Capítulo 5

Use of Automatic Speech Recognition Systems for Multimedia Applications

Marcos Valadão Gualberto Ferreira[1] and Jairo Francisco de Souza[1]

[1] Universidade Federal de Juiz de Fora. Departamento de Ciência da Computação (DCC/UFJF)

marcos.valadao@ice.ufjf.br, jairo.souza@ufjf.edu.br

Abstract

The need to retrieve information in multimedia content increases the demand for systems that use automatic speech recognition. A speech recognition system enables the computer to interpret audio signals, generating approximate textual transcriptions. These systems are based on probabilistic models that create a robust and correct model for human speech. In this paper we present a speech recognition systems architecture and we detail its basic components: the acoustic model, language model, lexical and decoder. The training process of acoustic and language models is also presented. Finally, we discuss how these systems can be used in some applications.

Resumo

A necessidade de recuperar informações em conteúdo multimídia aumenta a demanda por sistemas que usam o reconhecimento automático de fala. O sistema de reconhecimento de voz permite que o computador interprete sinais de áudio, gerando transcrições textuais aproximadas. Esses sistemas são baseados em modelos probabilísticos que criam um modelo robusto e correto para a fala humana. Neste artigo, é apresentada uma arquitetura de sistemas de reconhecimento de fala e uma descrição de seus componentes básicos: modelo acústico, modelo de linguagem, lexical e decodificador. O processo de treinamento de modelos acústicos e de linguagem também é apresentado. Finalmente, é apresentado como esses sistemas podem ser usados em algumas aplicações.

5.1. Introdução

Um sistema de Processamento de Linguagem Natural (PLN) abrange aspectos da comunicação humana como a fala, palavras, textos e sentenças considerando todo o contexto em que se encontra e se baseando em estruturas de linguagem [Gonzalez and Vera, 2003]. Estes sistemas são utilizados, de maneira geral, para possibilitar que o computador entenda e se comunique em linguagem humana de forma automática. O PLN é baseado em modelagens realistas que sempre necessitam de melhorias por parte das estruturas de comportamento de linguagem.

Existem diversas aplicações na área de PLN. Entre elas está a técnica de reconhecimento automático de fala ASR (do inglês *automatic speech recognition*), que será apresentado neste capítulo.

Tecnologias de ASR permitem que computadores interpretem a fala humana a partir da síntese de sinais de áudio, isto é, transcrever áudio em textos de linguagem natural [Coelho and Souza, 2014]. Este processo geralmente apresenta grandes dificuldades, pois possui limitações quanto ao reconhecimento por questões técnicas, tais como: tamanho do vocabulário, reconhecimento de fala contínua e quanto a estrutura complexa da voz humana, que depende de fatores como: sotaque, entonação, velocidade da voz, estado emocional etc [Gonzalez and Vera, 2003]. Na última década, o interesse nesta área aumentou, principalmente quando técnicas de *Deep Learning* começaram a ser utilizadas com sucesso em ASR [Dahl et al, 2012]. Desde então, diversos sistemas comerciais (i.e. Microsoft[1], IBM[2] e Google[3]) começaram a se popularizar.

Sistemas para reconhecimento de fala são construídos, geralmente, a partir de decodificadores e de modelagem acústica e de linguagem. Os decodificadores desempenham o papel de interpretação do sinal de áudio e de identificação, dentro da modelagem, por algum termo de equivalência. Já os modelos acústicos e de linguagem são responsáveis pelo mapeamento de toda a estrutura linguística [Cuadros, 2007]. É praticamente impossível que um sistema seja perfeito a ponto de abranger as diversas formas que a voz humana pode tomar e de como interpretá-la. Por isto, os sistemas de reconhecimento de voz conhecidos não possuem acurácia total. Porém, os avanços alcançados nos últimos anos tem permitido que essa tecnologia venha a ser utilizada com sucesso em diversos tipos de aplicação.

A questão central envolvendo um sistema para reconhecimento de fala é possuir um modelo acústico e de linguagem que seja satisfatório [Coelho and Souza, 2014]. Estes modelos são treinados utilizando uma base de dados pré-processada e técnicas de descrição probabilística para os termos da linguagem [Neto et al., 2005].

Este capítulo apresenta um resumo dos principais componentes de um sistema de reconhecimento de fala, entre eles modelo acústico, modelo de linguagem, léxico e

[1] https://docs.microsoft.com/pt-br/azure/cognitive-services/Speech/API-Reference-REST/BingVoiceRecognition

[2] https://www.ibm.com/watson/developercloud/speech-to-text.html

[3] https://cloud.google.com/speech/

decodificador. Também serão apresentadas as principais técnicas para treinamento de modelos, ferramentas de apoio e algumas aplicações destes sistemas.

5.2. Arquitetura de sistemas para reconhecimento de fala

Os sistemas ASR atuais são baseados no reconhecimento estatístico de padrões. Por trás desses sistemas podemos identificar minimamente quatro grandes componentes: **modelo acústico**, que mapeia o sinal original que está sendo processado em palavras e sentenças; o **modelo de linguagem**, que é o responsável por caracterizar a língua e condicionar a combinação de palavras, descartando frases agramaticais; o **léxico**, que representa as palavras do dicionário utilizado, com suas respectivas transcrições fonéticas; e o **decodificador**, que procura a melhor sequência de palavras num conjunto de hipóteses possíveis. A Figura 5.1. representa a arquitetura genérica de um sistema de reconhecimento de fala.

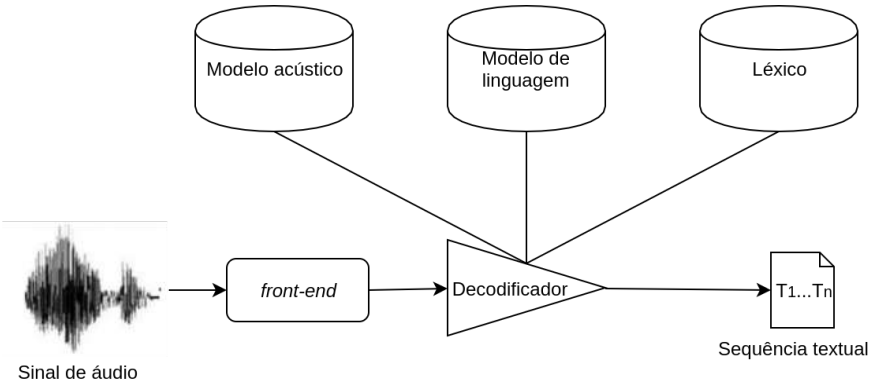

Figura 5.1. Arquitetura de sistemas ASR

Em alguns casos, devido às diferenças de arquiteturas, não se observa uma diferença tão rígida entre um ou mais desses componentes. Contudo, via de regra, as funcionalidades de cada um deles está sempre presente em um sistema de reconhecimento de fala. Nesta seção, vamos discuti-los separadamente.

5.2.1. Modelos Acústicos

Os modelos acústicos geralmente são HMM (*Hidden Markov Model*) [Rabiner and Juang, 1986] de fonemas com 3 estados. Podem ser modelos sem contexto, monofones, ou com contexto: difones (contexto à direita ou à esquerda), trifones (contexto à direita e à esquerda) ou n-fones. No *Hidden Markov Model*, o objetivo é encontrar parâmetros ocultos a partir de parâmetros observados.

Um HMM é definido pelos seguintes parâmetros: número de estados, matriz de probabilidade de função entre estados e, para cada estado, uma função de densidade de probabilidade, que tem como objetivo a caracterização acústica deste estado [da Veiga, 2013]. Por se tratar de um sistema com amplo vocabulário, as palavras modeladas são decompostas em suas respectivas transcrições fonéticas.

Cada fonema é representado por uma HMM que contém três estados emissores e uma topologia simples do tipo esquerda-direita que é comumente utilizada em sistemas ASR

[da Veiga, 2013]. Estados de entrada e saída, não emissores, são acrescentados para auxiliar na junção de modelos. Assim, o estado de saída de um fonema pode se juntar ao estado de entrada de outro fonema criando um HMM composto, como apresentado na Figura 5.2. Assim, é possível que os modelos se juntem formando palavras e estas sejam unificadas formando frases.

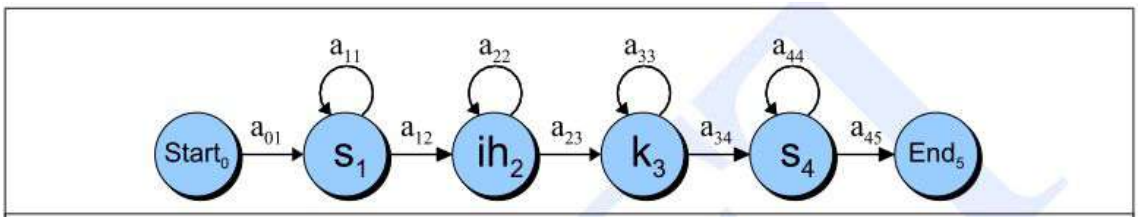

Figura 5.2. Junção de fonemas para representação da palavra "six" ("s ik k s") [Jurafsky, 2009]

O objetivo da modelagem acústica é definir quais as sequências de estados e, com isto, a sequência de fonemas, que possuem uma melhor verossimilhança a um vetor de características acústicas que está sendo processado, definindo assim uma série de probabilidades de palavras e sequências de palavras para o segmento.

Os modelos acústicos geralmente utilizam o HMM para lidar com a variabilidade temporal da fala e os Modelos de Misturas de Gaussianas (GMM - *Gaussian Mixture Model*) na representação da relação entre os estados do HMM e a entrada acústica. Uma abordagem alternativa de representação da relação é o uso de uma rede neural *feed-forward,* a qual recebe vários quadros de coeficientes de entrada e produz probabilidade sobre os estados HMM como saída. Esses modelos baseados em HMM e redes neurais profundas (DNN - *Deep Neural Networks*), mostraram superar o GMM em diversos *benchmarks* para reconhecimento automático de fala [Hinton et al., 2012].

Apesar de todas a vantagens que os GMM's trouxeram para o reconhecimento de fala, eles apresentam uma grave deficiência: são estatisticamente ineficientes para modelar dados que se situam sobre ou próximo de um coletor não-linear no espaço de dados. Isso implica diretamente no entendimento da estrutura que a fala produz. Com isso, acredita-se que outros tipos de modelos, como o de redes neurais, podem possuir resultados melhores para a modelagem acústico se for possível explorar de forma mais efetiva informações em uma grande janela de quadros da estrada acústica. As redes neurais têm a capacidade de extração de informação de dados sobre, ou próximo de um, coletor não-linear muito melhor que o GMM.

A técnica para treinamento de modelos acústicos usando redes neurais faz uso dos novos algoritmos de aprendizagem de máquinas e a capacidade de hardware para a formação de redes neurais profundas que contém muitas camadas de unidades não-lineares escondidas e uma camada de saída grande. Esta camada de saída grande é necessária para acomodar o grande número de estados do HMM que surgem quando cada fonema é modelado a partir das ligações dos modelos HMM's de fonemas que surgem no decorrer do treinamento.

O ajuste dos DNN's no treinamento é realizado em duas etapas. Na primeira etapa são inicializados os detectores de características, uma camada de cada vez, criando uma pilha de modelos, cada um dos quais tem uma camada de variáveis latentes. Na segunda fase, cada modelo da pilha é usado para inicializar uma camada de unidades ocultas em um DNN e então a rede é aperfeiçoada para prever os estados alvos. Estes alvos são obtidos utilizando um sistema-base HMM-GMM para o alinhamento forçado. Esta técnica é compartilhada por grupos de pesquisa da Universidade de Toronto, Microsoft Research, Google and IBM Research [Hinton et al., 2012].

Em linhas gerais, o modelo de treinamento com DNN funciona da seguinte forma:

- Uma janela com alguns milissegundos do sinal de fala é enviada para uma DNN.
- A DNN calcula a probabilidade daquela janela ser um determinado fonema e envia essa informação para o HMM.
- O HMM então "funde" essa informação com aquela que recebeu de outras janelas do sinal de fala para retornar qual o texto mais provável para aquele sinal de fala.

Esse modo de funcionamento está ilustrado na Figura 5.3. O sinal de áudio, indicado na figura com o rótulo *Observation*, está representado pelo seu espectro. Trechos desse espectro são enviados, um de cada vez, para a DNN, que os processa através das M camadas h. A saída da rede é então utilizada como probabilidade de observação (*Observation Probabilities*) pelo HMM. Ao final, o HMM produzirá qual a sequência de fonemas mais provável para aquela dada amostra de fala. Note nessa figura a presença dos pesos W da rede e as probabilidades de transição (*Transition Probabilities*) do HMM. Estes parâmetros precisam ser estimados no treinamento do modelo acústico.

