Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma análise probabilística do desempenho de sistemas ASR para rádios e tvs brasileira

Diego Marques de Azevedo

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Guilherme Souza Rodrigues - EST/UnB

Coorientador
Prof. Dr. Marcelo Ladeira - CIC/UnB

Brasília
2023

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Marques de Azevedo, Diego
MM357a Uma análise probabilística do desempenho de sistemas ASR para rádios e tvs brasileira / Diego Marques de Azevedo; orientador Guilherme Souza Rodrigues; co-orientador Marcelo Ladeira. -- Brasília, 2023.
46 p.

Dissertação(Mestrado Profissional em Computação Aplicada) - Universidade de Brasília, 2023.

1. Reconhecimento da Fala. 2. Wav2vec. 3. Kaldi. 4. Whisper. 5. Modelo Linear Generalizado. I. Souza Rodrigues, Guilherme, orient. II. Ladeira, Marcelo, co-orient. III. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma análise probabilística do desempenho de sistemas ASR para rádios e tvs brasileira

Diego Marques de Azevedo

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Guilherme Souza Rodrigues - EST/UnB (Orientador)

Prof. Dr. Arnaldo Candido Junior
UTFPR

Prof. Dr. Nelson Cruz Sampaio Neto
UFPA

do Programa de Pós-graduação em Computação Aplicada

Brasília, 8 de Fevereiro de 2023

Dedicatória

Dedico esse estudo e trabalho a Deus, que me conduziu e me deu forças para chegar até aqui, a Alyne Alves dos Santos – minha esposa e eterna companheira – e minha família, pelo apoio, paciência e incentivo.

Agradecimentos

Agradeço ao professor Guilherme Rodrigues e ao professor Marcelo Ladeira pelo empenho e paciência durante todo o trabalho.

Agradeço também a empresa Sérgio Machado Reis Epp pelo apoio durante a pesquisa, ao fornecer o ambiente para teste e os dados para pesquisa.

Resumo

Com o uso de tecnologias baseadas em redes neurais artificiais, os sistemas de Reconhecimento Automático de Fala (do inglês, *Automatic Speech Recognition* – ASR) para o Português Brasileiro (PB) têm apresentado importantes avanços nos últimos anos. Muitos dos trabalhos que alcançaram resultados no estado da arte utilizaram modelos de ponta-a-ponta de código aberto, como o Wav2vec 2.0 e o Whisper. Dentre outras alternativas, estão as ferramentas comerciais, também avaliadas em nosso trabalho. Foram incluídas no estudo as APIs do Google e da Microsoft para a transcrição da fala em texto e também o sistema da VoiceInteraction, chamado Audimus. Nós analisamos o desempenho relativo dessas ferramentas – utilizando como métrica o *Word Error Rate* (WER) – sobre a transcrição de áudios gravados de rádios e canais de TV. Um modelo linear generalizado (do inglês, *Generalized Linear Model* – GLM) foi criado para descrever de forma estocástica, para cada método em questão, o relacionamento entre as propriedades dos áudios utilizados no experimento (exemplo, formato do arquivo e duração do áudio) e a acurácia da transcrição. Dentre outros usos, essa estratégia permite a análise local da *performance* de cada método, sendo possível estimar o desempenho esperado para cada áudio. Assim, é possível identificar não apenas a ferramenta com melhor *performance* global, mas também diagnosticar em quais tipos de áudio cada método teria o melhor desempenho. Essa abordagem possibilita a construção de sistemas ASR otimizados a partir do uso composto de diversos transcritores. Os dados gerados e o código utilizado para construção do modelo estocástico neste experimento estão publicamente disponíveis.

Palavras-chave: reconhecimento da fala, Wav2vec 2.0, Kaldi, Google Speech-to-Text, Microsoft Azure Speech, Audimus.Media, Whisper, corpus, GLM, Português brasileiro.

Abstract

With the use of neural network-based technologies, Automatic Speech Recognition (ASR) systems for Brazilian Portuguese (BP) have shown great progress in the last few years. Several state-of-art results were achieved by open-source end-to-end models, such as the Kaldi toolkit, the Wav2vec 2.0 and the Whisper. Alternative commercial tools are also available, including the Google and Microsoft speech to text APIs and the Audimus System of VoiceInteraction. We analyse the relative performance of such tools – in terms of the so-called Word Error Rate (WER) – when transcribing audio recordings from Brazilian radio and TV channels. A generalized linear model (GLM) is designed to stochastically describe the relationship between some of the audio's properties (e.g. file format and audio duration) and the resulting WER, for each method under consideration. Among other uses, such strategy enables the analysis of local performances, indicating not only which tool shows the best overall performance, but when exactly it is expected to do so. This, in turn, could be used to design an optimized system composed of several transcribers. The data generated for conducting this experiment and the scripts used to produce the stochastic model are publicly available.

Keywords: speech recognition, Wav2vec 2.0, kaldi, Google Speech-to-Text, Microsoft Azure Speech, Audimus.Media, Whisper, text corpus, GLM, Brazilian Portuguese.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ASR *Automatic Speech Recognition.*

BI-LSTM *Bidirectional Long Short Term Memory.*

BIC *Bayesian Information Criteria.*

BRACIS *Brazilian Conference on Intelligent Systems.*

CNN *Convolution Neural Network.*

CTC *Connectionist Temporal classification.*

DNN *Deep Neural Network.*

GLM *Generalized Linear Model.*

GMM *Gaussian Mixture Model.*

HMM *Hidden Markov Model.*

IA Inteligência Artificial.

LAS *Listen, Attend, and Spell.*

LM Modelo de Linguagem.

LSTM *Long Short Term Memory.*

LVCSR *Large Vocabulary Continuous Speech Recognition.*

MFCC *Mel-Frequency Cepstral Coefficients.*

MLP *Multilayer Perceptron.*

PB Português Brasileiro.

PDF *Probability Density Function.*

RNN *Recurrent Neural Network.*

SNR *Signal-To-Noise Ratio.*

TDNN *Time Delay Neural Network.*

UFPA Universidade do Pará.

WER *Word Error Rate.*

WFST *Weighted Finite State Transducer.*

Capítulo 1

Introdução

O grande volume de notícias que é publicado diariamente na mídia impressa e audiovisual inviabiliza ações instantâneas de órgãos, empresas e pessoas caso a análise desse conteúdo seja feita de forma manual. O grande tempo requerido para análise impacta no tempo levado pelo órgão ou empresa citado em um possível cenário de denúncia, por exemplo. Com o objetivo de atender a essa demanda, surgiram serviços especializados no *clipping*, isto é, na seleção de conteúdo publicado a partir dos interesses do cliente.

A Sérgio Machado Reis EPP é uma empresa especializada no monitoramento e gestão de informações baseadas no *clipping* de imprensa e mídias sociais. O conteúdo selecionado pela empresa resulta em um apanhado de notícias de interesse dos clientes. Atualmente com mais de 20 anos no mercado, a empresa possui aproximadamente 342 clientes, dentre eles, órgãos públicos, empresas privadas e deputados. Além de coletar e selecionar notícias, a empresa faz um estudo analítico de narrativas e repercussão em mídia do conteúdo selecionado. Parte de seus colaboradores, dentre os quais estou incluso, tem como objetivo o auxílio e automatização de processos por meio da construção de ferramentas de gestão e seleção do conteúdo.

A seleção das notícias é feita a partir do cadastro em uma base de dados. O cadastro pode ser feito de forma automática por meio de *crawlers* que importam essas notícias a partir de sites requeridos em editais e contratos. Manualmente, essa seleção é feita a partir da marcação de trechos divulgados por veículos de rádio e tv. Atualmente, a empresa grava 640 veículos de rádio e 184 canais de tv durante 24 horas. Para o cadastro e revisão manual, a empresa conta com aproximadamente 92 colaboradores. O monitoramento de toda a programação dos veículos gravados é inviável devido ao período extenso de gravações e a quantidade limitada de colaboradores. Então, os trechos são restritos aos horários das editorias de maior importância definidos pelo cliente. Mesmo com o espaço reduzido de busca, nem todas as notícias publicadas são de interesse do cliente, o que obriga os colaboradores a ouvirem todas as editorias selecionadas.

Durante a contratação do serviço de *clipping*, uma lista contendo todas as palavras-chave de interesse é entregue pelo cliente. Assim, as matérias inseridas são destinadas ao cliente a partir da localização de uma ou mais palavras-chave no texto da notícia. Em rádios e TVs, os trechos destacados são transcritos e inseridos na descrição da notícia após a marcação feita pelo colaborador. A transcrição é realizada pelo software comercial Audimus.Media da VoiceInteraction[1]. A Tabela 1.1 apresenta a quantidade de notícias cadastradas manualmente e o percentual de transcrições dado o total de horas gravadas de rádio e tv.

	Rádio	TV
Horas gravadas	15360	4416
Notícias cadastradas	1895	2914
Transcrições	23%	62%

Tabela 1.1: Quantidade média de notícias cadastradas manualmente e transcritas por software comercial a cada dia.

Observa-se um número relativamente menor de notícias cadastradas de rádio comparadas às de TV, mesmo tendo esta uma quantidade superior de horas gravadas. O cadastro das notícias de rádio exige um maior esforço, pois é necessário que o colaborador ouça toda a programação gravada devido ao maior espaçamento dos trechos de interesse entre os anúncios. Isso impacta no tempo entre veiculação da notícia e a notificação do cliente pela empresa.

A transcrição das notícias cadastradas agiliza o processo de identificação do cliente e reduz o tempo entre a veiculação da notícia e sua notificação. Entretanto, a transcrição de todas as notícias cadastradas é inviável, tendo em vista o grande volume e o alto custo da aquisição de novas licenças do software comercial utilizado pela empresa.

Dentre os avanços apresentados no desenvolvimento de sistemas de ASR para o PB, Junior et al. [3] mostrou que o desempenho de cada modelo está limitado ao tipo de áudio em que é treinado. Em seu trabalho, o modelo treinado utilizando áudios gravados em ambientes não-controlados generaliza melhor para áudios com essas propriedades ao ser comparado com outro modelo, treinado sobre áudios gravados em um ambiente controlado. Em um ambiente real ou mesmo para áudios gravados a partir de rádios e TV, o uso ideal do modelo de transcrição deveria ser escolhido segundo as propriedades do áudio selecionado.

Em outros trabalhos, tais como o de Sampaio et al. [4], sistemas de ASR comerciais como Facebook Wit.ai, Microsoft Azure Speech e Google Cloud Speech-to-text foram comparados. Nos experimentos realizados, foram utilizados os conjuntos de dados Common Voice Mozilla e o VoxForge para testar a *performance*. Além disso, eles mensuraram

[1]https://voiceinteraction.ai/platforms/audimus media.html

a acurácia empírica *global* utilizando várias métricas, tais como, WER, BLEU, METEOR e similaridade de cosseno. Eles concluíram que esses sistemas de reconhecimento de fala, em geral, tinham desempenhos similares, com a vantagem do Microsoft Azure Speech.

