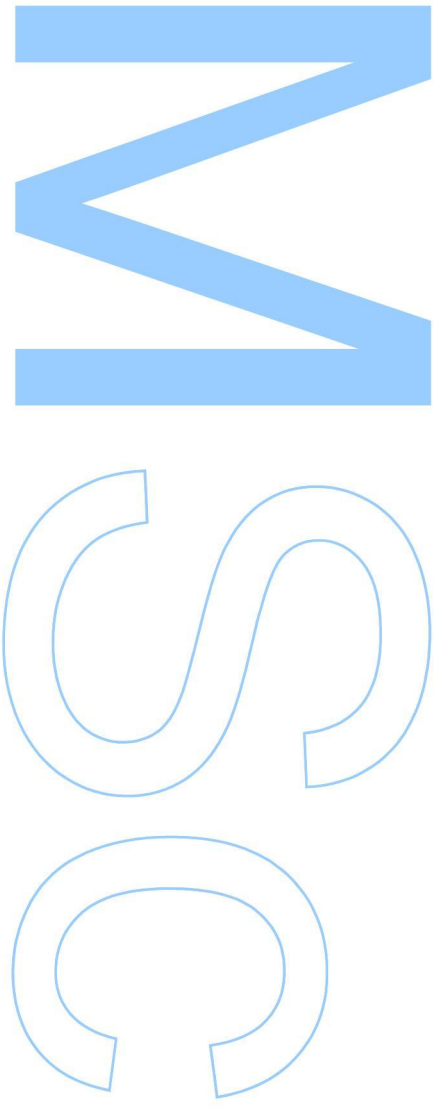
Reconhecimento de fala em português de Portugal num contexto com poucos recursos

João Manuel Alves Mourão de Sá
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciências de Computadores
2021

Orientador
Inês de Castro Dutra, Professora Auxiliar,
Faculdade de Ciências da Universidade do Porto



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

A speech recognition system converts human speech into text. Traditional speech recognition systems are comprised of several machine learning models. In the last decade, the focus of speech recognition research has shifted to the study of Deep Neural Networks architectures able to convert speech to text through a single model, in an end-to-end fashion. These new models require large amounts of data and computational resources. Portuguese is an under-resourced language, subsequently the performance of these new end-to-end (E2E) approaches is not well known in European Portuguese. Recently, cross-Language Transfer Learning techniques have been proven very effective in mitigating the high resource requirements for training E2E networks.

In this thesis, we compare the not so well known performance of E2E systems with that of traditional systems in the task of recognizing European Portuguese speech. To that end, we aggregated and processed two recent datasets; Europarl_st (E), with 32.8h of parliamentary sessions and 40 distinct voices, and Multilingual_TEDx (T) with 21.3h of TEDx talks and 109 voices in European Portuguese. These datasets were used to create traditional ASR systems and E2E systems. The E2E models are QuartzNet models pre-trained with 1000 hours of English speech, and subsequently retrained with the Portuguese datasets. We conducted a set of experiments to compare both architectures in the task of recognizing portuguese speech in the same domains in which the models were created (in-domain validation), and also in the other domain (out-of-domain validation). Additionaly, the results were also compared with the performance provided by the Google speech-to-text solution in transcribing the same datasets.

The results show that in such a low resourced scenario, the hybrid systems outperform the E2E solutions: in domain E, the hybrid system attained a word error rate (WER) of 12.32%, in contrast with the 15.47% attained by the E2E solution. In this dataset, the commercial Google Speech to text solution performed poorly, attaining a WER of 42.38%. These results demonstrate that it's viable to build an ASR system in European Portuguese with an acceptable performance, even if trained with very few hours, as long as the variability is close to the domain of aplicability. The systems didn't generalize to data outside the domain used for training (out-of-domain validation), meaning they wouldn't perform well in real life scenarios. This result can be explained by the limited data and variability present in the datasets. In the future, this problem could be solved by creating a new dataset with more data. Or perhaps trying other solutions, such as new data augmentation techniques.

Resumo

Sistemas de reconhecimento de fala têm como objetivo a conversão da fala humana em texto. Tradicionalmente, estes sistemas são compostos por vários modelos, treinados através de aprendizagem automática. Na última década, o foco da investigação em reconhecimento de fala (ASR) moveu-se para o estudo de arquitecturas baseadas em redes neuronais profundas que permitam a conversão da fala humana em texto de forma ponta-a-ponta (E2E). O treino destas soluções E2E exige quantidades muito elevadas de dados, fazendo com que a performance destas novas abordagens E2E seja pouco conhecida na língua portuguesa. Recentemente, técnicas de *Transfer Learning* mostraram-se eficazes para mitigar a elevada necessidade de recursos necessários para o treino de redes E2E. Estas técnicas baseiam-se no re-treino de uma rede pré-treinado com muitos dados numa língua, com dados de treino numa língua diferente.

Nesta tese, comparou-se a performance pouco conhecida de sistemas E2E com a dos sistemas tradicionais no reconhecimento de português de Portugal. Para esse efeito, processaram-se dois *datasets* recentes; Europarl_st (E), com 32.8h de sessões parlamentares e 40 vozes distintas, e Multilingual_TEDx (T) com 21.3h de apresentações de TEDx, e 109 vozes em português. Estes *datasets* serviram de base para a criação de sistemas de ASR tradicionais e E2E, estes últimos recorrendo a *Transfer Learning* a partir de um modelo QuartzNet pré-treinado com 1000 horas em inglês. Realizou-se um conjunto de experiências para comparar a performance das duas abordagens no reconhecimento de fala nos domínios em que foram criados (validação no domínio), e na capacidade de generalização para outros domínios (validação fora do domínio). Os resultados foram comparados com a performance da Google speech-to-text nos mesmos *datasets*.

Os resultados demonstram que num cenário de poucos recursos, os sistemas híbridos apresentam um desempenho superior ao dos modelos E2E: no domínio E, o sistema híbrido obteve uma percentagem de palavras erradas (WER) de 12.32%, inferior aos 15.47% da solução E2E. Nesse *dataset*, o Google Speech-to-text obteve um WER muito superior: 42.38%. Este resultado indica que é possível criar-se um sistema com um desempenho aceitável, treinado com um número reduzido de horas, desde que a variabilidade de vozes esteja próxima do domínio de aplicabilidade. Os sistemas obtiveram um mau desempenho quando confrontados com dados de outros domínios. Este resultado é explicável pelo número reduzido de horas e de variabilidade dos *datasets*, remetendo a aplicabilidade destes sistemas para os domínios a partir dos quais foram treinados. No futuro, este problema seria solucionado através da criação de *datasets* de maiores dimensões ou, talvez, através da incorporação de outras técnicas, como *data augmentation*.

Agradecimentos

À professora Inês Dutra pela disponibilidade e apoio.

À minha família e amigos que comigo escreveram esta página da minha vida.

Dedicado à minha família

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

ASR	Automatic Speech Recognition
ANN	Artificial Neural Network
CTC	Connectionist Temporal Classification
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
E2E	End-to-end
FFNN	Feedforward Neural Network
GMM	Gaussian Mixture Model
G2P	Grapheme-to-phoneme
HMM	Hidden Markov Model
MFCC	Mel Frequency Cepstral Coefficient
ML	Machine Learning
MLE	Maximum Likelihood Estimation
NLP	Natural Language Processing
LM	Language Model
OOV	Out-of-vocabulary
RNN	Recurrent Neural Network
STT	Speech-to-text
TDNN	Time Delay Neural Network
TL	Transfer Learning
WER	Word Error Rate
WFST	Weighted Finite-state Transducer

Capítulo 1

Introdução

Reconhecimento de fala automático, ou *Automatic Speech Recognition* (ASR), é uma área de processamento de linguagem natural que trata do estudo de modelos matemáticos que consigam converter a fala humana em texto. Este processo tem também o nome de conversão *speech-to-text* (STT) [9] .

Sistemas de ASR tradicionais dividem a tarefa de reconhecimento de fala em várias sub-tarefas. A primeira das quais tem o nome de extracção de atributos (*features*), e tem como objetivo converter o sinal acústico da fala num conjunto de observações capazes de discretizar os diferentes sons que foram pronunciados. Estas observações são dadas como entrada num descodificador, que recorre a vários modelos criados através de técnicas de aprendizagem automática para descodificarem a sequência de palavras com maior probabilidade de corresponder à sequência de observações.

Na última década, têm vindo a ser sugeridas abordagens baseadas em redes neuronais que procuram aprender um mapeamento direto entre as observações acústicas e as sequências de palavras mais prováveis. Estas abordagens, chamadas de *end-to-end* (E2E), ou ponta-a-ponta, simplificam o processo de criação de sistemas de ASR, dado que apenas requerem a criação de um único modelo, em vez de vários. Uma das desvantagens destas novas abordagens E2E, é que para proporcionarem um desempenho satisfatório, precisam de ser treinadas com uma quantidade muito mais elevada de dados de treino [10].

A aplicação de tecnologias de ASR é útil em várias situações. Por exemplo, para a transcrição de anotações médicas a pacientes com incapacidades auditivas. Nos últimos anos, têm vindo a ser desenvolvidas várias soluções comerciais que disponibilizam serviços de ASR. A maioria delas operam na *cloud*, como a *Google cloud speech-to-text*, ou o *Microsoft STT*. Estas soluções, para além de dispendiosas, impõem o envio das gravações para um servidor terceiro, onde é feito o processamento do áudio, e a partir do qual as transcrições são devolvidas para o cliente final. Subsequentemente, não são adequadas para o uso em contextos hospitalares, ou noutros contextos em que a preservação da privacidade seja crucial. Nestes casos, a criação de um sistema de ASR em português de Portugal *in-house*, que opere *offline*, afigurar-se-ia muito mais adequada.

A língua portuguesa, embora seja a sexta língua mais falada no mundo, é aquilo o que se pode considerar de *under-resourced language* [1]. Ou seja, uma língua com poucos recursos linguísticos que permitam o desenvolvimento de áreas afetas a processamento de linguagem natural. No contexto de ASR, estes recursos manifestam-se na forma de *corpus* paralelos de fala, constituídos por gravações de voz e nas respetivas transcrições em texto.

Este problema acentua-se ainda mais quando o foco se move para a variante de português de Portugal. Neste caso, a quantidade de recursos de acesso livre é ainda mais reduzida. Os poucos recursos existentes são, ou muito caros, ou feitos por investigadores que não os disponibilizam publicamente. Consequentemente, o desempenho das novas abordagens E2E não é muito conhecida no reconhecimento de português de Portugal.

Transfer learning (TL) é uma solução usada em aprendizagem automática que se define pela aplicação do conhecimento adquirido na resolução de um problema para a resolução de um problema semelhante. Em ASR, TL pode ser usada para transferir o conhecimento adquirido por um modelo no reconhecimento de fala numa língua, para o reconhecimento de fala numa língua diferente, num processo chamado de *cross-Language Transfer Learning*. Recentemente, trabalhos como aquele proposto por Huang et al. [8] demonstram a eficácia desta solução para mitigar um dos principais problemas inerentes à criação de um sistema de ASR E2E: falta de recursos linguísticos que possam ser usados como treino. A abordagem usada prende-se na adaptação de um modelo pré-treinado com muitas horas de gravação em inglês para uma língua diferente com menos recursos.

O objetivo desta tese consiste em comparar o desempenho proporcionado por um sistema de ASR E2E treinado via *cross-Language Transfer Learning* a partir de um modelo pré-treinado com 1000h na língua inglesa, com aquele que é proporcionada pelas soluções híbridas tradicionais, no reconhecimento de português de Portugal. Para o treino dos modelos, recorreu-se a dois *datasets* públicos muito recentes; Europarl_st [11], com 32.8 horas de sessões parlamentares, e Multilingual_TEDx [12], com 21.3 horas de apresentações de TEDx.

As seguintes questões de investigação serão abordadas neste trabalho:

1. Qual o desempenho de um sistema de ASR no reconhecimento de português de Portugal, treinado com uma quantidade muito limitada de recursos?
2. Qual a performance de um sistema de ASR tradicional e o de um sistema E2E treinado com *cross-Language Transfer Learning* no reconhecimento de português de Portugal?
3. Qual a capacidade das duas arquitecturas em reconhecerem dados imprevistos (*out-of-domain validation*)?

Organização

O Capítulo 2 faz uma exposição de alguns conceitos e tecnologias importantes e que estão subjacentes à construção de sistemas de ASR, e que são de relevo para a compreensão dos capítulos que se seguem. No Capítulo 3 é feito um levantamento de artigos, *datasets*, e soluções comerciais influentes em ASR. O Capítulo 4 descreve a metodologia e os passos tomados para a construção dos sistemas de ASR. O Capítulo 5 apresenta os testes efectuados para comparar os vários sistemas no reconhecimento de fala em português de Portugal. No Capítulo 6 é feita uma análise dos resultados obtidos por via das experiências realizadas, e a partir dos quais será possível responder às questões de investigação. O Capítulo 7 procura apresentar as principais ilações que foram possíveis de retirar com a elaboração desta tese, deixando também algumas sugestões de trabalhos futuros que possam servir de complemento a este trabalho.

Capítulo 2

Contexto teórico

Reconhecimento automático de fala, ou *Automatic Speech Recognition* (ASR) consiste na transcrição automática de sinais acústicos respeitantes à fala humana. Povey [13] descreve este conceito como um problema de procura por uma função $F(X)$ que consiga fazer o mapeamento de sinais acústicos X em sequências de texto W, com a máxima eficácia possível. Geralmente, a qualidade de F é aferida através da aplicação de $F(.)$ para um conjunto de gravações X, obtidos a partir de um conjunto de dados de testes, e calculo da taxa de erro resultante.

Encontrar a função F(.) é uma tarefa complexa, que envolve o estudo de áreas tão diversas como linguística, aprendizagem automática, estatística, ou processamento de sinal. Neste capítulo, é feito um enquadramento teórico de alguns dos conceitos mais importantes em ASR.

Fonética

A fonética é um ramo da linguística que procura classificar os diferentes sons que os humanos produzem durante a fala. Estes sons têm o nome de fonemas, e distinguem os sons que representam a forma de como as palavras são pronunciadas. Ao contrário das palavras, os fonemas não têm nenhum significado associado, e são universais à fala humana. A substituição de um fonema por outro, numa palavra, pode levar à transformação dessa palavra por outra.

p	f	m	R	O	6∼
b	v	n	i	o	o∼
t	s	J	e	u	u∼
d	z	l	E	@	aw
k	S	L	a	i∼	aj
g	Z	r	6	e∼	6∼j∼

Tabela 2.1: Fonemas característicos da língua portuguesa com acentuação de Portugal, no formato fonético SAMPA.

Por exemplo, a palavra cor pode ter dois significados: coloração ou memória, dependendo da forma de como é pronunciada - se se pronunciar o segundo fonema como /O/ ('ó', no alfabeto fonético SAMPA[1]), a palavra adquire o significado de memória, mas se se pronunciar /o/ ('ô') a palavra adquire o significado de coloração.

As palavras são mapeadas na sua representação fonética através de um dicionário fonético, que deverá conter uma entrada para cada palavra existente no vocabulário, incluindo várias entradas para palavras homógrafas que, escrevendo-se da mesma forma, são pronunciadas de forma diferente. Por exemplo, as palavras cor e colher, que adquirem diferentes significados (e pronúncias), consoante a sua utilização como nome ou como verbo.

dialética	d j 6 l E k t i k 6
dialético	d j 6 l E k t i k u
dialecto	d j 6 l E k t u
dialectos	d j 6 l E k t u S

Tabela 2.2: Exemplo de algumas entradas num dicionário léxico no formato SAMPA.

Aprendizagem automática

Aprendizagem automática, ou *Machine Learning* (ML), é uma área do ramo de Inteligência Artificial que visa o estudo de algoritmos que consigam aprender modelos treinados através de amostras de dados (dados de treino). Um modelo pode ser descrito como uma representação matemática mais abstrata de um fenómeno real. ML pode ser formulada como a optimização de uma função objetivo que seja capaz de generalizar para amostras de dados desconhecidas. A esta função objetivo é também dado o nome de função de custo quando o processo de optimização assenta na minimização do seu resultado.

Um modelo estatístico é uma distribuição de probabilidades construída com a finalidade de permitir a realização de inferências ou de decisões através dos dados. Uma distribuição normal, ou Gaussiana, é um exemplo simples de uma distribuição cuja forma é definida por dois parâmetros: a média e a variância. Nem todos os fenómenos seguem uma distribuição normal; nessas situações, é necessário procurar uma outra função que se afigure mais apropriada para modelar o fenómeno em causa. Essa função, chamada de função objetivo, é encontrada através de um processo de optimização.

Sistemas de ASR tradicionais assentam em dois modelos estatísticos: Hidden Markov Models (HMMs) e Gaussian Mixture Models (GMMs). HMMs e GMMs são modelos baseados em variáveis latentes: variáveis que não são diretamente observáveis mas que podem ser inferidas a partir de variáveis observáveis, geradas pelas variáveis escondidas. Em ASR, as sequências de fonemas que compõem as palavras podem ser interpretadas como sequências de variáveis latentes, inferíveis a partir de sequências de variáveis observadas, representadas por observações acústicas (ou atributos acústicos).

[1] https://www.phon.ucl.ac.uk/home/sampa/portug.htm

Mais recentemente, modelos estatísticos têm vindo a ser substituídos por Redes Neuronais Artificiais, ou Artificial Neural Networks (ANNs), para a classificação de observações acústicas em fonemas. ANNs podem ser visualizadas como um grafo dirigido cujos nós correspondem a neurónios artificiais, e em que cada neurónio retorna como saída o resultado de uma função de ativação que toma como entrada o somatório dos neurónios a ele ligados (ponderada pelo peso das respetivas arestas). Geralmente, os neurónios estão agregados em camadas, sendo que os sinais atravessam a primeira camada (*input layer*), até à última camada de saída (*output layer*). Uma rede neuronal profunda, ou Deep Neural Network (DNN), é uma ANN constituída por múltiplas camadas; uma de entrada, uma de saída, e múltiplas camadas intermédias.

Em ASR, o número de neurónios na camada de entrada pode corresponder à dimensão dos vetores de atributos que são dados como entrada na rede, e o número de neurónios na camada de saída ao número de fonemas passíveis de serem classificados. A utilização de uma função de ativação não linear, como *softmax*, permite que a rede consiga aprender uma função não linear capaz de capturar relações não lineares entre as observações acústicas e os fonemas, permitindo que a rede consiga classificar observações acústicas em classes de fonemas com maior eficácia daquela que seria possível através dos métodos estatísticos tradicionais.

Extração de atributos

Em áreas que envolvem o processamento de sinais acústicos, como em ASR, é comum extraírem-se um conjunto de atributos acústicos (*features* acústicas) que permitem caracterizar o sinal de som ao longo do tempo. Este processo consiste na divisão do sinal de áudio em T segmentos, que serão posteriormente convertidos em vetores de atributos, também chamados de observações acústicas: $O = o_1, o_2, ..., o_T$. Estas observações devem ser descritivas o suficiente para que possam fornecer informação útil sobre o sinal, e resilientes o suficiente às várias perturbações que possam surgir no ambiente acústico, como ruídos [14].

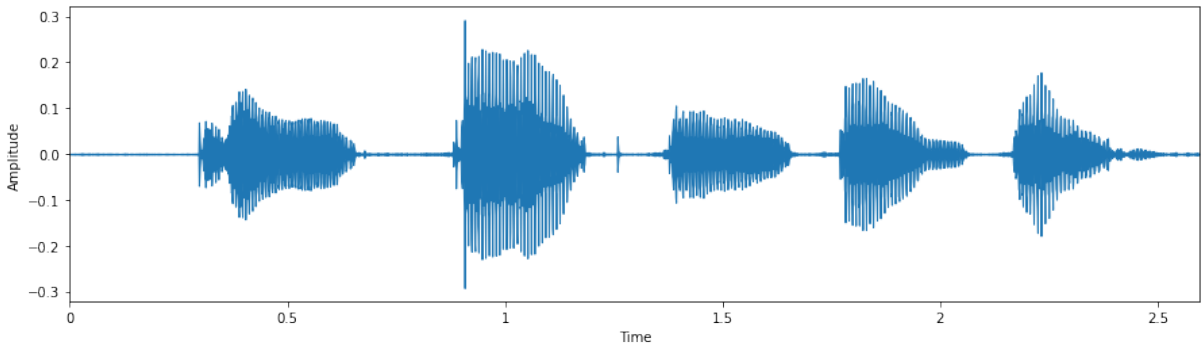


Figura 2.1: Sinal acústico na forma de onda.

O sinal acústico pode ser representado sob a forma de onda. No entanto, esta forma não permite capturar propriedades acústicas com o detalhe necessário para tornar possível identificar os fonemas que foram pronunciados. Para capturar propriedades acústicas com maior resolução, o sinal é convertido para o domínio das frequências, representado na forma de um espectrograma.

