Comparação de Serviços em Nuvem para Transcrição de Fala na Língua Portuguesa em áudios com Sotaques Regionais Brasileiros

Thalles Vargas Ribeiro Lopes[1], **Jefferson Oliveira Andrade**[2], **Karin S. Komati**[2]

[1]Sistemas de Informação

[2]Programa de Pós-graduação em Computação Aplicada (PPComp)
Campus Serra do Instituto Federal do Espírito Santo (IFES)

thallesvrl@gmail.com, {jefferson.andrade,kkomati}@ifes.edu.br

Abstract. *In this work, an analysis of two cloud services, Google Cloud and Wit.ai, which perform audio transcription in Brazilian Portuguese, was carried out in order to determine which is the best tool when submitted to different Brazilian accents. The Braccent database was used, a set of 1,648 audios, with seven accents: "nortista", "baiano", "fluminense", "mineiro", "carioca", "nordestino", and "sulista". The average of the Normalized Levenshtein metric for Wit.ai is 0.96, and for Google Cloud it is 0.89, and in both tools the worst results were for the "carioca" accent. Wit.ai showed better results in all scenarios, in addition to transcribing the punctuation symbols.*

Resumo. *Neste trabalho, foi realizada uma análise de dois serviços em nuvem, Google Cloud e Wit.ai, que realizam a transcrição de áudio em língua portuguesa, com o objetivo de determinar qual é a melhor ferramenta quando submetida aos diferentes sotaques brasileiros. Foi utilizada a base de dados Braccent, em um conjunto de 1.648 áudios, com sete sotaques: nortista, baiano, fluminense, mineiro, carioca, nordestino e sulista. A média da métrica de Levenshtein Normalizado para o Wit.ai é de 0,96, e para o Google Cloud é de 0,89, e em ambas as ferramentas os piores resultados foram para o sotaque carioca. Ao final, o Wit.ai apresentou resultados melhores em todos os cenários, além de transcrever os símbolos de pontuação.*

1. Introdução

O reconhecimento automático de fala (ASR, do inglês *Automatic Speech Recognition*), ou também conhecido por sistemas *Speech-To-Text* (literalmente, Fala-para-Texto), consiste no processo de conversão de um sinal acústico em um conjunto de palavras que foram ditas [Sharan and Moir 2016]. O primeiro sistema de reconhecimento de fala conhecido e documentado foi construído em 1952 por Davis, Biddulph e Balashek em Bell Laboratories, sistema era capaz de reconhecer dígitos, alcançando precisão de 97-99% quando era adaptado ao falante [Juang and Rabiner 2005].

Posteriormente, surgiram diversos usos para o reconhecimento de fala, como ferramenta de interação entre humano e máquina, por exemplo: aplicativos, como Siri, Cortana e Google Assistant, que permitem que o usuário acione funções ou realize buscas

através de comandos de voz [López Herrera et al. 2017]; atendimento ao cliente via de *chatbots* [Raju et al. 2018] e; Tecnologias Assistivas, em que é desenvolvido um ambiente virtual para pessoas cegas [Lima et al. 2015].

A Figura 1 ilustra o fluxo do ASR, que se inicia com o usuário falando uma frase ao sistema. A frase corresponde a um sinal de áudio (marcado por um retângulo verde), que é processado pelo motor de reconhecimento de fala. O resultado é a conversão do áudio para texto. No exemplo apresentado, nota-se que a frase resultante do sistema não é exatamente a mesma, a parte inicial "Oie...", que indica uma certa hesitação da pessoa não foi convertida, bem como os símbolos de pontuação ',' e '!' também não foram compreendidas pelo sistema. A vírgula que identifica uma forma de estabelecer uma parada no meio da frase e, a exclamação tem a ver com entonação da voz da pessoa. Pontuações em geral são sutis e dependendo de seu uso a frase pode mudar de sentido, assim, pontuações em ASR é um tema em estudo [Żelasko et al. 2018].