Esse trabalho tem como objetivo o desenvolvimento de uma comparação probabilisticamente estruturada do desempenho de sistemas de ASR – tanto comerciais quanto de código aberto – em áudios gravados em ambientes controlados e não-controlados. Para isso, é apresentado um conjunto de dados composto por 9 horas de áudios manualmente transcritos, gravados a partir de rádios e TV brasileiras. Esses dados são usados em um estudo detalhado que estima, dado algumas propriedades do áudio, o WER dos modelos desenvolvidos com o Kaldi, Wav2vec 2.0, Whisper e sistemas comerciais como *Google* e *Microsoft speech to text APIs* e o sistema Audimus da VoiceInteraction. Durante esse estudo, um *Generalized Linear Model* (GLM) é produzido. Esse modelo indica o melhor transcritor (dentre os que foram analisados) para cada áudio que será utilizado. A escolha desse transcritor é definida pela seleção do melhor resultado predito pelo GLM, dado o tipo de áudio observado. Isso é possível, a partir da mensuração da *performance esperada, local* de cada modelo para um determinado tipo de áudio. Os dados gerados para conduzir esse trabalho e os *scripts* utilizados para produzir o modelo estocástico estão disponíveis publicamente.

Esse trabalho se diferencia do trabalho desenvolvido por Sampaio et al. [4] em dois pontos significativos. Primeiramente, em [4], foi considerado diferentes sistemas de ASR e o conjunto de dados utilizado para a avaliação é composto por áudios gravados em um ambiente artificial e controlado. Além disso, em seu trabalho não houve uma tentativa de mensurar a *performance esperada, local*.

A condução desse trabalho, além de otimizar o uso da ferramenta de transcrição na empresa por meio do GLM, possui como diretiva as seguintes hipóteses:

1. O desempenho dos modelos melhora quando aplicado sobre áudios com tratamento de ruído;
2. O uso conjunto dos melhores transcritores para cada tipo de áudio, definido pelo modelo GLM, possui um desempenho melhor do que o uso dos transcritores de forma individual.

É importante ressaltar que este trabalho possui como cenário o uso de áudios já gravados, ou seja, as ferramentas de transcrição são utilizadas de forma *offline*. Portanto, o uso da solução proposta não deve considerar o tempo para a avaliação do melhor transcritor. Nesse contexto, o tempo necessário para a transcrição não é mais importante que a qualidade do texto transcrito.

Capítulo 2

Fundamentos e Estado da Arte

Nesse capítulo, serão descritos os modelos e ferramentas utilizadas na pesquisa. Além das ferramentas comerciais (Azure, Google Speech-to-Text e Audimus), este trabalho utiliza mais três abordagens (Kaldi, Wav2Vec 2.0 e Whisper); todas serão detalhadas a seguir. Dentre as razões pelas quais essas abordagens foram adotadas, está a facilidade de acesso ao código por meio de repositórios com código aberto. Isso permite sua modificação e extensão [1, 5, 6, 2]. Além disso, o grupo de pesquisa da Universidade do Pará (UFPA), FalaBrasil, disponibiliza de forma aberta seu repositório[1]. Ele possui *scripts* para o treinamento de modelos baseados no Kaldi, assim como outros recursos utilizados para a transcrição de áudio em PB. Junior et al. [3] e Stefanel Gris et al. [6], por sua vez, também disponibilizaram um conjunto de modelos pré-treinados com um conjunto de dados em PB usando Wav2Vec 2.0. Radford et al. [2], além de disponibilizarem de forma aberta o repositório[2], possuem uma série de modelos ajustados com dados em PB a partir do modelo geral inicialmente proposto[3].

Primeiramente, definiremos os sistemas de ASR segundo a ótica utilizada pelo Kaldi. Em seguida, descreveremos o processo de transcrição de áudio para texto segundo cada abordagem. Logo depois, apresentaremos a métrica utilizada em nossa análise para mensurar a *performance* de cada modelo, isto é, o *Word Error Rate* (WER).

2.1 Sistemas de ASR

Automatic Speech Recognition (ASR) é um processo de conversão do sinal da fala para uma sequência de palavras [7]. Esse processo é tipicamente composto por quatro componentes: processamento e extração de informações do sinal, modelo acústico (MA), modelo de

[1] https://gitlab.com/fb-asr/
[2] https://github.com/openai/whisper
[3] https://huggingface.co/models?language=pt&other=whisper-event&sort=downloads

linguagem (ML) e decodificador [8]. O primeiro (item B e C na Figura 2.1) remove ruídos e distorções do áudio, converte o sinal do domínio-tempo para o domínio-frequência e extrai informações que serão utilizadas pelo modelo acústico. O segundo (item D) integra o conhecimento acústico com o fonético. O MA é um modelo probabilístico que estima a probabilidade do áudio observado para cada sequência possível de palavras. Em seguida, no ML (item E), é estimada a probabilidade condicional de cada palavra, geralmente por meio do modelo de *N-grama*. O processo de decodificação (item F) estima a sequência hipotética de palavras mais prováveis dado os valores gerados pelo MA e ML.

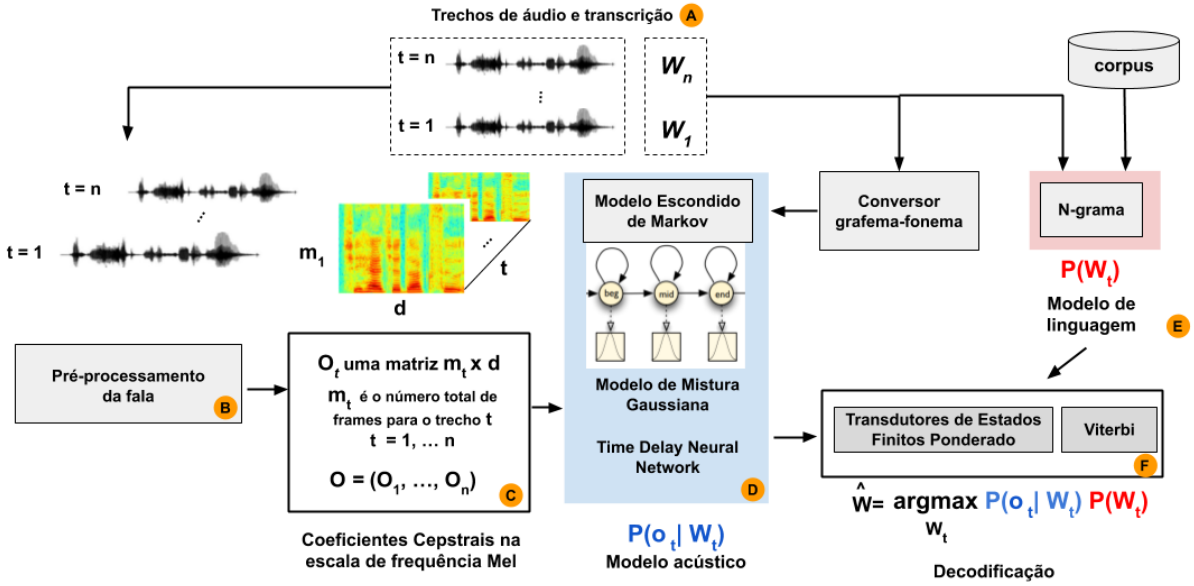


Figura 2.1: Componentes presentes em um sistema de ASR. Essa figura generaliza o funcionamento de um ASR tendo como referência o padrão desenvolvido no Kaldi. É importante ressaltar que esse sistema possui um funcionamento híbrido, ou seja, o treinamento do modelo com *Time Delay Neural Network* (TDNN) (mostrado em D) é feito a partir dos pesos definidos pelo treinamento de outro modelo baseado em *Hidden Markov Model* (HMM) com *Gaussian Mixture Model* (GMM). Outro destaque, seria o uso otimizado do modelo de linguagem durante o estágio de decodificação (mostrado em F). Modelos baseados no Wav2Vec e Whisper – por exemplo –, não utilizam o modelo de linguagem, nem um conversor grafema-fonema em sua arquitetura.

De modo geral, essa abordagem pode ser definida da seguinte forma. Seja $O = (O_1, O_2, ..., O_n)$ a sequência de observações formada a partir dos sucessivos trechos de áudio t. Cada observação O_t é definida por uma matriz $m_t \times d$, onde m_t é o número de frames que compõe o trecho t e $d = 39$, o número de dimensões do vetor MFCC extraído do *frame* (detalhado na Seção 2.1.1). Seja $W_t = w_1, w_2, w_3, ..., w_i$ a sequência de palavras que compõe uma sentença em cada trecho t. Considerando uma amostra observada O_t, é

possível identificar a sequência de palavras mais prováveis à posteriori, $\hat{W}_t$ desse trecho, tal como definido na Equação 2.1.

$$\hat{W}_t = \arg\max_{W_t \in \mathcal{L}} P(W_t|O = O_t). \tag{2.1}$$

Utilizando a regra de Bayes, a Equação 2.1 é redefinida tal como na Equação 2.2.

$$\hat{W}_t = \arg\max_{W_t \in \mathcal{L}} \frac{P(O_t|W_t)P(W_t)}{P(O_t)} = \arg\max_{W_t \in \mathcal{L}} P(O_t|W_t)P(W_t). \tag{2.2}$$

A probabilidade a priori $P(W_t)$ é determinada pelo modelo de linguagem baseado em *N-grama*. Assim, dado que $w_i \in W_t$, a probabilidade de w_i ocorrer é definida pelas N palavras anteriores, tal que $P(w_i|w_{i-(N-1)}, ..., w_{i-1})$. Já $P(O_t|W_t)$ será definido pelo modelo acústico. $P(O_t)$ é uma constante em relação à W_t. Por isso, é possível desprezar essa constante no processo de maximização.

2.1.1 Kaldi

Processamento do sinal

No campo linguístico e fonético da fala, a pronúncia de cada palavra é formada por uma sequência de fonemas [9]. Esse padrão de sons que compõe cada palavra pode ser representado de forma discreta por símbolos. A composição de palavras em fonemas é definida a partir de um dicionário fonético. Esse dicionário, conhecido como modelo léxico, mapeia cada palavra em fonemas. Isso é exemplificado na Tabela 2.1, que apresenta a transcrição fonética das palavras "catou"e "elevador". O mapeamento de fonemas para um símbolo utiliza como referência um modelo léxico e pode ser automatizado por meio do uso de um conversor grafema-fonema.

Palavra	Transcrição fonética
catou	k a t o w
elevador	e l e v a d o r

Tabela 2.1: Transcrição fonética de palavras em Português Brasileiro.

Os fonemas podem ser classificados em consoantes, vogais ou semivogais. No Português Brasileiro (PB), consoantes são os sons produzidos a partir do encontro do ar vindo dos pulmões com obstáculos (língua, dentes e lábios). Vogais são os fonemas produzidos apenas pela boca e fossas nasais, sem encontrar obstáculos.

Durante o processo da fala, a diferença fundamental entre os sons é definida pela pronúncia de fonemas sonoros e surdos. O primeiro é produzido a partir da vibração das pregas vocais. O segundo é formado quando as pregais vocais estão em repouso [9].

Os timbres são criados a partir das ressonâncias na cavidade oral (chamada também de formantes). Essa ressonância é criada pela passagem do som nas diferentes formas de cavidade determinadas pela posição da língua e dos lábios. A taxa relativa ao ciclo de abertura e fechamento das pregas vocais, durante um dado intervalo, para a produção de fonemas sonoros ou sons vocalizados é chamada de frequência fundamental (*F0*).