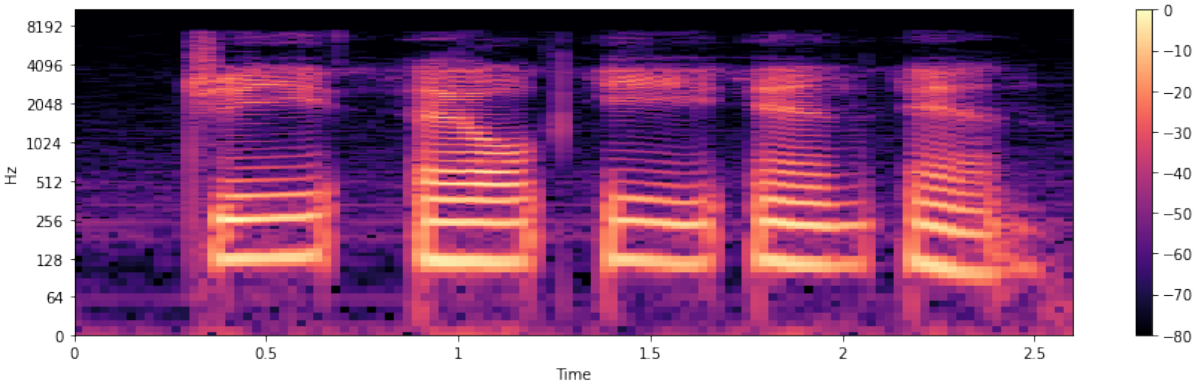


Figura 2.2: Espectrograma do sinal representado na Figura 2.1. O eixo horizontal representa a evolução do sinal ao longo do tempo numa janela temporal de 25ms, e o eixo vertical representa as frequências. A cor distingue a energia de cada frequência em cada momento.

Um espectrograma é um mapa de calor que representa a energia de cada frequência do sinal acústico ao longo de um período de tempo. Alguns sistemas de ASR *end-to-end* (secção 2.3) usam o espectrograma diretamente como entrada.

Vetores de atributos MFCC

Um método muito usado para representar sinais contínuos de áudio em vetores de atributos tem o nome de Mel-Frequency Cepstral Coefficients (MFCCs). O sucesso deste método deve-se, em parte, à sua capacidade em realizar um género de filtragem que se correlaciona com a forma de como os humanos interpretam os sinais acústicos [14].

Para a obtenção de vetores de atributos MFCC, o sinal é segmentado em janelas temporais mais pequenas (tipicamente, de 25ms). De seguida, cada janela é transformada para o domínio das frequências, através da aplicação de uma transformação discreta de Fourier (DFT), que permite obter informação relativa à quantidade de energia em cada gama de frequência.

A forma como os humanos interpretam o som não é linear: é possível distinguir melhor o som em frequências mais baixas do que em frequências mais elevadas. Por exemplo, uma diferença entre 1000Hz e 2000Hz é interpretada pelo sistema auditivo humana como sendo uma diferença maior do que entre 8000Hz e 9000Hz. Por esse motivo, a energia extraída usando a DFT é mapeada numa *Mel scale*, que consiste num conjunto de banco de filtros Mel que permitem que as frequências fiquem espaçadas na forma de como elas soam na realidade, ao ouvido humano.

A partir de cada um dos filtros, é aplicada uma Discrete Cosine Transform (DCT), que procura descorrelacionar os parâmetros espectrais. Esta transformação permite separar a informação relativa à fonte de excitação (*pitch*), do trato vocal (filtro), que corresponde à configuração exata dos articuladores como a língua ou os dentes.

Para a extração de MFCCs são, geralmente, considerados os 12 primeiros coeficientes

cepstrais, que representam informação sobre o filtro, que é mais útil para a detecção dos fonemas pronunciados. É também identificado um 13º atributo, a energia do *frame*, que também se correlaciona com os fonemas.

O discurso não é constante de *frame* para *frame*. Alterações como paragem gradual ou abrupta ajudam a identificar os fonemas. Assim, para cada um dos atributos cepstrais, podem ser adicionadas dois outros atributos: delta (ou velocidade), e delta-delta (aceleração), que contém informação relativa à alteração temporal entre os coeficientes cepstrais de dois *frames* consecutivos. O cálculo destes conjuntos de atributos é feito via a aplicação da primeira e da segunda derivada sobre os atributos MFCC.

Os atributos MFCC podem ser posteriormente transformados, por forma a torná-los mais robustos face a influências acústicas indesejáveis, como ruídos ou reverberações [15]. Uma transformação comum, neste contexto, tem o nome de Cepstral Mean and Variance Normalization (CMVN) [16], que normaliza os valores dos MFCCs por forma a terem média zero e uma variância unitária. A normalização baseia-se em estatísticas obtidas a partir de cada frase que foi pronunciada, ou através do conjunto de frases pronunciadas por cada pessoa.

2.1 Reconhecimento de fala

O objetivo de ASR consiste em descodificar sequências de observações (O) em sequências de palavras. A sequência de palavras mais provável W' pode ser estimada pela maximização de $P(W|O)$ para todas as possíveis sequências de palavras W no vocabulário V [14]. Esta procura pode ser formulada via:

$$W' = \arg\max_{W \in \mathcal{V}} P(W|O) \tag{2.1}$$

Onde $W = w_1, w_2, ..., w_n$ é uma sequência de palavras, $O = o_1, o_2, ..., o_m$ uma sequência de observações e V o conjunto de palavras admitidas na procura (vocabulário). A equação (2.1) pode ser expandida usando a regra de Bayes, obtendo-se:

$$W' = \arg\max_{W \in \mathcal{V}} \frac{P(O|W)P(W)}{P(O)} \tag{2.2}$$

Para se obter a sequência de palavras mais provável, o objetivo prende-se com maximização do numerador, pelo que $P(O)$ no denominador pode ser removido:

$$W' = \arg\max_{W \in \mathcal{V}} P(O|W)P(W) \tag{2.3}$$

Desta forma, ASR assenta na maximização do produto de dois componentes:

1. $P(W)$, que representa conhecimento à priori sobre a sequência de palavras W, e pode ser determinado através de um modelo de linguagem, obtido através de estatísticas relativas a frequências de palavras presentes num *corpus* de texto. Por exemplo, caso o texto deste capítulo fosse usado para a construção de um modelo de linguagem, a probabilidade atribuída à frase "Reconhecimento de fala" seria maior que a que seria atribuída à frase "Conhecimento de fala", dado que a primeira ocorre com maior frequência no *corpus*.

2. $P(O|W)$, que descreve a distribuição sobre observações acústicas O dada a sequência de palavras W, e é avaliada com recurso a um modelo acústico.

Tanto a modelação acústica como a modelação de linguagem são feitas com recurso a aprendizagem automática, via modelos treinados com dados de treino. Para a modelação acústica, é necessário um *corpus* paralelo com gravações de fala e das respetivas transcrições. Para a modelação de linguagem, apenas é necessário um *corpus* de texto. As abordagens usadas para a criação de sistemas de ASR têm evoluído ao longo dos anos. Hoje em dia é possível distinguir duas abordagens principais:

1. Métodos mais antigos, ditos tradicionais, e que consistem em abordagens híbridas que dependem da modelação de vários modelos em separado; por exemplo, um modelo acústico e um modelo de linguagem. Matematicamente, estes métodos são caracterizados pelo modelo generativo expresso na equação (2.3).

2. Métodos mais recentes, também chamados de *end-to-end*, que recorrem a *Deep Learning* para descodificar o texto diretamente do sinal acústico. Estes métodos são expressos matematicamente pela equação (2.1).

2.2 Métodos híbridos

Num sistema de ASR híbrido tradicional, o problema é apresentado como uma procura pela transcrição W com maior probabilidade de corresponder ao sinal de *input* O. Para esse processo, intervêm quatro módulos; um modelo acústico, um modelo de pronúncia, um modelo de linguagem, e um descodificador.

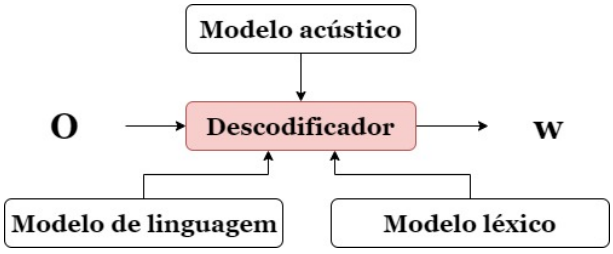


Figura 2.3: Componentes presentes num sistema de ASR híbrido clássico.

2.2.1 Modelo acústico

O modelo acústico tem como objetivo prever a sequência de fonemas através dos atributos acústicos, e é aprendido através de um conjunto de gravações de áudio e das respetivas transcrições. Tradicionalmente, os modelos acústicos são constituídos pela combinação de Hidden Markov Models com Gaussian Mixture Models (modelos HMM/GMM) e, mais recentemente, pela combinação de HMMs com Deep Neural Networks (modelos HMM/DNN).

2.2.1.1 Hidden Markov Models (HMMs)

Cadeias de Markov são frequentemente usadas para tarefas que envolvem a modelação de processos sequenciais [17]. Numa cadeia de Markov, a probabilidade de se estar num estado s_i, num qualquer tempo t, depende apenas dos k estados anteriores, e não da sequência inteira desde o tempo 1 até ao tempo $t-1$:

$$P(s_i|s_{t-1}, s_{t-2}, ..., s_{t_1}) = P(s_i|s_{t-1}, s_{t-2}, ..., s_{t-k}) \tag{2.4}$$

A cadeia de Markov mais simples depende apenas do estado mais recente ($k = 1$), e é representada por $P(s_i|s_{t-1})$.

Por vezes, os estados de uma cadeia de Markov estão escondidos, ou não são observáveis, mas produzem efeitos que são observáveis. Estas cadeias de Markov podem ser modeladas através de Hidden Markov Models (HMMs), compostos por uma cadeia de Markov de estados escondidos (variáveis latentes) S, e por uma sequência de observações O. Num HMM, cada observação o_t tem uma probabilidade $b_i(o_t)$ de ter sido gerada por um estado escondido s_i.

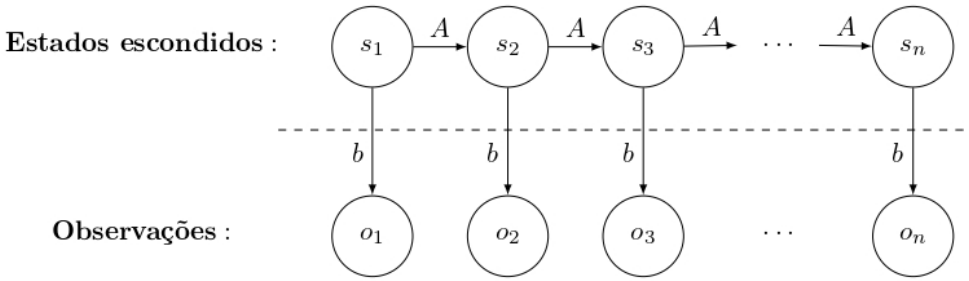


Figura 2.4: Hidden Markov Model

No contexto de ASR, a fala pode ser modelada através de HMMs, sendo que a sequência de estados escondidos corresponde à sequência de fonemas que desejamos descodificar a partir de observações acústicas (e.g., vetores de atributos MFCC).

Um HMM é definido através de um tuplo $M = (A, B, \pi)$ [14] , onde:

- A: corresponde a uma matriz de transição (NxN), que contém todas as probabilidades de transição entre estados escondidos.

- B: corresponde a uma matriz (MxN) com todas as probabilidades de emissão; e.g., a probabilidade de se observar um estado o_k, dado o estado escondido s_i.
- π: vetor de dimensão N que representa as probabilidades iniciais dos estados escondidos.

Onde M é o número de estados observáveis, e N o número de estados escondidos.

Em ASR, para se estimar as verosimilhanças das observações acústicas dados os fonemas, é necessário criar-se um estado HMM para cada um dos fonemas. A topologia HMM mais frequente para representar um fonema é um modelo *left-to-right*, também chamado de *Bakis model* . Nesta topologia, cada estado contém duas transições: um lacete, e uma transição para o estado seguinte [3].

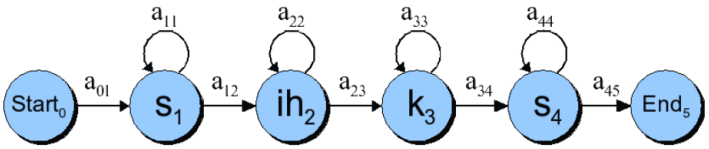


Figura 2.5: HMM corespondendo à palavra inglesa *six*, no formato Bakis [3].

Os mesmos fonemas podem soar de forma diferente consoante os fonemas que os sucedem ou precedem (contexto fonético), fazendo com que a representação ilustrada na Figura (2.6) seja insuficiente para representar as palavras corretamente. Por esse motivo, informação contextual é também modelada, tomando-se em consideração os fonemas que precedem e sucedem o fonema em questão. Estes fonemas "dependentes de contexto" à esquerda e à direita têm o nome de *triphones*. Com a utilização de uma topologia baseada em *triphones*, cada estado do HMM diz respeito a uma parte de um fonema; inicio, meio ou fim.

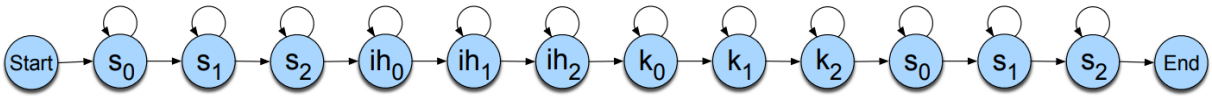


Figura 2.6: HMM corespondendo à palavra inglesa *six*, usando *triphones* [3].

Depois de se construir um HMM para cada fonema, os parâmetros dos modelos são estimados a partir de dados de treino. Cada um destes HMMs gera verosimilhanças dadas as observações que foram observadas (probabilidades de emissão), que podem ser usadas para mapear observações em fonemas. Por outras palavras, uma observação é uma função probabilística do estado escondido. Desta forma, a tarefa de aprendizagem consiste em estimar os parâmetros dos HMMs de forma a que cada um deles modele os fonemas associados de forma tão precisa quanto possível.

Num modelo acústico HMM/GMM, as probabilidades de emissão dos estados HMM, $P(o_t|s_t)$, são representadas através de funções de densidade, modeladas com Gaussian Mixture Models (GMMs).

2.2.1.2 Gaussian Mixture Models

Gaussian Mixture Models permitem modelar as distribuições de probabilidades dos vetores de atributos MFCC, extraídos de cada *frame* de áudio, corresponderem aos diferentes fonemas.

Uma distribuição Gaussiana é modelada por dois parâmetros, a média e a variância. Como cada observação é representada por um vetor com múltiplas dimensões (e.g., 39 dimensões), é feita uma generalização de um modelo Gaussiano com uma só dimensão (*univariate Gaussian distribution*) para um modelo com múltiplas dimensões (*multivariate Gaussian distribution*).

Aspectos como a entoação, género ou idade podem tornar a utilização de *multivariate Gaussian distribution* demasiado restritiva. Para resolver este problema, são usadas GMMs, que permitem que as distribuições possam ser multimodais:

$$p(x|\lambda) = \Sigma_{i=1}^{M} w_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{2.5}$$

A equação (2.5) representa um GMM. Corresponde à soma ponderada de M funções Gaussianas, onde onde w_i corresponde aos coeficientes de mistura (*mixing coefficints*), tal que $\Sigma w_i = 1$.

As probabilidades de transição e de emissão de modelos HMM/GMM são estimados a partir de dados de treino compostos por um *corpus* paralelo com gravações e as respetivas transcrições. O método usado para treino assenta na estimativa por máxima verosimilhança (MLE), que procura maximizar a verosimilhança de se observar os dados de treino, tendo em conta os parâmetros actuais do modelo. Em modelos que incluem variáveis latentes, a MLE pode ser calculada através de um algoritmo iterativo de *expectation-maximization*.

O algoritmo de Viterbi é um algoritmo de *expectation-maximization* frequentemente usado para estimar a MLE dos parâmetros de modelos HMM/GMM, num processo a que é dado o nome de alinhamento forçado de Viterbi [14]. Durante o alinhamento, o algoritmo procura alinhar as observações acústicas com os estados HMM, seguindo-se uma fase em que é feita a re-estimação dos parâmetros de distribuição dos GMMs. Este processo é repetido até os parâmetros convergirem, ou até um número fixo de iterações.

2.2.1.3 Deep Neural Networks

Na última década, DNNs têm sido aplicadas de forma bem sucedida em vários domínios. Em ASR, a utilização de DNNs para a modelação acústica permitiu a obtenção de resultados de estado de arte em várias tarefas de reconhecimento de fala, suplantando GMMs como método mais preciso para a classificação de observações acústicas em classes de fonemas.

Um modelo HMM/DNN é um modelo híbrido em que a componente de modelação acústica é feita através de uma Deep Neural Network.

Um dos motivos por detrás da popularidade de GMMs, devia-se ao facto de serem modelos generativos capazes de modelar diretamente a probabilidade de emissão dos estados HMM ($P(o_t|s_t)$). Na prática, as atributos podem ser fortemente não Gaussianos. DNNs, ao conseguirem aprender funções não-lineares, conseguem classificar observações em fonemas de forma mais precisa.

A formulação apresentada na equação (2.3) é apropriada para a descodificação de sistemas ASR que usam GMMs para a modelação acústica. Ao contrário de GMMs, DNNs são modelos discriminativos que classificam as observações de *input* em classes de fonemas: $P(s|o)$. Como a descodificação é baseada no cálculo das várias verosimilhanças $P(o|s)$, é aplicada a regra de Bayes para produzir pseudo-verosimilhanças, transformando o modelo probabilístico $P(o|s)$ num problema de classificação:

$$\prod_{t=1}^{T} P(o_t|s_t) \propto \prod_{t=1}^{T} \frac{P(s_t|o_t)}{p(s_t)} \tag{2.6}$$

O numerador, é uma DNN treinada para classificar um conjunto de atributos de *input* num estado HMM s_t. O denominador é a probabilidade à priori do estado s_t, e pode ser obtida através de um *corpus* de frequências (e.g., modelo de linguagem) [14].

Recurrent Neural Networks

Feed-Forward Neural Networks (FFNNs) são um sub-género simples de ANNs onde cada neurónio numa camada tem conexões diretas com os neurónios nas camadas subsequentes, sem ciclos na rede. Em ASR, os fonemas estão relacionados com os fonemas que foram pronunciados anteriormente ou posteriormente. Assim, é conveniente utilizar-se uma arquitectura de rede que contenha algum tipo de recursividade entre as várias camadas. Nesse sentido, *Recurrent Neural Networks* (RNNs), afiguram-se como uma solução mais adequada para a classificação de observações acústicas em fonemas, dado que são capazes de modelar dependências entre as observações através da incorporação de ciclos entre os neurónios, criando uma espécie de memória dos estados anteriores da rede.

Time Delay Neural Networks

Time Delay Neural Networks, também conhecidas por TDNNs, são um género de DNNs populares em ASR [4]. Tal como FFNNs, também se organizam em camadas. Uma diferença importante é que as camadas apresentam uma estrutura hierárquica, onde cada neurónio, em cada camada, é dotado do conhecimento do contexto de alguns passos temporais. Este contexto vai-se propagando pelas camadas, de forma a que as unidades na última camada possuem o máximo conhecimento contextual. A capacidade da rede em conseguir capturar informação contextual permite-lhe conseguir classificar as observações em fonemas com maior precisão que aquela proporcionada por uma FFNN. Outra vantagem, é que cada neurónio processa apenas um subconjunto de entrada

(uma janela), sendo que os pesos de cada janela são partilhados dentro da mesma camada. Este processo tem o nome de *sub-sampling*, e permite tornar o treino mais eficiente.

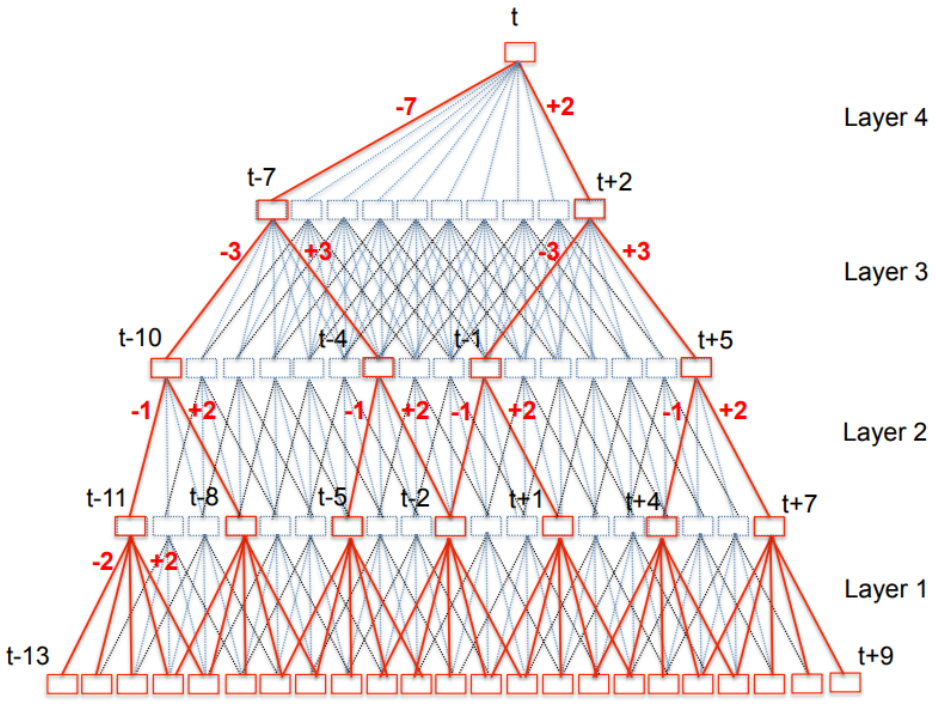


Figura 2.7: Estrutura de uma TDNN. Subsampling marcado a vermelho [4].