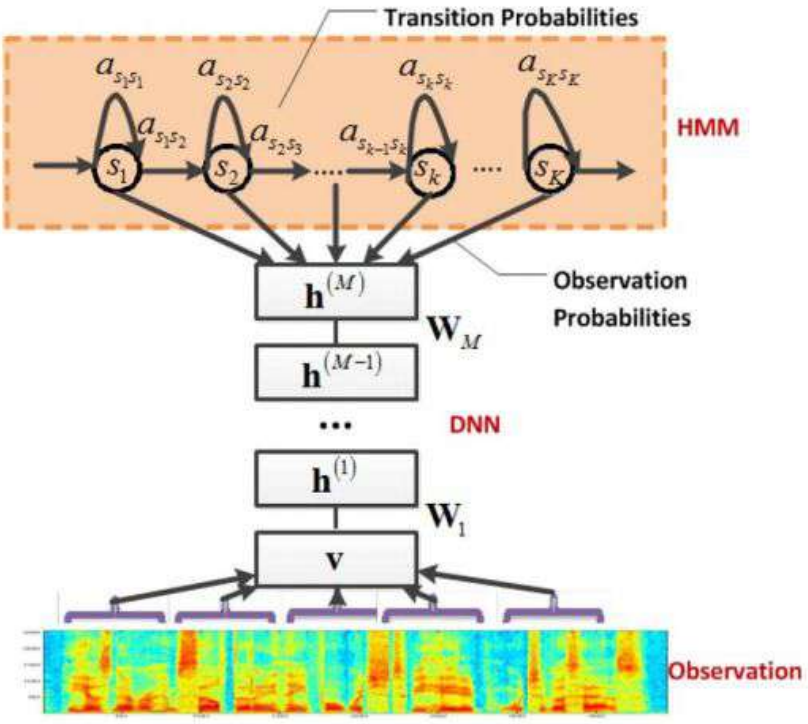

Figura 5.3. DNN aplicada para reconhecimento de fala [Dahl et al., 2012].

5.2.2. Modelos de Linguagem

O modelo de linguagem (LM - *Language Model*) não é necessário para o pleno funcionamento de sistemas ASR. Porém, utilizar somente informações acústicas não é suficiente para o desempenho satisfatório destes sistemas, uma vez que uma palavra poderia ser seguida por qualquer outra sem restrição [da Silva, 2010]. Assim, é consenso na área a utilização de modelos de linguagem para sistemas que lidam com amplo vocabulário, trazendo vantagens para o processo, como a redução do espaço de busca, redução do tempo de reconhecimento e a resolução de ambiguidades acústicas [de Sá Pessoa et al., 1999; da Silva, 2010].

O propósito do LM é prover a probabilidade de ocorrência de uma palavra $P(w_k)$ dada uma sequência de palavras que a antecedem, como é descrito na fórmula abaixo:

$$P(w_k) = P(w_k)P(w_k|w_{k-1})P(w_k|w_{k-1}w_{k-2})...P(w_k|w_{k-1}w_{k-2}...w_1)$$

Onde w_k, é uma palavra da sequência de palavras. Assim, a probabilidade de uma sequência de palavras w_k^1 ocorrer é dado por:

$$P(w_k^1) = P(w_1)\prod_{i=2}^{k} P(w_i|w_1...w_{i-1})$$

Pode-se observar que o cálculo destas probabilidades para grandes vocabulários é algo inviável, pois a probabilidade de ocorrência será calculada em relação a todas as palavras da frase. Para resolver este problema é utilizado n-gramas, onde se assume a probabilidade de uma dada palavra somente até n-1 palavras que a antecedem. Assim, o cálculo de $P(W_k^1)$ é dado por:

$$P(W_k^1) = P(w_1) \prod_{i=2}^{k} P(w_i|w_{i-n+1})$$

Os *n*-gramas são bastante eficientes em línguas onde a ordem das palavras é importante, pois o modelo de linguagem abrange características de sintaxe e semântica e evita a necessidade da criação de regras e de uma gramática formal. Em sistemas ASR é comumente utilizado bigramas (n = 2) e trigramas (n = 3), pois a maioria das palavras possuem forte dependências das duas que a antecedem.

Para avaliação de um modelo de linguagem, utiliza-se a medida de perplexidade. A perplexidade representa o quão indeciso o modelo pode ficar dada uma sequência textual. Quanto menor o valor da perplexidade, melhor é o modelo [Young et al., 2006]. Esta medida é obtida utilizando uma base de teste contendo frases, onde cada frase é avaliada segundo o modelo de linguagem construído. Com isso, um bom modelo de linguagem deve apresentar uma perplexidade alta para uma frase sintaticamente incorreta e uma perplexidade baixa para uma frase sintaticamente correta.

Como o modelo de linguagem é baseado em probabilidades geradas a partir da frequência de palavras, existem diversos métodos para amortizar os efeitos destes números. Isto é feito, geralmente, para diminuir a diferença de probabilidades entre palavras muito frequentes e pouco frequentes, como é demonstrado na Figura 5.4.

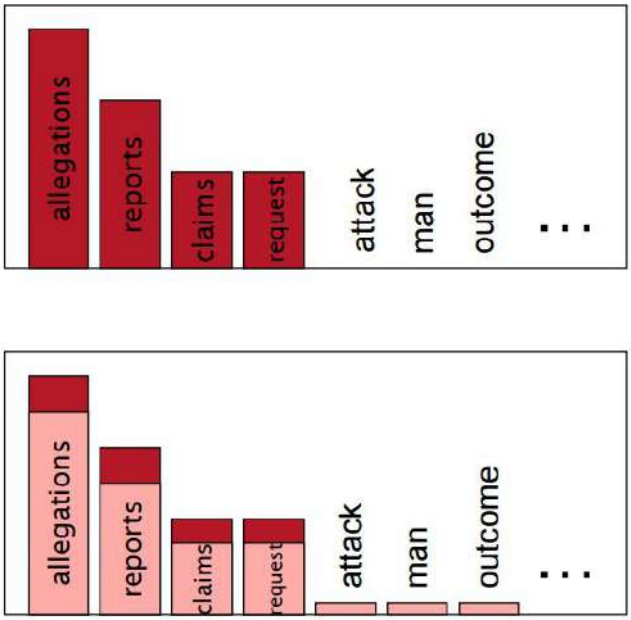

Figura 5.4. Efeito de descontos em modelos de linguagem. No primeiro gráfico é ilustrado as probabilidades sem o uso de métodos de desconto e no segundo com o uso de desconto [Jurafsky, 2017].

Na imagem é exemplificado os efeitos dos métodos de descontos no modelo de linguagem. A diferença entre as palavras com maior probabilidade (*allegations, reports claims* e *request*) e as com menor probabilidade (*attack, man* e *outcome*) diminui com a aplicação do desconto. Com isso, o modelo se torna mais justo para palavras com baixa frequência aumentando a chance de ocorrência das mesmas. Os métodos de descontos mais conhecidos são Add Estimation, Laplace (Add-one), Good-Turing [Katz, 1987], Kneser-Ney [Ney and Essen, 1991] e Witten-Bell [Witten and Bell, 1991].

Esta amortização do modelo de linguagem induz a diferenças no resultado da avaliação do modelo. Os métodos de desconto tanto podem aumentar a perplexidade (piorar o modelo) quanto diminuir a perplexidade (melhorar o modelo) dependendo do contexto de aplicação. Porém, uma prática que geralmente melhora os resultados do modelo de linguagem (diminui a perplexidade) é o aumento da base de dados para treino. Isto é avaliado em [Silva et al., 2004] onde é discutido, dentre outros aspectos, a influência no aumenta da base de texto para treinamento de modelos de linguagem para sistemas de reconhecimento de fala para grandes vocabulários, como é apresentado na Figura 5.5.

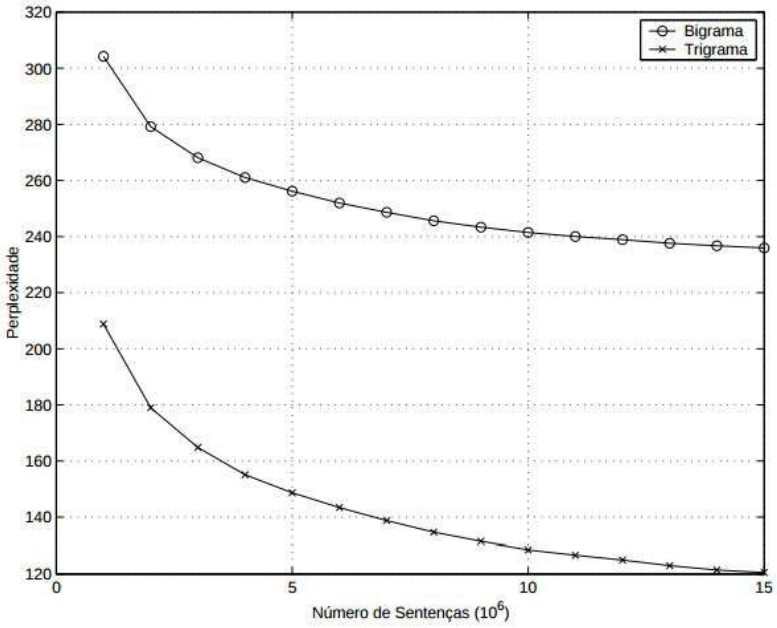

Figura 5.5. Evolução da perplexidade com aumento dos dado para treino, para bigramas e trigramas [Silva et al., 2004]

Pode-se observar a queda significativa que a perplexidade teve a medida que o número de sentenças aumentou. Isto influencia diretamente no sistema de reconhecimento de fala, pois a medida que o modelo de linguagem produz melhores probabilidades, os resultados serão melhores.

5.2.3. Léxico

O léxico é composto por um sequência de palavras e suas respectivas transcrições fonéticas. Esta técnica também é conhecida como conversão grafema fonema, ou seja, conversão de símbolos gráficos utilizados na construção de palavras da língua em unidades sonoras usadas para formar e distinguir palavras. A Tabela 5.1 mostra um exemplo de 5 palavras que compõem um léxico em um sistema de reconhecimento automático de fala.

Tabela 5.1. Componente léxico de um sistema ASR

Palavra	Transcrição fonética
catou	k a t o w
cisne	s i z n y
elevador	e l e v a d o r
maçãs	m a s ɐ̃ s
polímeros	p o l i m e ɾ ʊ s

Este componente é um importante pré-requisito para a construção de sistemas ASR. Porém, a conversão de uma sequência de caracteres em sequência de fones não é uma tarefa trivial. Sabe-se que a transcrição fonética gerada automaticamente apresenta algumas limitações. Uma dessas fraquezas é a dificuldade em tratar as combinações formadas pelo final de uma palavra com início de uma outra. Por exemplo, a palavra

"dois" termina com um "s" e soa como "ss" sempre, exceto quando a palavra falada após o "dois" começar com vogal. Nesse caso, o último fonema de dois se transforma em "z". Há diversos trabalhos que lidam com esses e outros problemas relacionados à transcrição fonética automática, como descrito apresentado em [Goel et. al.,2009].

Existem ainda trabalhos relacionados com a criação do dicionário fonético a as dificuldades da tarefa de conversão de sequência de caracteres em sequência de fones como em [Siravenha et al., 2008], onde são apresentadas as duas técnicas de destaques na literatura para a conversão grafema fonema: *data-driven* e baseadas em regras. O trabalho apresenta, primeiramente, a importância da elaboração de um dicionário fonético de grande vocabulário na construção de um sistema de reconhecimento automático de fala e como este componente, quando bem construído, impacta positivamente na acurácia dos modelos. Com isso, é proposto a contribuição de um algoritmo baseado em regras para a conversão grafema fonema. Os conversores baseados em regras apresentam a vantagem de não fazer uso de alinhamento lexical, visto que não há necessidade de treinamento para gerar as próprias regras. A proposta do trabalho é fornecer ao algoritmo regras fonológicas pré-estabelecidas de acordo com a língua no qual o sistema é baseado. Por fim são apresentados os desafios e resultados da abordagem proposta, bem como os ganhos atingidos em comparação com as abordagens tradicionais. Os melhores resultados foram obtidos em sistemas que utilizaram um dicionário baseado em regras, apresentando a menor taxa de WER (*Word Error Rate*).

Uma ferramenta interessante na construção do léxico é o Multilingual G2P[4], que é uma ferramenta de código aberto e que disponibiliza a criação de um léxico de forma simplificada. O arquivo de entrada para a criação do Léxico contém uma lista de palavras que representam o vocabulário de abrangência do sistema de reconhecimento de fala. Na seção 5.5.2., onde é apresentado o SRILM, é mostrado uma ferramenta simples para geração de vocabulário a partir de uma base de dados contendo frases.

O Multilingual G2P é baseado na ferramenta eSpeak[5] e, para utilização, é necessário a instalação do eSpeak. Em uma máquina Linux (Ex: Ubuntu 16.04), utilize o comando abaixo:

$ sudo apt-get install espeak

Após a instalação da dependência, utilize o arquivo executável, disponibilizado no GitHub do Multilingual G2P, usando o comando abaixo:

$ wget https://raw.githubusercontent.com/jcsilva/multilingual-g2p/master/g2p.sh

Após o término do download, utilize o comando abaixo para gerar o Léxico.

$ ./g2p.sh -w [vocabulario] > lexico.txt

[4] https://github.com/jcsilva/multilingual-g2p
[5] http://espeak.sourceforge.net/

O Multilingual G2P permite criar léxico para diversos idiomas. O idioma padrão é o Português do Brasil, porém existem outros idiomas que devem ser indicados como parâmetro na linha de execução acima. Um exemplo de execução é demonstrado abaixo para criação de um léxico em Inglês (en).

