%	36,67%
9	tri2a	5,54%	36,67%
9	tri2b	12,05%	51,67%
9	tri3b	6,68%	41,67%
10	mono0a	6,84%	36,67%
10	tri2a	7,33%	41,67%
10	tri2b	12,05%	60%
10	tri3b	9,61%	50%

O quadro abaixo apresenta os valores obtidos pelos demais tipos da biblioteca Kaldi para o corpus Constituição Federal.

ID	Tipo	WER	SER
1	RNN	6,12%	88,89%
1	MLP-HMM	72,78%	100%
1	RNN-HMM	2,22%	61,11%
2	MLP	5,17%	80,16%
2	MLP-HMM	1,06%	40,48%
2	RNN-HMM	1,24%	46,03%
3	MLP	7,68%	90,48%
3	MLP-HMM	1,22%	43,65%
3	RNN-HMM	1,19%	46,03%
4	N/A	N/A	N/A
4	N/A	N/A	N/A
4	N/A	N/A	N/A
5	N/A	N/A	N/A
5	N/A	N/A	N/A
5	N/A	N/A	N/A
6	mono0a	3,65%	79,37%
6	tri1	1,95%	53,97%
6	tri2a	1,67%	46,83%
6	tri2b	1,64%	50,79%
7	mono0a	3,65%	79,37%
7	tri2a	1,75%	47,62%
7	tri2b	2,04%	53,17%
7	tri3b	2,02%	53,97%
8	mono0a	2,33%	66,67%
8	tri2a	1,68%	46,03%
8	tri2b	2,81%	65,87%
8	tri3b	2,77%	60,32%
9	mono0a	2,33%	68,25%
9	tri2a	2,01%	50,79%
9	tri2b	3,73%	76,19%
9	tri3b	3,94%	73,81%
10	mono0a	2,13%	65,08%
10	tri2a	2,15%	55,56%
10	tri2b	4,82%	76,98%
10	tri3b	4,76%	72,22%