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) são um género de ANNs muito populares em visão computacional. Uma arquitectura típica consiste na disposição de várias camadas convolucionais e de *pooling* alternadas entre si, terminando numa camada completamente ligada para classificação [18]. A camada convolucional é constituída por um conjunto de *kernels*, ou filtros, que são usados para a extração de atributos relevantes do *input* através de operações de convolução sobre as entradas, originando *feature maps*. Como as convoluções são operações lineares (produto escalar entre matrizes), e para que a rede consiga aprender mapeamentos mais complexos, é comum aplicar-se uma função de ativação não linear, como ReLU, sobre cada pixel no *feature map*. A camada de *pooling* procura reduzir a dimensionalidade de cada *feature map*, destacando as *features* mais relevantes. CNNs também podem ser usadas em ASR. Nestes casos, é possível utilizar a convolução mais comum (com duas dimensões), construindo-se *feature maps* a partir de uma representação bidimensional do sinal acústico (e.g., o espectrograma). Outra solução consiste no uso de CNNs com uma dimensão. Neste caso, os *feature maps* são vetores (e.g., MFCCs).

2.2.2 Modelo léxico

Uma sequência de estados, descodificada através de um modelo acústico, corresponde a uma sequência de fonemas. No entanto, o objetivo de um sistema ASR consiste em prever sequências de palavras. Uma solução para este problema dá-se através da utilização de um modelo de pronúncia, também chamado de léxico (ou dicionário de pronúncia), que faz o mapeamento de sequências de fonemas nas respetivas palavras.

A criação de um dicionário léxico para uma língua como a portuguesa apresenta vários desafios. A língua portuguesa não segue uma ortografia fonética - não existe um mapeamento de um para um entre as letras (ou grafemas), e os respectivos fonemas. Daqui decorre que os modelos de pronunciação têm que ser criados manualmente, idealmente por linguistas, que façam esse mapeamento na forma de dicionários fonéticos. A Tabela 2 exemplifica algumas entradas de um dicionário fonético em português.

Outro inconveniente prende-se com o tamanho do vocabulário. É impraticável construir-se um dicionário fonético cobrindo todo o vocabulário existente na língua portuguesa; todas as palavras possíveis em todas as variantes (como o género, número, tempo verbal ou pronunciações regionais). Uma possível solução para este problema dá-se através do uso de modelos criados via ML que consigam produzir pronunciações fonéticas para as palavras que não estão presentes no dicionário. Na literatura é possível encontrar referência a modelos que fazem conversão de grafemas em fonemas, chamados de modelos Grapheme-to-phoneme (G2P) [19].

2.2.3 Modelo de linguagem

Um modelo de linguagem estatístico permite definir o vocabulário e quais as sequências de texto que o sistema de ASR pode produzir, tendo por base a probabilidade de cada sequência surgir num *corpus* de texto, usado como treino. A utilização de um modelo de linguagem permite, também, introduzir alguma coerência sintática à sequência de palavras que fora detetada pelos modelos acústicos e de pronunciação. O vocabulário refere-se ao conjunto de unidades textuais, ou *tokens*, usados na criação do modelo de linguagem.

Na equação (2.3), o modelo de linguagem $P(W)$ é usado para modelar a probabilidade de uma dada sequência de palavras $P(W) = P(w_1, w_2, ..., w_N)$, tendo em conta estatísticas obtidas a partir de um *corpus* de texto. Tipicamente, $P(W)$ é calculado com o produto das probabilidades de cada subsequência de palavras na sequência, e pode ser reescrito via:

$$P(W) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2) * ... * P(w_m|w_1, ..., w_{m-1}) \tag{2.7}$$

Para limitar o tamanho da sequência de palavras que podem ser tidas em consideração na criação do modelo, é aplicada a propriedade de Markov para determinar um contexto. Por exemplo, utilizando um modelo de Markov de primeira ordem, com um contexto de tamanho 1, são

usadas estatísticas referentes apenas às frequências de pares de palavras adjacentes, permitindo obter-se um modelo bigram (ou 2-gram). Nesse caso, $P(W)$ poderia ser reescrito como:

$$P(W) = P(w_1) * P(w_2|w_1) * P(w_3|w_2) * ... * P(w_m|w_{m-1}) \tag{2.8}$$

Modelos de linguagem n-gram

Modelos de linguagem n-gram são uma generalização da equação (2.8), em que cada palavra está condicionada pela sequência de $N-1$ palavras anteriores. Por exemplo, a probabilidade da palavra w_t surgir numa dada sequência, deve ter em consideração apenas a sequência de palavras $w_{t-n}, w_{t-n+1}, ..., w_{t-1}$, que precedem w. Estas probabilidades poderiam ser estimadas diretamente através do cálculo do número de ocorrências no *corpus* de treino, via:

$$P(W) = P(w_n|w_1, w_2, ..., w_{n-1}) = \frac{c(w_1, ..., w_n)}{c(w_1, ..., w_{n-1})} \tag{2.9}$$

Onde W corresponde à sequência de palavras e $c(w_1, ..., w_n - 1)$ ao número de ocorrências da sequência W.

No entanto, a abordagem anterior, ao depender apenas e só da contagem de frequências de palavras presentes no *corpus*, atribui probabilidade zero a todos os n-grams que não ocorrem no *corpus*, descartando todas as outras combinações de palavras possíveis. Um dos objetivos de aprendizagem automática é precisamente usar os dados de treino para produzir um modelo mais genérico que consiga, também, prever padrões em dados que não foram observados. Por esse motivo, é comum aplicar-se uma técnica de *discounting* à contagem de frequência das sequências, por forma a reservar alguma da massa da probabilidade total a n-grams que não foram observados.

Num modelo n-gram, à medida que o n aumenta, a frequência de sequências com a ordem n diminui. Uma solução para este problema dá-se através do uso de técnicas de *backoff*, em que, no caso de existir um número insuficiente de exemplos de sequências com a ordem n desejada, o modelo pode retroceder para uma ordem mais reduzida, aproveitando informação estatística de n-grams de menor ordem. Por exemplo, se num modelo 3-gram não existirem trigrams para uma dada sequência, são usados bigrams. Se não existirem bigrams, são usados unigrams.

A utilização de palavras como unidades textuais possui um inconveniente: o modelo é incapaz de inferir palavras que não ocorram nunca no *corpus* de treino. Para solucionar este problema, o vocabulário pode ser definido por sub-palavras. Nesse caso, as palavras são segmentadas em tamanhos mais pequenos (no limite, até ao nível dos grafemas). Esta solução permite, também, reduzir substancialmente o tamanho do modelo de linguagem e, consequentemente, aumentar a eficiência no processo de descodificação.

2.2.4 Descodificador

O descodificador é um módulo de procura sobre todas as possíveis transcrições W geradas pelos modelos descritos anteriormente. Sistemas de ASR híbridos, geralmente, codificam HMMs num único *Weighted Finite-State Transducer* (WFST), também chamado de *decoding graph*, a partir do qual a descodificação é realizada.

Weighted Finite State Transducer

Um *finite-state transducer* é um autómato de estados finito cujas transições entre estados estão etiquetadas com símbolos de entrada e de saída. Dessa forma, um caminho através do *transducer* codifica um mapeamento de uma sequência de símbolos de entrada em símbolos de saída.

Um *weighted finite state transducer*, ou WFST, é um *finite-state transducer* em que as transições têm um peso associado. Os pesos codificam probabilidades, durações, ou qualquer unidade quantificável que possa ser acumulada ao longo dos caminhos do grafo, por forma a permitir calcular o custo geral do mapeamento de uma *string* de entrada numa *string* de saída. Dessa forma, *weighted finite state transducers* afiguram-se como uma forma natural para representar modelos probabilísticos usados em ASR [5], nomeadamente, os modelos acústicos, de linguagem e de pronunciação.

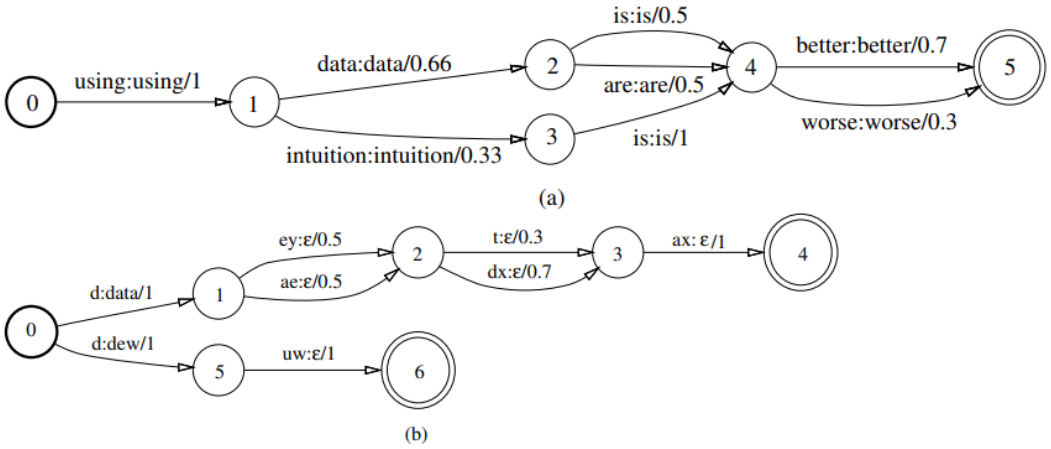


Figura 2.8: Exemplos de WFSTs [5]. Os números dentro dos círculos identificam os estados. Cada transição é identificada por *input*:*output*/peso.

A Figura (2.8) exemplifica dois WFSTs. O WFST (a) é um exemplo simples de um modelo de linguagem que representa as possíveis frases que podem ser formadas através dos símbolos de entrada (palavras). Neste modelo de linguagem, é atribuída uma maior probabilidade à frase 'using data is better' que à frase 'using intuition is worse'. O WFST (b) representa um modelo léxico que faz o mapeamento de sequências de fonemas em palavras.

A codificação dos modelos em WFSTs permite que os modelos possam ser criados de forma separada, e depois agregados num único grafo compacto a partir do qual a pesquisa pode ser realizada de forma eficiente. Existem várias propriedades que podem ser usadas para manipular WFSTs. Em ASR, há três que se sobressaem:

1. **Composicionalidade:** permite a combinação de diferentes WFSTs, construídos de forma independente, num único WFST. Por exemplo, o WFST correspondente ao modelo léxico pode ser combinado com o WFST do modelo de linguagem, permitindo a criação de um WFST que mapeia sequências de fonemas em frases.
2. **Determinização:** assegura que cada sequência de *input* tem apenas um único caminho no WFST, permitindo reduzir o espaço de procura.
3. **Minimização:** permite transformar um WFST num equivalente com o menor número de estados e de transições possíveis, combinando estados redundantes. Esta operação permite reduzir o tamanho do autómato e tornar a procura mais eficiente.

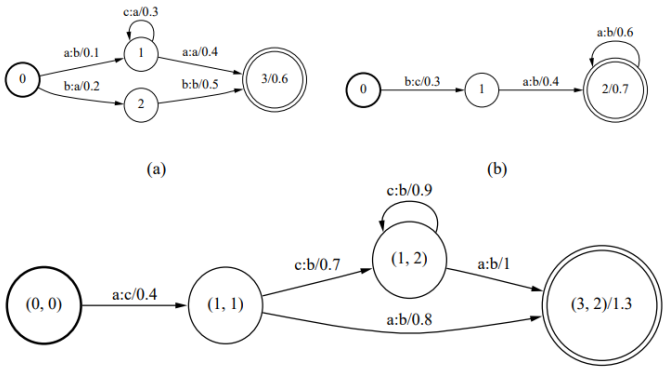


Figura 2.9: Composição de WFSTs. O autómato em baixo é formato pela composição do autómato (a) com o autómato (b). E.g., se no estado 0 do autómato (a) for consumido o símbolo a, é produzido como *output* o símbolo b e o autómato transita para o estado 1. O autómato (b), inicialmente no estado 0, consome como *input* o símbolo de *output* gerado por (a), gerando como *output* o símbolo c, e transitando para o estado 1, formando o estado $(1, 1)$ no autómato final.

Assim, os vários modelos que compõem um sistema de ASR híbrido, podem ser representados por um WFST composto por quatro WFSTs:

1. **H:** Mapeia estados HMM em fonemas dependentes de contexto, e.g., *triphones*.
2. **C:** Mapeia fonemas dependentes contexto em fonemas independentes de contexto.
3. **L:** Corresponde ao modelo léxico, e faz o mapeamento de fonemas em palavras.
4. **G:** Gramática, ou modelo de linguagem, que faz o mapeamento de sequências de palavras em sequências de palavras mais prováveis.

Geralmente, os WFSTs L e G são compostos num $L \circ G$ que resulta num WFST que mapeia fonemas em palavras restringidas pela gramática G. As operações de determinização e de minimização são depois aplicadas neste novo WFST para produzir um grafo mais compacto. De seguida, este WFST é combinado com os WFSTs C e H, resultando num grafo de procura $HCLG$ que faz o mapeamento de sequências de estados HMM em sequências de palavras:

$$HCLG = min(det(H \circ min(det(C \circ min(det(L \circ G)))))) \tag{2.10}$$

2.3 Métodos *End-to-end*

Na secção anterior foram descritos alguns conceitos importantes associados à modelação de sistemas de ASR híbridos. Uma desvantagem desta abordagem é que os vários componentes do sistema precisam de ser treinados em separado. A separação do processo de treino pode ter como consequência resultados sub-óptimos, devido à falta de propagação de erro entre os modelos. Em ASR, estes problemas tendem a manifestarem-se na sensibilidade a ruídos e variações entre vozes [20]. Watanabe et al. [21] enumeraram alguns dos principais defeitos afectos às soluções híbridas:

- O sistema é muito complexo, envolvendo a construção de vários modelos. O treino do modelo acústico divide-se em múltiplas iterações de treino e de alinhamento das observações acústicas com os fonemas.
- O mapeamento de fonemas em palavras está dependente da criação de um modelo de pronunciação, baseado num dicionário léxico. O facto de o dicionário léxico ter que ser criado à mão por alguém que possua conhecimentos linguísticos torna todo este processo propenso a erros humanos.
- Os modelos acústicos e de linguagem n-gram assumem uma independência condicional entre os estados (na forma de Markov *assumptions*), levando a que os modelos não consigam capturar as dependências existentes em dados sequenciais.
- O processo de descodificação é complexo, exigindo a integração de vários módulos. Embora a integração possa ser feita de forma eficiente via WFSTs, a construção e optimização dos *transducers* é um processo complexo.
- Os módulos são optimizados de forma separada, o que pode resultar numa optimização incoerente e sub-óptima do sistema de ASR, no seu global.

Nos últimos anos, Deep Learning (DL) tem vindo a ganhar popularidade em várias áreas de investigação. No domínio de reconhecimento de fala, têm vindo a ser sugeridas abordagens que recorrem a DL para a criação de modelos que consigam aprender um mapeamento direto entre o áudio e a transcrição, sem dependerem da criação de módulos intermédios. Matematicamente, esta abordagem procura optimizar a equação (2.1).

Estas novas abordagens de treino ponta a ponta, ou *end-to-end* (E2E), permitem simplificar o processo de treino, dado que apenas é necessário treinar um único modelo, em vez de vários. Consequentemente, o modelo pode ser aprendido de uma só vez, e optimizado conjuntamente de uma forma mais coesa.

Algo que obstante, quando comparadas com arquitecturas híbridas, abordagens E2E precisam de quantidades muito mais elevadas de dados de treino para produzir os mesmos resultados. Isto deve-se ao facto de sistemas E2E serem propensos a *overfitting*, quando treinados com um número reduzido de dados [22].

Uma rede E2E só precisa de ser treinada através de um *corpus* de fala paralelo. Existem várias formas linguísticas que podem ser usadas para representar as transcrições, como grafemas ou palavras. A classificação das observações em palavras é impraticável: exigiria que a camada de saída da rede fosse enorme, e imporia a necessidade de existir um elevado número de exemplares de cada palavra nas transcrições usadas para treino. Por estes motivos, as abordagens E2E tendem a focar-se na classificação de observações acústicas em grafemas.

Infelizmente, o número de janelas temporais a partir das quais se extraem as observações, tende a ser muito superior ao número de grafemas na sequência de saída de referência. Como resultado, a classificação de cada observação acústica resulta num número muito mais elevado de grafemas. Por exemplo, se um áudio com 1 segundo de gravação da palavra 'letras' fosse dado como entrada numa rede que classificasse observações acústicas a cada 50ms, seriam retornados $\frac{1000}{50} = 20$ grafemas, um valor consideravelmente superior ao número de grafemas na referência (|'letras'|= 6). Na realidade, o tamanho das janelas temporais tendem a rondar os 20-25ms, e as gravações são muito mais longas. Consequentemente, o número de classificações na saída é muito superior ao número de grafemas real.

Uma solução mais ou menos imediata para este problema seria colapsar previsões de grafemas repetidos. Remetendo-nos para o exemplo anterior, caso a rede retornasse como saída a sequência 'lllleeeetttrrraaasss', os grafemas repetidos poderiam ser colapsados em 'letras'. Infelizmente, esta solução tem um problema fundamental: não consegue classificar corretamente palavras com os mesmos grafemas repetidos de forma consecutiva. Por exemplo, a sequência de grafemas 'vvoo' seria mapeada em 'vo', e não na palavra correta 'voo'.

Uma solução para este problema, introduzida em 2006 por Graves et al. [23], tem o nome de Connectionist Temporal Classification. Trata-se de um módulo composto por uma camada de saída e por uma função de custo, que pode ser incorporado no topo de uma ANN para permitir a classificação de observações sequenciais.

Connectionist Temporal Classification

Numa rede E2E baseada em CTC, o módulo de saída é uma camada de *softmax* que gera uma distribuição de probabilidades sobre todos os grafemas de saída. Uma das particularidade do CTC é que os vetores de saída prevêem, para além do número de classes que se pretende classificar,

uma classe extra, a que é dado o nome de delimitador nulo ('-'). Este delimitador permite definir onde é que as previsões podem ser colapsadas, permitindo resolver a ambiguidade associada à presença de grafemas repetidos consecutivamente. Este processo opera em dois passos:

1. Associar a cada observação acústica uma distribuição de probabilidades sobre os grafemas possíveis. Ou seja, cada janela temporal é mapeada com um vetor de tamanho $k+1$, onde k é o número de grafemas, mais o delimitador nulo ('-').
2. Descodificação. Um caminho (uma possível transcrição), corresponde às classes de cada janela temporal. A descodificação é feita da seguinte forma:
   (a) Concatenar grafemas repetidos; e.g., 'vooo-o' $\longrightarrow$ 'vo-o'
   (b) Remover delimitadores nulos (-); e.g., 'vo-o' $\longrightarrow$ 'voo'

Para que a rede possa ser treinada, é necessário conhecer-se a probabilidade do modelo produzir a transcrição correta: $P('voo', O)$. Para calcular este valor, é necessário calcular as probabilidades de todas as sequências (de comprimento igual ao número de janelas temporais) que, quando colapsadas, resultam na mesma transcrição. Por exemplo:

$$P('voo'|O) = P('voooo\text{-}o'|O) + P('v\text{-}oo\text{--}o'|O) + P('v\text{-}o\text{-}ooo'|O) + ...$$

Existem múltiplas combinações possíveis. Consequentemente, é seguida uma abordagem usando programação dinâmica que acumula as probabilidades de diferentes caminhos em cada intervalo de tempo. Estas probabilidades são depois usadas para calcular o custo e atualizar os pesos da rede, via *backpropagation*.

A tarefa de determinar os grafemas de saída da rede tem o nome de descodificação. A descodificação mais simples chama-se *greedy decoding*, e retorna sempre o grafema com maior probabilidade, de acordo com o resultado da aplicação do *softmax* na camada de saída da rede.

Um dos defeitos desta abordagem, é que é assumido que os grafemas de saída, em cada intervalo de tempo, são independentes dos grafemas anteriores. Este é um problema estrutural afecto à própria arquitectura CTC, dado que não prevê a existência de ciclos na camada final da rede. Como consequência, a rede é incapaz de aprender um modelo de linguagem ao nível dos grafemas.

Como forma de solucionar este problema, Hannun et al. [24], em 2014, propuseram o algoritmo *Prefix Beam Search* que admite a incorporação de um modelo de linguagem externo no processo de descodificação.

Tal como num algoritmo de *beam search*, o processo de procura é feito de forma a que em cada intervalo de tempo (na camada de saída da rede CTC) sejam considerados apenas um número fixo de candidatos, definido pelo tamanho do *beam* (*beam width*), e a partir dos quais é possível expandir o processo de procura no intervalo de tempo seguinte.

Neste caso, os candidatos são prefixos que vão sendo expandidos ao longo da pesquisa, sendo o prefixo inicial uma *string* vazia. Sempre que um prefixo tentar expandir-se para um "espaço vazio", a última palavra é avaliada pelo modelo de linguagem, restringindo a selecção de candidatos ao conjunto de palavras mais prováveis. Desta forma, todas as sequências de grafemas do prefixo são forçosamente sequência de palavras previstas pelo modelo de linguagem. Com esta solução, é possível adicionar um certo contexto linguístico e evitar que a rede retorne transcrições não previstas pela gramática. Este método popularizou-se e foi integrado em várias arquitecturas influentes baseadas em CTC [25].