$ ./g2p.sh -w [vocab] -l en

Outros idiomas podem ser encontrados na página do Multilingual G2P no GitHub.

5.2.4. Extração de Parâmetros (*front-end*)

A extração de parâmetros é um processo recorrente em todos os sistemas que envolvem reconhecimento de padrões. Esse processo tem como principal objetivo extrair somente as informações do áudio que serão utilizadas no reconhecimento, de forma que seja um processo objetivo e robusto [da Veiga, 2013]. Assim, a função prioritária deste componente é dividir o sinal que está sendo processado em blocos e de cada um destes blocos derivar uma estimativa suavizada do espectro [Tevah, 2006].

Em sistemas ASR, os parâmetros mais utilizados são coeficientes *mel-cepstrais* (MFCC – *Mel-Frequency Cepstral Coefficients*) [Davis and Mermelstein, 1980]. Algumas outras abordagens de parâmetros conhecidas são: coeficientes cepstrais de predição linear (LPCC – *Linear Predictive Cepstral Coefficients*) e coeficientes dinâmicos (parâmetros delta e delta-delta) que também podem ser usados em conjunto com as demais abordagens.

Um exemplo de aplicação da extração de parâmetros em sistemas ASR é compensar o efeito de longa duração de espectros, causado principalmente pela forma acústica como o áudio foi construído (equipamentos, canais de áudio, distância entre o locutor e o microfone etc). Neste caso é utilizada uma técnica de atualização da média espectral para cada segmento de forma a proteger o sistema contra variações do canal [Tevah, 2006].

5.2.5. Decodificador

O decodificador é o componente dos sistemas ASR responsável por manipular todos os outros componentes descritos anteriormente e gerar a melhor sequência textual aproximada para um dado segmento de áudio de entrada. O decodificador terá a disposição todas as informações de dados observados que foram usados na construção do modelo e, desta forma, poderá decidir qual sequência textual mais adequada para determinada entra acústica.

A Figura 5.6. representa um sistema de reconhecimento de dígitos e ilustra a rede de probabilidades que o decodificador possui a partir dos estados HMM e do Léxico.

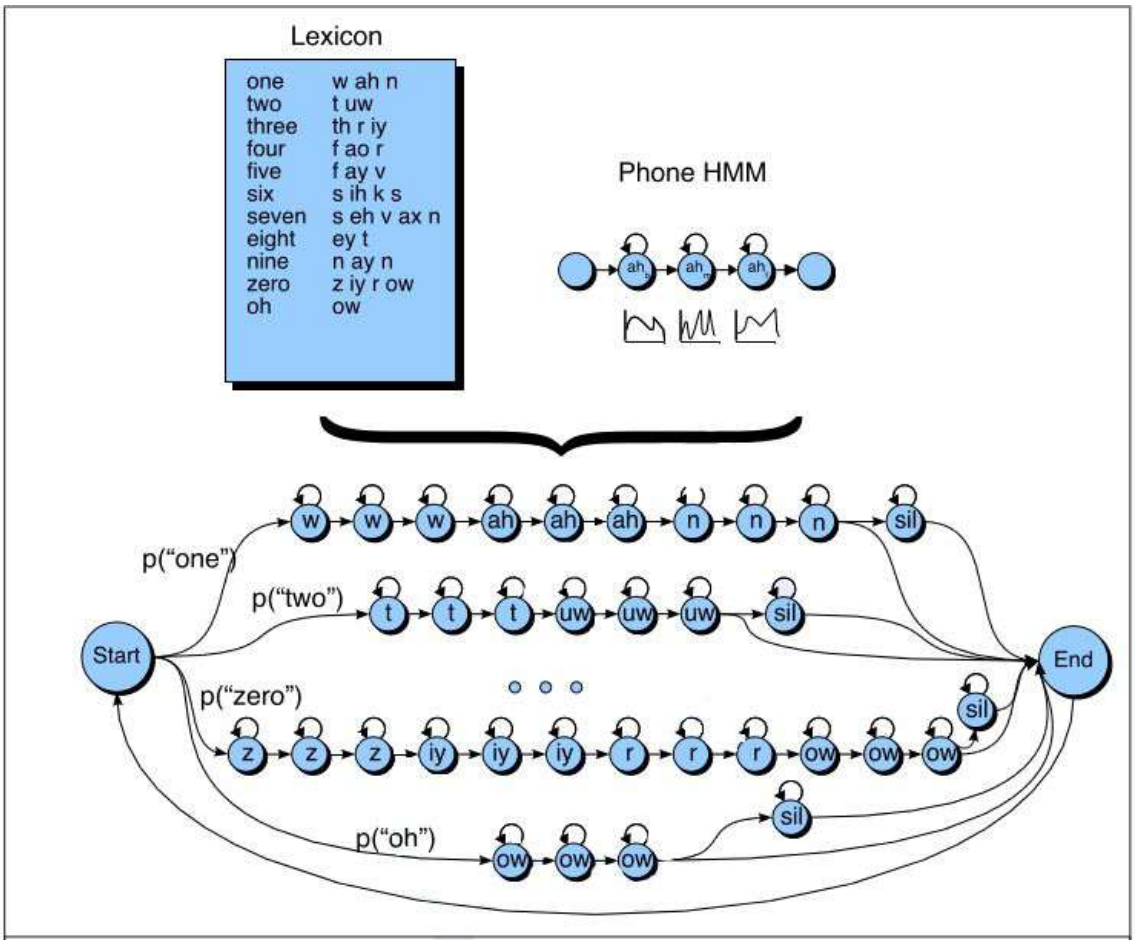

Figura 5.6. Rede de probabilidade de palavras para um sistema de reconhecimento de dígitos. As palavras são representadas por fonemas descritos no léxico [Jurafsky, 2009]

A tarefa do decodificador é obter a melhor sequência de palavras que descreve uma dada sequência de características acústicas (gerada pelo *front-end* a partir de um dado áudio). O processo de decodificação é construído por um rede de palavras geradas pela modelagem do sistema. Esta rede é definida por um conjunto de nós (palavras) conectados por arcos, onde cada arco possui uma probabilidade de ocorrência (transição) [da Silva, 2010]. As palavras são representadas por sequências de modelos acústicos e a escolha das palavras obedece ao modelo de linguagem que define um universo de hipóteses para as sequências de palavras [da Veiga, 2013]. Após encontrar o melhor caminho na rede (aquele cuja probabilidade é mais alta), o decodificador transcreve a sequência de fonemas encontrados em suas respectivas representações grafemáticas no léxico e por fim gera a sequência textual, em linguagem natural, e finaliza o sistema. Com todas as informações do modelo acústico, modelo de linguagem e léxico, forma-se um grafo, com ilustrado na Figura 5.7. Cabe ao algoritmo implementado no decodificador percorrer esse grafo em busca do melhor caminho para um dado sinal de fala.

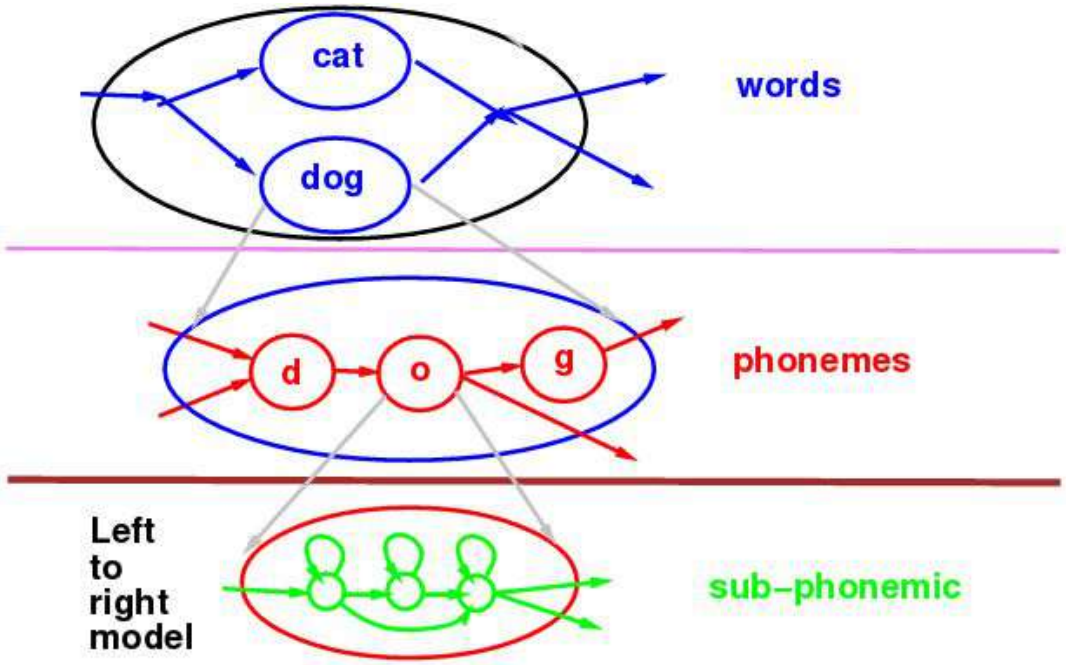

Figura 5.7. O decoder executa um algoritmo de busca em um grafo que agrega conhecimentos do modelo acústico, do modelo de língua e do léxico [Bengio, 1999].

5.3. Avaliação de sistemas de reconhecimento de fala

Os sistemas de reconhecimento de fala são comumente avaliados segundo a taxa de erro de palavras (WER - *Word Error Rate*). Esta métrica representa o quanto a transcrição gerada pelo sistema de reconhecimento de fala (hipótese) difere da transcrição original (referência) [Jurafsky, 2009].

A WER é calculada a partir da definição de quais palavras da hipótese estão corretas, quais foram inseridas incorretamente, quais foram excluídas e quais foram substituídas em comparação com a referência. A WER é calculada pela seguinte fórmula:

$$WER = 100 \times \frac{Inserções + Substituições + Deleções}{Total\ de\ Palavras\ na\ Referência}$$

Outra métrica, menos utilizada, mas também importante é a Taxa de Erro de Frases (SER - *Sentence Error Rate)*, que representa quantas frases possuem pelo menos um erro.

$$SER = 100 \times \frac{Qtd\ de\ sentenças\ com\ pelo\ menos\ um\ erro}{Qtd\ total\ de\ sentenças}$$

Existem ferramentas próprias para o cálculo da WER, como por exemplo a ferramenta Sclite[6], que é um programa para avaliar a saída de sistemas de reconhecimento de fala. Este programa recebe como entrada dois arquivos, um contendo a hipótese e o outro a

[6] http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm#sclite_name_0

referência. A Figura 5.8 apresenta um exemplo de avaliação através da ferramenta Sclite

```
id: (bit_bytes-1215)
Scores: (#C #S #D #I) 8 3 2 0
REF: conversão de caracter para BITS ESSA  primeira parte QUE é UMA parte padrão
HYP: conversão de caracter para BIPS NESSA primeira parte *** ** A   parte padrão
Eval:                             S    S                   D   D  S
```

Figura 5.8. Alinhamento entre referência (REF) e hipótese (HYP) para avaliação de sistemas de reconhecimento de fala. Nesse caso, foram 8 acertos de palavras, 3 substituições, 2 deleções e 0 inserções.

Para este caso, o cálculo da taxa de erro de palavra seria:

$$WER = 100 \times \frac{3+2+0}{13} = 38,46\%$$

O padrão para os arquivos de hipóteses e referências é [ID + "\t" + texto]. É possível enviar também um repositório de arquivos de hipótese e referência. Desta forma, é possível enviar diversos segmentos de transcrição para avaliar a WER. O Sclite também disponibiliza outros arquivos com alguns dados sobre a avaliação feita. Um deles apresenta um resumo geral de todos os arquivos que foram enviados e a avaliação final do sistema. A Figura 5.9. representa um exemplo deste arquivo.

SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err
bit_bytes	99	2633	61.8	25.1	13.1	4.9	43.1	100.0
complexidade_de_algoritmos	63	1768	67.6	23.5	8.9	7.0	39.4	100.0
dnd_e_redes_p2p	112	3139	56.7	28.3	15.0	4.6	47.9	100.0
espacos_vetoriais	42	1399	50.7	30.8	18.5	3.3	52.6	100.0
introducao_a_computacao_grafica	48	1484	56.3	26.5	17.3	3.2	47.0	100.0
introducao_a_engenharia_sf	21	798	57.6	28.6	13.8	2.3	44.6	100.0
introducao_a_linguagem_html	21	654	52.3	27.8	19.9	2.8	50.5	100.0
introducao_ao_computador	21	569	75.0	15.8	9.1	4.4	29.3	100.0
introducao_a_redes_de_computadores	28	773	63.9	24.3	11.8	3.9	40.0	100.0
processo_agil	47	1454	41.7	30.8	27.4	1.0	59.2	100.0
subsistemas_de_memoria	21	804	47.5	39.8	12.7	5.1	57.6	100.0
Sum/Avg	523	15475	57.4	27.4	15.2	3.9	46.5	100.0
Mean	48.5	1289.6	57.4+	27.4+	15.2+	3.9+	46.5+	100.0
S.D.	32.1	829.5	9.4+	5.9+	5.4+	1.6+	8.6+	0.0
Median	42	1399	56.7+	27.8+	13.8+	3.9+	47.0+	100.0

Figura 5.9. Porcentagens fornecidas para avaliação de saídas do sistemas de reconhecimento de fala.