Como visto na Figura 2.1 (item A, B e C), durante o treinamento do modelo ASR o conjunto de sinais de áudio na base é dividido em segmentos denominados *frames* a partir de uma técnica chamada janelamento. Cada *frame* deve ser pequeno o suficiente, tal que, o sinal da fala delimitado no *frame* esteja próximo de uma onda estacionária. É comum que a janela possua 25ms e deslize por um intervalo de 10ms. A sobreposição é necessária caso exista alguma transição ao longo da evolução do espectro. Logo após, é extraído de cada *frame* os vetores paramétricos (pré-processamento) comumente representados por meio de Coeficientes Cepstrais na escala de frequência Mel, do inglês *Mel-Frequency Cepstral Coefficients* (MFCC)[10]. O pré-processamento do áudio bem como as técnicas aplicadas para a extração de vetores em MFCC são detalhas em [10, 11]. Ao final, é produzido uma lista de matrizes $O = (O_1, \ldots, O_n)$, onde O_t é uma matriz $m_t \times d$, para $t = 1, \ldots, n$ e d um vetor em MFCC de dimensão 39 extraído de cada *frame*.

Modelo Acústico

No modelo acústico (item D na Figura 2.1), $P(O_t|W_t)$ é calculado a partir da aplicação de Modelo Escondido de Markov, do inglês *Hidden Markov Model* (HMM), com topologia do tipo *"left-to-right"*. Cada estado em um HMM será um subfonema definido pelo conversor grafema-fonema. Isso é possível, pois cada fonema pode ser dividido em três subfonemas (trifones). Essa divisão é praticada como uma forma de capturar o contexto determinado pela influência dos fonemas vizinhos durante a pronúncia do fonema central. A transição entre um estado e outro, por exemplo, o estado i para o estado j, é representado por a_{ij}. A probabilidade de emissão é definida por $b_j(O_{t_j})$ e expressa a probabilidade dos vetores MFCC em O_{t_j} serem gerados pelo subfonema do estado j. Na Figura 2.1 item D em amarelo, cada nó é um subfonema e cada retângulo embaixo representa uma *Probability Density Function* (PDF) que estima a $b_j(O_{t_j})$. Na abordagem representada, a PDF pode ser definida por um *Gaussian Mixture Model* (GMM) e *Time Delay Neural Network* (TDNN). O uso do GMM e TDNN são detalhados em [12].

Modelo de Linguagem

O modelo de linguagem (item E na Figura 2.1) estima a probabilidade $P(w_i|w_{i-(N-1)}, ..., w_{i-1})$ de uma palavra dado a sequência de palavras $w_1, ..., w_i$ que a antecedem. O cálculo de $P(w_t)$ é definido pelo produto das probabilidades condicionais de cada palavra e pode ser

determinado pelo uso de *N-gramas*. A aplicação do modelo de linguagem em sistemas ASR é acompanhada de uma constante empírica. Seu uso pondera a determinação de uma palavra a partir da formação dos trifones que a compõe, tal como detalhado em [10]. A construção do ML não está restrita ao vocabulário utilizado no ASR. Ele pode ser construído a partir de outro vocabulário e reutilizado. Essa abordagem foi adotada durante os experimentos realizados no Capítulo 3.

Decodificador

O processo de decodificação (item F na Figura 2.1) busca encontrar a sequência de estados (fonemas) que maximiza a probabilidade de observação, dado um trecho de áudio. Devido ao número extremamente grande de possíveis sequências, é necessário que a busca por essa sequência seja computacionalmente viável. Nesse sentido, é utilizado um algoritmo de otimização, denominado Viterbi. Para o suporte a vocabulários de larga escala, sistemas de ASR (como o Kaldi) combinam recursos, tais como o dicionário fonético, modelo de linguagem e HMM em uma estrutura denominada *Weighted Finite State Transducer* (WFST) [13]. Ela funciona como uma máquina de estados finita que mapeia uma sequência de estados definida no HMM para uma sequência de palavras. Esse mapeamento é dividido em quatro etapas, realizadas por componentes denominados transdutores e é capaz de otimizar ainda mais o processo de busca pela sequência de palavras. Esse processo é detalhado em [14, 10].

Os corpus de fala contínua com um grande número de palavras são denominados *Large Vocabulary Continuous Speech Recognition* (LVCSR). O aumento do vocabulário estende o espaço de busca formado pelos fonemas. Consequentemente, o tempo necessário para encontrar o fonema relativo a cada observação é maior. A busca em LVCSR é otimizada com a aplicação do WFST e o algoritmo de Viterbi, tal como descrito em [10].

Estrutura Kaldi

O Kaldi é um pacote de ferramentas desenvolvido em 2009 por um grupo de pesquisadores da Universidade Johns Hopkins. Seu uso foi adotado pelo grupo de pesquisa da Universidade do Pará (UFPA), FalaBrasil, que disponibiliza de forma aberta seu repositório[4]. Além disso, ele possui uma ampla documentação[5], um repositório[6] contendo exemplos de *scripts* com aplicação em diferentes base de dados, dentre elas o *Linguistic Data Consortium*, VoxForge e *Common Voice Mozilla*.

[4]https://gitlab.com/fb-asr/
[5]http://kaldi-asr.org/doc/
[6]https://github.com/kaldi-asr/kaldi

Com ele, a criação de um novo sistema de ASR aplicado a um determinado cenário se torna mais simples. Sua estrutura é apresentada na Figura 2.2 por meio da organização de suas pastas. Para o treinamento de um novo modelo, basta criar um diretório e seguir o padrão ditado pelo projeto de referência denominado wsj. Os dados de entrada são compostos por: arquivos de áudio do tipo wav (wav.scp); arquivos de texto, correspondente às transcrições dos áudios (corpus.txt) e arquivo léxico, contendo o vocabulário e sua composição fonética (lexicon.txt). O Kaldi utiliza como padrão *sil* e *spn* para fonemas silenciosos, tais como ruídos e pausas. Além disso, sua estrutura permite informar o gênero e a nomeação do locutor (spk2gender.txt).

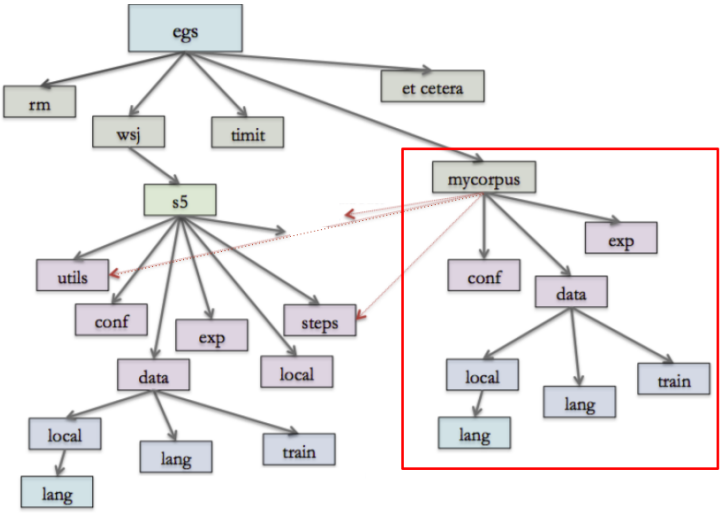


Figura 2.2: Organização dos diretórios do Kaldi. A estrutura que utilizamos para o uso do Kaldi em nosso estudo é destacada em vermelho. Basicamente, replicamos toda a árvore de diretórios definida a partir da pasta *mycorpus*. Utilizamos como referência, o projeto desenvolvido e disponibilizado pela grupo Fala Brasil da Universidade do Pará. O pré-processamento do texto e dos áudios foram realizados por meio de *scritps* criados durante o uso dessa ferramenta e inseridos na pasta *local*.

O Kaldi utiliza o WFST durante o processo de decodificação. O suporte é dado pelo uso da biblioteca de código aberto OpenFst[7]. A estrutura necessária para aplicação da biblioteca é criada a partir de novos arquivos do tipo fst, formados durante a preparação dos dados de entrada.

2.1.2 Wav2Vec

O Wav2Vec 2.0 é um framework para aprendizagem auto-supervisionada de representações da fala. Sua arquitetura é vista na Figura 2.3. Nessa abordagem, o modelo é pré-treinado de forma auto-supervisionada e ajustado (*fine-tuning*) em uma tarefa supervisionada.

[7]https://kaldi-asr.org/doc/fst_algo.html

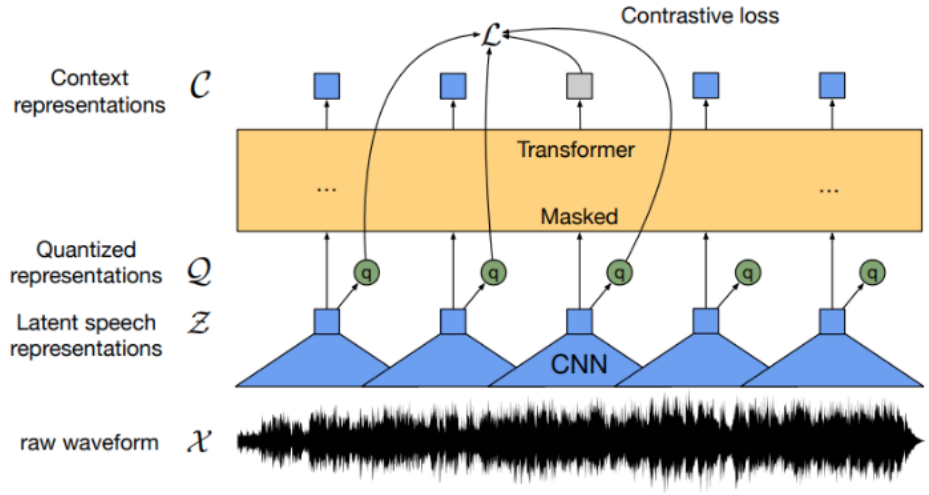


Figura 2.3: Arquitetura do Wav2Vec 2.0 [1]. Ela é composta por um *encoder* convolucional multi-camada definido por um CNN (mostrado em azul) que gera representações latentes do sinal da fala. Esses dados são inseridos em uma rede de contexto (mostrado em amarelo) formada por um *Transformer*, treinado para aprender representações contextualizadas a partir do uso de mascaramento de representações quantizadas (em verde) formadas com os dados gerados pelo *encoder*. Durante o processo de *fine-tunning*, uma camada é adicionada no topo da rede de contexto. De forma supervisionada, o modelo é ajustado para predizer caracteres ou fonemas que compõe a transcrição a partir de representações do áudio bruto geradas pela rede de contexto.

O primeiro estágio é composto por um *encoder* convolucional multi-camada (CNN) $f : X \longrightarrow Z$. Nesse processo, o áudio bruto X é mapeado em representações latentes da fala $z_1, ..., z_T$ em T *time-steps*. Os parâmetros definidos em cada bloco de convolução são detalhados em [1]. Em seguida, os dados representados por Z são aplicados em uma rede de contexto $g : Z \longrightarrow C$, que segue a arquitetura *Transformer*, e que mapeia as representações latentes Z em representações contextualizadas $c_1, ..., c_T$.

Um conjunto finito de dados gerados pelo *encoder* é discretizado em $q_1, ..., q_t$ por um módulo de quantização $h : Z \longrightarrow Q$, detalhado em [1, 15]. Cada representação latente quantizada q_t é utilizada no processo de aprendizagem de representações contextualizadas. Isso é feito por uma tarefa contrastiva de diferenciação similar ao que ocorre no BERT [16]. Nela, um conjunto de representações latentes são mascaradas. Nesse processo, o objetivo é fazer com que o modelo seja capaz de predizer tais partes. Durante essa etapa, o modelo deve identificar representações quantizadas q_t verdadeiras a partir de um conjunto falso de *time-steps* presentes em um contexto mascarado.