2.4 *Transfer Learning*

Transfer Learning (TL) é uma técnica usada em aprendizagem automática que procura aplicar o conhecimento obtido por um modelo na realização de uma tarefa, na realização de uma tarefa diferente, embora com o seu quê de semelhante. No contexto de ASR, TL pode ser usado para transferir o conhecimento de um modelo treinado com muitas horas numa determinada língua, para aprender o reconhecimento de fala numa língua com menos recursos. Esta solução tem o nome de *Cross-Language Transfer Learning*, e foi aplicada de forma bem sucedida em sistemas de ASR baseados em DNNs.

A lógica por detrás deste conceito é que as atributos aprendidas pelas camadas inferiores da DNN tendem a ser algo universais a todas as línguas, permitindo a sua reutilização para línguas diferentes. Esta hipótese foi confirmada experimentalmente por Kunze et al. [6]. Neste estudo, os autores congelaram os pesos das camadas inferiores de um modelo treinado em inglês, e treinaram apenas os pesos das camadas mais superficiais com um número mais reduzido de horas na língua alemã. O estudo demonstrou que a aplicação de TL permitiu superar os resultados que seriam obtidos caso o modelo fosse treinado de raiz, diretamente na língua alemã, com menos dados.

Outros estudos reportam resultados semelhantes. É o caso do estudo feito por Huang et al. [8]. Neste caso, os autores utilizaram uma DNN treinada em inglês, e adaptaram-na para o reconhecimento de línguas tão díspares como o russo e castelhano.

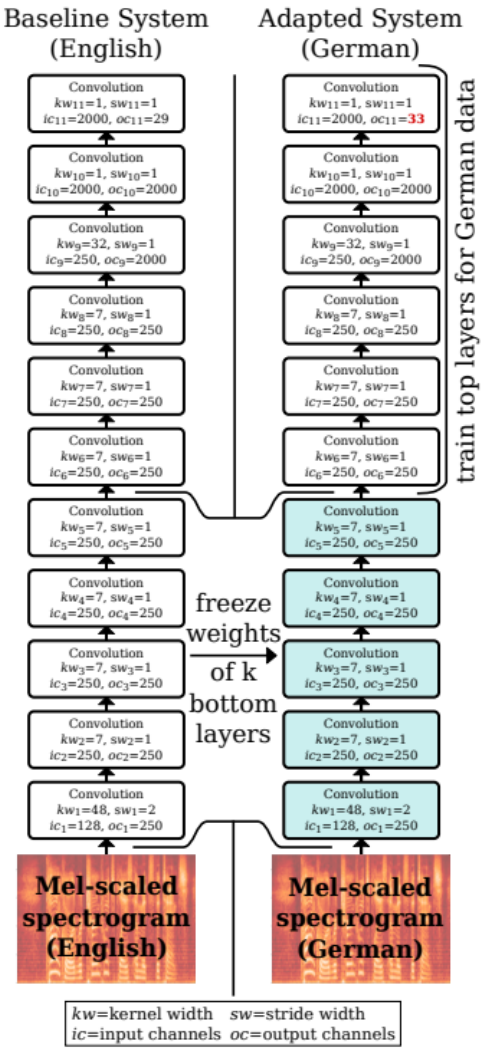


Figura 2.10: Abordagem usada em [6].

Capítulo 3

Estado da Arte

Na última década a investigação no domínio de ASR tem-se focado maioritariamente em arquitecturas E2E baseadas em DNNs. Esta secção começa pela apresentação dos *datasets* mais usados neste domínio. Depois são apresentados alguns trabalhos influentes, seguindo-se um apanhado de alguns *datasets* e de trabalhos realizados no domínio de ASR na língua portuguesa. Segue-se uma descrição de algumas *frameworks* disponíveis para a criação de sistemas de ASR. No fim são apresentadas algumas soluções comerciais populares para o fornecimento de serviços de STT.

Datasets

A generalidade da investigação no domínio de ASR recai no estudo de abordagens que consigam obter resultados de estado de arte em seis *datasets* principais, que procuram avaliar o reconhecimento de fala (em inglês) em diferentes contextos; reconhecimento de fala em ambientes de maior ou menor ruído, conversas fluentes e informais ou fala sucinta:

1. **LibriSpeech** [26] composto por 1000 horas de gravações de leituras de *audiobooks*.
2. **WSJ** [27] contém 400h de gravações de publicações narradas do *Wall Street Journal*.
3. **Switchboard** [28] contém aproximadamente 260 horas de gravações de conversas telefónicas. Tem um custo de 3000 USD.
4. **TED-LIUM** [29] é composto por 452h de gravações de apresentações de TED Talks.
5. **CHiME**[1] é um corpus com gravações de conversas informais em ambientes ruidosos. É gratuito para fins académicos, caso contrário tem um custo de 2000£.
6. **TIMIT** [30] tem a particularidade das transcrições serem compostas pelos fonemas. É gratuito para membros do *Linguistic Data Consortium*. Caso contrário, custa 250 USD[2].

[1] https://chimechallenge.github.io/chime6/
[2] https://catalog.ldc.upenn.edu/LDC93S1

3.1 Trabalhos recentes

Em 2014, Hannun et al. [22] propuseram a arquitectura Deep Speech. Trata-se de uma arquitectura de DNN, composta por camadas convolucionais e RNNs bidireccionais, treinadas para preverem a sequência de grafemas mais prováveis de corresponderem aos atributos acústicos dados como entrada na rede, minimizando uma função de custo CTC. Este trabalho popularizou-se por ter obtido um resultado de estado de arte, à data, no *dataset* Switchboard (um *dataset* considerado "difícil", pela variabilidade de vozes presente). O bom resultado deve-se, em parte, ao elevado número de horas de gravação usadas para treino: 5000, divididas por 9600 pessoas diferentes.

Em 2015, Amodei et al. [25] introduziram a arquitectura Deep Speech 2, uma expansão do trabalho desenvolvido em Deep Speech [22], com algumas optimizações que permitiram acelerar drasticamente o processo de treino (um *speedup* de 7) e melhorar a taxa de erro. A rede é composta por 11 camadas; 3 convolucionais, 7 RNNs bidireccionais, e uma última camada completamente ligada. Com a incorporação de um modelo de linguagem n-gram, com base na estratégia de *beam search pruning* [24], a arquitectura conseguiu obter uma diminuição relativa de 43.3% de taxa de palavras erradas, face à implementação original. Mais uma vez, um fator chave que explica esta melhoria de desempenho prende-se com o número de horas usado para treino: 12000h (mais do dobro da implementação original). Tanto o Deep Speech como o Deep Speech 2, foram muito influentes, e tornaram inequívoco que modelos E2E beneficiam de grandes quantidades de dados de treino.

No mesmo ano, Chan et al. introduziram a arquitectura Listen, Attend and Spell (LAS) [31], que conseguiu superar o resultado obtido pelo Deep Speech 2. Trata-se de uma DNN do tipo *encoder-decoder*, onde o *encoder* toma o nome de *listener*, e consiste numa RNN que codifica *filter bank spectra* como entrada, em vez de MFCCs. O descodificador (*speller*), é uma RNN baseada em *attention*, que emite grafemas na saída. A grande inovação desta arquitectura prende-se com o mecanismo de *attention*, que permite que o descodificador gere sequências de grafemas sem assumir a existência de independência condicional entre eles, conseguindo aprender uma espécie de modelo de linguagem inter-grafemas. Este factor foi determinante para explicar a melhoria de desempenho relativamente àquela proporcionada pelos modelos E2E, até então, do estado de arte [22][25], baseados em CTC.

Em 2016, Collobert et al. introduziram a arquitectura wav2letter [7] que, ao contrário das abordagens até então, utiliza exclusivamente redes convolucionais. A substituição de RNNs por CNNs permitiu tornar o processo de treino muito mais rápido, mantendo um desempenho semelhante àquele proporcionado pelo Deep Speech 2. Neste trabalho, os autores introduziram o conceito de *Auto Segmentation Criterion* (ASG) como alternativa para CTC. A Figura (3.1) ilustra a arquitectura desta rede.

Em 2019, Li et al. [32], introduziram o Jasper, uma arquitectura baseada em CTC que, tal como o Wav2letter, recorre exclusivamente a redes convolucionais. Os autores propuseram várias

versões da mesma rede; a versão maior é bastante profunda, contendo 54 camadas convolucionais com 333M de parâmetros no total, e obteve resultados de estado de arte no *dataset* LibriSpeech sem necessitar da introdução de dados extra no processo de treino, ou seja, recorrendo apenas ao conjunto de treinos do *dataset*.

No mesmo ano, Kriman et al. sugeriram o modelo QuartzNet [2], um modelo baseado no Jasper que consegue obter um desempenho semelhante necessitando de um número muito mais reduzido de parâmetros (19 milhões, em vez dos 333 milhões do Jasper).

Data augmentation é uma técnica usada em ML que consiste na criação de novos dados sintéticos com base em dados existentes. Tradicionalmente, *data augmentation* era aplicado em ASR através da modificação dos sinais acústicos. Por exemplo, através da introdução de ruídos de fundo, aumentando ou diminuindo a amplitude dos sinais acústicos, ou tornando o som mais agudo ou mais grave por via da alteração das frequências.

Park et al., em 2019, introduziram o SpecAugment [33], um método para *data augmentation* que permite a criação de novos dados directamente a partir dos atributos dados como entrada na rede. Nesse trabalho, foram ainda introduzidas novas técnicas de *data augmentation*, como "esconder" algumas frequências ou intervalos de tempo no espectrograma, permitindo que os modelos conseguissem generalizar melhor em tarefas de ASR em ambientes com ruídos. Estas técnicas foram aplicadas numa rede Listen, Attend and Spell [31], e permitiram obter um desempenho de estado de arte (à data), nos *datasets* LibriSpeech e Switchboard.

Recentemente, Georgescu et al. [34] fizeram uma comparação exaustiva destas e de outras abordagens que foram surgindo, que pode servir de guia para a recente literatura. No estudo, é feita uma comparação entre o desempenho e as exigências computacionais necessárias para implementação das várias arquitecturas.

Wer_are_we[3] é um projeto interessante que procura manter um registo atualizado de resultados de estado de arte obtidos nos principais *datasets* usados em ASR.

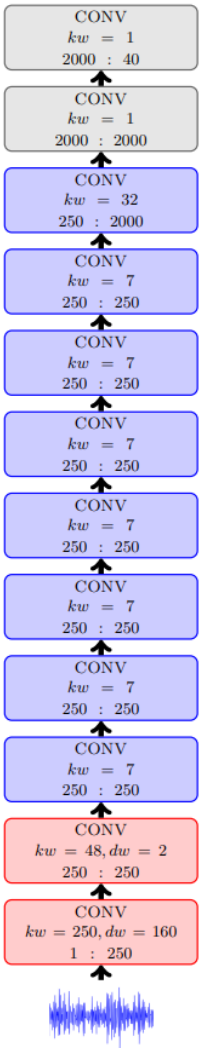


Figura 3.1: Arquitectura wav2letter [7]

[3] https://github.com/syhw/wer_are_we

E em português?

Lima et al. [1] fez um acervo dos *datasets* existentes em português que podem ser usados para ASR. O estudo identificou a existência de 24 corpus de fala, entre os quais 3 de acesso livre. Segundo o estudo, a maioria dos corpus não tem um número abundante de horas de gravação nem um balanceamento adequado em termos de idades e de género das vozes.

Corpus	# Vozes	Géneros (F/M)	Idades	Horas	#Frases	Variante
HESITA	-	-	-	27h	-	PT-PT
LapsMail	25	4/21	19-30	-	2150	PT-BR
CORALBR	362	183/179	18-60	21h	-	PT-BR

Tabela 3.1: Características de corpus públicos, adaptado de [1]. Campos que não foram possíveis de obter estão preenchidos com -. A variante PT-PT refere-se à pronunciação em português de Portugal, e a variante PT-BR refere-se à pronunciação com acentuação do Brasil.

O único que está na variante de português de Portugal é o corpus HESITA. No entanto, este corpus tem um propósito muito específico, encontrando-se anotados apenas eventos de hesitação, como pausas ou repetições [35]. Recentemente surgiram três *datasets* públicos e de acesso livre:

- **Europarl-ST** [11] é um corpus de 2019, composto por gravações anotadas de intervenções de deputados no parlamento europeu, compiladas entre 2008 e 2012.

  Uma das línguas presentes no *dataset* é português na variante de Portugal, com cerca de 32h de gravações divididas por 40 vozes diferentes.

- **Multilingual-TEDx** [12] é um corpus multilingual compilado em 2021 com gravações de palestras de TEDx em várias línguas.

  Português de Portugal é uma das línguas presentes no corpus, com 21.2 horas de gravações distribuídas por 109 pessoas diferentes.

- **Common Voice** [36] é um corpus tutelado pela Mozilla, composto por gravações submetidas por voluntários. As gravações são, depois, validadas de acordo com a avaliação feita por outros voluntários.

  O corpus contém, também, gravações em português, embora não seja feita uma diferenciação entre a pronunciação do Brasil e a de Portugal. Consequentemente, a grande maioria das gravações estão na variante de português do Brasil. Na versão 7.0, o número de horas validadas em português são 84h, divididas por 2038 vozes diferentes.

A falta de *datasets* em português disponíveis ao público limita o desenvolvimento de novas tecnologias de ASR, desincentivando a investigação de técnicas mais em voga, baseadas em DNNs, como aquelas que foram mencionadas neste capítulo.

É, no entanto, possível encontrar referências a vários artigos que recorrem a ASR para o desenvolvimento de sistemas de ASR para outros fins que não exclusivamente a transcrição de

áudio em português. Isto acontece porque o objectivo difere da conversão da fala em texto, e a necessidade de um maior número de horas de gravações acaba por ser aliviado, dado que o domínio de aplicabilidade prevê um vocabulário mais reduzido.

É o caso do trabalho proposto por Ribeiro et al. [37], em 2018, que recorre a um conjunto de ANNs para o reconhecimento de comandos de voz em português. Ou o trabalho proposto por Gonçalves et al. [38] em 2017, que usa um sistema de ASR baseado em GMMs e FFNNs para verificação de voz (*speaker verification*), na prevenção de intrusos em sistemas de identificação por voz.

Na área médica é possível encontrar referências a algumas abordagens que recorrem a ASR. Por exemplo, o trabalho desenvolvido por Rocha et al. [39], em 2016, que implementa um módulo de ASR em português por forma a aumentar a eficiência na obtenção de dados clínicos.

Algumas patologias neurológicas afectam o sinal da fala. Consequentemente, Proença et al. [40], em 2014, desenvolveram um mecanismo para identificar características acústicas do sinal da fala que pudessem indiciar a manifestação de Parkinson. Num registo semelhante, Pompili et al. [41], em 2020, propõem um sistema de ASR para atestar se doentes de aphasia (uma doença que afecta a comunicação), pronunciaram correctamente um conjunto de palavras interposto por uma plataforma *online* de terapia de fala.

No domínio multimédia, Abad et al. [42] propõem um sistema de ASR baseado em ANNs e HMMs para a legendagem de programas televisivos em português com acentuação de Portugal, do Brasil, e de países Africanos de língua oficial portuguesa. Dentro desta linha, Alvarez et al. [43] desenvolveram uma aplicação para legendagem automática de conteúdo multimédia (em directo e em diferido) em diferentes línguas europeias, entre as quais o português. O sistema é baseado em ANNs e HMMs e apresenta uma percentagem de palavras erradas a rondar os 30% no reconhecimento de português.

Veiga et al., em 2014 [44] desenvolveram um sistema de ASR baseado em HMMs para a detecção de um vocabulário mais reduzido de palavras (*keyword spotting*) em português. O *dataset* usado não é de acesso público.

3.2 *Frameworks* e soluções de STT comerciais

Para o desenvolvimento de um sistema de ASR, é comum recorrer-se a algum tipo de *framework* como ponto de partida para a criação de modelos, por forma a que se possa tirar partido de todo um ecossistema de bibliotecas, *scripts* ou adaptação de modelos pré-treinados. Nesta secção, é feita uma breve descrição de algumas das principais *frameworks* usadas para a criação de sistemas de ASR.

Criação de sistemas híbridos

Ao longo dos anos foram surgindo várias *frameworks* para a construção de sistemas híbridos. A primeira que se popularizou foi a HTK[4]. Trata-se de uma *framework* escrita em C, que disponibiliza *scripts* que permitem a construção de *templates* de sistemas baseados em HMMs. Apesar de influente, caiu em desuso, sendo que a última *release* foi feita em 2018.

Outra *framework* que teve alguma influência é o Julius[5], desenvolvida pela Universidade de Quioto, e permite a construção de sistemas baseados em HMMs. Inicialmente, foi concebida para o reconhecimento da língua japonesa, mas mais tarde foi adaptada para o reconhecimento de outras línguas. Um exemplo dessa adaptação é o projecto LaPSAM[6], que incorpora um modelo acústico em português do Brasil no descodificador usado pelo Julius.

O Sphinx[7] é uma *framework* escrita em Java que continua a ser desenvolvido de forma activa, com um ciclo de *releases* frequente.

O Kaldi[8] é a *framework* mais popular para o desenvolvimento de sistemas ASR híbridos. Foi desenvolvida maioritariamente pela Johns Hopkins University, com vista à sua utilização para fins de investigação. É escrita em C++, e recorre a *shell scripts* para ligar os vários componentes das bibliotecas entre si. Utiliza colecções de *scripts*, chamados de "receitas", para facilitar a criação de modelos de ASR do estado de arte. Algumas das principais características:

- Utiliza a toolkit OpenFST como biblioteca externa para integração ao nível do código com Weighted Finite State Transducers (WFSTs).
- Os algoritmos, funções e bibliotecas implementadas pelo Kaldi são construídas de forma bastante genérica e modular, permitindo que o código possa ser facilmente expandido.
- Processos de extracção de atributos ou a modelação de HMMs e DNNs são implementados de forma eficiente, em C++.

Criação de sistemas End-to-end

Nos últimos anos tem surgido uma vasta quantidade de *frameworks* que procuram auxiliar o processo de criação de ANNs e DNNs. Uma *framework* popular é o PyTorch, escrito em Python, e que permite prototipar redes neuronais de forma expedita. O TensorFlow é um exemplo de uma outra *framework* muito popular, escrita em Python, e que se baseia no conceito de *flow graphs*: grafos cujos nós representam operações matemáticas e em que as arestas representam os tensores que fluem entre eles. Outras opções populares incluem o Chainer, escrito em Python

[4] https://github.com/open-speech/HTK
[5] https://github.com/julius-speech/julius
[6] https://github.com/atilaromero/julius-lapsam
[7] https://github.com/sphinx-doc/sphinx
[8] https://github.com/kaldi-asr/kaldi

e que também oferece capacidades de *flow graphs*, ou o Keras, uma biblioteca simples que recorre ao TensorFlow como *backend* e que procura simplificar ao máximo possível o processo de criação de redes. Estas *frameworks* são muitas vezes usadas como *backend* de outras *frameworks* mais específicas para a criação de redes neuronais E2E, usadas em ASR. Na Tabela (3.2) estão destacadas algumas delas.

Framework	**Linguagem/*backend***
Wav2Letter++	C++
EspNet	PyTorch e Chainer
Mozilla DeepSpeech	TensorFlow
OpenSeq2Seq	TensorFlow
NVIDIA NeMo	PyTorch Lightning

Tabela 3.2: Frameworks populares para a criação de sistemas End-to-End

Soluções comerciais

A Tabela (3.3) apresenta as soluções comerciais mais populares para o serviço de STT. Tratam-se de soluções SaaS.

Solução	**Preço (por hora de gravação, em USD)**
Google Speech-to-Text	1.44 (com registo de dados, i.e, sem privacidade) 2.16 (sem registo de dados)
Amazon Transcribe	1.44 (primeiras 4h) 0.9 (12.5h seguintes) 0.612 (>16.7 h)
Microsoft STT	1
IBM Watson	0.6

Tabela 3.3: Soluções comerciais

Capítulo 4

Metodologia e desenvolvimento

Neste capítulo são descritos os passos tomados para a implementação de sistemas de ASR para português de Portugal. A secção metodologia descreve os passos de tomada de decisão na escolha das arquitecturas, *frameworks*, *datasets* e plano de desenvolvimento. A secção de desenvolvimento, detalha o processo de criação dos sistemas ASR.

4.1 Metodologia

Arquitecturas e *Frameworks*

O objetivo desta tese assenta na comparação entre arquitecturas híbridas mais tradicionais, com as novas abordagens E2E treinadas com recurso a *Cross-Language Transfer Learning* no reconhecimento de português de Portugal.

Para o desenvolvimento das arquitecturas híbridas, optou-se pelo uso da *framework* Kaldi, por ser a mais popular, e por permitir a criação de sistemas de ASR robustos, com recurso a receitas de estado de arte.

No que se refere à arquitectura E2E, tomámos dois requisitos como fundamentais: que fosse do estado de arte e que possuísse um *checkpoint* pré-treinado, que pudéssemos tomar como ponto de partida para o treino em português. Encontrámos várias opções que preenchiam estes critérios: as implementações do Deep Speech 2 [25], do wav2letter [7] e do Jasper[32] pela *framework* OpenSeq2Seq[1], e as implementações do Jasper e do QuartzNet[2] pela NVIDIA NeMo[2]. De entre as várias implementações disponíveis, escolhemos o QuartzNet, por apresentar um desempenho competitivo com uma fração do tamanho. Subsequentemente, recorremos à *framework* NVIDIA NeMo para a configuração destas redes.