Na Figura 5.9, a primeira coluna representa cada arquivo que foi enviado para o Sclite avaliar a transcrição. A segunda coluna representa o número de sentenças, a terceira o número de palavras, a quarta a média de acerto em cada arquivo, a quinta a média substituições depois deleções depois inserções e, por fim, a WER ("Err") por arquivo e a SER ("S.Err"). Nas duas últimas linhas temos o desvio padrão da média e a média, respectivamente. Portanto, o valor final de WER dessa amostra de dados é de 47% e o valor de SER é de 100%.

Um outro arquivo do Scilite disponibiliza os principais erros que ocorrem na amostra de dados, como entrada para inserção, remoção e substituição. A Figura 5.10. ilustra uma parte deste arquivo.

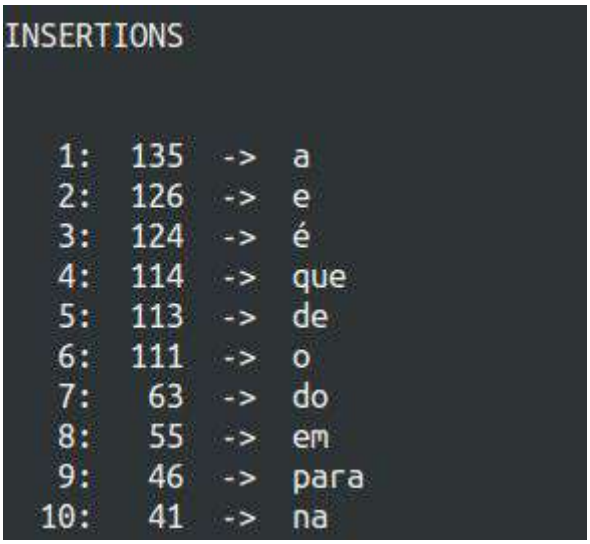

Figura 5.10. Principais erros de inserção em uma dada amostra de dados de hipóteses.

Assim, o principal erro de inserção foi "a" com 135 ocorrências, seguido de "e" com 126 ocorrências e assim por diante. Estes dados também são fornecidos para os erros de deleções e substituições.

Com estes dados é fácil avaliar um sistema de reconhecimento de fala, extrair informações de principais erros e adaptar o sistema para diminuí-los.

5.3. Treinamento de modelos

O principal fator de relevância na criação de sistemas de reconhecimento automático de fala é a elaboração de uma modelagem robusta e que seja adequada para o contexto de aplicação. Para a elaboração dos modelos acústico e de linguagem, é utilizada a técnica de treinamento supervisionado. O treinamento supervisionado constrói a modelagem a partir da observação de dados que possua as características que se deseja obter.

O treinamento do modelo acústico faz uso de uma base de áudio com texto associado. Esta base deve estar bem alinhada, o que quer dizer que áudio e transcrição devem ser bem segmentados. Já o treinamento do modelo de linguagem faz uso de uma base de textos (frases) que estejam sintaticamente corretos.

Para o treinamento de modelos de linguagem, algumas ferramentas implementam os principais algoritmos presentes na literatura. Entre elas, podemos citar o CMU-SLM [Rosenfeld and Clarkson, 1997] e o SRILM [Stolcke, 2002] como poderosas ferramentas para construção de modelos de linguagem.

No treinamento de modelos acústicos, os parâmetros do modelo são ajustados de forma que, para os áudios da base acústica, a saída gerada pelo sistema de reconhecimento de fala seja o mais próximo possível das transcrições originais. Esses ajustes são feitos por algoritmos (como Baum-Welch e Viterbi) que estão implementados em diversas ferramentas de uso livre, como HTK[7], RASR[8] e Kaldi[9].

O Kaldi é um dos *toolkits* para treinamento de modelo acústico mais usados pela comunidade. A ferramenta está sendo constantemente atualizada e contém os algoritmos do estado da arte. Além disso, o Kaldi divulga os procedimentos para o treinamento de sistemas de reconhecimento de fala feitas por grandes grupos de pesquisa ao redor do mundo. Dessa forma, estes procedimentos permitem que usuários inexperientes possam se orientar nos primeiros passos para criar modelos mais simples.

O pacote de ferramentas disponibilizado pelo Kaldi é um instrumento na elaboração de elementos básicos para um sistema de reconhecimento de fala. O *toolkit* possui ferramentas para treinamento de modelos utilizando os mais diversos métodos, incluindo o uso de Redes Neurais [Miao, 2014]. Este também possui métodos para alinhamento de texto e áudio [Ochshorn and Hawkins, 2017], ferramentas para criação de decodificadores e métodos para processamento de áudio.

5.3.1. Treinamento de modelos acústicos

O treinamento de modelos acústicos tem como objetivo a geração de modelos para sistemas ASR influenciando em um acurácia satisfatória destes. Para o treinamento de modelos acústicos HMM é necessário o fornecimento das seguintes informações: conjunto de arquivos de áudio bem segmentados; conjunto de arquivos de texto com as respectivas transcrições originais dos arquivos de áudio; dicionário de abrangência do modelo e respectivas transcrições fonéticas.

A partir disto, os passos para o treinamento de modelos acústicos são:

1. Extração de parâmetros dos arquivos separados para treino: nesta etapa são geradas características acústicas importantes para o treinamento
2. Inicialização dos modelos de fonemas: neste ponto, utiliza-se da técnica *Flat Start* para segmentação automática da base. Esta técnica é dividida em três etapas: a primeira é desconsiderar a pausa existente entre as palavras, gerando uma primeira estimativa dos modelos de fonemas. O segundo é a criação de um modelo de pausa a partir do modelo de silêncio de início e fim de frase (justificando um base de dados bem segmentada, ou seja, respeitando início e fim de frase), após isso re-estimam-se os modelos. O terceiro e último é o realinhamento da base de treino em função dos modelos e estes são re-estimados novamente [Tevah, 2006; Young et al., 2006].

[7] http://htk.eng.cam.ac.uk/
[8] https://www-i6.informatik.rwth-aachen.de/rwth-asr/
[9] http://kaldi-asr.org/

3. Conversão do modelo de fonemas para trifones: os arquivos de texto são novamente transcritos agora com trifones, levando-se ou não em conta, os trifones entre-palavras. O fones centrais dos trifones referenciam os fonemas treinados no passo anterior [da Silva, 2010].
4. Compartilhamento dos estados a partir de árvores de decisão: a lista de todos os trifones necessários ao modelo acústico é criada e no final é gerado um arquivo mapeando modelos que compartilham distribuições [Tevah, 2006; Young et al., 2006].

A Figura 5.11. apresenta, de forma simplificada, como é estruturada as características extraídas da base de treinamento e o resultado do treinamento do modelo acústico.

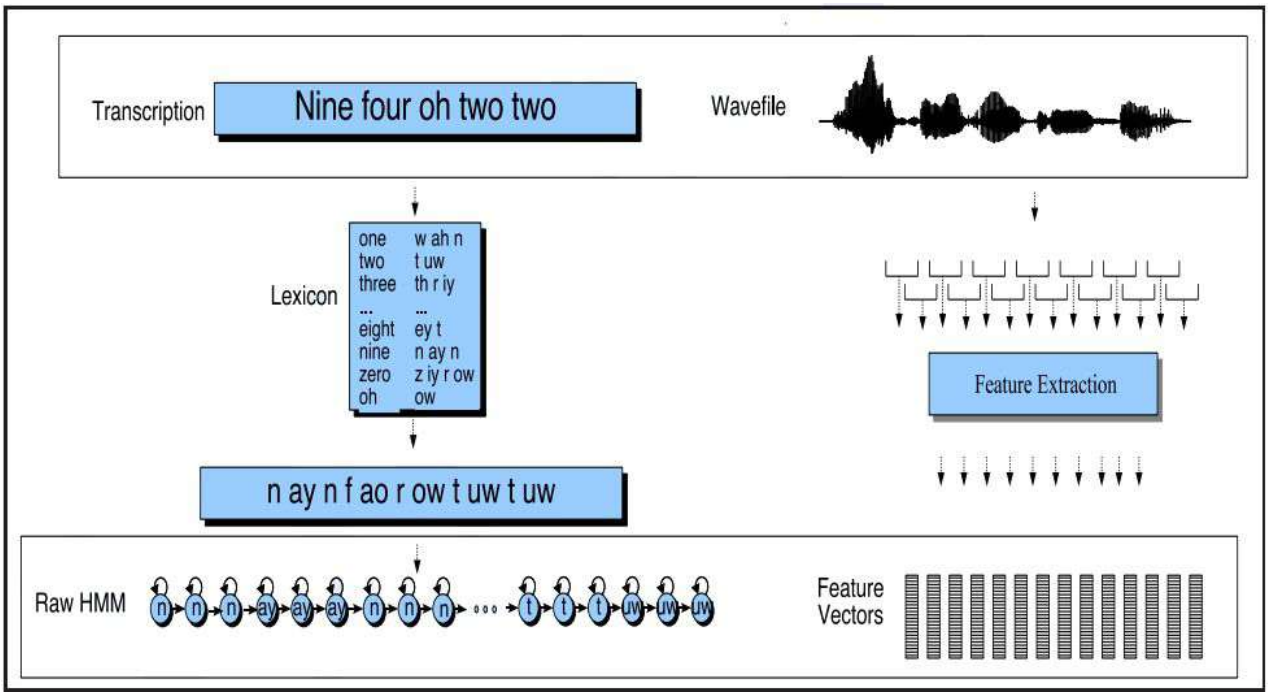

Figura 5.11. Entrada para o algoritmo de treinamento contando com, arquivo de áudio (Wavefile) e respectiva transcrição associada. O arquivo de áudio é processado para extração de características acústicas enquanto a transcrição passa pelo processo de conversão grafemática através do Léxico. Por fim é gerado um vetor de característica acústicas e uma rede de estados HMM [Jurafsky, 2009].

5.3.2. Treinamento de modelos de linguagem

O treinamento de modelos de linguagem tem como objetivo a geração de modelos para sistemas ASR que auxiliarão no reconhecimento de fala e na construção gramatical dos resultados. Para o treinamento do modelo de linguagem são requeridos o dicionário fonético e os arquivos de texto de onde serão extraídas as frequências de palavras e o relacionamento entre elas.

Algumas alterações são necessárias nos arquivos de texto para que o treinamento seja satisfatório. É de fundamental importância que os arquivos de texto tenha marcadores de início e final de frase (<s>, </s>) e que as frases estejam normalizadas (números e siglas), livres de pontuações(. , ! ? … etc) e caracteres especiais ($ % # @ etc) [Young et al., 2006].

A seguir serão descritos os passos para a criação de modelos de linguagem do tipo unigrama, bigrama e trigrama.

1. Geração da frequência das palavras existentes nos arquivos de texto e do relacionamento entre cada palavra com até duas anteriores, incluindo início e final de frase [Tevah, 2006; Young et al., 2006].
2. Identificação de palavras fora do vocabulário (OOV - *Out of Vocabulary*) que são excluídas do treinamento.
3. Treinamento dos n-gramas: Computação dos modelos unigrama, bigrama e trigrama incluindo probabilidades de sequências de palavras e fatores de escalonamento.
4. Medição da perplexidade dos modelos n-gramas gerados.

5.4. Bases de dados para treinamento

Para obter sistemas ASR que sejam robustos e com acurácia satisfatória é necessário uma base de dados bem estruturada e de grande porte para o treinamento de modelos acústicos e de linguagem.

O termo *corpus* (plural *corpora*) de voz será utilizado neste capítulo para representar a base de dados de áudio com suas respectivas transcrições textuais, utilizadas no treinamento de modelos acústicos, e o *corpus* de texto para representar a base de dados de sentenças (frases) estruturadas, utilizadas para treinar o modelo de linguagem.

Uma grande dificuldade no treinamento de modelos acústicos e de linguagem é a obtenção destes *corpora* de grande porte e gratuitos [Silva et al., 2005]. Isto é uma carência na área que difere de outras línguas como o inglês. Além disso, é importante que estes *corpora* sejam especializados, ou seja, que áudios e textos contendo informações de diferentes áreas de conhecimento sejam utilizados no treinamento. Isto para que o sistema não seja "viciado" em termos de uma única área (a não ser que este seja o foco) ou de nenhuma.

A dificuldade de possuir o *corpus* de voz e texto adequados é sem dúvida um dos principais problemas encontrados pelos pesquisadores da área.

Sistema típicos de reconhecimento de fala são treinados usando centenas de horas de dados de treinamento acústico cuidadosamente transcritos, como é dito em [Evermann et al., 2005]. Este trabalho descreve as dificuldades no reconhecimento automático de fala para sistemas CTS (*Conversation Telephone Speech*) e como o aumento da base de dados em milhares de horas melhora a acurácia dos modelos acústicos e de linguagem. Mais especificamente o trabalho compara dois sistemas de transcrição treinados com as mesmas técnicas e quantidades de dados diferentes, um com 360 e o outro com 2000 horas de conversas telefônicas e descreve quais os métodos de limpeza nos dados de áudio e de transcrições associados. Foram aplicados regras para normalizar o texto, corrigir ortografia e descartar palavras que foram ditas somente uma vez. Além disso foi mapeado, na base de áudio, blocos de silêncio e falas masculinas e femininas. O trabalho conclui apresentando as melhorias que foram adquiridas no incremento da base e os valores de ganho na acurácia e taxa de erro de palavras (WER).