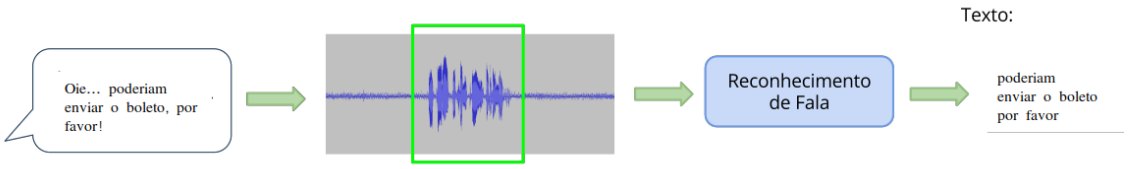

Figura 1. Fluxo geral de um ASR.

Também devemos levar em consideração a língua falada, pois dependendo da língua, os recursos disponíveis variam, há diferentes sotaques para a mesma língua dependendo da região. A língua inglesa é a que possui mais estudos [Shi et al. 2021, Ahmed et al. 2019, Wang et al. 2020, Zhang et al. 2021] e por consequência uma diversidade de base de dados, modelos e resultados quando comparado à língua portuguesa, por exemplo. Encontram-se artigos sobre sotaque árabe [Biadsy et al. 2009], sotaque francês [Lazaridis et al. 2014], sotaque em mandarim [Weninger et al. 2019], dialetos da Nigéria [Salau et al. 2020], dentre outros.

Em português, destacam-se o trabalho realizado por [Ynoguti 1999] e por [Batista 2019, Batista et al. 2018]. No caso do Brasil, a variedade de sotaques é apresentada no mapa da Figura 2, com as divisões territoriais para cada um dos 16 sotaques: 1 - Caipira; 2 - Costa norte; 3 - Baiano; 4 - Fluminense; 5 - Gaúcho; 6 - Mineiro; 7 - Nordestino; 8 - Nortista; 9 - Paulistano; 10 - Sertanejo; 11 - Sulista; 12 - Florianopolitano; 13 - Carioca; 14 - Brasiliense; 15 - Serra amazônica e; 16 - Recifense. Cada sotaque pode ter uma diferente pronúncia para algumas palavras, o que pode confundir o sistema automático de reconhecimento de fala [Viglino et al. 2019].

A tecnologia de transcrição de fala tem evoluído bastante e se popularizado devido ao crescimento das plataformas de serviço em nuvem. Como existem diversos serviços que disponibilizam essa funcionalidade, surge a pergunta de qual delas é a mais adequada para processar áudios em diferentes sotaques brasileiros. As APIs (*Applications Programming Interface*) foram selecionadas a partir da lista das mais usadas no mercado [Opidi 2019], com o requisito de processar a língua portuguesa (pt-BR) e considerando o custo de sua utilização. A fim de reduzir o custo dos experimentos do trabalho, todas as APIs usadas disponibilizam de forma gratuita uma quantidade suficiente de período/uso.