[1] https://nvidia.github.io/OpenSeq2Seq/html/speech-recognition.html#models
[2] https://ngc.nvidia.com/catalog/models/nvidia:nemospeechmodels/files

Modelo	**# parâmetros**	**WER**
Deep Speech 2	49M	6.71
Wav2Letter	208M	6.67
Jasper10x5	333M	3.37
QuartzNet15x5	**18M**	**3.96**

Tabela 4.1: Percentagem de palavras erradas (WER) dos vários modelos no LibriSpeech dev-clean (*greedy decoding*). Os modelos Jasper e QuartzNet correspondem às implementações pelo NVIDIA NeMo, os modelos wav2letter e Deep Speech 2 correspondem aos *checkpoints* implementados e disponibilizados pelo OpenSeq2Seq.

Hardware e *Software*

Todo o desenvolvimento e experiências foram produzidas na mesma máquina:

- Placa Gráfica: NVIDIA GeForce RTX 3060
- Processador: Intel Core i5-9400F
- Memória RAM: 16GB
- Sistema operativo: Fedora 30

No caso do desenvolvimento dos sistemas E2E, recorreu-se ao docker (versão 19.03.12) para a instalação e utilização do NVIDIA NeMo.

Datasets

Para a criação dos modelos acústicos e dos modelos E2E, e com base no acervo descrito no capítulo de estado de arte, optou-se pela utilização dos dois *datasets* de fala públicos, existentes em português:

- Europarl-ST [11]
- Multilingual-TEDx [12]

Para tornar o texto desta tese mais sucinto, os *corpus* Europarl-ST e Multilingual-TEDx serão, doravante, referidos por E e por T, respectivamente.

Por forma a obter-se mais dados de treino, optou-se por criar um novo *dataset*, composto pela junção dos *datasets* E e T. Este *dataset* será referido por E+T, nos textos que se seguem.

Para a criação dos modelos de linguagem, recorreu-se às transcrições presentes nos *corpus* E, T e E+T. Como estes *datasets* são pequenos, recorreu-se também ao *corpus* de texto CETEMPúblico [45][3], para a criação de modelos de linguagem com maior variabilidade linguística e maior capacidade de generalização.

Nos textos que se seguem, o *corpus* CETEMPúblico será denominado por P. A forma exacta de como os *datasets* E, T, E+T e P foram processados é descrita em maior pormenor na secção de Desenvolvimento.

Plano de desenvolvimento

Para cada um dos *datasets* de fala, construiu-se um conjunto de modelos híbridos e um modelo E2E.

A Figura (4.1) apresenta um diagrama do processo de desenvolvimento para o *corpus* T. A mesma lógica aplica-se aos *corpus* E e E+T. Por uma questão de tempo, optou-se por por integrar cada modelo acústico apenas com o modelo de linguagem do mesmo domínio, e com o modelo de linguagem mais genérico P.

Na fase de desenvolvimento dos modelos de linguagem, constatou-se que o tamanho dos modelos, em disco, crescia de forma mais ou menos linear com o aumento da ordem usada. Como o *corpus* CETEMPúblico é muito grande, por uma questão prática, optámos por nos limitar ao uso de *3-grams*.

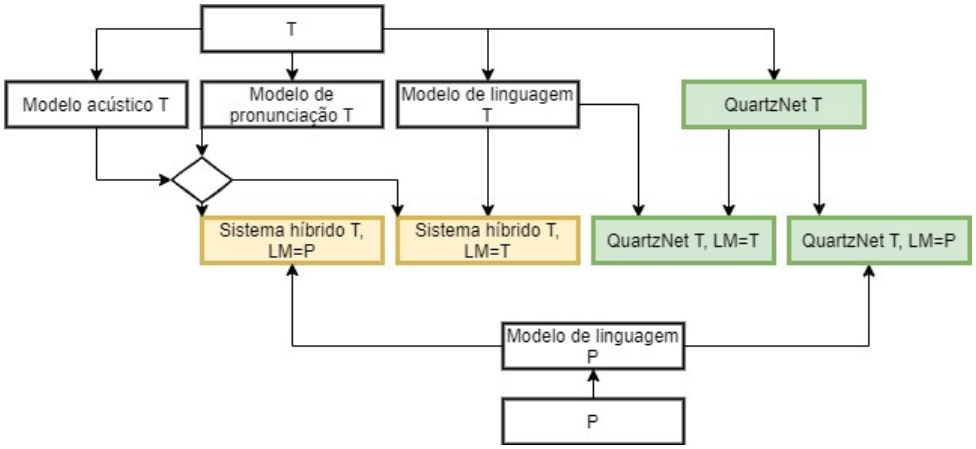


Figura 4.1: Os rectângulos preenchidos a amarelo representam sistemas de ASR híbridos treinados. Os rectângulos a verde representam modelos E2E QuartzNet. LM refere-se a modelo de linguagem.

[3] https://www.linguateca.pt/CETEMPublico

4.2 Desenvolvimento

4.2.1 Processamento dos *datasets*

Europarl-ST (E)

O primeiro *corpus* é o Europarl-ST [11], um *corpus* multi-lingual de acesso livre, com gravações de intervenções de deputados no parlamento europeu compiladas entre 2008 e 2012. O *corpus* foi construído com vista à sua utilização em tarefas relacionadas com *Spoken Language Translation* (SLT): transcrição automática de gravações numa dada língua fonte, ou *source*, para uma língua diferente, chamada de alvo, ou *target*.

O *dataset* vem segmentado em vários directórios, relativos a várias línguas europeias, incluindo a portuguesa. Dentro de cada um dos directórios, estão contidas as gravações nessa língua (*source*), e um conjunto de vários sub-directórios referentes a cada uma das línguas *target*. Por sua vez, cada um dos sub-directórios surge dividido em directórios de treino, teste e desenvolvimento. Dentro de cada *split*, estão contidos vários ficheiros. A combinação destes ficheiros permite fazer o mapeamento dos segmentos de áudio (representam um ou parte de um discurso parlamentar) com o respectivo ficheiro de áudio no qual o segmento está contido (o mesmo ficheiro de áudio contém vários segmentos), com o identificador do orador que proferiu o segmento, e com a respectiva transcrição.

Como o objetivo deste projecto é ASR em português, e não SLT, os *splits* originais deixam de fazer sentido: estamos interessados em agregar o maior número de pares de gravação-transcrição em português (independentemente da língua *target*), e não o maior número de gravações em português traduzidas para uma dada língua. Assim, para agregar a totalidade das transcrições em português, fez-se uma pesquisa em profundidade sobre toda a hierarquia de directórios, e guardou-se todos tuplos (gravação, segmento, orador, transcrição), sem duplicados.

Após este processo, aplicou-se o seguinte processamento sobre cada transcrição: letras maiúsculas foram convertidas em letras minúsculas, símbolos de pontuação foram removidos, e valores numéricos foram convertidos nas palavras correspondentes. Eg., “O dia 21 de Junho marca o início do Verão.” é convertido em “o dia vinte e um de junho marca o início do verão”. Todo este processamento foi feito através de um *script* em python construído para o efeito. Para a conversão dos números na sua representação textual (em extenso), recorreu-se à biblioteca num2words[4].

Após este passo, recriaram-se novos conjuntos de treino (*train*), teste (*test*) e desenvolvimento (*dev*), com as seguintes proporções: 75% dos tuplos para treino, 15% para teste, e 10% para desenvolvimento.

[4] https://pypi.org/project/num2words

Multilingual TEDx (T)

O segundo *dataset* , Multilingual TEDx [12], é semelhante ao anterior. É constituído por gravações, em várias línguas, de participantes em sessões de TEDx. Tal como o Europarl-ST, o *dataset* também vem preparado para SLT. Dessa forma, extraiu-se apenas o conteúdo relativo às intervenções em português de Portugal. Depois deste passo, foi aplicado o mesmo pré-processamento que tinha sido usado no *dataset* anterior. No fim, dividiu-se o *dataset* em conjuntos com a mesma proporção escolhida para o *dataset* anterior: 75% dos dados para treino, 15% para teste, e 10% para desenvolvimento.

Europarl-ST + Multilingual TEDx (E+T)

Criou-se um novo *dataset* agregando os dois *datasets* Europarl-ST e Multilingual-TEDx. Neste *dataset*, concatenaram-se os conjuntos criados nos dois *datasets* anteriores, mantendo-se a mesma proporção de amostras de treino, teste e desenvolvimento.

	Horas			# frases		
Dataset	*train*	*test*	*dev*	*train*	*test*	*dev*
E	24.6	4.87	3.32	8534	1707	1136
T	15.98	3.14	2.18	9812	1962	1308
E+T	40.58	8	5.5	18346	3669	2444

Tabela 4.2: Distribuição de horas e frases por cada conjunto

Dataset	# vozes distintas
E	40
T	109
E+T	149

Tabela 4.3: Número de vozes de cada *dataset*

CETEMPúblico (P)

Recorreu-se ao *corpus* CETEMPúblico para a modelação de linguagem. O *corpus* é composto por, aproximadamente, 180 milhões de palavras em português de Portugal, divididas ao longo de mais de 1 milhão e meio de extractos de notícias publicadas num jornal português. No total, o *corpus* ocupa 1.1 GB em disco.

A elevada dimensão do *corpus* e o facto dos ficheiros oriundos do jornal incluírem material em formatos diversos (e.g., imagens ou outras categorias não legíveis), torna impraticável fazer uma revisão manual do *corpus* por forma a ser possível assegurar que todo o conteúdo é limpo. Como forma de mitigar este problema, optou-se por aplicar o seguinte processo de filtragem:

- Excluir todos os títulos e autores das notícias

- Remover todas as frases contendo números ou caracteres não previstos no alfabeto latino simples português (sem contar com símbolos de pontuações, como vírgulas ou pontos finais)

Mais tarde, na fase de desenvolvimento do modelo de linguagem através do Kaldi, verificou-se que o computador usado nestas experiências não suportava a compilação do WFST correspondente ao modelo de linguagem através do *corpus*. Após vários testes, constatou-se que o tamanho máximo do *corpus* que permitia a criação do modelo de linguagem sem incorrer em problemas de falta de memória, rondava os 400 MB. Consequentemente, removeram-se aleatoriamente 2064226 entradas, por forma a que o tamanho do *corpus* resultante rondasse os 400MB.

Corpus	# palavras	# frases	Tamanho em disco
E	223042	8534	1.4 MB
T	154651	9798	0.881 MB
E+T	377693	18332	2.3 MB
P	67180524	3058112	401 MB

Tabela 4.4: *Corpus* E, T, E+T são consideravelmente mais pequenos que o *corpus* P, mas assemelham-se mais aos respectivos conjuntos de testes e de desenvolvimento.

4.2.2 Modelos híbridos

Um sistema de ASR híbrido envolve a criação de um modelo acústico para mapear observações acústicas em fonemas, um modelo de pronúncia para mapear fonemas em palavras, e um modelo de linguagem que mapeia palavras em sequências de palavras. A modelação de cada um destes componentes foi feita com base na receita fornecida pelo Kaldi para o *dataset* mini_librispeech. Nesta secção, é feita a descrição dos passos efetuados.

Pré-processamento

Para a modelação acústica, o Kaldi requer um mapeamento entre as gravações (identificadas por um identificador), a localização das mesmas no sistema de ficheiro, a respectiva transcrição, e um identificador da pessoa que a pronunciou. No Kaldi, este mapeamento tem que ser configurado manualmente em três ficheiros:

1. *wav.scp*: <identificador de uma gravação> <localização da gravação> (...)
2. *text*: <identificador de uma gravação> <transcrição correspondente> (...)
3. *utt2spk*: <identificador de uma gravação> <identificador da pessoa que a proferiu> (...)

No apêndice é possível encontrar algumas amostras dos ficheiros criados.

4.2.2.1 Modelação acústica

No Kaldi, os parâmetros de um modelo HMM/GMM são estimados em iterações de treino de Viterbi: cada iteração é composta por uma fase de treino e uma fase de alinhamento forçado, que procura estimar quais os estados fonéticos com maior probabilidade de terem sido gerados por cada observação acústica, dados os parâmetros actuais do modelo.

A criação dos modelos segue uma lógica sequencial: os parâmetros do primeiro modelo servem de base para a criação do segundo modelo, e assim sucessivamente, até ao quarto e último modelo. Desta forma, os alinhamentos gerados por cada modelo podem ser aproveitados e aprimorados à medida que se vão introduzindo algoritmos de treino mais sofisticados, no treino dos modelos subsequentes. O aumento do grau de sofisticação dá-se através da incorporação de fonemas dependentes de contexto (eg., *triphones*), do aumento do número de parâmetros gaussianos e de estados HMM passíveis de serem modelados, e no aumento da dimensionalidade dos vectores de atributos dados como entrada no treino dos modelos. Através deste processo de treino sequencial é possível ir-se guardando o estado do processo de treino na forma de um modelo intermédio e, no fim, atestar a melhoria de performance produzida por cada um deles. Modelos criados:

1. **mono:** o primeiro modelo HMM/GMM que foi treinado, foi um modelo *monophone*, onde cada fonema é modelado por um único estado HMM - existe um mapeamento de um para um entre fonemas e estados HMM. Inicialmente os atributos são alinhados igualmente com os estados. Depois os parâmetros do modelo são estimados via treino de Viterbi durante 40 *epochs*. O modelo toma como entrada vectores de atributos MFCC com 13 dimensões, correspondentes aos 13 coeficientes espectrais, normalizados via CMVN por voz. O treino dos modelos *monophone* é feito através do script train_mono.sh. Este modelo serve de bloco de construção inicial para a criação de modelos mais sofisticados, que irão ser introduzidos a seguir.

2. **tri1:** no modelo anterior, apenas eram tidos em conta os parâmetros acústicos de um único fonema. Como os fonemas podem variar consideravelmente em consequência do contexto em que se inserem (qual o fonema que o precede e qual o que o sucede), é criado um modelo baseado em *triphones*. O modelo toma como entrada vectores de atributos com 13 coeficientes espectrais, mais os respectivos atributos delta e delta-delta, totalizando um vector com 39 dimensões.

3. **tri2:** o segundo modelo triphone é semelhante ao anterior, com a única diferença de que neste caso o número total de parâmetros Gaussianos foi incrementado para 15000, e o número de estados HMM para 2500.

4. **tri3:** neste modelo, é adicionado contexto temporal aos vectores de atributos MFCC via *frame splicing*: cada vetor (com 39 dimensões MFCC) é concatenado com os vetores relativos aos quatro *frames* que o precedem e que o sucedem, resultando num vector com $39 + 39 * 8 = 351$ dimensões. Para permitir que os parâmetros possam ser estimados de forma robusta, a dimensionalidade dos vectores é reduzida para 40.

5. **tdnn:** Com base nos alinhamentos gerados pelo último modelo (tri3), é criado um modelo DNN/HMM. No Kaldi, este modelo é referido como um *chain model*[5], e é descrito com detalhe em [46]. A DNN é uma TDNN com 6 camadas, com um *time-stride* (*sub-sampling*) de 1 para as primeiras três camadas, 0 para quarta camada, e 3 para as restantes camadas. A TDNN foi treinada durante 20 *epochs*, com um *learning rate* inicial de 0.002, e um *learning rate* final de 0.0002.

4.2.2.2 Modelação de pronúncia

A sequência de estados escondidos, modelada pelo modelo acústico, corresponde a uma sequência de fonemas. No entanto, o objetivo principal em ASR consiste em gerar uma sequência de palavras para uma sequência de observações. No Kaldi, o modelo de pronunciação é um *Weighted Finite State Transducer* (WFST) que mapeia uma sequência de fonemas de *input* numa sequência de palavras de *output*.

Como foi mencionado no Capítulo 2, a criação de um modelo de pronunciação requer um dicionário léxico. Dessa forma, optou-se pelo uso de um dicionário fonético disponível ao público[6]. com as pronunciações em português de Portugal, no formato SAMPA, das 50 mil palavras mais frequentes no *corpus* CETEMPúblico.

Existe, no entanto, um pequeno número de palavras nas transcrições dos *datasets* de treino, que não constam no dicionário. Estas palavras, caso ocorressem nos conjuntos de teste e de desenvolvimento, não seriam transcritas correctamente. Para resolver este problema, procurou-se uma transcrição fonética para este conjunto de palavras. Para línguas não-fonéticas, mas onde existem certos padrões de pronunciação das palavras, como é o caso do português, é possível seguir uma abordagem de ML para aprender um mapeamento entre as palavras e os fonemas que as compõem. Estes modelos têm o nome de conversores grapheme-to-phoneme (G2P). Consequentemente, aplicou-se um modelo G2P[7] para obter a sequência de fonemas para as restantes palavras.

O Kaldi requer, também, que sejam especificados os fonemas que queremos modelar através do modelo de pronunciação num ficheiro nonsilence_phones.txt. Neste caso, corresponde a uma lista com os fonemas usados em português, no formato SAMPA. Para além dos fonemas ditos reais, presentes no dicionário léxico, o Kaldi precisa de ter uma forma de conseguir representar os restantes sons, modelados através do modelo acústico, e que podem corresponder a sons como ruídos ou silêncios. Estes sons são representados por um conjunto de fonemas especiais, como SPN (*spoken noise*) ou SIL (*silence*), e devem ser especificados num ficheiro silence_phones.txt. No apêndice é possível encontrar exemplos dos ficheiros de mapeamento fonético usados nesta tese. Esta informação léxica é depois usada pelo Kaldi para compilar o WFST correspondente ao léxico, no formato OpenFST, num ficheiro L.fst.

[5] https://kaldi-asr.org/doc/chain.html
[6] https://lsi.co.it.pt//spl/resources/dic_CETEMP_50k_mc_v9.txt
[7] https://lsi.co.it.pt/g2p/index.html

4.2.2.3 Modelação de linguagem

Para a construção dos modelos de linguagem, utilizaram-se quatro *corpus*: as transcrições dos conjuntos de treino dos *corpus* de fala E, T e E+T, e o *corpus* de texto P. Os quatro *corpus* foram usados para gerar quatro modelos *3-gram* diferentes.

```
[nan@localhost europarl]$ head data/local/tmp/lm.arpa && echo "(...)" && tail -n 5 data/local/tmp/lm.arpa

\data\
ngram 1=14538
ngram 2=92922
ngram 3=24314

\1-grams:
-1.460143       </s>
-99     <s>     -0.4242287
-1.432815       a       -0.5228534
(...)
-1.041393       o único país
-0.1760913      procedimento único de
-0.544068       tempo útil </s>

\end\
[nan@localhost europarl]$
```

Figura 4.2: Amostra do modelo de linguagem *3-gram* criado usando os dados de treino do *corpus* E, no formato ARPA. <s> específica o inicio de uma frase, </s> é usado para denotar o fim de uma frase.

Os modelos de linguagem são criados com recurso ao programa *ngram-count*, que gera um ficheiro no formato ARPA contendo as probabilidades logarítmicas (na base 10) dos diferentes *n-grams*. De seguida, é aplicado o programa *arpa2fst* para compilar os modelos de linguagem no formato ARPA, em WFSTs (no formato OpenFST). Neste processo, as probabilidades logarítmicas são negadas, por forma a que o custo total de uma sequência de palavras (ao longo do caminho do WFST), indique o grau de improbabilidade dessa sequência. No Kaldi, estes WFSTs correspondentes ao modelo de linguagem são designados por G.fst.

4.2.3 Modelos end-to-end

Nesta secção é feita a descrição do desenvolvimento de sistemas de ASR E2E para português. Os *datasets* usados nesta parte são os mesmos que foram usados no desenvolvimento dos sistemas híbridos; E, T, e E+T. Para esta fase, recorreu-se a *Transfer Learning* para adaptar um modelo E2E QuartzNet [2], pré-treinado em inglês, para português.

QuartzNet é uma rede convolucional, do tipo codificador-descodificador (*encoder-decoder*), treinada com CTC. A parte da rede correspondente ao codificador contém a maioria dos pesos da rede. O descodificador toma como entrada a saída do descodificador e gera as sequências dos grafemas que melhor classificam as representações geradas pelo *encoder*.

A primeira camada da rede, C1, é uma camada convolucional. Depois desta primeira camada, seguem-se B blocos compostos por quatro camadas: duas camadas convolucionais, uma camada

de normalização e uma camada de activação (ReLU). Cada bloco é repetido R vezes. Depois, seguem-se duas camadas convolucionais, C3 e C4. A parte da rede desde a camada C1 até à camada C3 corresponde ao *encoder*. A camada C4 é o *decoder*.

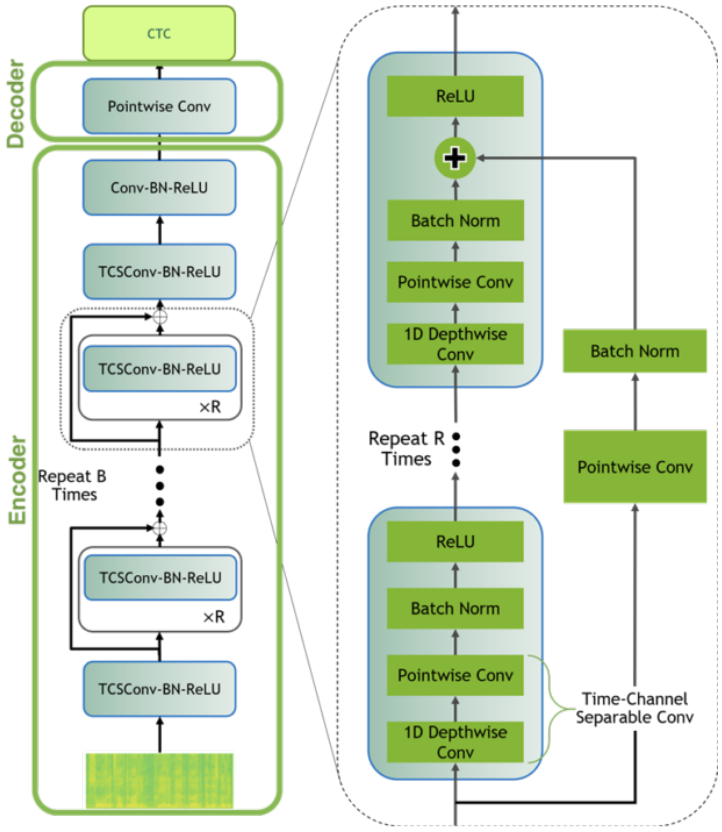


Figura 4.3: Arquitetura do modelo QuartzNet [8].