Em seguida, o modelo é ajustado *fine-tuning* por meio de uma tarefa supervisionada. Nessa etapa, o áudio bruto e sua respectiva transcrição são utilizados. Uma camada de saída randomicamente inicializada é adicionada no topo da rede de contexto. O modelo é, então, ajustado para predizer um conjunto finito n de alvos (caracteres ou fonemas)

determinado pelo áudio utilizado como entrada. A aprendizagem ocorre por meio do *Connectionist Temporal classification* (CTC) [17].

Apesar do Wav2Vec não precisar de um modelo de linguagem ou do uso de um dicionário fonético para a geração de transcrições gramaticalmente coerentes, sua combinação com esses recursos é – potencialmente – capaz de promover um aumento significativo na *performance* final do modelo de transcrição [1]. Para isso, é possível que a cada *time-step* – durante o processo de decodificação – o *beam search* seja aplicado sobre as probabilidades de todos os possíveis caracteres de saída definidos pelo modelo ajustado. Durante a escolha do carácter mais provável em um determinado *time-step*, a probabilidade definida pelo modelo de linguagem (N-grama) para o possível carácter é considerada. Isso previne o transcritor de gerar uma sequência de letras que não formem palavras válidas (de acordo com o LM).

2.1.3 Whisper

Whisper é um sistema ASR de uso geral, proposto pela equipe da OpenAI [2]. Em seu estudo, os autores têm como foco explorar as capacidades do pré-treinamento supervisionado em larga escala para o reconhecimento de fala e desenvolver um sistema de processamento da fala robusto, capaz de atingir resultados com alta qualidade sem a necessidade de ajuste (*fine-tunning*), tal como o Wav2Vec. Para isso, a arquitetura vista na Figura 2.4 é adotada.

Esse modelo utiliza um *Transformer encoder-decoder* [18], treinado em várias tarefas relacionadas ao processamento da fala, tais como ASR em múltiplas linguagens, tradução de fala, reconhecimento da linguagem e detecção de voz. O treinamento do modelo foi realizado com 680 000 horas de dados coletados da web. Dentre eles, 117 000 horas possuem 96 linguagens diferentes do inglês americano e 125 000 horas são de traduções para o inglês. Durante a coleta dos dados, os autores removeram as transcrições geradas por outros sistemas de ASR a partir da aplicação de várias heurísticas, detalhas em [2]. Esse conjunto de dados abrange uma grande variedade de áudios gravados em diferentes ambientes, com diferentes locutores e falas em múltiplas linguagens.

O treinamento do modelo em múltiplas tarefas foi condicionado a uma sequência de *tokens* de entrada específicos para cada tipo de tarefa e submetidos ao *decoder*. A composição dessa sequência é detalhada na parte inferior da Figura 2.4. Os dois primeiros blocos são formados por textos transcritos, anteriores ao início da transcrição do segmento de áudio. Ao adicionar esse trecho precedente, os autores visam definir o contexto no qual o trecho de áudio será transcrito, e assim, resolver um possível problema relativo à ambiguidade de áudios. O início da tarefa de transcrição é indicado pelo *token* `<|startoftranscript|>`. Em casos em que não há fala, é utilizado o *token* `<|nospeech|>`. A

detecção da linguagem é realizada pelo modelo VoxLingua107 [19]. Tarefas como transcrição e tradução são indicadas pelos *tokens* `<|transcribe|>` e `<|translate|>`. Tarefas como a predição ou não de *timestamps* é definida por `<|notimestamps|>`. O final da transcrição é indicado pelo *token* `<|endoftranscript|>`.

No treinamento, foram utilizados áudios com 16 kHz, transformados em espectrograma Mel em escala logarítmica (log-Mel-espectrogramas) com 80 canais. A segmentação dos áudios foi realizada por meio do janelamento, a partir de janelas de 25ms e deslize por um intervalo de 10ms. Em seguida, os dados foram normalizados (para uma escala de -1 a 1, com média igual a 0) e processados pelo *encoder*. O *encoder* é composto por duas camadas de convolução e função de ativação GELU. Os autores adicionaram às *embeddings* de saída *embeddings* geradas por posição sinusoidal, e então, aplicaram estes dados aos blocos de *Transformer encoder*. O *decoder* utiliza *embeddings* de posição aprendida gerada a partir dos *tokens* de entrada. Tanto o *encoder* quanto o *decoder* possuem o mesmo número de blocos de *Transformer*. Detalhes relacionados aos hiperparâmetros utilizados podem ser vistos em [2].

2.1.4 Google Speech-to-text

As metodologias utilizadas no processo de construção de um sistema de ASR comercial não são totalmente reveladas por razões comerciais. Entretanto, a equipe de Inteligência Artificial (IA) do Google propôs um modelo de ponta-a-ponta baseado na arquitetura do *Encoder-Decoder* com *Recurrent Neural Network* (RNN), chamada *Listen, Attend, and Spell* (LAS) [20]. Depois, foi proposto uma melhoria no modelo. Ele passou a utilizar um *Multi-headed Attention Layer*, isto é, uma nova métrica de treinamento baseada na taxa mínima de palavras erradas com um modelo de linguagem externo. Os detalhes dessa arquitetura são definidos em [21]. Os dois trabalhos foram treinado e avaliados sobre conjunto de dados *Google Voice Search*.

2.1.5 Microsoft Azure Speech

A equipe de IA da Microsoft apresentou uma versão de seu sistema de ASR treinado sobre os conjuntos de dados *Switchboard* e *CallHome* [22]. Nessa versão, o modelo utilizado é formado por um conjunto de arquiteturas detalhado em [23]. Junto a essa estrutura, também é utilizado um modelo baseado no CNN com um *Bidirectional Long Short Term Memory* (BI-LSTM) e um modelo de linguagem baseado no *Long Short Term Memory* (LSTM).

2.1.6 Audimus.Media

O Audimus.Media é um sistema de ASR desenvolvido pela VoiceInteraction. Esse sistema foca em serviços de captação de legendas (*closed captioning*). Inicialmente, ele foi criado para o Português Europeu [24]. Atualmente, esse sistema pode ser aplicado em mais de 30 línguas, incluindo o PB. No Brasil, ele é utilizado pelas principais empresas de mídia[8]. Esse sistema é resultado dos trabalhados publicados em [25, 26, 27]. Ele é um sistema híbrido de reconhecimento de fala que combina o *Hidden Markov Model* (HMM) com o *Multilayer Perceptron* (MLP). Sua arquitetura HMM-MLP é usada no modelo acústico para gerar a probabilidade *a posteriori* dos fonemas, dado um determinado segmento de áudio observado. Durante o processo de decodificação, assim como no Kaldi, é utilizado um WFST. Seu uso integra a saída do modelo acústico a um dicionário fonético e um modelo de linguagem. Os dados gerados a partir do processo de decodificação são utilizados para mensurar a confidência de cada palavra. Ao final, um classificador baseado no uso de entropia máxima estima a certeza associada a cada palavra de saída.

2.2 WER

A avaliação de trechos transcritos por sistemas de ASR pode ser medida pelo *Word Error Rate* (WER). Ela é definida pela divisão entre o somatório do número de erros, contabilizado pelas operações de substituição, inserção e deleção em cada trecho, e o somatório do número de acertos, contabilizado pelas operações de substituição, exclusão e acertos. A Figura 2.5 apresenta um trecho de áudio transcrito, no qual a primeira linha corresponde a sequência de palavras utilizadas como referência, a segunda linha o trecho transcrito pelo sistema de ASR e a última linha as operações necessárias para que o trecho transcrito seja semelhante ao trecho utilizado como referência.

A partir do conjunto de trechos transcritos, é possível calcular o WER pela Equação 2.3, onde S é o número de substituições, D é o número de exclusões, I é o número de inserções e C é o número de palavras corretas, tal que:

$$WER = \frac{S + D + I}{S + D + C}. \tag{2.3}$$

Em nossa análise, o *Word Error Rate* (WER) será calculado de forma *global* e *local*. O uso do WER *global* é definido pelo somatório das operações S, D, I e C em cada sequência reconhecida pelo transcritor. O WER *local* é estimado para cada sentença transcrita que foi reconhecida a partir de cada áudio, individualmente. De forma geral, tem-se em

[8] https://voiceinteraction.ai/platforms/audimus_media.html

vista que, a medida que o WER é menor, melhor será o desempenho do sistema de ASR avaliado.

2.3 Estado da Arte

Os sistemas de *Automatic Speech Recognition* (ASR) baseados em técnicas de *deep learning* fizeram progresso ao treinar os modelos com diferentes níveis de ruído e distância de microfones [28], o que reduziu o problema de degradação de *performance* em áudios com grande variação de ruído. Entretanto, essa abordagem requer um grande montante de dados para o treinamento, o que para modelos em Português Brasileiro (PB), torna-se um obstáculo impeditivo.

Em 2020, um conjunto de dados abertos de aproximadamente 376 horas foi publicado. A maior parte desses dados é formado por falas em ambientes controlados. Eles são compostos por: Common Voice Corpus versão 6.1[9], Sid dataset[10], VoxForge[11], LapsBM1.5[12], Multilingual LibriSpeech (MLS) [29] e CETUC [30]. Em 2021, o corpus Multilingual TEDx [31] e o Common Voice Corpus versão 7.0[13] incrementaram o montante de dados publicamente disponíveis, atingindo um total de 574 horas. Ainda em 2021, o conjunto de dados CORAA [3] também foi publicado, contribuindo com 290.77 horas de gravações de discursos em espaços não controlados. Esse conjunto de dados é composto por 5 corpus: ALIP, C-ORAL Brasil I, NURC-Recife, SP2010 and TEDx em português.

Vários trabalhos contribuíram para o avanço do ASR em PB. Batista et al. [32] treinou um modelo com *Hidden Markov Model* (HMM) e *Deep Neural Network* (DNN)(HMM-DNN) utilizando o pacote de ferramentas Kaldi [5], alcançando um *Word Error Rate* (WER) de 4.75% no corpus LapsBM1.4. Quintanilha et al. [33] propôs um modelo de *ponta-a-ponta* baseado na topologia do DeepSpeech 2 [34]. Esse trabalho obteve um WER de 25.45% usando os seguintes corpus: Sid, Voxforge, LapsBM1.4, CSLU Spoltech [35] e CETUC.

Com o advento da aprendizagem auto-supervisionada, a partir da publicação do Wav2Vec 2.0 [1], um novo avanço foi alcançado. A partir do uso de um modelo pré-treinado (`Wav2Vec-XLSR-53` [36]), Stefanel Gris et al. [6] e Junior et al. [3] alcançaram um WER de 10.5% e 24.18%. Ambos ajustaram seus modelos (*fine-tunning*) com um conjunto de dados em PB. Enquanto Stefanel Gris et al. [6] obteve esse resultado com um conjunto de dados composto por: Sid, Voxforge, LapsBM1.4, CSLU Spoltech, MLS

[9] https://commonvoice.mozilla.org/pt/datasets
[10] https://doi.org/10.17771/PUCRio.acad.8372
[11] http://www.voxforge.org/pt/downloads
[12] https://laps.ufpa.brfalabrasil/
[13] https://commonvoice.mozilla.org/pt/datasets

e Common Voice Corpus; Junior et al. [3] utilizou o CORAA. Uma base de dados com áudios gravados tanto em ambientes controlados quanto em ambientes não-controlados, isto é, com uma maior diversidade.