Para o Português do Brasil existe uma dificuldade grande em possuir uma base extensa e gratuita. Visando contornar este problema, [Wessel and Ney, 2005] expõem um estudo da influência da quantidade de dados transcritos, inicialmente disponíveis, no desempenho dos modelos acústicos. Neste trabalho é proposto um método diferente para o treinamento de modelos com poucos dados iniciais. Visto que existe muita quantidade de dados acústicos e poucos desses estão associados à transcrições, o trabalho utiliza-se de poucos dados transcritos manualmente para treinar um sistema de base que será usado para reconhecer outros dados acústicos e assim utilizá-los também no treinamento. Para validação da proposta é feita uma marcação de confiança nas palavras transcritas, ou seja, cada palavra transcrita pelo sistema base é associada a uma medida de confiança, identificando que somente palavras acima de um limiar serão utilizadas no treinamento. Nessa abordagem, primeiramente, são estimados os parâmetros de treinamento de um modelo acústico com pequenas quantidades de fala transcritas manualmente. Este modelo acústico é então usado para reconhecer *corpora* de voz grandes que não possuem transcrições associadas e gerar um gráfico de palavras, que é computado a fim de gerar as medidas de confiança. Assim, um novo *corpus* de voz é criado unindo aquele utilizado no primeiro treinamento e o novo gerado. Com isso, o mesmo procedimento padrão de treinamento é executado com o *corpus*, combinado às medidas de confiança geradas. É ressaltado ainda que este processo gera melhorias nas medidas de confiança conforme novos modelos vão surgindo. Ainda, no trabalho, é apresentado os ganhos obtidos na acurácia dos modelos e na taxa de erro de palavras (WER - *Word Error Rate*) a partir dos experimentos realizados.

Existem ainda outros trabalhos que abordam o problema de treinar modelos com poucos recursos. Em [Riccardi and Hakkani-Tür, 2003] é apresentada também uma técnica para transcrever e gerar uma medida de confiança para as transcrições. Os segmentos que não possuem transcrições com boa confiança são transcritos manualmente construindo assim uma base de dados com qualidade melhor. Neste trabalho é descrito um método para combinar aprendizagem ativa e não supervisionada para um sistema de reconhecimento de fala. O objetivo é minimizar a supervisão humana para o treinamento de modelos acústicos e de linguagem e maximizar o desempenho com os dados de áudio com transcrições e sem transcrições associadas. A aprendizagem ativa visa diminuir o número de dados de áudio a serem transcritos manualmente processando-os, e selecionando os mais precisos com relação a uma função de custo. Para a aprendizagem não supervisionada foi utilizada as transcrições automáticas e uma medida de confiança por palavra. O trabalho é concluído apresentando os ganhos e benefícios da combinação de aprendizagem ativa e não supervisionada.

Há também trabalhos para o aproveitamento de mídias legendadas como em [Chan e Woodland, 2004], onde é aproveitada a disponibilidade de legendas em noticiários para treinar modelos acústicos e gerar frases foneticamente balanceadas para o modelo de linguagem. Neste trabalho é apresentado os problemas em se utilizar legendas como base para treino. O principal deles é que, as transcrições geralmente são oriundas de *closed caption* e legendas de comerciais, que são construídas manualmente e são parcialmente corretas. Embora estes transcritos contenham uma série de erros e não possam ser usados diretamente como dados para treino, é proposto no trabalho o usos destes dados como fonte de supervisão para o treino de modelos acústicos, técnica

conhecida como treinamento levemente supervisionado. Na abordagem proposta do treinamento levemente supervisionado, foi utilizado modelos de linguagem tendenciosos para as transcrições de *closed caption* que foram usados no processamento da base de áudio. Com isso, todas as transcrições adquiridas foram utilizadas para treinamento com a técnica de máxima verossimilhança ou para fornecer as transcrições corretas no treinamento discriminativo. Os resultados apresentados mostram que é possível a diminuição da taxa de erro de palavras utilizando esta abordagem, e que a construção de uma base de treinamento com este tipo de mídia é viável para um sistema de reconhecimento dentro desse contexto.

Outro trabalho relacionado a mídias legendas é apresentado em [Panayotov et al., 2015], onde é aproveitado o conteúdo de audiobooks para alimentar a base de dados de treino. Neste trabalho é proposto a criação de uma base de treino para modelos de sistemas de reconhecimento de fala, utilizando conteúdo de audiobooks em inglês disponibilizados gratuitamente pelo projeto LibriVox[10]. A grande questão de se utilizar audiobooks para o treinamento de modelos, é o alinhamento correto do áudio com o texto escrito. Os processos de treinamento, geralmente, requerem que os dados venham segmentados até algumas dezenas de segundos e contenham o texto relacionado ao segmento. A abordagem proposta no trabalho é composta de dois estágios. No primeiro estágio é utilizado um algoritmo de alinhamento para encontrar a melhor região entre o áudio reconhecido e o texto do capítulo. A partir disso é tomada a maior região de similaridade, que geralmente representa um capítulo inteiro, e descartado o resto. Nessa região de similaridade é destacadas palavras que, a partir do reconhecimento, tenham uma alta confiança. Depois disso o áudio é dividido em segmentos de algumas dezenas de segundos utilizando um algoritmo de programação dinâmica. Esta divisão é feita a partir do reconhecimento de regiões de silêncio. Assim é gerado um texto candidato para um segmento de áudio. Na segundo etapa é filtrado os segmentos onde o texto candidato tem uma alta probabilidade de ser impreciso. Também, é criado um gráfico de decodificação para cada segmento. Nesta etapa é rejeitado qualquer segmento que tenha desvio da transcrição original. Isto é feito utilizando técnicas para comparar palavras com blocos de áudio de fala e identificar variações bruscas dos fones de cada palavra e do sinal processado. Além disso foi filtrado segmentos com áudios barulhentos (ruído). O trabalho conclui demonstrado o resultado superior do sistema construído a partir de audiobooks em comparação a outro sistema treinado com bases de dados tradicionais, transcritos manualmente, porém de menor tamanho.

Existem ainda, bases gratuitas que foram construídas por trabalhos acadêmicos, como no desenvolvido pelo Laboratório de Processamento de Sinais (LaPS[11]) da Universidade Federal do Pará, onde foi construído um *corpus* de voz de aproximadamente 11 horas de áudio e um *corpus* de texto com 120 mil frases que são disponibilizados gratuitamente. Outra base disponibilizada gratuitamente é o CETENFolha, que consiste em textos extraídos do jornal Folha de São Paulo e disponibilizados gratuitamente em formato adequado para processamento de texto. A base é mantida pelo projeto Linguateca[12] e

[10] https://librivox.org
[11] http://www.laps.ufpa.br/falabrasil/descricao.php
[12] http://www.linguateca.pt/ACDC/

contém quase 2 milhões de frases em formato ideal para o processamento de modelos de linguagem.

Além de possuir *corpus* que sejam suficientemente grandes, outras questões são levantadas sobre este fator de vital importância na construção de sistemas ASR. É necessário que os textos estejam normalizados, ou seja, dígitos e siglas sejam traduzidos para suas respectivas representações textuais, também é de vital importância realizar uma varredura palavra por palavra para validação ortográfica (são utilizados corretores ortográficos para esta tarefa) [Tevah, 2006] e que as frases sejam foneticamente balanceadas (distribuição fonética similar à encontrada na fala) [Ynoguti, 1999]. Isto deve ser respeitado pois o modelo de linguagem precisa possuir um padrão de escrita. Por exemplo, se no *corpus* de texto usado no treinamento existir tanto a *string* "2" quanto a *string* "dois", a probabilidade será dividida entre as duas *strings* sendo que elas possuem o mesmo significado ("dois" é a representação textual do dígito 2). Outro exemplo clássico é com relação erros de ortografia (ou reformas na escrita das palavras, como aconteceu no Brasil, em 2009, com o Novo Acordo Ortográfico). Se no *corpus* de texto possuir as *strings* "ideia" e "idéia", a probabilidade também será dividida entre estas palavras, gerando contextos diferentes. Além de dígitos e erros de ortografia, é importante observar outros fatores no *corpus* de texto: devem ser removidos caracteres especiais e de pontuação; conversão de número romanos em representação textual; conversão de números ordinais em representação textual; conversão de abreviações e representação textual completa; conversão de valores monetários e de temperatura; demarcação de início e fim de frase (<s> e </s> respectivamente); capitalização de texto em minúsculo. A Figura 5.12. é um exemplo de uma parte de arquivo, ideal para o modelo de linguagem, pertencente ao *corpus* de texto.

```
<s> o clima na minha cidade é chuvoso chove a maior parte do tempo e pouco sol </s>
<s> bom dia hoje é dia vinte e sete de janeiro de dois mil e quartoze e o sol já está predominante </s>
<s> o preço do arroz está três vezes mais caro que o normal chegando a dezoito reais e treze centavos </s>
<s> a ideia principal do jogo é que ele possa simular a realidade </s>
<s> maria foi aprovada em décimo lugar no curso de medicina na universidade de são paulo </s>
<s> a temperatura máxima prevista para hoje é de trinta e seis graus <s>
```

Figura 5.12. Arquivo para treinamento de modelos de linguagem normalizados.

Ainda, para a *corpus* de voz é necessário que o começo e fim de segmentos sejam respeitados com silêncio e que os arquivos de áudio tenham uma mesma configuração (amostragem, quantidade de canais e *codec*). As transcrições manuais são as mais adequadas para serem utilizadas no *corpus* de voz, pois uma série de regras devem ser respeitadas na escrita do texto. A utilização de legendas, por exemplo, gera uma série de dificuldades pois, geralmente, o tempo de início e fim de segmentos de legenda estão em desordem com o áudio, o autor das legendas omite algumas palavras que foram faladas pelo locutor por questões sintáticas da frase e não existe um padrão de escrita. Com relação às configurações de áudio, é necessário que exista um padrão pois palavras iguais ditas em áudios diferentes com configurações diferentes iriam gerar um vetor de características diferentes para uma mesma palavra, diminuindo assim a probabilidade de ocorrência da palavra. Portanto, é importante que os áudios possuam a melhor configuração de áudio possível e que seja comum a todos.

5.5. Ferramentas para construção de componentes básicos de sistemas ASR

A partir da definição de conceitos básicos para o reconhecimento de fala, pode-se iniciar o processo de construção destes sistemas. Os recursos e o tempo computacional necessários para a criação destes sistemas estão diretamente ligados à base de dados utilizada no treinamento. Por ser um processo de geração de probabilidades a partir de dados observados, o custo computacional é levado em consideração. O principal componente envolvido neste custo computacional é o modelo acústico. Desta forma, dependendo do tamanho *corpus* de voz utilizado, será necessário o uso de equipamentos melhores e até recursos específicos (GPUs por exemplo).

Para treinar, por exemplo, um modelo acústico com um *corpus* de voz de aproximadamente 100 horas é desejável contar com pelo menos uma máquina Linux (ex: Ubuntu 16.04) com processador Intel Core i7, 32 GB de RAM, 1 TB de HD e uma boa placa GPU (ex: Nvidia GTX-970). Com esta infraestrutura de hardware é possível treinar um modelo acústico em 5 dias.

Nesta seção, será apresentada ferramentas para criação dos componentes básicos descritos neste capítulo, bem como processo de instalação, dependências, receitas de treinamento e arquivos gerados. O kaldi, será a ferramenta responsável pelo treinamento do modelo acústicos e pela geração do sistema de decodificação. O SRILM, será a ferramenta responsável pela criação do modelo de linguagem. E por fim o Multilingual G2P, será a ferramenta responsável para construção do léxico em PT-BR.

5.5.1. Kaldi

O Kaldi é um *toolkit* que é usado principalmente por pesquisadores da área de reconhecimento de fala. Ele é semelhante a outro *toolkit* muito conhecido da área que é o HTK[13], porém o Kaldi propõe algumas facilidades e a simplificação dos processos envolvendo sistemas de reconhecimento de fala, principalmente com relação ao treinamento de modelos acústicos e decodificação. Suas principais características são: Integração a nível de código com Finite State Transducers (FSTs); Suporte a álgebra linear; algoritmos fornecidos de forma genérica; licença aberta; disponibilização de receitas completas para a construção de sistemas de reconhecimento de fala [Povey et al., 2011].

As ferramentas apresentados pelo Kaldi nesta seção, serão aquelas utilizadas para treinamento de modelos acústicos e na construção do decodificador Kaldi GStreamer [Alumäe, 2014].