Na fase de desenvolvimento, tentou-se utilizar as arquitecturas 15x5 e 10x5, por produzirem melhores resultados (3.98% e 4.14% de palavras erradas no conjunto *dev-clean* do *dataset* LibriSpeech, respectivamente) [2]. No entanto, não foi possível reutilizar esses modelos, uma vez que a placa gráfica usada não tinha capacidade para carregar esses modelos em memória. Consequentemente, optou-se pelo uso da arquitectura mais pequena, QuartzNet_5x5, que obteve uma WER de 5.39% no mesmo *dataset*. Mais especificamente, tomou-se como base o modelo QuartzNet_5x5LS-EN[8] pré-treinado com 1000 horas de gravações do *dataset* LibriSpeech.

Pré-processamento e configuração da rede

A configuração e treino foi feito através da *framework* NVIDIA NeMo. O pré-processamento baseia-se na criação de um ficheiro no formato JSON, com o mapeamento entre a localização das gravações no sistema de ficheiros, a respectiva transcrição e a duração das mesmas. Criou-se

[8] wget https://api.ngc.nvidia.com/v2/models/nvidia/nemospeechmodels/versions/1.0.0a5/files/QuartzNet5x5LS-En.nemo

Bloco	**C**	**S**		
		5x5	**10x5**	**15x5**
C1	256	1	1	1
B1	256	1	2	3
B2	256	1	2	3
B3	512	1	2	3
B4	512	1	2	3
B5	512	1	2	3
C2	512	1	1	1
C3	1024	1	1	1
C4	||alfabeto||	1	1	1
#params		6.7M	12.8M	18.9M

Tabela 4.5: Descrição dos modelos QuartzNet-15x5, 10x5 e 15x5, adaptado de [2]. Existem cinco blocos únicos: B1-B5. Os diferentes modelos repetem estes blocos um número diferente de vezes, representado por S_i; e.g., na arquitectura 5x5 cada bloco é repetido uma vez, na arquitectura 15x5 cada bloco é repetido 3 vezes $(B_1, B_1, B_1, B_2, ..., B_4, B_5, B_5, B_5)$. C refere-se ao número de canais de saída.

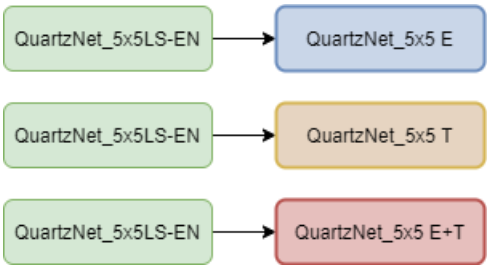


Figura 4.4: Modelos QuartzNet criados via Transfer Learning. O modelo QuartzNet_5x5LS_EN é re-treinado usando os *datasets* E, T, e E+T. Os pesos do *decoder* são inicializados de forma aleatória.

um *script* para produzir estes ficheiros com base nos ficheiros de mapeamento acústico usados pelo Kaldi: agregaram-se os ficheiros de áudio identificados em *wav.scp* com as transcrições em *text* e, recorreu-se ao programa *ffmpeg* para obter a duração de cada gravação. Exemplos destes ficheiros podem ser encontrados no apêndice.

O *decoder* original do QuartzNet5x5LS-En tem 28 canais de saída, correspondentes às 26 letras do alfabeto latino simples usado na língua inglesa (mais o espaço e o apóstrofe). A língua portuguesa utiliza um alfabeto latino mais extenso, incluindo vogais acentuadas e cê cedilhado. Como resultado, o número de canais de saída da camada de *output* foi alterada para 45.

Depois, fixou-se um *learning rate* em 0.001, um *batch size* de 32, e treinou-se a rede durante 1000 *epochs*. Os conjuntos de desenvolvimento (dev) foram usados para avaliar o modelo durante o processo de treino.

Em apêndice (NeMo) é possível encontrar um exemplo de um ficheiro configuração usado para configurar e treinar os modelos através do NVIDIA NeMo. Também é possível encontrar as curvas de *loss* e de erro na validação durante o processo de treino.

Modelo de linguagem

Modelos E2E ditos puros não prevêem a incorporação de um modelo de linguagem - a rede deve ser capaz de retornar as transcrições a partir dos atributos acústicos dadas como *input* de forma directa, ponta-a-ponta. No entanto, foi demonstrado que a integração de um modelo de linguagem externo ajuda a diminuir a percentagem de palavras erradas, principalmente em redes do tipo CTC [22][25], que assumem uma independência condicional entre os grafemas de saída.

Assim, recorreu-se à biblioteca KenLM[9], suportado pelo *beam search decoder* integrado no NVIDIA NeMo, para a criação de quatro modelos de linguagem *3-gram*, usando os mesmos *corpus* usados na criação dos modelos de linguagem para os sistemas híbridos (transcrições dos conjuntos de treino dos *corpus* de fala E, T e E+T, e o *corpus* de texto P).

[9] https://github.com/kpu/kenlm

Capítulo 5

Experiências e Testes

Neste capítulo são descritas as experiências tomadas para avaliação da qualidade dos modelos que foram desenvolvidos, e a partir das quais se procurará encontrar respostas às questões de investigação introduzidas na introdução.

Métrica de erro

Optou-se pelo uso da Word Error Rate (WER) para medir o desempenho e qualidade dos sistemas de ASR. Trata-se de uma métrica muito popular em ASR que mede a distância de edições entre a transcrição retornada pelo sistema (hipótese), e a referência (*ground truth*).

A distância de edições refere-se ao número de inserções (I), eliminações (E), e substituições (S) de palavras necessárias para transformar a hipótese na referência, e é calculada através da seguinte fórmula:

$$WER = 100 * \frac{I + E + S}{N} \tag{5.1}$$

Onde:

- **S**: número de substituições de palavras
- **E** : número de eliminações de palavras
- **I**: número de inserções de palavras
- **C** : número de palavras corretas
- **N** : Número de palavras na referência ($N = S + E + C$)

Por exemplo:

- **Frase original:** reconhecimento de fala
- **Frase obtida:** conhecimento fala
- **Erro:** S E C

O valor de WER para a frase em cima é: $100 * \frac{2}{3} = 66.67\%$.

Resumo dos modelos que foram treinados

Recorde-se que foram treinados os seguintes modelos:

Dataset	**Modelos híbridos**					**Modelos E2E**
de treino	**mono**	**tri1**	**tri2**	**tri3**	**tdnn**	**QuartzNet_5x5**
E	mono E	tri1 E	tri2 E	tri3 E	tdnn E	QuartzNet_5x5 E
T	mono T	tri1 T	tri2 T	tri3 T	tdnn T	QuartzNet_5x5 T
E+T	mono E+T	tri1 E+T	tri2 E+T	tri3 E+T	tdnn E+T	QuartzNet_5x5 E+T

Tabela 5.1: Identificação dos 18 modelos acústicos que foram criados

5.1 Experiência 1) Aferir qual o melhor modelo de linguagem

Para aferir se é preferível utilizar um modelo de linguagem mais específico, composto pelas transcrições do conjunto de treinos, embora mais pequeno, ou um modelo de linguagem maior mas mais genérico (P), conjugou-se cada um dos modelos acústicos, descritos na Tabela (5.1), com dois modelos de linguagem: um construído usando as transcrições do respectivo conjunto de treinos, e outro usando o modelo de linguagem genérico P.

No caso dos modelos híbridos, recorreu-se ao *script* do Kaldi *mkgraph.sh* para conjugar os WFSTs correspondentes aos modelos acústicos e de pronunciação com o WFST do modelo de linguagem construído usando as transcrições do conjunto de treinos do modelo acústico, ou com o WFST do corpus P. No caso dos modelos QuartzNet_5x5, utilizou-se os modelos de linguagem KenLM.

Após este processo, descodificou-se o conjunto de desenvolvimento (*dev*), usando cada um dos sistemas resultantes do processo de agregação mencionado no parágrafo anterior, e calculou-se os valores de WER. No caso dos modelos híbridos, recorreu-se ao script score_kaldi_wer.sh[1]. Para os modelos E2E, utilizou-se o script fornecido pelo NVIDIA NeMo para o efeito[2].

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/steps/scoring/score_kaldi_wer.sh

[2] https://github.com/NVIDIA/NeMo/blob/v1.0.0/nemo/scripts/asr_language_modeling/ngram_lm/eval_beamsearch_ngram.py

5.2 Experiência 2) Encontrar os parâmetros de inferência ideais

A experiência 1 permite encontrar qual o modelo de linguagem a partir do qual é possível obter resultados mais aproximados das referências.

Em modelos que recorrem a CTC, como o QuartzNet, existem três parâmetros que podem ser usados no processo de *prefix beam search*; um parâmetro *alpha*, que específica a importância atribuída ao modelo de linguagem, um parâmetro *beta* que penaliza hipóteses mais compridas, e um parâmetro *beam width*, que define o número de candidatos que podem ser considerados na procura *beam*: valores de *beam width* maiores resultam em previsões mais aproximadas das referências, mas tornam o processo de procura mais demorado.

Quanto maior for o valor de *alpha*, maior a importância atribuída ao modelo de linguagem (e consequentemente, menor a importância atribuída ao modelo acústico). No caso do *beta*, um valor negativo penaliza sequências maiores, enquanto que um valor positivo induz o descodificador a privilegiar sequências mais compridas.

Motivado pela escolha do modelo de linguagem obtido na experiência 1, aplicou-se uma *grid search* para encontrar os parâmetros de inferência que produzem melhores resultados (um WER menor na inferência). Por omissão, os valores de *alpha*, *beta* e *beam width* estão definidos como 1, 0, e 128. Assim, optou-se por considerar 20 valores de *alpha* $(0, 0.25, ..., 4.75)$ e 20 valores de *beta* $(-5, -4.5, ..., 4.5)$ distintos.

Para acelerar o processo de pesquisa, fixou-se o tamanho do *beam* num valor relativamente baixo (64), e testou-se a descodificação do *dataset* E+T usando todas as combinações de valores de *alpha* e *beta* $(20 * 20 = 400)$ descodificações. A combinação que produziu melhores resultados foi escolhida para os restantes testes que se seguem.

5.3 Experiência 3) Avaliar a performance nos conjuntos de avaliação

A partir das experiências anteriores foi possível inferir o modelo de linguagem e o modelo acústico híbrido que produz melhores resultados no conjunto de desenvolvimento (*dev*) dos *datasets* E, T e E+T. Foi também possível encontrar os parâmetros de inferência que produzem melhores resultados no conjunto de desenvolvimento do *dataset* E+T.

Nesta experiência, são usados os valores de *alpha* e *beta* obtidos na experiência anterior para avaliar o desempenho dos sistemas no conjunto de avaliação final, o conjunto *test*, para os diferentes *datasets*. No caso dos modelos QuartzNet, o tamanho do *beam* foi incrementado para 2048, por forma a que se possa obter os resultados mais próximos dos ideais.

Note-se que os parâmetros *alpha* e *beta* encontrados através da experiência 2 dizem respeito ao *dataset* E+T. Por uma questão de tempo, optou-se por não procurar os parâmetros ideais

para os *datasets* E e T, assumindo-se que os valores obtidos em E+T estariam relativamente próximos dos valores ideais para E e T individualmente.

5.4 Experiência 4) Comparar o desempenho com o Google speech-to-text

Esta experiência pretende contrastar o desempenho obtido através dos modelos desenvolvidos nesta tese com aquela proporcionada com uma das soluções comerciais mais populares usadas em ASR, a Google Speech-to-text. O serviço é gratuito para os primeiros 60 minutos de gravações[3]. Consequentemente, optou-se por transcrever apenas os conjuntos *test* dos *datasets* E e T. O conjunto *dev* não foi considerado nesta experiência.

Assim, criou-se uma função para fazer uma chamada à API da Google Speech-to-text por forma a gerar transcrições para as gravações dos conjuntos *test* dos *datasets* E e T, em português de Portugal. Os resultados obtidos foram guardados em dois ficheiros de texto, relativos aos dois *datasets*. Após este processo, aplicou-se a função usada pela NVIDIA NeMo[4] para o cálculo da WER, para calcular o WER das transcrições retornadas pela Google Speech-to-text com as referências de *ground truth*.

5.5 Experiência 5) Avaliar o desempenho dos modelos em inferirem novos dados

Os diferentes modelos foram treinados em vários domínios (E, T e E+T). Nesta experiência, os modelos treinados em cada um dos domínios serão validados noutros domínios (e.g., modelo tdnn E (treinado no domínio E), será avaliado no conjunto de testes de T).

A partir desta experiência, será possível avaliar se os modelos conseguem generalizar para domínios distintos. O resultado desta experiência servirá de aproximação para aferir a viabilidade da aplicação destes modelos em contextos reais.

[3] https://cloud.google.com/speech-to-text/pricing
[4] https://github.com/NVIDIA/NeMo/blob/v1.0.0/nemo/collections/asr/metrics/wer.py

Capítulo 6

Resultados e análise

6.1 Experiência 1) Aferir qual o melhor modelo de linguagem

Modelo de	**Modelo acústico (E)**					
linguagem	**mono**	**tri1**	**tri2**	**tri3**	**tdnn**	**QuartzNet_5x5**
E	40.9	25.53	24.17	22.88	16.36	19.52
P	37.76	20.75	19.10	17.63	**12.15**	**18.39**

Tabela 6.1: Valores de WER nos conjuntos *dev* dos modelos acústicos treinados com o *dataset* E quando combinados com os modelo de linguagem E ou P.

Modelo de	**Modelo acústico (T)**					
linguagem	**mono**	**tri1**	**tri2**	**tri3**	**tdnn**	**QuartzNet_5x5**
T	68.46	50.97	49.65	47.05	38.71	**48.98**
P	66.93	45.87	44.24	40.89	**32.69**	49.93

Tabela 6.2: WER nos conjuntos *dev* dos modelos acústicos treinados com o *dataset* T quando combinados com os modelo de linguagem T ou P.

Modelo de	**Modelo acústico (E+T)**					
linguagem	**mono**	**tri1**	**tri2**	**tri3**	**tdnn**	**QuartzNet_5x5**
E+T	54.04	37.03	35.52	33.11	23.55	28.54
P	52.18	32.96	31.33	28.96	**19.89**	**28.18**

Tabela 6.3: WER nos conjuntos *dev* dos modelos acústicos treinados com o *dataset* E+T quando combinados com os modelo de linguagem E+T ou P.

A partir dos resultados expostos nas tabelas 6.1, 6.2 e 6.3, é possível aferir que quase todos os modelos acústicos apresentam um melhor resultado, em termos de WER, quando combinados com a utilização do modelo de linguagem construído através do corpus P. O único modelo onde isso não ocorre é o modelo QuartzNet_5x5 T (tabela 6.2), embora a diferença seja praticamente insignificante (0.05 WER). Na verdade, os modelos QuartzNet foram os que apresentaram uma

menor diferença em termos de WER com a mudança do modelo de linguagem. Isto deve-se ao facto de se ter usado um *beam* relativamente pequeno (64), e de se terem usados valores de *alpha* e *beta* pré-definidos, que podem estar longe dos valores ideais para os *datasets* em causa.

Apesar de o corpus P não corresponder ao domínio de nenhum dos *datasets*, foi o que produziu melhores resultados. Estima-se que isso aconteça devido ao facto de ser um modelo consideravelmente maior e, consequentemente, tenha agregado estatísticas mais representativas dos 3-grams presentes nos conjuntos de desenvolvimento dos vários datasets. Por outras palavras, os conjuntos de treino dos modelos de linguagem mais pequenos tinham menor variabilidade linguística, e consequentemente terão feito *overfitting* nos conjuntos de treino.

Relativamente aos modelos híbridos, é possível comprovar que o processo de treino iterativo permitiu obter uma melhoria de performance, sendo que todos os modelos apresentaram um melhor desempenho relativamente aos modelos treinados na iteração anterior. Nestes casos, as maiores diferenças deram-se com a alteração de topologia fonética (de *monophones* para *triphones*) e de modelo acústico (de HMM/GMM para HMM/TDNN).

Em conclusão: é possível concluir que os modelos de linguagem baseados no *corpus* P são os que apresentam melhores resultados para a generalidade dos modelos acústicos, pelo que serão usados nas experiências que se seguem.

6.2 Experiência 2) Encontrar os parâmetros de inferência ideais

A combinação de valores *alpha* e *beta* que gerou melhores resultados foi 3.5 e 4, respetivamente. Estes valores estão bastante díspares dos valores que vinham pré-definidos na implementação original do *beam search decoder* suportado pelo NVIDIA NeMo: 1 e 0.

		b																			
		-5	-4.5	-4	-3.5	-3	-2.5	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2	2.5	3	3.5	**4**	4.5
a	0	33.1	33.0	33.0	33.0	33.0	33.1	33.2	33.3	33.5	33.7	34.0	34.3	34.8	35.2	35.8	36.5	37.3	38.2	39.2	40.4
	0.25	31.5	31.4	31.3	31.3	31.4	31.4	31.3	31.5	31.6	31.7	31.9	32.1	32.4	32.8	33.4	33.8	34.5	35.1	35.9	36.8
	0.5	30.4	30.3	30.3	30.2	30.1	30.1	30.1	30.0	30.1	30.2	30.3	30.6	30.8	31.1	31.4	31.8	32.4	32.9	33.5	34.2
	0.75	29.6	29.5	29.3	29.3	29.2	29.2	29.1	29.2	29.2	29.2	29.3	29.4	29.6	29.8	30.0	30.4	30.8	31.2	31.6	32.3
	1	29.0	28.8	28.7	28.6	28.5	28.5	28.3	28.3	28.4	28.4	28.5	28.6	28.7	28.8	29.0	29.3	29.6	29.8	30.3	30.7
	1.25	28.6	28.4	28.3	28.2	28.0	27.9	27.9	27.9	27.8	27.8	27.8	27.8	27.9	28.1	28.2	28.4	28.6	28.8	29.1	29.4
	1.5	28.3	28.1	27.9	27.8	27.6	27.6	27.5	27.5	27.4	27.4	27.3	27.3	27.4	27.5	27.6	27.7	27.9	28.1	28.3	28.6
	1.75	28.3	28.0	27.8	27.5	27.4	27.3	27.2	27.1	27.1	27.1	27.0	26.9	27.0	27.0	27.1	27.2	27.4	27.5	27.7	27.9
	2	28.3	28.0	27.8	27.5	27.3	27.2	27.0	26.8	26.8	26.7	26.8	26.6	26.6	26.7	26.7	26.7	26.9	27.0	27.2	27.3
	2.25	28.3	28.0	27.8	27.5	27.3	27.1	27.0	26.8	26.7	26.6	26.5	26.4	26.4	26.4	26.4	26.4	26.4	26.6	26.7	26.9
	2.5	28.5	28.1	27.9	27.7	27.4	27.1	26.9	26.8	26.6	26.5	26.4	26.4	26.3	26.3	26.2	26.2	26.2	26.3	26.4	26.5
	2.75	28.7	28.4	28.1	27.8	27.6	27.4	27.1	26.9	26.6	26.5	26.4	26.3	26.2	26.1	26.0	26.1	26.1	26.1	26.1	26.2
	3	29.0	28.7	28.4	28.1	27.7	27.5	27.2	27.0	26.8	26.6	26.5	26.3	26.2	26.1	26.0	26.0	25.9	26.0	26.0	26.0
	3.25	29.5	29.1	28.7	28.4	28.1	27.8	27.5	27.2	26.9	26.8	26.5	26.4	26.3	26.2	26.0	26.0	25.9	25.8	26.0	26.0
	3.5	29.9	29.5	29.2	28.7	28.4	28.1	27.8	27.5	27.3	27.1	26.8	26.5	26.4	26.2	26.2	26.1	25.9	25.9	**25.8**	25.9
	3.75	30.5	30.1	29.7	29.3	28.9	28.6	28.2	27.9	27.6	27.4	27.2	26.9	26.6	26.4	26.3	26.1	26.1	26.0	25.9	25.9
	4	31.2	30.7	30.3	29.7	29.4	29.0	28.7	28.3	28.0	27.7	27.5	27.2	27.0	26.7	26.6	26.4	26.3	26.1	26.0	26.0
	4.25	31.9	31.3	31.1	30.4	30.0	29.6	29.3	28.9	28.5	28.3	27.9	27.7	27.4	27.1	26.9	26.7	26.5	26.4	26.3	26.2
	4.5	32.7	32.1	31.6	31.2	30.7	30.2	29.8	29.6	29.2	28.9	28.6	28.2	27.9	27.5	27.3	27.0	26.8	26.7	26.7	26.5
	4.75	33.6	33.0	32.5	32.0	31.5	31.1	30.5	30.1	29.8	29.4	29.1	28.8	28.4	28.1	27.8	27.5	27.2	27.1	26.9	26.8

Figura 6.1: *heatmap* com os valores de WER para as várias combinações de valores de *alpha* (a) e *beta* (b) consideradas. *Beam size* foi fixado em 64.