Em 2022, o surgimento do Whisper [2] possibilitou não somente a transcrição do áudio para texto feita por um modelo, mas também a execução de múltiplas tarefas, tais como: o reconhecimento de fala em múltiplas linguagens, tradução de fala, reconhecimento da linguagem e detecção de voz. Embora o objetivo dos autores fosse desenvolver um sistema de processamento da fala robusto, capaz de atingir resultados com alta qualidade sem a necessidade de ajuste (*fine-tunning*), existe a possibilidade de alcançar resultados melhores a partir do *fine-tunning* do modelo inicialmente proposto em uma linguagem específica, como o PB [2]. Nesse sentido, um grupo formado por vários pesquisadores e com grande destaque devido à promoção do uso de modelos baseados em *deep learning*, chamado HuggingFace[14], realizou um evento visando promover o uso do Whisper. A partir desse evento, vários modelos para o PB foram ajustados com o Common Voice Corpus. Entre eles, está o desenvolvido por Botero Jose [37], com um WER de 6.58%.

Além disso, fazem parte desse avanço e pesquisa, ferramentas de transcrição comerciais tais como: *Google Cloud Speech-to-Text*[15], *Microsoft Azure Speech*[16] e o sistema Audimus da VoiceInteraction[17].

[14]https://huggingface.co/
[15]https://cloud.google.com/speech-to-text?hl=pt-br
[16]https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/
[17]https://www.voice-interaction.com/br/audimus-media-legendagem-automatica-em-tempo-real/

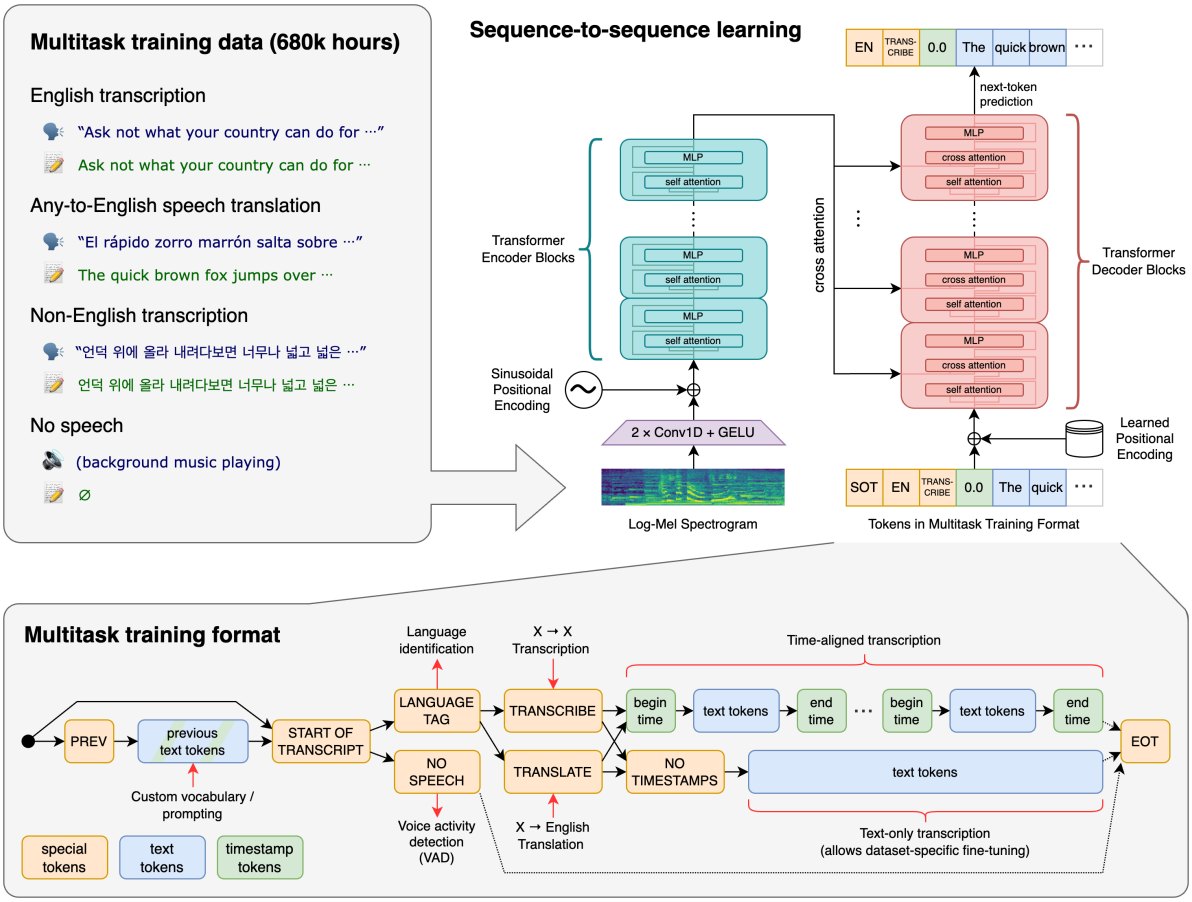


Figura 2.4: Arquitetura do Whisper [2]. Ela possui um *encoder* composto por duas camadas de convolução, com função de ativação GELU e blocos definidos por um *Transformer* (lado superior direito, em azul). O mesmo número de blocos é utilizado no *Transformer decoder* (lado superior direito, em rosa) que utiliza, durante o treinamento, *embeddings* de posição aprendida formada a partir de uma sequência de *tokens* que caracteriza o tipo de tarefa a ser executado. A composição da sequência de *tokens* é definida na parte inferior da figura. O dado de saída indica qual linguagem é utilizada no texto e o tipo de tarefa que foi executado.

```
common_voice_pt_19275111 ref   ***   você  poderia  ter  morrido  depois  que   a    paz   fosse  declarada
common_voice_pt_19275111 hyp  folha  você  poderia  ter  morrido  depois  ***  ***  capaz   foi   declarada
common_voice_pt_19275111 op     I     C      C      C      C        C      D    D     S      S       C
```

Figura 2.5: Exemplificação das operações que compõe o WER. A substituição (S) ocorre a partir da predição incorreta de uma palavra pelo transcritor. A inserção (I) ocorre quando existe uma palavra predita pelo transcrito, porém, no texto de referência, não existe uma palavra para a mesma posição. A operação de remoção ou deleção (D) seria o caso inverso da operação de inserção. Quando tanto a palavra predita quanto a palavra de referência são iguais e ocupam a mesma posição, então é marcado como correto (C).

Capítulo 3

Metodologia

Nesse capítulo, vamos apresentar a abordagem usada para comparar os sistemas de ASR descritos inicialmente e gerar um modelo estocástico capaz de estimar o WER *esperado*, dado as propriedades do áudio.

3.1 Conjunto de Dados de Teste

O WER *esperado, local* e *global* foi calculado sobre um *corpus* de teste em que os áudios apresentam uma grande diversidade acústica. Os dados utilizados no treinamento dos outros modelos não são compostos por gravações de rádio. Para que o ambiente no qual o modelo desenvolvido nessa pesquisa pudesse ser simulado, foi necessário a criação de uma nova base de dados.

Esse novo conjunto de dados é formado por áudios transcritos manualmente e possuem revisão realizada em pares por uma empresa especializada[1]. As gravações utilizadas foram coletadas ao longo do ano de 2020 a partir de rádios e canais de TV brasileiros. Elas possuem discursos espontâneos presentes em entrevistas, diálogos gravados em ambientes naturais, ambientes com ruído, fundo musical e ambientes controlados (estúdios para apresentação de notícias). Foram coletados áudios de 91 canais (43 de rádio e 48 de TV) das regiões norte, sudeste e nordeste do Brasil. Eles totalizam 418 arquivos de áudio, que combinados somam 9 horas de duração e 1.225.785 palavras. A duração de cada arquivo varia de 3 a 892 segundos. Eles representam uma amostra estratificada da base de dados da empresa de *clipping* descrita inicialmente. Esse conjunto de dados está publicamente acessível[2] e representa um sub-produto importante desse trabalho.

Durante essa pesquisa, todas as transcrições (tanto as geradas pelos modelos quanto a manualmente realizada) passaram por um estágio de pré-processamento. Foram remo-

[1] https://www.audiotext.com.br/
[2] https://github.com/diegomarq/BRTVRAD

Propriedades do Áudio	Min	Média	Max
max volume	-41.4	-18.8	-7.1
avg volume	-22.6	-3.1	0.0
bitrate (Kbs)	8.0	40.7	134.0
snr	-0.369	-0.012	0.381
max volume nr	-84.3	-25.6	-22.3
avg volume nr	-60.8	-8.3	-2.4
bitrate nr (Kbs)	64.0	64.0	68.0
snr nr	-0.007	-0.000	0.005

Tabela 3.1: Os valores apresentados foram calculados tanto para os áudios originais, como para os áudios com tratamento de ruído. As propriedades que possuem tratamento de ruído foram destacadas por meio do termo "nr".

vidos do texto caracteres especiais, tais como pontuação, parenteses e hífen; o símbolo de porcentagem (%) foi transcrito para (porcento); números cardinais e ordinais foram expandidos para sua forma literal utilizando a biblioteca *num2words*[3]; os textos foram normalizados para caracteres minúsculos.

Dentre as hipóteses desse trabalho, foi observado se o tratamento de ruído sobre os áudios poderia impactar positivamente a *performance* dos modelos de ASR. Nesse sentido, passamos a utilizar duas categorias de áudio em nossas análises, são elas:

1. Áudios em sua forma original, ou seja, não foi aplicado nenhum tratamento;
2. Áudios que possuem um tratamento básico de remoção de ruído. Em nosso experimento, esse conjunto de áudios possui o sufixo "nr pre-processing technique"ou apenas "nr".

O processo de remoção de ruído utilizado em nossa análise é feito por meio da aplicação de alguns filtros sobre a frequência dos áudios. Esses filtros permitem que a frequência de cada áudio esteja somente entre 200Hz e 1500Hz. Após isso, foi aplicado o método de normalização definido no FFMPEG[4].

O treinamento do modelo estocástico desenvolvido nesse trabalho utiliza como dado de entrada propriedades do áudio capazes de definir o perfil do áudio observado. Para isso, alguns metadados foram extraídos, tais como o volume máximo e médio, *bitrate* e o *Signal-To-Noise Ratio* (SNR). Por meio dos valores obtidos, visualizados na Tabela 3.1, é possível destacar a grande variedade nos dados utilizados para o teste dos modelos.

[3]https://github.com/savoirfairelinux/num2words
[4]http://ffmpeg.org/ffmpeg-all.html#loudnorm

3.2 Conjunto de Dados de Treino

O conjunto de dados CORAA foi usado para treinar o nosso modelo baseado no Kaldi e o modelo baseado no Wav2Vec apresentado em [3]. O CORAA é composto por 290.77 horas de áudios gravados em ambientes controlados e não-controlados. As gravações compreendem entrevistas, diálogos, monólogos, conferências e aulas, leituras e conversas de palco. A duração de cada áudio varia entre 2.4 e 7.6 segundos. Eles foram manualmente transcritos em pares. A divisão dos dados para o processo de aprendizado do Kaldi seguiu como proposto por [3]. Utilizamos 273 horas de áudio para treinamento e o restante para validação.