O processo de instalação do Kaldi conta com os seguintes passos:

1. Faça o *clone* do repositório do Kaldi no GitHub
    a. *$ git clone https://github.com/kaldi-asr/kaldi*
2. Acesse a pasta kaldi/tools/
    a. Execute o comando abaixo para verificar dependências do Kaldi
        i. *$ ./extras/check_dependencies.sh*
    b. Se alguma dependência precisar ser instalada, o script irá indicá-la, caso contrário lançará “all OK”
    c. É importante que o passo de verificação de dependências seja seguido para evitar erros futuros

[13] http://htk.eng.cam.ac.uk/

   d. Execute o comando abaixo:
      i. $ *make*
   e. Este processo pode ser demorado. Caso esteja disponível múltiplos núcleos na máquina, é possível utilizar o parâmetro "-j" para paralelizar o processo. Ex: uso de 4 CPU's
      i. $ make -j 4
3. Agora, acesse a pasta kaldi/src/
   a. Execute os comandos abaixo:
      i. $ ./configure --shared
      ii. $ make depend
      iii. $ make
   b. Este processo pode ser demorado. Caso esteja disponível múltiplos núcleos na máquina, é possível utilizar o parâmetro "-j" para paralelizar o processo. Ex: uso de 8 CPU's
      i. $ make depend -j 8
      ii. $ make -j 8
4. Ainda, é necessário a anexação do SRILM ao Kaldi, pois algumas receitas disponibilizadas fazem uso da ferramenta para modelos de linguagem de forma integrada. Assim, baixe o código fonte do SRILM e o deposite na pasta *kaldi/tools/*. Após isto, execute o script dentro da pasta *$ ./install_srilm.sh*
5. Após estes passos, o kaldi está instalado
6. Para testar se tudo ocorreu bem, é recomendável o teste com algum dos exemplos disponíveis na pasta do kaldi. A maioria deles são receita prontas de treinamento para ajuda os desenvolvedores. Um destes exemplo de execução de treinamento é apresentado abaixo:
   a. Acesse a pasta kaldi/egs/yesno/s5/
   b. Execute o comando abaixo
      i. $ ./run.sh
   c. O treinamento de um sistema de transcrição simples será iniciado e ao final apresentará a WER deste sistema para uma base de testes. Caso a instalação tenha ocorrido com sucesso, o aparecimento da WER indicará que o Kaldi está funcionando plenamente para treinamento de modelos acústicos

Com o Kaldi instalado, é importante entender os principais passos de uma receita de treinamento de modelos acústicos, quais arquivos de entrada utilizar e em qual formato. Estes detalhes podem ser encontrados na seção 5.5.1.1. Com relação ao decodificador, a seção 5.5.1.2., apresentará um sistema de decodificação simples e os principais elementos para criação deste componente do sistema de reconhecimento de fala.

5.5.1.1. Kaldi como ferramenta para criação de modelos acústicos

Para treinar o modelo acústico são necessários alguns arquivos principais de indicação do *corpus* de voz. É importante enfatizar que o *corpus* de voz esteja bem segmentado para que o sistema de reconhecimento de fala possua uma acurácia melhor. Ainda, uma boa prática para preparação do *corpus* de voz, é aconselhável que cada arquivo de áudio tenha um identificador (id) próprio e que este seja padronizado. A Figura 5.13. ilustra um exemplo destes arquivos pertencentes ao *corpus* de voz.

```
A001.wav  A088.wav  A175.wav  A263.wav  A361.wav  A449.wav  A537.wav
A002.wav  A089.wav  A176.wav  A264.wav  A362.wav  A450.wav  A538.wav
A003.wav  A090.wav  A177.wav  A265.wav  A363.wav  A451.wav  A539.wav
A004.wav  A091.wav  A178.wav  A266.wav  A364.wav  A452.wav  A540.wav
A005.wav  A092.wav  A179.wav  A267.wav  A365.wav  A453.wav  A541.wav
A006.wav  A093.wav  A180.wav  A268.wav  A366.wav  A454.wav  A542.wav
A007.wav  A094.wav  A181.wav  A269.wav  A367.wav  A455.wav  A543.wav
A008.wav  A095.wav  A182.wav  A270.wav  A368.wav  A456.wav  A544.wav
A009.wav  A096.wav  A183.wav  A271.wav  A369.wav  A457.wav  A545.wav
A010.wav  A097.wav  A184.wav  A272.wav  A370.wav  A458.wav  A546.wav
A011.wav  A098.wav  A185.wav  A273.wav  A371.wav  A459.wav  A547.wav
```

Figura 5.13. Organização do *corpus* de voz

Neste caso, por exemplo, o identificador é constituído de um carácter alfabético e de um número de três dígitos, gerando assim mais de 18 mil combinações possíveis de identificadores. O áudios estão no formato .WAV.

Como foi descrito na seção 5.3., o *corpus* de voz é composto, além dos arquivos de áudio, das respectivas transcrições exatas de cada arquivo de áudio. Para esta informação, o Kaldi faz uso de quatro arquivos principais, três deles essenciais. Os arquivos principais são:

- **wav.scp**: arquivo de texto com várias linhas, sendo que cada linha contém um identificador de arquivo e o caminho do arquivo de áudio identificado.

  O arquivo **wav.scp** tem a seguinte estrutura:

  <identificador_do_audio> <caminho_para_audio>

  A Figura 5.14. ilustra um exemplo de arquivo **wav.scp**.

```
A001 /media/dados/resources/database/speech/pt-BR/rnp/8k/A001.wav
A002 /media/dados/resources/database/speech/pt-BR/rnp/8k/A002.wav
A003 /media/dados/resources/database/speech/pt-BR/rnp/8k/A003.wav
A004 /media/dados/resources/database/speech/pt-BR/rnp/8k/A004.wav
A005 /media/dados/resources/database/speech/pt-BR/rnp/8k/A005.wav
A006 /media/dados/resources/database/speech/pt-BR/rnp/8k/A006.wav
A007 /media/dados/resources/database/speech/pt-BR/rnp/8k/A007.wav
```

Figura 5.14. Arquivo wav.scp

  Pode-se observar que, por exemplo, o arquivo de áudio cujo identificador para o Kaldi é *A001*, é representado pelo arquivo de áudio *A001.wav*. Repare que o nome do identificador e do arquivo de áudio são iguais, porém isto não é regra. O arquivo de áudio cujo identificador é *A001* para o Kaldi poderia estar relacionado a um arquivo *videoaula.wav*, por exemplo;

- **segments**: arquivo de texto com várias linhas, sendo que cada linha é composta por um identificador de segmento, identificador do arquivo de áudio onde existe esse segmento e a região do arquivo de áudio onde esse segmento existe. Este arquivo não é necessário, quando cada arquivo de áudio representa um pequeno segmento de fala. Se cada arquivo de áudio tiver mais de 30 segundos, é

necessário que use o arquivo **segments** para indicar uma possível segmentação para o áudio.

Caso seja necessário utilizar o arquivo **segments**, é importante que os segmentos de áudio sejam pequenos (10~30 segundos) e que a transcrição em cada segmento seja bem alinhada. Caso as transcrições exatas estejam mal alinhadas, o modelo acústico vai ser construído associando característica acústicas da pronúncia de uma palavra à outra. Existem ferramentas para segmentação de áudio a partir da identificação de silêncio[14]. Estas ferramentas podem auxiliar na segmentação do áudio diminuindo o esforço manual para esta tarefa de preparação de dados. Existe também um projeto intitulado Gentle[15], que utiliza o Kaldi, para realizar o alinhamento forçado de palavras com áudio. Está técnica também pode auxiliar na preparação da base, caso as informações de tempos dos segmentos não estejam disponíveis.

O arquivo **segments** tem a seguinte estrutura:

<identificador_do_segmento> <identificador_do_audio> <início> <fim>

A Figura 5.15., ilustra um exemplo de arquivo **segments**.

```
A001-000080-041481-042021 A001 414.81 420.21
A001-000081-042054-042264 A001 420.54 422.64
A001-000082-042477-042960 A001 424.77 429.60
A002-000000-000000-000522 A002 0.00 5.22
A002-000001-000603-001479 A002 6.03 14.79
A002-000002-001503-001659 A002 15.03 16.59
A002-000003-001716-002106 A002 17.16 21.06
```

Figura 5.15. Arquivo segments

Neste exemplo, temos a primeira coluna que representa o identificador do segmento de áudio (ex: *A001-000080-041481-042021*), a segunda coluna representa o identificador do áudio onde aquele segmento se encontra (ex: *A001*), a terceira coluna o tempo (em segundos) onde aquele segmento se inicia no áudio (ex: *414.81*) e a última coluna, o tempo (em segundos) onde o segmento termina no áudio (ex: *420.21*). Repare que os segmentos de áudio são sempre pequenos com relação ao tempo de duração. O segmento, exemplificado acima, tem apenas 5.4 segundos (420.21s - 414.81s = 5.4 s).

- **text**: arquivo de texto com várias linhas. Cada linha contém o identificador de segmento seguido de sua transcrição textual. Não deve haver pontuação, dígitos, siglas, abreviações, etc, na transcrição de cada segmento. Todo o texto deve estar padronizado, ou tudo maiúsculo ou tudo minúsculo.

  O arquivo **text** tem a seguinte estrutura:

  <identificador_do_segmento> <frase>

[14] https://github.com/wiseman/py-webrtcvad
[15] https://lowerquality.com/gentle/

A Figura 5.16. ilustra um exemplo de arquivo **text**

```
A001-000080-041481-042021 grupos aglomerados e superaglomerados de galáxias
A001-000081-042054-042264 de fato nosso universo é extraordinariamente grande
A001-000082-042477-042960 e universo tão extraordinariamente grande
                          é difícil imaginar que estamos sós
A002-000000-000000-000522 a vida
A002-000001-000603-001479 como a conhecemos hoje na terra é espetáculo que assistimos
                          praticamente todos os ambientes conhecidos pelo homem
A002-000002-001503-001659 mas do que ela depende
A002-000003-001716-002106 ela é o resultado das condições e recursos disponíveis
```

Figura 5.16. Arquivo text

Neste exemplo, temos a primeira coluna que representa o identificador do segmento e a segunda o texto contido dentro daquele segmento. Comparando com o arquivo segments, exemplificado na Figura 5.15., a frase "*grupos aglomerados e superaglomerados de galáxias*" é dita no aúdio A001 no segmento que começa no tempo *414.81*s e termina no tempo *420.21*s. Este segmento é representado pelo identificador *A001-000080-041481-042021*.

- **utt2spk**: arquivo de texto com várias linhas contendo o identificador do segmento e a identificação do locutor que gravou tal segmento. Se essa informação não estiver disponível, não há problemas.

  O arquivo **utt2spk** tem a seguinte estrutura:

  <identificador_do_segmento> <identificador_de_locutor>

  A Figura 5.17. ilustra um exemplo de arquivo **utt2spk**

```
A001-000080-041481-042021 A00180
A001-000081-042054-042264 A00181
A001-000082-042477-042960 A00182
A002-000000-000000-000522 A00200
A002-000001-000603-001479 A00201
A002-000002-001503-001659 A00202
A002-000003-001716-002106 A00203
```

Figura 5.17. Arquivo utt2spk

Neste exemplo cada segmento tem um locutor diferente. Este locutor é representado por um identificador único. No exemplo da Figura 5.17., está informação não é conhecida para cada segmento de áudio, por isso cada segmento possui um identificador diferente. Mas esta informação é interessante quando se deseja treinar um sistema de reconhecimento de voz para uso próprio ou para separar fala de homens, mulheres, crianças e idosos por exemplo.

Estes quatro arquivos principais e os arquivos de áudio, são os principais elementos para o treinamento de modelos acústicos. Para executar o processo, deve se utilizar algumas das receitas disponibilizadas pelo Kaldi no GitHub[16]. As receitas de treinamento são todas padronizadas (arquivos executáveis, pastas de *corpus* e o

[16] https://github.com/kaldi-asr/kaldi

caminho dos arquivos gerados), assim é necessário apenas anexar o *corpus* de texto à receita e executar o seguintes comandos:

1. Execute o comando *$ ./run.sh*
2. O resultado do treinamento pode ser encontrado na pasta *exp/*

As receitas de treinamento podem ser encontrada em *kaldi/egs/*. Mais detalhes de implementação e ferramentas para sistemas de reconhecimento de fala podem ser encontradas na documentação do Kaldi.

5.5.1.2. Kaldi como ferramenta para criação de decodificador

O Kaldi GStreamer[17], é um sistema para decodificação de sinal de fala em sequência textual aproximada. Sua principal característica é ser um servidor full-duplex de reconhecimento de fala em tempo real onde é possível indicar qual modelo será utilizado pelo sistema. Neste passo, com o modelo acústico já implementado, é possível indicar o modelo para ser utilizado pelo servidor de reconhecimento de fala. O decodificador utiliza o Kaldi e GStreamer para implementação, porém foi implementado um *dockerfile*[18] do projeto do decodificador, com todas as dependências compiladas.

Além dos arquivos gerados pelo Kaldi, no treinamento de modelos, o decodificador faz uso de um arquivo principal de configuração (.yaml), para inicialização do serviço de reconhecimento de fala. Existem alguns exemplos[19] [20] [21] deste tipo de arquivo. A Figura 5.18. apresenta uma parte deste arquivo de configuração, onde deve ser indicado cada arquivo do modelo.

```
model : test/models/english/tedlium_nnet_ms_sp_online/final.mdl
word-syms : test/models/english/tedlium_nnet_ms_sp_online/words.txt
fst : test/models/english/tedlium_nnet_ms_sp_online/HCLG.fst
mfcc-config : test/models/english/tedlium_nnet_ms_sp_online/conf/mfcc.conf
ivector-extraction-config : test/models/english/tedlium_nnet_ms_sp_online/conf/ivector_extractor.conf
```

Figura 5.18. Arquivo de configuração do decodificador

Cada linha dessa, deve indicar o arquivo gerado pelo treinamento.