Isto significa que atribuir um peso maior ao modelo de linguagem foi benéfico (parâmetro *alpha*), i.e., o modelo de linguagem, treinado a partir de um corpus com uma dimensão considerável, conseguiu auxiliar o modelo acústico relativamente fraco, treinado com poucas horas. O valor de *beta* é muito diferente do pré-definido, o que significa que foi útil permitir que o descodificador privilegiasse transcrições mais longas.

Em conclusão: os parâmetros *alpha* e *beta* ideais para o *dataset* E+T andarão próximos de 3.5 e 4. Com esta combinação de parâmetros, foi possível obter um WER de 25.8% (com um *beam* de 64), um resultado consideravelmente melhor que o que tinha sido obtido anteriormente, 28.18% (com um *beam* de 128). Assim, estes valores serão usados para as experiências que se seguem.

6.3 Experiência 3) Avaliar a performance nos conjuntos de avaliação

Modelo	**Dataset de avaliação**	**WER (*dev*)**	**WER (*test*)**
tdnn E	E	**12.15**	**12.32**
QuartzNet_5x5 E	E	15.13	15.47
tdnn T	T	**32.69**	**32.45**
QuartzNet_5x5 T	T	42.0	42.32
tdnn E+T	E+T	**19.89**	**19.64**
QuartzNet_5x5 E+T	E+T	24.08	24.12

Tabela 6.4: Valores de WER dos vários modelos nos respectivos conjuntos de desenvolvimento e de teste. Modelo de linguagem P. Melhores resultados marcados a negrito.

A partir dos resultados expostos na Tabela (6.4), é possível concluir que mesmo após a optimização dos parâmetros *alpha* e *beta*, e com o aumento do tamanho do *beam* para 2048, os modelos híbridos baseados em TDNNs continuam a obter melhores resultados, em todos os *datasets*. Este resultado vai ao encontro dos resultados esperados na literatura, que concluem que modelos E2E não conseguem superar modelos híbridos quando treinados com um número insuficiente de horas [10].

Os modelos treinados no domínio E apresentam resultados muito superiores que os modelos treinados no domínio T, independentemente do modelo. Existem vários motivos que explicam este resultado. O *dataset* T apresenta uma variabilidade acústica consideravelmente maior que aquela que está presente no *dataset* E. Esta variabilidade manifesta-se ao nível de quantidade de vozes (109, em contraste com as 40 do E), diferentes ambientes acústicos com ruídos de fundo, risos, hesitações, mudanças de ritmo, alguns aplausos esporádicos, e outras imprevisibilidades típicas no contexto de uma apresentação do TEDx. Pelo contrário, o *dataset* E é composto por discursos parlamentares, onde os deputados seguem uma linha de discurso mais linear, com

menos imprevisibilidades. Não menos importante, os modelos treinados no domínio E foram treinados com mais horas de gravação, ou seja, tiveram mais amostras de treino.

Os modelos treinados no E+T apresentam um WER inferior àquele que se obteria através do cálculo média dos valores de WER obtidos nos *dataset* E e T ponderada pelo número total de horas de treino. Analisando em maior detalhe: o modelo tdnn E, treinado com 24.6 horas, tem um WER de 12.15% no conjunto *dev*. O modelo tdnn T, treinado com 15.98 horas, obteve um WER de 32.69% no conjunto *dev*. Daqui, resulta que a média de WER destes dois modelos, no conjunto *dev*, é de $\frac{12.15*24.6+32.69*15.98}{24.6+15.98} = 20.24$, que é superior àquela que foi obtida pelo modelo tdnn E+T no conjunto *dev* (19.89%).

O mesmo resultado verifica-se no conjunto *test* $\frac{12.32*24.6+32.45*15.98)}{24.6+15.98} = 20.25 > 19.64$. Daqui, podemos verificar que o aumento do número de horas teve um impacto positivo no desempenho do modelo.

O mesmo se verifica nos modelos QuartzNet; o modelo treinado no E+T apresenta um WER no conjunto *dev* de 24.08%, inferior ao esperado: 25.71%. No conjunto *test* a tendência mantém-se, a média corresponde a 26.04% e o resultado obtido foi 24.12%. Há, no entanto, uma ressalva a fazer: na experiência 2, realizou-se a pesquisa dos parâmetros *alpha* e *beta* que optimizaram o resultado na descodificação do conjunto *dev* de E+T. Por uma questão de tempo, não se realizou esta pesquisa para os *dataset* E e T. Assumiu-se, que estes valores iriam servir de aproximação para os valores óptimos de E e de T, dado que que E+T resulta da concatenação de E com T. Pode dar-se o caso de os parâmetros óptimos para E+T diferirem dos parâmetros óptimos para E e/ou T. Isso implicaria que os resultados do QuartzNet E e/ou QuartzNet T estariam ligeiramente inferiores aos resultados óptimos, e explicaria o porquê dos resultados obtidos pelo QuartzNet em E+T superarem os resultados esperados.

Em conclusão, neste cenário de poucos recursos, os modelos híbridos apresentam um desempenho superior ao dos modelos QuartzNet.

6.4 Experiência 4) Comparar o desempenho com o Google speech-to-text

Os modelos treinados nos domínios E e E+T apresentam um desempenho superior àquele proporcionado pelo Google speech-to-text, independentemente da arquitectura usada (híbrida ou E2E).

A diferença mais marcante ocorre nos modelos treinados no *dataset* E. Nestes caso, o melhor modelo híbrido apresenta uma percentagem de erro 3.4 vezes inferior ao Google speech-to-text. O serviço da Google é genérico, i.e., foi treinado para o reconhecimento de fala num vasto conjunto de domínios, por assim dizer. Por outro lado, os modelos criados nesta tese estão a ser testados no mesmo domínio em que foram treinados. Este resultado indica-nos que é possível criar um sistema com uma boa taxa de acerto num domínio específico, mesmo com um número reduzido

Modelo	**Dataset de avaliação**	**WER *(test)***
tdnn E	E	**12.32**
QuartzNet_5x5 E	E	15.47
Google Speech-to-text	E	42.38
tdnn T	T	32.45
QuartzNet_5x5 T	T	42.32
Google Speech-to-text	T	**25.75**
tdnn E+T	E+T	**19.64**
QuartzNet_5x5 E+T	E+T	24.12
Google Speech-to-text	E+T	35.96

Tabela 6.5: Valores de WER nos conjuntos de teste. Modelo de linguagem P, n-gram=3. Melhores resultados marcados a negrito.

de horas de treino. Neste caso, o modelo híbrido tdnn E foi o que apresentou melhores resultados, embora o modelo E2E QuartzNet E não tenha tido um desempenho muito inferior.

É interessante verificar que o desempenho da google speech-to-text foi muito inferior no *dataset* E em comparação com o *dataset* T. A diferença de resultados é tão díspar, que não podemos descartar a hipótese de o serviço da google incorporar o *dataset* T no treino do seu modelo em português, sendo que, nesse caso, estaríamos a testar o modelo num sub-conjunto de dados que o próprio modelo usou para treino.

Outra possível explicação: os dados de treino usados pela Google assemelham-se muito mais ao *dataset* T do que ao *dataset* E. Esta explicação parece mais lógica, atendendo a que T tem maior variabilidade que E, é provável que o modelo genérico da Google se intersecte mais com o tipo de conteúdo presente no *dataset* T.

6.5 Experiência 5) Avaliar o desempenho dos modelos em inferirem novos dados

Na experiência 3 já tinha ficado explícito que, no mesmo domínio, os modelos híbridos baseados em TDNNs produzem sempre melhores resultados que os modelos QuartzNet. Em diferentes domínios, os modelos treinados noutros domínios obtiveram uma má performance, não tendo sido capazes de generalizar.

Modelo	**Domínio de avaliação T**
tdnn E	**58.52**
QuartzNet E	61.53

Tabela 6.6: Valores de WER no conjunto *test* do domínio T.

Na Tabela (6.6) é possível verificar que no domínio T, os modelos treinados no domínio E

Modelo	**Domínio de avaliação E**
tdnn T	**35.92**
QuartzNet T	44.51

Tabela 6.7: Valores de WER no conjunto *test* do domínio E.

(tdnn E e QuartzNet E) têm um desempenho mau: 58.52% e 61.53%. O facto de terem sido treinados com poucas horas e com pouca variabilidade acústica, pode explicar este mau resultado.

Em contraste, os modelos treinados no domínio T (tabela (6.7)), obtiveram um desempenho consideravelmente superior no domínio E. O *dataset* E é um *dataset* mais fácil (tem menos ruídos e menor variabilidade). Este resultado parece indicar que, para efeitos de generalização, o aumento do número de vozes tem um impacto muito positivo.

Neste caso, o modelo QuartzNet T teve um desempenho inferior ao do tdnn T. Sistemas E2E são conhecidos por terem dificuldade em generalizar na presença de novos dados [22], e que em comparação com sistemas híbridos, precisam de uma quantidade muito superior de dados de treino para obterem valores de WER semelhantes, uma vez que têm tendência a fazer *overfitting* quando treinados com um número limitado de dados [10]. Efectivamente, os modelos híbridos apresentaram uma maior capacidade de generalização para outros domínios (ainda que baixa), mesmo quando comparados com os modelos QuartzNet, pré-treinados com um número muito superior de horas.

Em conclusão: o número de horas e a variabilidade de vozes foi insuficiente para permitir que os modelos conseguissem generalizar para novos dados. Daqui, concluímos que as potenciais aplicabilidades destes modelos remetem-se para os cenários de baixos recursos a partir dos quais foram treinados.

Capítulo 7

Conclusões

Existem poucos recursos linguísticos, de acesso livre, a partir dos quais seja possível treinar modelos de ASR para o reconhecimento de português de Portugal. Por forma a colmatar este problema, derivaram-se e processaram-se dois *datasets* recentes, no âmbito de *Spoken Language Translation*, para o domínio de ASR em português de Portugal.

A partir destes *datasets*, implementaram-se sistemas de ASR usando-se os dois paradigmas de construção mais proeminentes nesta área: arquitecturas híbridas baseadas em HMMs, e arquitecturas E2E baseadas em DL. As primeiras, foram construídas através do Kaldi. As segundas foram criadas usando *Cross Language Transfer Learning* a partir de um modelo QuartzNet_5x5LS pré-treinado com 1000 horas em inglês. De seguida, os sistemas foram avaliados por forma a que fosse possível encontrar respostas às questões de investigação, elencadas na introdução desta tese.

Não é possível fazer uma análise comparativa dos resultados obtidos com os resultados disponíveis na literatura; os *datasets* usados são muito recentes, e pelo que foi possível apurar, não existem resultados públicos em português para estes *datasets*.

Quando comparados com o desempenho da Google speech-to-text, os resultados não são conclusivos: os modelos criados com base em discursos parlamentares superaram largamente os resultados obtidos através da Google speech-to-text, enquanto que os modelos criados com base em palestras de TEDx tiveram um desempenho muito inferior.

Estima-se que o motivo desta disparidade se deva à conjugação de dois factores: número de horas de treino e variabilidade de vozes. O *dataset* de discursos parlamentares é maior e tem menor variabilidade de vozes. Consequentemente, o modelo tem um maior número de dados de treino para que possa aprender, sendo que a menor variabilidade implica que o conjunto de testes se assemelha mais ao conjunto de treinos, resultando num erro menor.

Nesse sentido, podemos verificar que é possível criar-se um sistema de ASR com um desempenho aceitável treinado com um número reduzido de horas, mas com uma variabilidade de vozes próxima do domínio de aplicabilidade. É o caso dos modelos treinados em E. Neste caso, o

Google Speech-to-text obteve um resultado fraco (42.38%), em contraste com os dos modelos tdnn E e QuartzNet_5x5 E (12.32% e 15.47%), treinados nesta tese.

Nessa situação, o modelo híbrido construído através do Kaldi apresentou um desempenho modestamente superior (12.32%) ao do QuartzNet (15.47%). O resultado não é muito díspar, e podemos argumentar que o ligeiro aumento de desempenho não suplanta toda a engenharia subjacente à construção de um sistema híbrido através do Kaldi - preparação de dicionário léxico, utilização de um G2P para gerar transcrições fonéticas para as palavras não presentes no léxico, necessidade de criar múltiplos ficheiros de mapeamento, processo de treino iterativo, e a integração dos vários módulos é um processo complexo, propenso a erros, e claramente concebido para uso por profissionais ou por investigadores da área[1]. Em sentido inverso, o processo de treino dos modelos QuartzNet é muito mais simples e directo.

Os modelos tiveram um mau desempenho no reconhecimento de fala noutros domínios. Existem múltiplas razões que podem explicar esta falha. A principal, deve-se à falta de variabilidade nos dados. Aspectos como o timbre, o tom, a acentuação, a cadência, hesitações, inflexões de voz, diferentes sotaques, entre outras idiossincrasias, exigem que o conjunto de treinos tenha uma grande quantidade de dados (com elevada variabilidade) para que consiga reconhecer a fala noutros domínios. Isto responde à questão de investigação final. Caso o objectivo seja a construção de um sistema de ASR para um domínio mais genérico, é essencial agregar-se um *dataset* com uma elevada quantidade de horas e de variabilidade para que possa generalizar para dados não vistos.

Trabalho futuro

Daqui, existem vários caminhos que podem ser seguidos para prosseguir este trabalho:

Português é uma língua com poucos recursos em NLP, o que é um factor limitador para o desenvolvimento de tecnologias de ASR utilizando abordagens baseadas em DNNs, cada vez mais em voga. Neste sentido, seria importante criar-se *corpus* de fala, com uma grande quantidade de horas e variabilidade de vozes, e que pudessem ser disponibilizados ao público em geral. Uma sugestão, seria a criação de uma aplicação para dispositivos móveis que permitisse a submissão de gravações de voz de pequenas frases, interpostas pela aplicação. As gravações podiam ser, depois, validadas por outros utilizadores, numa lógica de *crowd-sourcing*, semelhante àquela que é usada pela Mozilla Common Voice[2].

Nesta tese, aplicou-se *Transfer Learning* de um modelo QuartzNet pré-treinado com 1000 horas do Librispeech, para treinar um conjunto de redes em português com um número muito mais reduzido de horas. No futuro, seria interessante experimentar treinar modelos QuartzNet de raiz com os *datasets* usados nesta tese, e contrastar o desempenho dos mesmos com aquele que foi obtido usando *Transfer Learning*. Este teste seria importante para medir o impacto da

[1] https://kaldi-asr.org/doc/index.html
[2] https://commonvoice.mozilla.org/en

utilização de *Transfer Learning* no desempenho dos modelos.

Também poderia ser interessante experimentar a aplicação de *Transfer Learning* a partir de um modelo pré-treinado numa língua com mais semelhanças com o português, como o castelhano[3], e comparar o desempenho com aquele que foi obtida nesta tese. Outro caminho possível, seria testar o treino de um modelo E2E com o *corpus* público Common Voice, com cerca de 84h de dados em português do Brasil, e fazer *Transfer Learning* desse modelo para português de Portugal.

Outra solução, seria treinar um modelo que retornasse como *output* sequências de fonemas, e depois converter as sequências fonéticas em grafemas em português, seguindo-se uma abordagem parecida com aquela proposta por Polák et al. [47]. Com esta abordagem, os autores conseguiram classificar com maior resolução os sons que foram pronunciados (ao nível dos fonemas), e depois utilizaram uma rede para traduzir sequências de fonemas em sequências de grafemas, numa abordagem parecida com aquela que é usada em *Machine Translation*.

Outra solução que também poderia ser usada para mitigar a falta de dados em português, seria através do uso de várias técnicas de *data augmentation*, por forma a conseguir produzir-se mais dados sintéticos em português de Portugal durante o processo de treino. Neste sentido, poderia ser testado o uso das várias técnicas referidas em SpecAugment [33].

Nesta tese recorreu-se a modelos *n-gram* por serem fáceis de criar e de integrar na construção dos modelos híbridos. Mais especificamente, utilizaram-se modelos *3-gram*. Talvez fosse interessante testar o impacto do uso de modelos de linguagem *n-gram* com uma ordem diferente. Por exemplo, modelos *2-gram*, ou *4-gram*. Dentro deste tópico, também se poderia testar o uso de modelos baseados em sub-palavras, ou modelos baseados em redes neuronais. Outra sugestão: testar a incorporação de modelos de linguagem com um tamanho equiparável àquele que foi usado nesta tese (P), mas modelados através de *corpus* pertencendo aos domínios de treino dos modelos acústicos. No caso do *dataset* E, isto poderia ser feito através de um *crawler* que agregasse actas de parlamentos europeus.

[3] https://ngc.nvidia.com/catalog/models/nvidia:nemo:stt_es_quartznet15x5

Apêndice A

Kaldi

Para o Kaldi conseguir gerar os modelos de linguagem, é necessário preparar-se manualmente um directório dict para o dicionário léxico, contendo os seguintes ficheiros:

- nonsilence_phones.txt
- silence_phones.txt
- lexicon.txt

Para o treino e avaliação dos modelos acústicos, é necessário criar-se directórios de treino e teste com os seguintes ficheiros:

- utt2spk: identifica quem pronunciou cada gravação.
- text: identifica cada transcrição
- wav.scp: contém um caminho para a localização de cada gravação

```
[nan@localhost dict]$ cat nonsilence_phones.txt && echo -ne "#fonemas: " && wc -l nonsilence_phones.txt
j~
w~
i~
i
e
E
a
6
@
O
o
u
j
w
p
t
k
b
e~
6~
o~
u~
d
g
f
s
S
v
z
Z
l
L
r
R
m
n
J
#fonemas: 37 nonsilence_phones.txt
```

Figura A.1: nonsilence_phones.txt: Contém a lista dos fonemas que são usados nas transcrições.

```
[nan@localhost dict]$ cat silence_phones.txt
sil
spn
```

Figura A.2: silence_phones.txt: lista os fonemas que não correspondem a fonemas ditos reais, presentes no dicionário léxico. spn é usado para representar *spoken noise*, i.e., ruídos imperceptíveis. sil é usado para representar silêncios.

```
[nan@localhost dict]$ head lexicon.txt && echo "(...)" && tail lexicon.txt && echo "" && echo -ne "#palavras: " && wc -l lexicon.txt
a 6
á a
à a
aba a b 6
abacateiro 6 b 6 k 6 t 6 j r u
ábaco a b 6 k u
ábacos a b 6 k u S
abade 6 b a d @
abadia 6 b 6 d i 6
abafada 6 b 6 f a d 6
(...)
zoroastrianos z u r u 6 S t r i 6 n u S
zorro z o R u
zulmira z u l m i r 6
zulus z u l u S
zulu z u l u
zurique z u r i k @
zurrou z u R o
zurzir z u r z i r
<UNK> spn
!SIL sil

#palavras: 52326 lexicon.txt
```

Figura A.3: Dicionário léxico. Podem existir entradas repetidas para as mesmas palavras, mas com sequências de fonemas diferentes, representando pronunciações distintas das mesmas palavras. O som *spoken noise* é mapeado com o marcador <UNK>, que representa todas as palavras desconhecidas pelo léxico. No Kaldi, todas as palavras que surjam nos dados de treino mas que não constem no léxico são mapeadas com <UNK>.

```
[nan@localhost train]$ head -n3 utt2spk && echo "(...)" && wc -l utt2spk
eumember111589en20120201163262000014722929x eumember111589
eumember111589en20120201163262000048066454x eumember111589
eumember111589en20120201163262000650477 0xx eumember111589
(...)
8533 utt2spk
```

Figura A.4: utt2spk relativo ao conjunto de treinos do dataset E

```
[nan@localhost train]$ head -3 text && echo "(...)" && echo -ne "#transcricoes: " && wc -l text
eumember111589en20120201163262000014722929x      separadamente temendo a sua unidade e o que ela po
derá significar para o enfraquecimento da influência e do domínio da tríade capitalista que nós co
nsideramos ser os estados unidos da américa o japão e a união europeia nas instituições do capital
ismo internacional
eumember111589en20120201163262000048066454x      hoje tornam-se progressivamente grandes potências
mundiais rivalizando com a tríade e alguns deles fazem mesmo um percurso inverso ao da união europ
eia no plano do combate à fome e à pobreza que o rumo e as políticas erradas da união europeia e d
os estados-membros farão aumentar entre nós
eumember111589en20120201163262000650477 0xx      pela nossa parte defendemos o estabelecimento de r
elações com todos os países recusando a sua catalogação e independente de diferenças de conceções
políticas económicas sociais e culturais assegurando o interesse mútuo
(...)
#transcricoes: 8533 text
```

Figura A.5: text relativo ao conjunto de treinos do dataset E

```
[nan@localhost train]$ head -n3 wav.scp && echo "(...)" && echo -ne "#gravacoes: " && wc -l wav.scp
en20080924313260008 95   /home/nan/kaldi/egs/europarl/audio/en.20080924.31.3-260.m4a.wav
en200809243132602302261 7        /home/nan/kaldi/egs/europarl/audio/en.20080924.31.3-260.m4a.wav
en200809243132602302310 /home/nan/kaldi/egs/europarl/audio/en.20080924.31.3-260.m4a.wav
(...)
#gravacoes: 8533 wav.scp
```