3.3 Processamento do Modelo de ASR

3.3.1 Kaldi

Durante o treinamento do Kaldi, utilizamos como modelo de linguagem o modelo pertencente ao Audimus.Media. O dicionário fonético foi criado a partir da ferramenta disponibilizada pelo grupo de pesquisa da Universidade do Pará, FalaBrasil. A ferramenta chamada grafema-para-fonema (G2P) mapeou para fonemas aproximadamente 23 mil palavras presentes no vocabulário do Audimus.Media LM. Por padrão, o Kaldi utilizada áudios monofónicos com 16kHz no formato WAVE. Para atender esse padrão, utilizamos o FFMPEG.

Após a padronização dos áudios, eles foram codificados em um vetor baseado em MFCC para, em seguida, serem utilizados pelo modelo acústico. O treinamento do modelo utilizando o HMM-GMM foi interrompido após a obtenção do menor WER com 42000 *leaves* e 40000 *gaussians*. Os parâmetros obtidos foram manualmente estabelecidos e tiveram como referência o trabalho anterior de qualificação do mestrado e [32]. A partir desse modelo, foi realizado o treinamento de outro modelo acústico baseado em uma *Deep Neural Network* (DNN) utilizando 10 épocas. Nessa etapa, foram utilizados os mesmos parâmetros definidos no recipiente do Kaldi para o treinamento do conjunto de dados do Common Voice[5].

O equipamento utilizado para o treinamento desse modelo foi alugado. Contratamos uma *Google Cloud Computer Engine* com 96 CPUs e 210GB RAM durante aproximadamente 90 horas. A configuração do ambiente foi facilitada por meio da criação de um Docker. Dessa forma, o tempo gasto nessa etapa diminuiu consideravelmente, permitindo

[5]https://github.com/kaldi-asr/kaldi/blob/master/egs/commonvoice

um gasto mais focado no processamento do modelo. Disponibilizamos os *scripts* criados nessa etapa de forma pública. Eles podem ser acessados no repositório do github[6].

3.3.2 Wav2Vec

A limitação dos recursos necessários para realizar o processo de *fine-tunning* sobre um modelo inicial do Wav2Vec, nos impulsionou a optar pela escolha de outro modelo. Nesse sentido, adotamos o modelo `wav2vec2-large-xlsr-coraa-portuguese`, disponível publicamente no repositório do HuggingFace[7]. Esse modelo utiliza como base o `Wav2Vec 2.0 XLSR-53`.

Em nosso experimento, tanto o modelo baseado no Wav2Vec quanto o modelo baseado no Kaldi foram analisados a partir de seu uso combinado com o mesmo modelo de linguagem, isto é, o Audimus.Media LM.

Durante a análise, o modelo baseado no Wav2Vec não suportou áudios com duração maior que 30 segundos. Para o teste, utilizamos uma *Google Cloud Computer Engine* com GPU NVIDIA K80. Em áudios mais longos, nossa aplicação apresentava erros relacionados a falta de memória. Nesse sentido, nós adaptamos uma solução proposta pela equipe do HuggingFace. Eles propuseram um tipo de fragmentação do áudio com *strides*. Cada fragmento é composto por uma quantidade pré-definida de *frames* do áudio que serão mapeados para um carácter (chamado *logit*) por meio do algoritmo CTC. O processo de formação da cadeia de caracteres definida a partir do mapeamento dos *logits* é detalhado em [38].

3.3.3 Whisper

Em nosso experimento, utilizamos o modelo `whisper-medium-pt`, disponível publicamente no repositório do HuggingFace[8]. Esse modelo foi ajustado sobre o conjunto de dados do Common Voice versão 11[9] a partir do modelo inicial `whisper-medium`[10]. A escolha desse modelo para o nosso experimento baseou-se em sua ampla utilização pela comunidade e limitação do hardware utilizado. Para o teste, utilizamos uma *Google Cloud Computer Engine* com GPU NVIDIA K80.

[6] https://github.com/diegomarq/docker-kaldi-coraa-pt
[7] https://huggingface.co/Edresson/wav2vec2-large-xlsr-coraa-portuguese
[8] https://huggingface.co/jlondonobo/whisper-medium-pt
[9] https://huggingface.co/datasets/mozilla-foundation/common_voice_11_0
[10] https://huggingface.co/openai/whisper-medium

3.3.4 Comercial

O uso do Audimus.Media requer a compra de uma licença. O uso dessa aplicação foi realizado a partir da licença adquiria pela Sérgio Machado Reis Epp. Em nossa análise, o sistema foi executado em um Windows Server 2019 com processador Intel Core i7 e 8GB RAM. Os arquivos de áudio utilizados foram transcritos a partir de uma pasta pré-definida de entrada. Para cada arquivo de áudio, foi produzido um arquivo de saída no formato `txt` referente a sua transcrição.

A transcrição dos áudios via *Microsoft Azure Speech* e *Google Speech-to-text* ocorre a partir do uso de uma API. Em nosso trabalho, foi desenvolvido um *script* em Python no Google Colab. A configuração definida para o uso do *Google Speech-to-text* foi composta pelo uso da linguagem padrão `pt-BR` e `LINEAR16` como codificação. Esses parâmetros foram submetidos a um objeto definido como `SteamingRecognitionConfig`. A configuração definida para o uso do *Microsoft Azure Speech* também foi composta pelo uso da linguagem padrão `pt-BR`. A configuração dos áudios foi realizada por meio da criação do objeto `PushAudioInputStream`.

3.4 Modelo estocástico

Em nosso experimento, ajustamos um modelo de regressão binomial (disponível publicamente[11]), com o numerador da Equação 2.3 atuando como número de falhas e seu denominador como número de ensaios. Em princípio, o primeiro pode ser maior que o segundo. Entretanto, a probabilidade de tal ocorrência é desprezível no contexto aplicado, de modo que essa decisão não impõe qualquer preocupação prática.

A construção do modelo estocástico é representada na Figura 3.1 e está associada a interpretação de que o WER corresponde aproximadamente à probabilidade de uma palavra ser corretamente transcrita. Nesse modelo, as *features* de regressão foram selecionadas por meio do *Bayesian Information Criteria* (BIC). O procedimento utilizado busca retroativamente, a partir de um conjunto inicial formado por todas as co-variáveis (incluindo o transcritor), interações lineares pareadas e termos polinomiais de segunda ordem (para as variáveis quantitativas). Embora esse seja um modelo de baixa capacidade, com apenas 75 graus de liberdade, a interpretação e inferência dos parâmetros não é direta. Nesse sentido, ainda não seria possível associar de forma direta uma boa*performance* esperada local de um determinado áudio, por exemplo, com um baixo volume.

[11] https://github.com/diegomarq/glm-asr-brtvrad

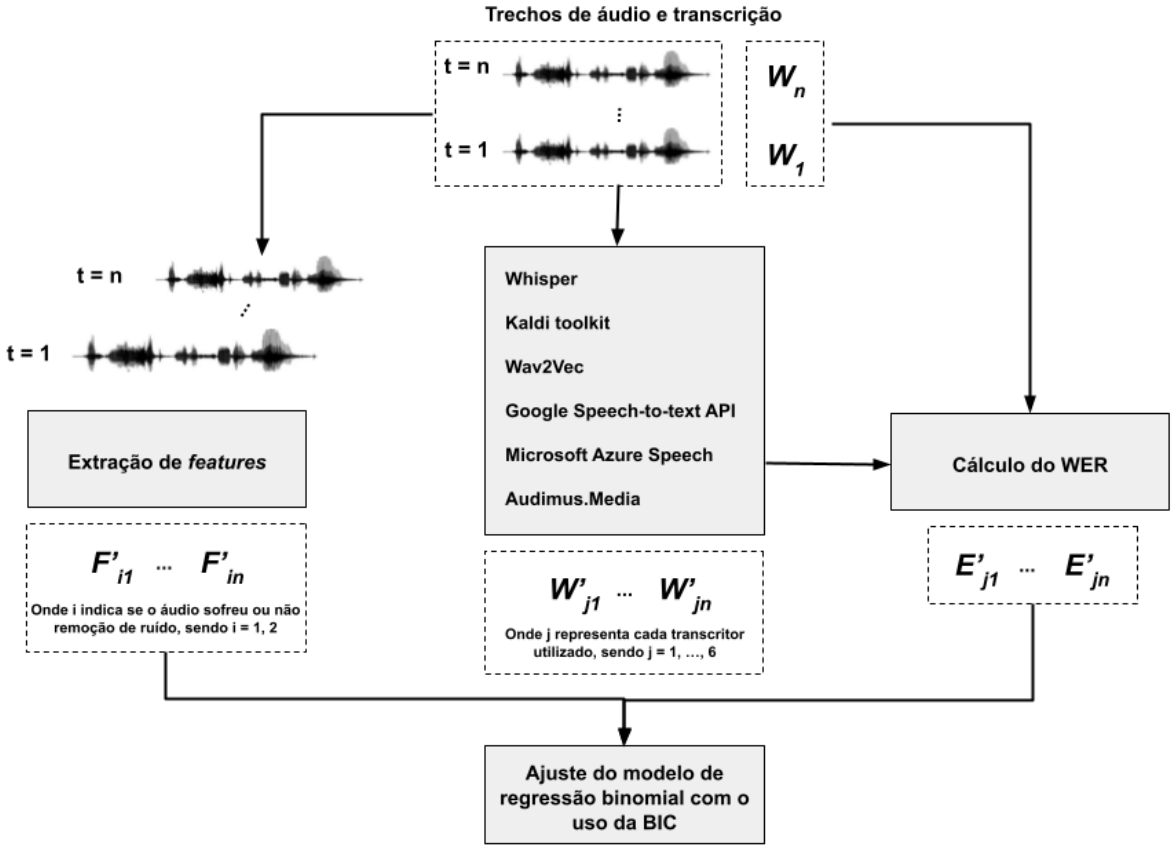


Figura 3.1: Abordagem utilizada na pesquisa para construção do modelo estocástico. Nesse processo, as propriedades de cada áudio t são extraídas. Isso gera um conjunto de vetores F_{it}, cada qual representa o conjunto de *features* extraídas do áudio observado t e i, onde i indica se o áudio sofreu ou não remoção de ruído. Em seguida, cada áudio observado t é transcrito por cada sistema de ASR j utilizado na pesquisa, gerando uma sequência de palavras W'_{jti}. Em seguida, calculamos o WER para cada áudio (t, i) a partir da transcrição gerada W'_{ti} e sua transcrição manual W_t. Ao final, o modelo de regressão é ajustado, sobre o WER, a partir da aplicação do método de seleção de *features Bayesian Information Criteria* (BIC) usando como conjunto inicial de busca as *features* e as suas interações de segunda ordem.

Capítulo 4

Experimentos

Em nosso experimento, iremos utilizar o WER como métrica para mensurar a *performance* de cada modelo. Para compreender melhor a análise que será realizada, dois aspectos cruciais precisam ser observados.

1. Alguns áudios gravados são mais fáceis de transcrever que outros. A *performance* de qualquer método varia em função das características do áudio que será avaliado. O WER definido na Equação 2.3 atua como uma medida de acurácia global, mas não oferece uma descrição detalhada de quando exatamente o sistema utilizado se destaca ou apresenta baixa *performance*.

2. O interesse de nossa pesquisa reside sobre a estimativa esperada do modelo, dado um áudio com determinadas características. Essa expectativa está além da análise realizada com apenas as métricas relativas à *performance* observada do modelo sobre o áudio. Nosso interesse não é simplesmente avaliar o desempenho dos modelos em um conjunto de dados em particular. Sabemos mais sobre o modelo ao estimar sua *performance* esperada quando submetido a um conjunto de áudios com determinado perfil, isto é, um conjunto de características específico.