Para instalação e construção do decodificador, deve-se seguir os passos abaixo:

1. O Docker[22] deve estar instalado na máquina onde o decodificador será construído, para instalá-lo, siga os passo na página do Docker Docs.
2. Com o Docker instalado, dê um *pull* na imagem do Docker Hub
    a. *$ docker pull jcsilva/docker-kaldi-gstreamer-server*
3. Crie uma pasta, onde ficará localizado seus modelos. Uma sugestão de criação é:
    a. *$ sudo mkdir /media/kaldi_models/*
4. Execute o comando abaixo para criar o *container* e inicializar o serviço para o arquivo .yaml gerado

[17] https://github.com/alumae/kaldi-gstreamer-server
[18] https://github.com/jcsilva/docker-kaldi-gstreamer-server
[19] https://github.com/alumae/kaldi-gstreamer-server/blob/master/sample_worker.yaml
[20] https://github.com/alumae/kaldi-gstreamer-server/blob/master/estonian_worker.yaml
[21] https://github.com/alumae/kaldi-gstreamer-server/blob/master/sample_english_nnet2.yaml
[22] https://docs.docker.com/engine/installation/

   a. *$ docker run -it -p 8080:80 -v /media/kaldi_models:/opt/models jcsilva/docker-kaldi-gstreamer-server:latest /bin/bash*
   b. *$ /opt/start.sh -y /opt/models/model.yaml*
5. Após este passo, volte para a máquina *host* (Ctrl + P e Ctrl + Q). O serviço de reconhecimento de fala está ouvindo na porta 8080
6. Um exemplo de execução do serviço pode ser feito utilizando o *script* de cliente
   a. *$ wget https://raw.githubusercontent.com/alumae/kaldi-gstreamer-server/master/kaldigstserver/client.py -P /tmp*
7. Caso seja necessário parar o serviço, execute o comando abaixo dentro do *container*
   a. *$ /opt/stop.sh*

5.5.2. SRILM

O SRILM (do Inglês SRI Language Modeling toolkit) é um *toolkit* para construção e aplicação de modelos estatísticos de linguagem que é utilizado principalmente em Reconhecimento de Fala, Tradução Automática e Correção de Palavras.

No contexto de reconhecimento de fala, um exemplo de aplicação seria a utilização errada das palavras "sela" e "cela". Embora o som reproduzido pela fala para ambas seja idêntico, elas possuem significados completamente diferentes. A palavra "sela" é relacionado a sela de cavalo e a palavra "cela" é relacionado a cela de prisão. Assim, a probabilidade da frase "Pedro está em uma **cela** separada" é maior que a probabilidade da frase "Pedro está em uma **sela** separada". Esta informação é disponibilizada pelo modelo de linguagem e apresenta um reconhecimento de fala mais coerente.

Como descrito na seção 5.2.2., a probabilidade de ocorrência de uma palavra é dada a partir da probabilidade de ocorrência desta palavra dado que uma outra palavra ocorreu anteriormente. Com isso, a probabilidade de ocorrência de uma frase é dada pela produtório da ocorrência de cada uma das palavras desta frase. Por exemplo, $P(\text{"}Pedro\ está\ em\ uma\ cela\ separada\text{"})$ pode ser definido como[23]:

$$P(\text{"}Pedro\ está\ em\ uma\ cela\ separada\text{"}) = P(Pedro|<s>) \times$$
$$P(está|Pedro) \times P(em|está) \times P(uma|em) \times$$
$$P(cela|uma) \times P(separada|cela) \times P(</s>|separada)$$

A probabilidade de uma palavra é definido pela probabilidade de ocorrência da palavra dado que uma outra ocorreu. Isto quer dizer que somente um palavra, que aconteceu anteriormente, é levada em consideração na geração das probabilidades. Este modelo, então, é chamado Unigram. Existem também modelos Bigram, Trigam e N-gram. Para está mesma frase, um exemplo de modelo Bigram, resultaria no seguinte:

$$P(\text{"}Pedro\ está\ em\ uma\ cela\ separada\text{"}) =$$
$$P(está|<s>\ Pedro) \times P(em|Pedro\ está) \times P(uma|\ está\ em) \times$$
$$P(cela|\ em\ uma) \times P(separada|uma\ cela) \times P(</s>|cela\ separada)$$

[23]Os termos "<s>" e "</s>" representam início e fim de frase respectivamente

Para os modelos N-gram, é importante ressaltar que a medida que N cresce, o gasto computacional se torna muito grande e muitas vezes inviável para geração dos modelos. Assim é comum utilização de modelos Trigram, 4-gram e 5-gram.

O SRILM pode ser utilizado para treinar esses diversos modelos, com diferentes valores de n-gram, métodos de desconto, interpolação de modelos etc.

Para o processo de instalação do SRILM é necessário alguns passos descritos abaixo. As etapas descritas abaixo são para instalação em um ambiente Linux 64 bits.

1. Faça o download de todas as dependências descritas na página de Download do SRI
2. Faça o download da versão mais recente do SRILM preenchendo o formulário na página de Download o SRI
3. Extraia o arquivo na pasta onde deseja instalar o SRILM
4. Abra para edição, o arquivo "Makefile" dentro da pasta onde o SRILM foi descompactado.
5. Encontre uma linha comentada como esta:
    a. *# SRILM = /home/speech/stolcke/project/srilm/devel*
    b. Remova o caracter "*#*"
    c. Substitua "/home/speech/stolcke/project/srilm/devel" pelo caminho para a pasta onde o SRILM foi descompactado
6. Usando o terminal do Linux, navegue até a pasta onde o SRILM foi descompactado
7. Execute o comando **make**
8. Os arquivos binários compilados do SRILM, devem ser encontrados em /[caminho_para_SRILM]/bin/i686-m64/

Os passos para execução de um treinamento simples com o SRILM são:

1. Navegue até a pasta /[caminho_para_SRILM]/bin/i686-m64/
2. Execute o comando:
    a. *./ngram-count -text [corpus_texto] -order 3 -lm [arquivo_de_saida]*

O executável binário, *ngram-cout*, é a principal ferramenta do SRILM. Ele é responsável por treinar os modelos de linguagem e possuem uma série de parâmetros de configuração de treinamento.

- ***-text*** *[corpus_texto]*: Este parâmetro representa a entrada de texto para o treinamento do modelo de linguagem. Este arquivo deve respeitar os padrões discutidos na seção 5.4. para *corpus* de texto
- ***-order*** *n*: Este parâmetro representa a ordem de *n*-gram que deseja que o modelo seja treinado. No exemplo acima foi treinado o modelo Trigram para o *corpus* de texto de entrada
- ***-lm*** *[arquivo_de_saida]:* arquivo de saída do treinamento onde será descrito as probabilidades de cada gram treinado. Um exemplo de arquivo será apresentado nas próximas páginas.

Existem ainda outros parâmetros para extrair informações e melhorar o modelo. Entre eles podemos citar alguns mais importantes como:

- ***-write-vocab*** *file:* Escrever no arquivo *file* o vocabulário de um dado *corpus* de texto
    - Ex: *./ngram-count -text [corpus_texto] write-vocab [arquivo_de_saida]*
- ***-prune*** *threshold:* Efetuar um corte no modelo final, definido por um *threshold*
    - Ex: *./ngram-count -text [corpus_texto] -order 3 -prune 1e-10 -lm [arquivo_de_saida]*
- ***-wbdiscount****n:* Parâmetro para uso do método de desconto Witten-Bell. Onde *n* é o valor do gram onde se deseja aplicar o desconto
    - Ex: *./ngram-count -text [corpus_texto] -order 3 -wbdiscount3 -wbdiscount2 -lm [arquivo_de_saida]*
- ***-ndiscount****n:* Parâmetro para uso da lei de desconto natural de Ristad. Onde *n* é o valor do gram onde se deseja aplicar o desconto
    - Ex: *./ngram-count -text [corpus_texto] -order 3 -ndiscount3 -ndiscount2 -ndiscount1 -lm [arquivo_de_saida]*
- ***-kndiscount****n:* Parâmetro para uso do método de desconto Kneser-Ney modificado. Onde *n* é o valor do gram onde se deseja aplicar o desconto
    - Ex: *./ngram-count -text [corpus_texto] -order 3 -kndiscount2 -ndiscount1 -lm [arquivo_de_saida]*
- ***-ukndiscount****n:* Parâmetro para uso do método de desconto Kneser-Ney original. Onde *n* é o valor do gram onde se deseja aplicar o desconto
    - Ex: *./ngram-count -text [corpus_texto] -order 3 -ukndiscount2 -lm [arquivo_de_saida]*
- ***-interpolate****n:* Parâmetro para interpolação de gram de ordem inferior a *n*.
    - Ex: *./ngram-count -text [corpus_texto] -order 3 -interpolate3 -kndiscount2 -ndiscount1 -lm [arquivo_de_saida]*

O resultado final de um treinamento de modelo de linguagem é um arquivo do tipo ARPA que é criado no caminho que for indicado no parâmetro *-lm [arquivo_de_saida]*. A cara do arquivo final de treinamento de modelo é apresentado na Figura 5.19., onde o comando utilizado para geração foi: *./ngram-count -text [corpus_texto] -order 4 -lm [arquivo_de_saida]*.

```
\data\
ngram 1=70
ngram 2=98

\1-grams:
-1.22617        </s>
-99     <s>     -0.8230478
-1.305351       a       -0.756101
-2.004321       aprovada        -0.2967086
-2.004321       arroz   -0.2923438
-2.004321       bom     -0.2923438
-2.004321       caro    -0.2923438
-2.004321       centavos        -0.2744322
-2.004321       chuvoso -0.2967086
-2.004321       cidade  -0.2834803

\2-grams:
-0.90309        <s> </s>
-0.60206        <s> a
-0.90309        <s> bom
-0.90309        <s> maria
-0.60206        <s> o
-0.7781513      a dezoito
-0.7781513      a ideia
-0.7781513      a maior
-0.7781513      a realidade
-0.7781513      a temperatura
-0.30103        aprovada em

\end\
```

Figura 5.19. Parte do arquivo ARPA de saída do treinamento de modelo de linguagem treinado com SRILM. Para este modelo foi utilizado um *corpus* de texto com 6 frases e 70 palavras sem repetição. Um *corpus* deste tamanho é considerado muito pequeno e deve ser utilizado apenas para exemplificação e não para aplicação.

Outra ferramenta importante do SRILM é o executável binário *ngram*. Ele é responsável por avaliação do modelo, medição de perplexidade e interpolação entre dois ou mais modelos treinados. Um exemplo de utilização para medição da perplexidade pode ser feito com o comando: *./ngram -lm [arquivo_ARPA] -ppl [corpus_teste]*. Para este comando, o resultado é como ilustrado na Figura 5.20. O *corpus* de teste do modelo de linguagem tem o mesmo formato do *corpus* de texto apresentado na seção 5.4., porém com tamanho bem menor. Um valor de referência para o tamanho do *corpus* de teste é de 20% do tamanho no *corpus* de texto no qual o modelo foi treinado, porém não existe um valor fixo para isto. Além disso, é importante que as sentenças que estiverem presentes no *corpus* de texto não façam parte do *corpus* de teste pois a avaliação não reproduziria o que acontece de fato, numa situação real de utilização do modelo de linguagem. Portanto, é aconselhável que separe o *corpus* de texto e *corpus* de teste antes

do treinamento, caso a avaliação seja um ponto de interesse. Repare que medir a perplexidade do modelo de linguagem não é um ponto vital para o funcionamento, somente um indicativo para avaliar o quão bom é um modelo de linguagem.

```
file corpus_teste: 6 sentences, 96 words, 11 OOVs
1 zeroprobs, logprob= -183.195 ppl= 108.517 ppl1= 151.667
```

Figura 5.20. Medição de perplexidade com SRILM

Na Figura 5.20, pode-se observar alguns resultados para o arquivo de teste do modelo de linguagem. A primeira linha apresenta o nome do arquivo de teste, o número de sentenças que ele possui, quantidade de palavras e quantas dessas não fazem parte do vocabulário do modelo de linguagem que está sendo avaliado (OOV - *Out of Vocabulary).* Na segunda linha alguns valores numéricos são representados, porém o valor mais importante é o *ppl* que representa a perplexidade do modelo de linguagem para o *corpus* de teste fornecido. Neste caso a perplexidade é de 108.517. É importante ressaltar que a perplexidade é o inverso da probabilidade, com isso quanto menor o valor da perplexidade melhor é o modelo de linguagem.