Figura A.6: wav.scp relativo ao conjunto de treinos do dataset E

Apêndice B

NeMo

```
root@d5ebdf839dac:/workspace/nemo/europarl/manifests# head -n3 train_manifest.json && echo "(...)" && echo -ne "#gravacoes: " && wc -l train_manifest.json
{"audio_filepath": "/workspace/nemo/europarl/audio/en20120201163262000014722929.wav", "duration": 14.57, "text": "separadamente temendo a sua unidade e o qu
e ela poderá significar para o enfraquecimento da influência e do domínio da tríade capitalista que nós consideramos ser os estados unidos da américa o jap
ão e a união europeia nas instituições do capitalismo internacional"}
{"audio_filepath": "/workspace/nemo/europarl/audio/en20120201163262000048066454.wav", "duration": 16.48, "text": "hoje tornam-se progressivamente grandes po
tências mundiais rivalizando com a tríade e alguns deles fazem mesmo um percurso inverso ao da união europeia no plano do combate à fome e à pobreza que o
rumo e as políticas erradas da união europeia e dos estados-membros farão aumentar entre nós"}
{"audio_filepath": "/workspace/nemo/europarl/audio/en20120201163262000065047770.wav", "duration": 11.95737, "text": "pela nossa parte defendemos o estabeleci
mento de relações com todos os países recusando a sua catalogação e independente de diferenças de conceções políticas económicas sociais e culturais assegu
rando o interesse mútuo"}
(...)
#gravacoes: 8533 train_manifest.json
```

Figura B.1: train_manifest.json relativo ao dataset E

```python
import pytorch_lightning as pl
from nemo.core.config import hydra_runner
from nemo.utils import logging
from nemo.utils.exp_manager import exp_manager
import nemo
import nemo.collections.asr as nemo_asr
from nemo.collections.asr.models import EncDecCTCModel
from omegaconf import OmegaConf
from omegaconf import DictConfig
import copy

@hydra_runner(config_path="configs", config_name="config_5x5")
def main(cfg):

    quartznet = nemo_asr.models.EncDecCTCModel.from_pretrained(model_name="QuartzNet5x5LS-En")

    from ruamel_yaml import YAML
    config_path = './configs/config.yaml'

    yaml = YAML(typ='safe')
    with open(config_path) as f:
        params = yaml.load(f)

    params['model']['train_ds']['manifest_filepath'] = "./europarl/train_manifest.json"
    params['model']['validation_ds']['manifest_filepath'] = "./europarl/dev_manifest.json"

    new_vocabulary=[
            ' ', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
            'v', 'w', 'x', 'y', 'z', '-', 'á', 'é', 'í', 'ó', 'ú', 'à', 'è', 'ì', 'ò', 'ù', 'â', 'ê', 'ô', 'ã', 'ẽ', 'õ', 'ç'
        ]

    quartznet.change_vocabulary(new_vocabulary=new_vocabulary)
    params["labels"]=new_vocabulary

    new_opt = copy.deepcopy(params['model']['optim'])
    new_opt['lr'] = 0.001

    quartznet.setup_optimization(optim_config=DictConfig(new_opt))
    quartznet.setup_training_data(train_data_config=params['model']['train_ds'])
    quartznet.setup_validation_data(val_data_config=params['model']['validation_ds'])
    quartznet.cfg.labels = new_vocabulary

    trainer = pl.Trainer(**cfg.trainer)
    exp_manager(trainer, cfg.get("exp_manager", None))

    trainer.fit(quartznet)

if __name__ == '__main__':
    main()  # noqa pylint: disable=no-value-for-parameter
```

Figura B.2: Adaptação do modelo QuartzNet_5x5LS para português

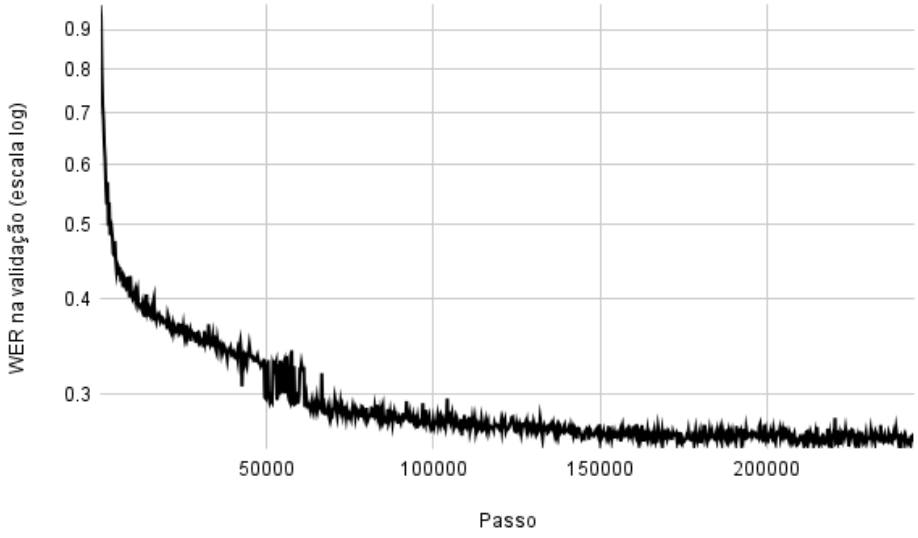


Figura B.3: WER na validação do modelo QuartzNet_5x5 E. O treino demorou cerca de 2 dias até atingir o limite máximo de 1000 epochs, correspondentes a 240000 passos. Na prática, a rede não aprendeu praticamente nada após um dia de treino (aproximadamente, após o passo 120000).

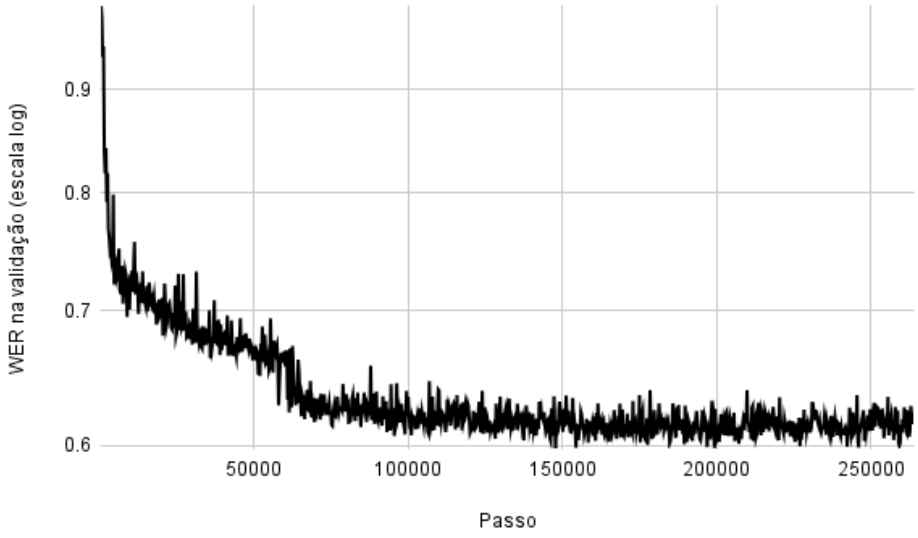


Figura B.4: WER na validação do modelo QuartzNet_5x5 T. O treino demorou sensivelmente 2 dias, equivalentes a 264000 passos de treino, embora não tenha havido progresso após o primeiro dia: o modelo aprendeu quase tudo nas primeiras 18h (78000 passos)

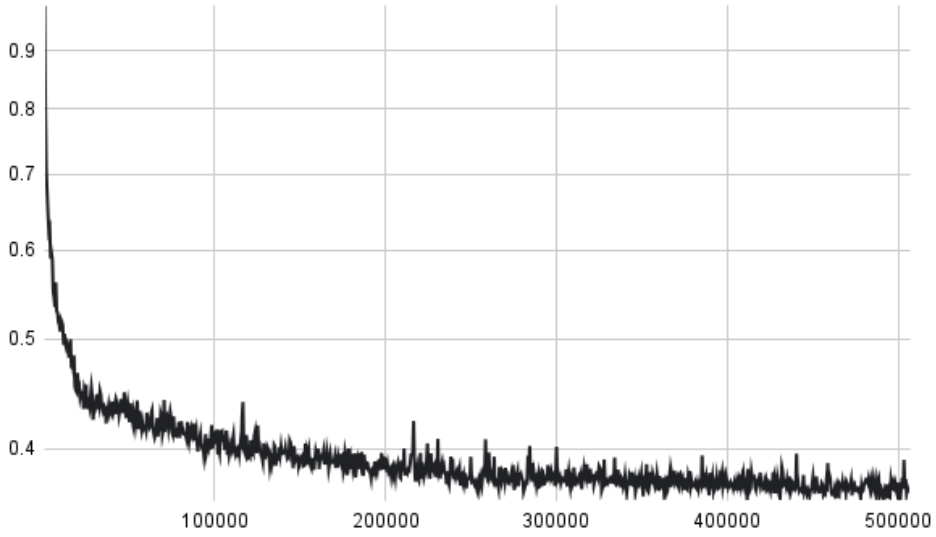


Figura B.5: WER na validação do modelo QuartzNet_5x5 E+T. Demorou cerca de quatro dias a treinar, embora não tenha havido progresso após os primeiros 2 dias de treino (244 000 passos).

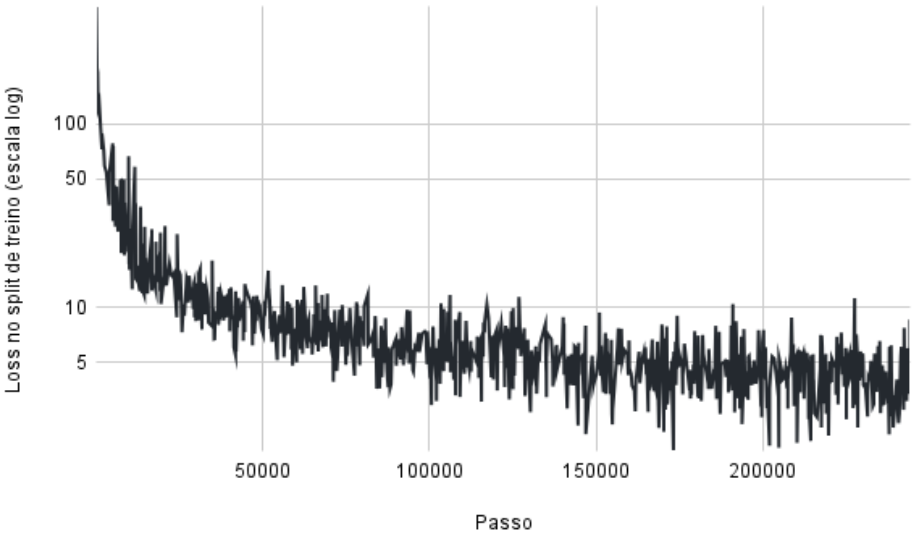


Figura B.6: Curva de loss do modelo QuartzNet_5x5 E

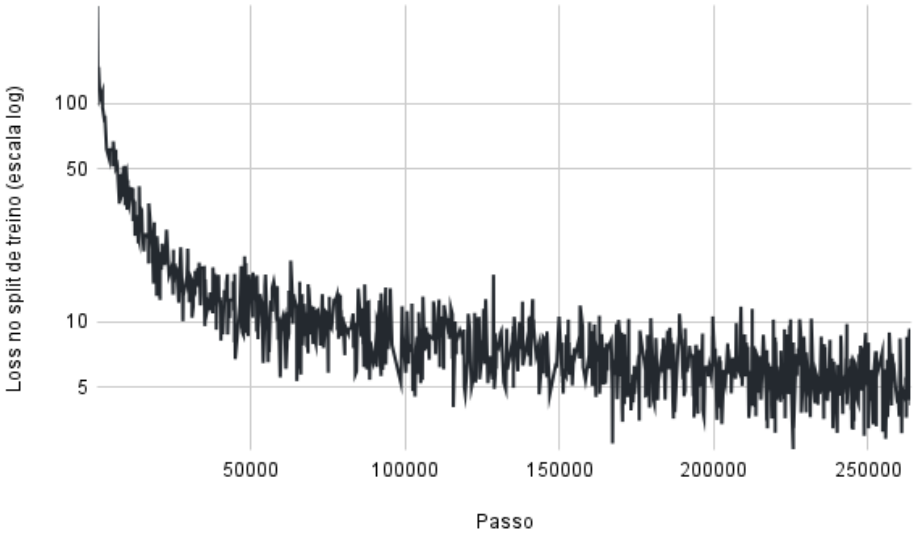


Figura B.7: Curva de loss do modelo QuartzNet_5x5 T

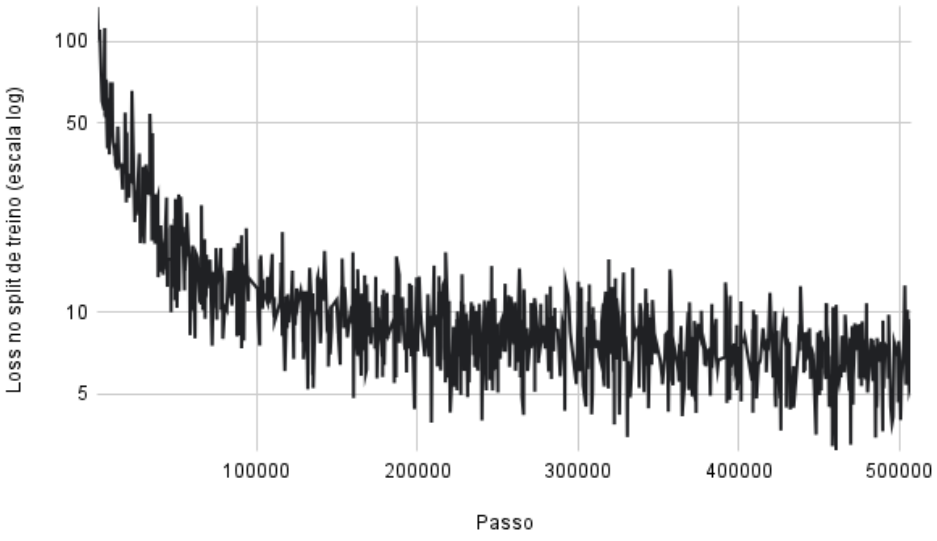


Figura B.8: Curva de loss do modelo QuartzNet_5x5 E+T

Bibliografia

[1] Thales Aguiar de Lima and Márjory Da Costa-Abreu. A survey on automatic speech recognition systems for portuguese language and its variations. *Computer Speech & Language*, 62:101055, 2020.

[2] Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6124–6128. IEEE, 2020.

[3] Andrew Maas. http://web.stanford.edu/class/cs224s/lectures/224s.20.lec8.pdf. Online, February 2021. Accessed on September 2021.

[4] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*, pages 3214–3218, 2015.

[5] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer handbook of speech processing*, pages 559–584. Springer, 2008.

[6] Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. Transfer learning for speech recognition on a budget. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 168–177, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[7] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. abs/1609.03193, 2016.

[8] Jocelyn Huang, O. Kuchaiev, Patrick K. O'Neill, Vitaly Lavrukhin, Jason Li, Adriana B. Flores, G. Kucsko, and Boris Ginsburg. Cross-language transfer learning, continuous learning, and domain adaptation for end-to-end automatic speech recognition. *arXiv: Audio and Speech Processing*, 2020.

[9] Wiqas Ghai and Navdeep Singh. Literature review on automatic speech recognition. *International Journal of Computer Applications*, 41:42–50, 03 2012.

[10] Gakuto Kurata, Kartik Audhkhasi, and Brian Kingsbury. IBM research advances in end-to-end speech recognition at INTERSPEECH 2019, 10 2019.

[11] Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, Javier Jorge, Nahuel Roselló, Adrià Giménez, Albert Sanchis, Jorge Civera, and Alfons Juan. Europarl-st: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8229–8233. IEEE, 2020.

[12] Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W. Oard, and Matt Post. The multilingual tedx corpus for speech recognition and translation, 2021.

[13] Daniel Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University Engineering Dept, 2003.

[14] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*, chapter Automatic Speech Recognition, pages 369–404. Springer, 2019.

[15] Robert Rehr and Timo Gerkmann. Cepstral noise subtraction for robust automatic speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 375–378, 2015.

[16] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1):133–147, 1998.

[17] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*, chapter Basics of Machine Learning, pages 39–86. Springer, 2019.

[18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[19] Arlindo Veiga, Sara Candeias, and Fernando Perdigão. Generating a pronunciation dictionary for european portuguese using a joint-sequence model with embedded stress assignment. *Journal of the Brazilian Computer Society*, 19(2):127–134, 2013.

[20] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*, chapter End-to-End Speech Recognition, pages 537–574. Springer, 2019.

[21] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

[22] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.

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

[24] Awni Y. Hannun, Andrew L. Maas, Daniel Jurafsky, and Andrew Y. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. 2014. arXiv: 1408.2873.

[25] Dario Amodei, 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. In *International conference on machine learning*, pages 173–182. PMLR, 2016.

[26] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[27] Douglas B. Paul and Janet M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, page 357–362, USA, 1992. Association for Computational Linguistics.

[28] J.J. Godfrey, E.C. Holliman, and J. McDaniel. Switchboard: telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520 vol.1, 1992.

[29] François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. Ted-lium 3: Twice as much data and corpus repartition for experiments on speaker adaptation. *Lecture Notes in Computer Science*, page 198–208, 2018.

[30] J. Garofolo, Lori Lamel, W. Fisher, Jonathan Fiscus, D. Pallett, N. Dahlgren, and V. Zue. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, November 1992.

[31] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[32] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An end-to-end convolutional neural acoustic model, 2019.

[33] Daniel Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Cubuk, and Quoc Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech*, pages 2613–2617, 09 2019.

[34] Alexandru-Lucian Georgescu, Alessandro Pappalardo, Horia Cucu, and Michaela Blott. Performance vs. hardware requirements in state-of-the-art automatic speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2021(1), July 2021.

[35] Sara Candeias, Dirce Celorico, Jorge Proença, Arlindo Veiga, and Fernando Perdigão. Hesita(tions) in portuguese: a database. In *Sixth Workshop on Disfluency in Spontaneous Speech*, 2013.

[36] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus, 2020.

[37] Fábio Cisne Ribeiro, Raphael Torres Santos Carvalho, Paulo César Cortez, Victor Hugo C De Albuquerque, and Pedro Pedrosa Rebouças Filho. Binary neural networks for classification of voice commands from throat microphone. *IEEE Access*, 6:70130–70144, 2018.

[38] André R Gonçalves, Ricardo PV Violato, Pavel Korshunov, Sebastien Marcel, and Flavio O Simoes. On the generalization of fused systems in voice presentation attack detection. In *2017 International conference of the biometrics special interest group (BIOSIG)*, pages 1–5. IEEE, 2017.

[39] Ricardo Sousa Rocha, Pedro Ferreira, Inês Dutra, Ricardo Correia, Rogerio Salvini, and Elizabeth Burnside. A speech-to-text interface for mammoclass. In *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–6, 2016.

[40] Jorge Proença, Arlindo Veiga, Sara Candeias, João Lemos, Cristina Januário, and Fernando Perdigão. Characterizing parkinson's disease speech by acoustic and phonetic features. In Jorge Baptista, Nuno Mamede, Sara Candeias, Ivandré Paraboni, Thiago A. S. Pardo, and Maria das Graças Volpe Nunes, editors, *Computational Processing of the Portuguese Language*, pages 24–35, Cham, 2014. Springer International Publishing.

[41] Anna Pompili, Alberto Abad, Isabel Trancoso, José Fonseca, and Isabel P. Martins. Evaluation and extensions of an automatic speech therapy platform. In Paulo Quaresma, Renata Vieira, Sandra Aluísio, Helena Moniz, Fernando Batista, and Teresa Gonçalves, editors, *Computational Processing of the Portuguese Language*, pages 43–52, Cham, 2020. Springer International Publishing.

[42] Alberto Abad, Hugo Meinedo, Isabel Trancoso, and Joao Neto. Transcription of multi-variety portuguese media contents. In *International Conference on Computational Processing of the Portuguese Language*, pages 409–420. Springer, 2012.

[43] Aitor Álvarez, Carlos Mendes, Matteo Raffaelli, Tiago Luís, Sérgio Paulo, Nicola Piccinini, Haritz Arzelus, João Neto, Carlo Aliprandi, and Arantza del Pozo. Automating live and batch subtitling of multimedia contents for several european languages. *Multimedia Tools and Applications*, 75(18):10823–10853, Sep 2016.

[44] Arlindo Veiga, Carla Lopes, Luís Sá, and Fernando Perdigão. Acoustic similarity scores for keyword spotting. In Jorge Baptista, Nuno Mamede, Sara Candeias, Ivandré Paraboni, Thiago A. S. Pardo, and Maria das Graças Volpe Nunes, editors, *Computational Processing of the Portuguese Language*, pages 48–58, Cham, 2014. Springer International Publishing.

[45] Paulo Alexandre Rocha and Diana Santos. "cetempúblico: Um corpus de grandes dimensões de linguagem jornalística portuguesa". In *V Encontro para o processamento computacional da língua portuguesa escrita e falada (PROPOR 2000)*, pages 131–140, November 2000.

[46] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755, 2016.

[47] Peter Polák, Sangeet Sagar, Dominik Macháček, and Ondřej Bojar. Cuni neural asr with phoneme-level intermediate step for non-native slt. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 191–199, 2020.