Para cada áudio presente no conjunto de teste, foram extraídos alguns metadados, tais como *bitrate*, *duração* (em segundos), *volume médio*, *volume máximo* e *taxa* de SNR. Outras informações como *estado de gravação* (São Paulo, Goiás ou outros), *fonte* (TV ou rádio) também foram acrescentadas. Submetemos esses áudios a todos os sistemas de ASR analisados nesse trabalho (Azure, Kaldi, Whisper, Google Speech-to-Text e Audimus). Então, geramos uma *performance local* observada para cada um.

Após a geração das transcrições, acrescentamos ao nosso conjunto de dados mais algumas informações, tais como: número de palavras (definido pela transcrição de referência), número de erros (definido pelo numerador da Equação 2.3), número de acertos (calculado

como a diferença entre o número total de palavras e o número de erros) e WER (obtido pelo método para cada áudio).

Durante o treinamento do modelo binomial, utilizamos como dado de entrada uma tabela gerada a partir das informações observadas relativas às propriedades do áudio e da *performance local* de cada método. Essa tabela possui 5434 observações (disponível publicamente [1]), sendo composta pelos seguintes campos:

1. Código da notícia;
2. Tipo de veículo (TV ou rádio);
3. Data da notícia;
4. Duração do áudio (em segundos);
5. UF (estado do veículo onde o áudio foi gravado);
6. Método (sistema de ASR utilizado);
7. Filtro (áudio com ou sem tratamento de ruído);
8. Volume médio (em decibéis);
9. Volume máximo (em decibéis);
10. *Bitrate* (em kbs);
11. SNR;
12. Transcrição Método (transcrição realizada pelo sistema de ASR observado);
13. Transcrição referência;
14. Número de acertos;
15. Número de palavras;
16. Número de erros.

No modelo binomial escolhido via BIC, utilizamos como co-variáveis as seguintes informações:

1. Método;
2. UF;
3. Bitrate (os valores foram segmentados – maiores que 100, serão "Alto"; menores que 100, serão "Baixo");

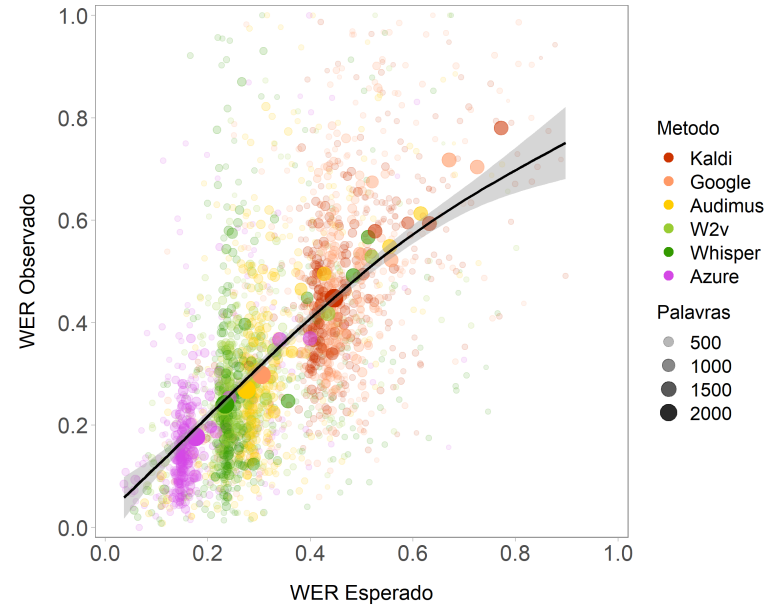
[1] https://github.com/diegomarq/glm-asr-brtvrad

4. Duração do áudio;
5. Volume médio;
6. Volume máximo;
7. SNR.

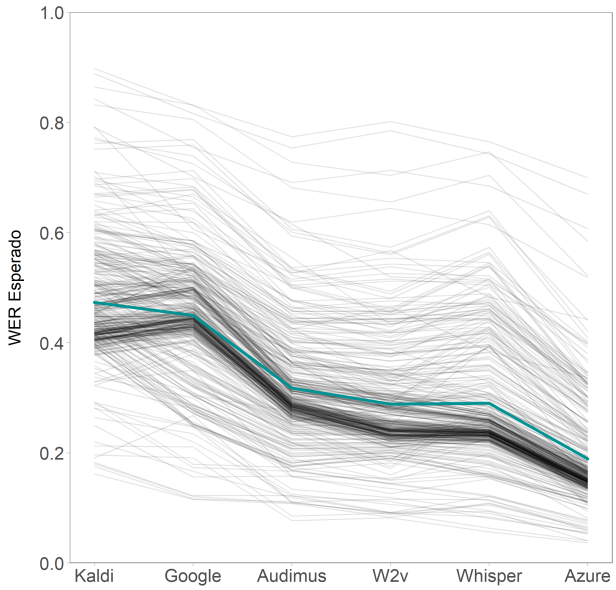
A Figura 4.1a e a Tabela 4.1 ilustram a qualidade do modelo binomial após ser ajustado. O modelo está relativamente bem calibrado, embora não esteja particularmente preciso. Note, na Figura 4.1a, que os áudios com pequena duração, com poucas palavras para serem transcritas, apresentam uma maior volatilidade; os áudios curtos estão mais espaçados em torno da linha ajustada em destaque. É possível destacar uma maior dispersão dos áudios transcritos pelo Whisper, Wav2Vec (W2v), Audimus, Google e Kaldi. As transcrições realizadas pelo Azure possuem uma menor dispersão em termos de *performance* esperada/observada, refletindo sua maior robustez ao avaliar áudios com diferentes características. Ainda assim, observamos uma crescente diferença entre o WER observado e o WER esperado apartir do melhor transcritor global até o pior transcritor global.

A Figura 4.1b indica que o WER esperado é fortemente dependente em relação às propriedades do áudio (utilizadas como co-variáveis). Nessa figura, cada linha corresponde a um arquivo de áudio. A dispersão das linhas em todos os métodos indica que, mesmo no melhor método médio esperado (menor ponto na linha azul), existem determinados áudios que poderão apresentar um resultado ruim. No gráfico, é possível visualizar que para estes áudios, o modelo irá prever um resultado ruim em todos os métodos. Neste caso, é possível afirmar que a *performance* ruim esperada se deve a qualidade ruim do áudio observado. Chegamos a essa conclusão também ao ouvirmos individualmente os áudios com uma maior dispersão. O contrário também é visto. Nosso modelo previu uma boa *performance* para todos os métodos em áudios com uma boa qualidade.

Em nossos experimentos, observamos vários pontos onde há um cruzamento de linhas (Figura 4.1b). Isso indica que existe uma concorrência entre os métodos observados. Para 14.1% das gravações analisadas, o Audimus.Media apresenta *performance* esperada melhor que o modelo baseado no Wav2Vec. Isso ocorre apesar do Wav2Vec alcançar um WER global menor. Também é possível comparar o Kaldi e o Google. Neste caso, o Kaldi apresenta uma *performance* esperada melhor em praticamente metade (52.2%) dos áudios de teste. As implicações para esse caso são relevantes; por exemplo, uma empresa que utilizada o *Google Speech-to-test* poderia definir, a partir do WER esperado, o conjunto de áudios que seria diretamente transcrito pelo Kaldi. Tal implementação poderia ser vista como um *emsemble*, cuja *performance* seria superior aos sistemas de transcrição individuais que o compõe. Nesse cenário, o WER global alcançaria 44%, uma diminuição de 1% com relação ao WER global esperado. Apesar de o número ser pequeno, o impacto



(a)



(b)

Figura 4.1: A Figura 4.1a compara o WER observado e o esperado. Cada ponto representa um arquivo de áudio. A área do ponto é proporcional ao número de palavras presentes em cada transcrição. A cor é determinada pelo sistema de ASR analisado. A Figura 4.1b mostra o WER esperado a partir do método sob consideração. Nesta figura, cada linha corresponde a um arquivo de áudio e a linha em destaque representa a média com relação a todos os áudios.

Sistemas de ASR	WER Médio Esperado	WER Médio Observado
Azure nr	18.9%	17.9%
W2v	28.9%	27.2%
Whisper	29.0%	26.8%
Audimus	31.8%	33.2%
Google nr	45.0%	45.8%
Kaldi nr	47.4%	45.6%

Tabela 4.1: Os valores (WER esperado e observado) apresentados correspondem a média calculada sobre todo o conjunto de teste analisado. Para conveniência do leitor, esses valores são visualmente indicados na Figura 4.1b por meio da linha azul em destaque. O sufixo "nr"significa que o valor indicado foi calculado a partir da aplicação do modelo sobre o conjunto de áudios de teste com tratamento de ruído. Ao visualizar a tabela, é possível verificar que o Azure nr apresenta o menor WER médio esperado e observado, e que a dispersão entre os valores dessas duas naturezas é pequeno, indicando que nosso modelo está bem ajustado.

Metodo A	Metodo B	Audios escolhidos A	Audios escolhidos B	WER Esperado
Audimus	W2v	13.7%	86.3%	28.7%
Kaldi nr	Google nr	52.2%	47.8%	44.0%
Whisper	W2v	61.7%	38.3%	27.8%

Tabela 4.2: Possível uso combinado de dois métodos onde a melhor escolha é definida pelo modelo binomial. Na coluna "WER Esperado", é indicado a *performance* global para o uso combinado dos métodos. Nessa tabela é importante destacar a competição entre o Kaldi nr e o Google nr, e uma melhora no WER esperado ao utilizar esses dois métodos de forma conjunta.

econômico que tal sistema poderia gerar seria maior que sua melhora esperada no WER. O Kaldi seria escolhido em 52.2% dos áudios, o que levaria a empresa, nesse caso hipotético, a usar menos o *Google Speech-to-test*.

Na Tabela 4.2, é mostrado outras possíveis combinações entre os métodos utilizados nesse trabalho. De modo geral, ao considerar todos os sistemas de ASR analisados, o nosso modelo binominal indica, para todos os possíveis tipos de áudios analisados, *Microsoft Azure Speech* como a melhor alternativa.

Ao analisar o desempenho de cada modelo, é possível verificar que o uso de tratamento de ruído, não necessariamente, implica em uma melhora na transcrição. Esse fato é observado no Audmus.Midia, Wav2Vec e Whisper. Para estes modelos, a *performance* esperada foi inferior para áudios com tratamento de ruído. Embora, tanto o Kaldi quanto o Google Speech-to-Text e o *Microsoft Azure Speech* apresentaram desempenho superior.

É necessário destacar que, dentre os métodos analisados, apenas o Wav2vec apresenta uma maior limitação quanto ao ambiente computacional no qual deve ser implantado. Sendo possível utilizá-lo apenas em uma ambiente com GPU. Os outros possuem maior

flexibilidade, podendo optar pela GPU ou CPU. Ao escolhermos pela CPU, foi observado que a velocidade de transcrição foi afetada, se tornando mais lenta do que quando optamos pela GPU, o que era de se esperar.

Capítulo 5

Conclusão

A implantação do modelo estocástico desenvolvido nesse trabalho irá beneficiar tanto do ponto de vista econômico, quanto relativo à qualidade da transcrição, a empresa foco desse estudo. Em nossos experimentos, é possível concluir que o uso dos métodos de transcrição comerciais (como o Audimus.Midia) em conjunto com ferramentas *open-source*, é vantajoso. O uso do Wav2Vec nos parece a melhor opção (exceto, talvez, pela necessidade de usar GPU).