O executável binário *ngram* também possui outros parâmetros que são ferramentas importantes para melhorias e experimentos com modelo de linguagem. Entre eles podemos citar:

- ***-mix-lm*** *file:* Parâmetro utilizado quando se deseja interpolar dois modelos. O *file* deve ser o arquivo ARPA que se deseja interpolar.
    - Ex: *./ngram -lm [lm_ARPA_1] -order 4 -mix-lm [lm_ARPA_2]*
- ***-lambda*** *weight:* O parâmetro lambda define o peso do modelo principal na interpolação de modelos. O valor *default* é 0.5
    - Ex: *./ngram -lm [lm_ARPA_1] -order 4 -mix-lm [lm_ARPA_2] -lambda 0.6*
- ***-mix-lm2*** *file:* Este parâmetro deve ser usado quando se deseja interpolar 2 modelos ou mais (*-mix-lm3, -mix-lm4 …).* O *file* deve ser o arquivo ARPA que se deseja interpolar.
    - Ex: *./ngram -lm [lm_ARPA_1] -order 4 -mix-lm [lm_ARPA_2] -mix-lm2 [lm_ARPA_3]*
- ***-mix-lambda2*** *weight:* Esse parâmetro deve ser usado quando se deseja interpolar 2 modelos ou mais (*-mix-lambda3, -mix-lambda4 …).* O valor *weight* representará o peso de cada modelo na interpolação.O valor *default* é 0.5
    - Ex: *./ngram -lm [lm_ARPA_1] -order 4 -mix-lm [lm_ARPA_2] -lambda 0.4 -mix-lm2 [lm_ARPA_3] -mix-lambda2 0.6*

Os modelos de linguagem são sempre dependentes do contexto de treinamento em que foram construídos, com isso não é possível afirmar qual o melhor método de treinamento (“qual método de desconto utilizar?”; “qual número de grams no treinamento?”; “é necessário interpolar os grams?”; “é ideal criar dois modelos e interpola-los?”; “qual peso definir para cada modelo na interpolação?”). Portanto, executar experimentos de treinamento avaliando os modelos a partir da perplexidade é sempre uma forma interessante de construir um bom modelo de linguagem para a aplicação desejada.

5.5. Aplicações

A viabilidade de extrair informações de arquivos multimídia com áudio e vídeo possibilita uma série de aplicações em diversos contextos.

Uma das aplicações mais conhecidas para reconhecimento automático de fala é o uso de transcrição simultânea na criação de legendas ocultas para sistemas de televisão. O AUDIMUS.MÉDIA [Meinedo et al., 2003] é um sistema que gera legendas ao vivo para emissoras de TV e de *streaming*. O objetivo do sistema de reconhecimento de fala, neste contexto, é gerar a legenda sem intervenção humana evitando custo com serviços de legendagem manual.

O reconhecimento de fala também é utilizado para questões de aprendizado e acessibilidade. O CineAD [Campos et al., 2014] é um sistema para geração automática de roteiros de audiodescrição (AD), que é um recurso essencial para pessoas cegas e com baixa visão para acesso ao cinema. Este trabalho utiliza, dentre outras técnicas, o reconhecimento de fala para geração de legendas que serão utilizadas na criação de AD. O reconhecimento de fala também pode ser utilizado para questões de aprendizado, como descrito em [Higgins and Raskind, 1999], onde foi avaliado o uso de um sistema ASR para auxiliar na educação de crianças e adolescentes com dificuldade de aprendizagem. O reconhecimento de fala também pode ser usado, na área da educação, para melhorar o desempenho de crianças e adolescentes como descrito em [Hämäläinen et al., 2013], onde é apresentado um jogo educacional que utiliza a técnica de reconhecimento de fala com o objetivo de melhorar a coordenação física e as habilidades em matemática e música dos estudantes.

Ainda, podemos citar aplicações práticas e cotidianas para sistemas de reconhecimento de fala. O uso de assistentes virtuais em *smartphones* está presente no sistema das principais empresas da área de telefonia móvel. Os assistentes são elaboradas para permitir que o usuário interaja com o dispositivo a partir da voz, utilizando assim um sistema de reconhecimento de fala e técnicas para processamento do resultado do reconhecimento. Estes sistemas de reconhecimento também podem ser encontrados em diversos dispositivos que interagem com usuários, como por exemplo carros, eletrodomésticos, sistemas de autenticação, casas inteligentes etc. No âmbito dos negócios, o CALO [Tur et al., 2008] é um assistente de reunião, que fornece informações automáticas de reuniões utilizando, dentre outras técnicas, o uso de reconhecimento de fala para legendagem, geração de ata e busca por conteúdos de conversas.

Também podemos citar aplicações diretas para sistemas de reconhecimento de fala, como os esforços de alguns trabalhos [Raimond and Lowis, 2012; Coelho and Souza, 2015; Yang and Meinel, 2014] que utilizam estes sistemas para auxiliar na anotação, busca e recomendação de vídeos. Este processo utiliza o texto de resultado da transcrição, como entrada para algoritmos de anotação semântica na intenção de identificar termos para classificar o conteúdo multimídia. Estes termos podem ser relacionados a bases de conhecimento e desta forma criar uma estrutura de dados ligados podendo assim, utilizar técnicas para recomendação de conteúdos. Assim, pode-se perceber que os cenários para aplicação de sistemas de reconhecimento de fala

são amplos e que este tipo de sistema pode auxiliar tanto na automatização de tarefas quanto na extração de informações para aplicações.

5.5. Conclusões

Neste trabalho, apresentamos a arquitetura básica de sistemas de reconhecimento automático de fala, descrevendo os principais elementos, funcionalidades e métodos para treinamento e criação de sistemas ASR. Também apresentamos aplicações diretas e indiretas para estes sistemas com o objetivo de demonstrar como estes sistemas se comportam nos mais diversos contextos.

O reconhecimento de fala é um tema recorrente na área de processamento de linguagem natural e conta com uma comunidade ativa e em constante processo de melhorias paras os diversos elementos desses sistemas. Assim, o reconhecimento de fala é um tema promissor e pode ser aplicado a diversos problemas, principalmente da área de multimídia.

Referências

Alumäe, Tanel. "Full-duplex Speech-to-text System for Estonian." Baltic HLT. 2014.

Arlindo Oliveira da Veiga. Treino não supervisionado de modelos acústicos para reconhecimento de fala. Ph.D. Dissertation. Universidade de Coimbra. 2013.

Bengio, Yoshua. "Markovian models for sequential data." Neural computing surveys 2.199 (1999): 129-162.

Campos, Virginia Pinto, T. M. U. Araujo, and G. L. Souza Filho. "CineAD: Um Sistema de Geração Automática de Roteiros de Audiodescrição." Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia) (2014).

Chan, Ho Yin, and Phil Woodland. "Improving broadcast news transcription by lightly supervised discriminative training." Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on. Vol. 1. IEEE, 2004.

Coelho, S. A. ; Souza, J. F. . Anotação semântica de transcritos para indexação e busca de vídeos. In: Conferência Ibero Americana WWW/INTERNET, 2014, Porto, Portugal. 12ª Conferência Ibero Americana WWW/INTERNET, 2014. v. 1. p. 51-58.

Cuadros, Carlos Daniel Riquelme. "Reconhecimento de voz e de locutor em ambientes ruidosos: comparação das técnicas MFCC e ZCPA." Dissertação de mestrado. Programa de Pós-Graduação em Engenharia de Telecomunicações da Escola de Engenharia da Universidade Federal Fluminense. (2007).

Dahl, George E., et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." IEEE Transactions on audio, speech, and language processing 20.1 (2012): 30-42.

Davis, Steven, and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences." IEEE transactions on acoustics, speech, and signal processing 28.4 (1980): 357-366.

de Sá Pessoa, Luis A., Fábio Violaro, and Plínio A. Barbosa. "Modelos da Língua Baseados em Classes de Palavras para Sistema de Reconhecimento de Fala Contínua 10.14209/jcis. 1999.10." Journal of Communication and Information Systems 14.2 (1999).

Evermann, Gunnar, et al. "Training LVCSR systems on thousands of hours of data." Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on. Vol. 1. IEEE, 2005.

Gonzalez, Marco, and Vera LS Lima. "Recuperação de informação e processamento da linguagem natural." XXIII Congresso da Sociedade Brasileira de Computação. Vol. 3. 2003.

Hämäläinen, Annika, et al. "A Multimodal Educational Game for 3-10-year-old Children: Collecting and Automatically Recognising European Portuguese Children's Speech." Speech and Language Technology in Education. 2013.

Higgins, Eleanor L., and Marshall H. Raskind. "Speaking to read: The effects of continuous vs. discrete speech recognition systems on the reading and spelling of children with learning disabilities." Journal of Special Education Technology 15.1 (1999): 19-30.

Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." IEEE Signal Processing Magazine 29.6 (2012): 82-97.

Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics". 2nd edition. Prentice-Hall. 2009.

Jurafsky, Dan. "Language Modeling". Natural Language Processing Group of Stanford University (2017). Disponível em https://web.stanford.edu/~jurafsky/ . Acesso em 3 de Setembro de 2017. Jurafsky

Katz, Slava. "Estimation of probabilities from sparse data for the language model component of a speech recognizer." IEEE transactions on acoustics, speech, and signal processing 35.3 (1987): 400-401.

Meinedo, Hugo, et al. "AUDIMUS. media: a Broadcast News speech recognition system for the European Portuguese language." Computational Processing of the Portuguese Language (2003): 196-196.

Miao, Yajie. "Kaldi+ PDNN: building DNN-based ASR systems with Kaldi and PDNN." arXiv preprint arXiv:1401.6984 (2014).

Neto, Nelson, Ênio Silva, and Erick Sousa. "Software usando reconhecimento e síntese de voz: o estado da arte para o Português brasileiro." Proceedings of the 2005 Latin American conference on Human-computer interaction. ACM, 2005.

Ney, Hermann, and Ute Essen. "On smoothing techniques for bigram-based natural language modelling." Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on. IEEE, 1991.

Panayotov, Vassil, et al. "Librispeech: an ASR corpus based on public domain audio books." Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015.

Povey, Daniel, et al. "The Kaldi speech recognition toolkit." IEEE 2011 workshop on automatic speech recognition and understanding. No. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

Rabiner, Lawrence, and B. Juang. "An introduction to hidden Markov models." ieee assp magazine 3.1 (1986): 4-16.

Raimond, Yves, and Chris Lowis. "Automated interlinking of speech radio archives." LDOW 937 (2012).

Riccardi, G.; Hakkani-Tür, D. Z. Active and unsupervised learning for automatic speech recognition. In: INTERSPEECH, 2003.

Robert M Ochshorn and Max Hawkins. 2017. “Gentle forced aligner [computer program]. (2017). https://lowerquality.com/gentle/ Access date: 25 ago. 2017.

Rosenfeld, Roni, and Philip Clarkson. "Statistical language modeling using the CMU-Cambridge toolkit." (1997).

Silva, E., et al. "Modelos de linguagem n-grama para reconhecimento de voz com grande vocabulário." III workshop em tecnologia da informação e da linguagem humana. 2004.

Silva, E., et al. "Desenvolvimento de um sistema de reconhecimento automático de voz contínua com grande vocabulário para o Português Brasileiro." XXV congresso da sociedade Brasileira de computação. 2005.

Silva, Carlos Patrick Alves da. Um software de reconhecimento de voz para português brasileiro. MS thesis. Universidade Federal do Pará, 2010.

Siravenha, Ana, et al. "Uso de regras fonológicas com determinação de vogal tônica para conversão grafema-fone em português brasileiro". 7th International

Information and Telecommunication Technologies Symposium. 2008.

Stolcke, Andreas. "SRILM-an extensible language modeling toolkit." Interspeech. Vol. 2002. 2002.

Tevah, Rafael Teruszkin. Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro. Diss. Dissertação (mestrado). COPPE/UFRJ, M. Sc., Engenharia Elétrica, 2006.

Tur, Gokhan, et al. "The CALO meeting speech recognition and understanding system." Spoken Language Technology Workshop, 2008. SLT 2008. IEEE. IEEE, 2008.

Wessel, Frank, and Hermann Ney. "Unsupervised training of acoustic models for large vocabulary continuous speech recognition." IEEE Transactions on Speech and Audio Processing 13.1 (2005): 23-31.

Witten, Ian H., and Timothy C. Bell. "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression." Ieee transactions on information theory 37.4 (1991): 1085-1094.

Yang, Haojin, and Christoph Meinel. "Content based lecture video retrieval using speech and video text information." IEEE Transactions on Learning Technologies 7.2 (2014): 142-154.

Ynoguti, Carlos Alberto. "Reconhecimento de fala contínua usando modelos ocultos de Markov.". Ph.D. Dissertation. Universidade Estadual de Campinas. 1999.

Young, S., et al. "The HTK book (v3. 4)." Cambridge University (2006).

Biografia Resumida dos Autores

Jairo Francisco de Souza é graduado em Ciência da Computação (UFJF), possui mestrado em Engenharia de Sistemas e Computação (UFRJ) e Doutorado em Informática (PUC-RJ). Atualmente é professor adjunto da Universidade Federal de Juiz de Fora e docente do programa de pós-graduação em Ciência da Computação da UFJF, onde atua em pesquisas nas áreas de recuperação de informação, processamento de linguagem natural e integração semântica.

Marcos Valadão Gualberto Ferreira é graduando do oitavo período do curso de Ciência da Computação na Universidade Federal de Juiz de Fora (UFJF) e estagiário da Rede Nacional de Ensino e Pesquisa (RNP). Atua há quase dois anos no projeto de Busca Avançada por Vídeos baseada em transcrição de áudio, metadados, anotação semântica e recomendação (GT-BAVi).