A comparação realizada nesse estudo não é comum na literatura relacionada a sistema de reconhecimento de fala. Esse fato foi destacado na publicação desse trabalho no BRACIS 2022 [39]. Geralmente, estudos nessa área enfatizam uma comparação entre os sistemas de ASR baseado no WER global (todo o conjunto de dados) obtido por cada método analisado.

É possível destacar como contribuição a comunidade científica a publicação do conjunto de teste utilizado nesse trabalho, composto por áudios com uma alta variabilidade em termos de suas propriedades. Além disso, os *scripts* utilizados para o treinamento do kaldi (por meio do Docker), para o treinamento e seleção das co-variaveis, bem como os dados utilizados no treinamento do modelo binomial, estão disponíveis publicamente.

Embora seja observado que a *performance* esperada do *Microsoft Azure Speech* seja superior para todos os áudios analisados, esse sistema de ASR não é capaz de transcrever todos os áudios de forma plenamente acurada. Em alguns casos, ele apresenta erros na transcrição de algumas palavras.

Em alguns conjuntos de áudios como propriedades semelhantes, o Audimus.Media e o modelo baseado no Wav2Vec são competitivos, assim como o sistema baseado no Kaldi e o *Google Speech-to-text*. Nesse contexto, é possível a construção de um transcritor *ensemble*.

Essa analise não pode ser vista como uma comparação definitiva entre os sistemas de ASR analisados. Os sistemas de código-aberto baseados no Kaldi, Wav2Vec e Whisper

podem ser construídos inúmeras vezes a partir do treinamento/ajuste sobre novos conjuntos de dados e a otimização de hiper-parâmetros. Isso inviabiliza uma comparação definitiva entre os modelos, pois é possível que apresentem uma nova *performance*.

É notária a limitação desse trabalho quanto ao cenário em que pode ser utilizado. O uso do modelo estocástico para a escolha do melhor transcritor, dado as características do áudio, inviabiliza sua aplicação em ambientes onde se exige que o tempo de transcrição seja mínimo, como por exemplo *streaming*. Outro ponto a ser considerado na aplicação desse trabalho é a manutenção dos métodos de ASR. É possível que, ao longo do tempo, o modelo deva englobar ou corrigir palavras em seu vocabulário. Caso isso não ocorra, a *performance* definida pelo modelo binomial será prejudicada.

Como trabalho futuro, um modelo estocástico mais sofisticado pode ser construído. Dimensões práticas como preço da licença, custo operacional e de manutenção e agilidade na transcrição podem ser contempladas na comparação dos sistemas.

Referências

[1] Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed e Michael Auli: *wav2vec 2.0: A framework for self-supervised learning of speech representations*. Advances in Neural Information Processing Systems, 33:12449–12460, 2020. viii, 4, 10, 11, 14

[2] Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey e Ilya Sutskever: *Robust speech recognition via large-scale weak supervision*. arXiv preprint arXiv:2212.04356, 2022. ix, 4, 11, 12, 15, 16

[3] Junior, Arnaldo Candido, Edresson Casanova, Anderson Soares, Frederico Santos de Oliveira, Lucas Oliveira, Ricardo Corso Fernandes Junior, Daniel Peixoto Pinto da Silva, Fernando Gorgulho Fayet, Bruno Baldissera Carlotto, Lucas Rafael Stefanel Gris e Sandra Maria Aluísio: *Coraa: a large corpus of spontaneous and prepared speech manually validated for speech recognition in brazilian portuguese*, 2021. 2, 4, 14, 15, 19

[4] Sampaio, Matheus Xavier, Regis Pires Magalhães, Ticiana Linhares Coelho da Silva, Lívia Almada Cruz, Davi Romero de Vasconcelos, José Antônio Fernandes de Macêdo e Marianna Gonçalves Fontenele Ferreira: *Evaluation of automatic speech recognition systems*. Em *Anais do XXXVI Simpósio Brasileiro de Bancos de Dados*, páginas 301–306. SBC, 2021. 2, 3

[5] Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz *et al.*: *The kaldi speech recognition toolkit*. Em *IEEE 2011 workshop on automatic speech recognition and understanding*, número CONF. IEEE Signal Processing Society, 2011. 4, 14

[6] Stefanel Gris, Lucas Rafael, Edresson Casanova, Frederico Santos de Oliveira, Anderson da Silva Soares e Arnaldo Candido Junior: *Brazilian portuguese speech recognition using wav2vec 2.0*. Em *International Conference on Computational Processing of the Portuguese Language*, páginas 333–343. Springer, 2022. 4, 14

[7] Rabiner, Lawrence: *Fundamentals of speech recognition*. Fundamentals of speech recognition, 1993. 4

[8] Yu, Dong e Li Deng: *Automatic Speech Recognition*. Signals and Communication Technology. Springer London, London, 2015, ISBN 978-1-4471-5778-6 978-1-4471-5779-3. `http://link.springer.com/10.1007/978-1-4471-5779-3`, acesso em 2020-04-01. 5

[9] Faraco, Carlos Emílio: *Gramática: fonética e fonologia, morfologia, sintaxe, estilística*. Ed. Atica, 1996. 6

[10] Jurafsky, Dan e James H. Martin: *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, N.J, 2000, ISBN 978-0-13-095069-7. 7, 8

[11] Taylor, Paul Alexander: *Text-to-speech synthesis*. Cambridge University Press, Cambridge, UK ; New York, 2009, ISBN 978-0-521-89927-7. OCLC: ocn221147648. 7

[12] Peddinti, Vijayaditya, Daniel Povey e Sanjeev Khudanpur: *A time delay neural network architecture for efficient modeling of long temporal contexts*. Em *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. 7

[13] Mohri, Mehryar, Fernando Pereira e Michael Riley: *Weighted finite-state transducers in speech recognition*. Computer Speech & Language, 16(1):69–88, janeiro 2002, ISSN 08852308. `https://linkinghub.elsevier.com/retrieve/pii/S0885230801901846`, acesso em 2020-08-03. 8

[14] Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer e Karel Vesely: *The kaldi speech recognition toolkit*. Em *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, dezembro 2011. IEEE Catalog No.: CFP11SRW-USB. 8

[15] Jegou, Herve, Matthijs Douze e Cordelia Schmid: *Product quantization for nearest neighbor search*. IEEE transactions on pattern analysis and machine intelligence, 33(1):117–128, 2010. 10

[16] Devlin, Jacob, Ming Wei Chang, Kenton Lee e Kristina Toutanova: *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805, 2018. 10

[17] Graves, Alex, Santiago Fernández, Faustino Gomez e Jürgen Schmidhuber: *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks*. Em *Proceedings of the 23rd international conference on Machine learning*, páginas 369–376, 2006. 11

[18] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser e Illia Polosukhin: *Attention is all you need*. Advances in neural information processing systems, 30, 2017. 11

[19] Valk, Jörgen e Tanel Alumäe: *Voxlingua107: a dataset for spoken language recognition*. Em *2021 IEEE Spoken Language Technology Workshop (SLT)*, páginas 652–658. IEEE, 2021. 12

[20] Chan, William, Navdeep Jaitly, Quoc Le e Oriol Vinyals: *Listen, attend and spell: A neural network for large vocabulary conversational speech recognition*. Em *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, páginas 4960–4964. IEEE, 2016. 12

[21] Chiu, Chung Cheng, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina *et al.*: *State-of-the-art speech recognition with sequence-to-sequence models*. Em *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 4774–4778. IEEE, 2018. 12

[22] Xiong, Wayne, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang e Andreas Stolcke: *The microsoft 2017 conversational speech recognition system*. Em *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, páginas 5934–5938. IEEE, 2018. 12

[23] Xiong, Wayne, Jasha Droppo, Xuedong Huang, Frank Seide, Michael L Seltzer, Andreas Stolcke, Dong Yu e Geoffrey Zweig: *Toward human parity in conversational speech recognition*. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25(12):2410–2423, 2017. 12

[24] Meinedo, Hugo, Diamantino Caseiro, Joao Neto e Isabel Trancoso: *Audimus. media: a broadcast news speech recognition system for the european portuguese language*. Em *International Workshop on Computational Processing of the Portuguese Language*, páginas 9–17. Springer, 2003. 13

[25] Meinedo, Hugo, Alberto Abad, Thomas Pellegrini, I Trancoso e J Neto: *The l2f broadcast news speech recognition system*. Proc. Fala, páginas 93–96, 2010. 13

[26] Meinedo, Hugo, Nuno Souto e João P Neto: *Speech recognition of broadcast news for the european portuguese language*. Em *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU'01.*, páginas 319–322. IEEE, 2001. 13

[27] Neto, J, Hugo Meinedo e Márcio Viveiros: *A media monitoring solution*. Em *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 1813–1816. IEEE, 2011. 13

[28] Yu, Dong e Li Deng: *Automatic speech recognition*, volume 1. Springer, 2016. 14

[29] Pratap, Vineel, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve e Ronan Collobert: *MLS: A large-scale multilingual dataset for speech research*. Em *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, páginas 2757–2761. ISCA, 2020. 14

[30] Alencar, VFS e Abraham Alcaim: *Lsf and lpc-derived features for large vocabulary distributed continuous speech recognition in brazilian portuguese*. Em *2008 42nd Asilomar conference on signals, systems and computers*, páginas 1237–1241. IEEE, 2008. 14

[31] Salesky, Elizabeth, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W. Oard e Matt Post: *The multilingual tedx corpus for speech recognition and translation*. Em *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, páginas 3655–3659. ISCA, 2021. 14

[32] Batista, Cassio T, Ana Larissa Dias e Nelson C Sampaio Neto: *Baseline acoustic models for brazilian portuguese using kaldi tools.* Em *IberSPEECH*, páginas 77–81, 2018. 14, 19

[33] Quintanilha, Igor Macedo, Sergio Lima Netto e Luiz Wagner Pereira Biscainho: *An open-source end-to-end asr system for brazilian portuguese using dnns built from newly assembled corpora.* Journal of Communication and Information Systems, 35(1):230–242, 2020. 14

[34] Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen *et al.*: *Deep speech 2: End-to-end speech recognition in english and mandarin.* Em *International conference on machine learning*, páginas 173–182. PMLR, 2016. 14

[35] Schramm, M, LF Freitas, A Zanuz e D Barone: *Cslu: Spoltech brazilian portuguese version 1.0 ldc2006s16*, 2006. 14

[36] Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer e Veselin Stoyanov: *Unsupervised cross-lingual representation learning at scale.* Em *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, páginas 8440–8451, Online, julho 2020. Association for Computational Linguistics. https://aclanthology.org/2020.acl-main.747. 14

[37] Jose, Botero: *Whisper medium portuguese brpt*, 2022. https://huggingface.co/jlondonobo/whisper-medium-pt. 15

[38] Patry, N: *Making automatic speech recognition work on large files with wav2vec2 in transformers*, 2022. https://huggingface.co/blog/asr-chunking. 20

[39] Azevedo, Diego Marques de, Guilherme Souza Rodrigues e Marcelo Ladeira: *A probabilistically-oriented analysis of the performance of asr systems for brazilian radios and tvs.* Em *Intelligent Systems: 11th Brazilian Conference, BRACIS 2022, Campinas, Brazil, November 28–December 1, 2022, Proceedings, Part II*, páginas 169–180. Springer, 2022. 29