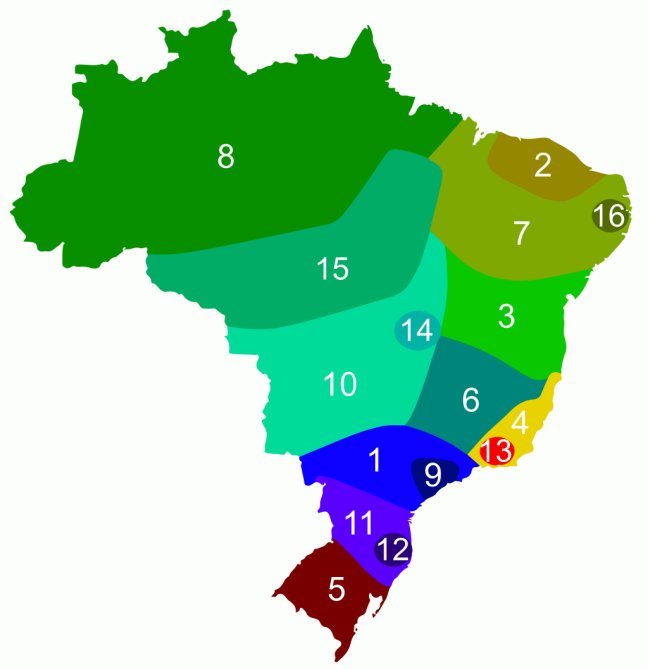

Figura 2. Mapa de sotaques do português brasileiro. Imagem sob licença CC BY-SA 4.0.

Assim, a proposta deste trabalho é comparar os resultados de dois serviços em nuvem de ASR para a língua portuguesa, as APIs: Google Cloud Speech API[1] e Wit.ai[2]. Foram selecionadas ferramentas que fornecessem o serviço de forma gratuita, *online*, e que fizessem a transcrição em língua portuguesa. Assim, ferramentas *offline*, como Vosk e o CMU Sphinx, não foram selecionadas. A seleção das duas ferramentas foi baseada na ferramenta que obteve os melhores resultados no trabalho de Iinuma e de Oliveira Igarashi [Iinuma and de Oliveira Igarashi 2019], o Google Cloud, e na ferramenta que obteve o melhor resultado no trabalho de Lima *et al.* [de Lima et al. 2021], o Wit.ai. Ambos os trabalhos citados fazem comparação de ferramentas de ASR, mas sem avaliar a questão dos sotaques regionais do português brasileiro.

Como fonte de dados foi utilizado a base de dados Braccent, criada por [Batista 2019], que é uma base de áudios com os seguintes sotaques: nortista, baiano, fluminense, mineiro, carioca, nordestino e sulista. Foi usado um subconjunto de 1.648 áudios da base de dados Braccent. Para analisar a taxa de acerto da transcrição da fala foram utilizadas as métricas Levenshtein e Levenshtein Normalizado.

No que se refere à estrutura do artigo, na Seção 2 são apresentados alguns trabalhos correlatos ao tema. Os materiais e métodos são descritos na Seção 3, construindo o ambiente de experimentos para a apresentação e discussão dos resultados na Seção 4. A Seção 5 encerra o artigo com os comentários finais e trabalhos futuros.

[1]https://cloud.google.com/speech-to-text
[2]https://wit.ai/

2. TRABALHOS CORRELATOS

Nesta seção detalhamos três artigos, um artigo de [Iinuma and de Oliveira Igarashi 2019] que faz uma comparação entre ferramentas da IBM, Google e Amazon para conversão de voz para texto de ligações telefônicas; um segundo artigo de Lima *et al.* [de Lima et al. 2021] que fez uma comparação entre 5 (cinco) serviços de reconhecimento de fala em termos de usabilidade, limitação e precisão; e um terceiro artigo de [Żelasko et al. 2018] que avalia a pontuação de transcrições de chamadas telefônicas.

Segundo [Iinuma and de Oliveira Igarashi 2019], as tecnologias, "Plataformas como serviço", "infraestrutura como serviço" e "Inteligência Artificial", estão sendo utilizadas para desenvolver sistemas que substituem pessoas em *call centers*. Existem casos até de fechamento completo de um *call center* para trocar por uma *startup* que utiliza programas de computador que substituem os seres humanos.

No artigo [Iinuma and de Oliveira Igarashi 2019] foram comparadas as ferramentas: Google Cloud Speech to Text, IBM Watson Speech to Text e Amazon Transcribe. A base de dados foi composta de gravações de áudio com 6 (seis) diferentes interlocutores brasileiros que realizaram a leitura de uma parte do livro "Divina Comédia". São 3 (três) pessoas do sexo feminino e 3 (três) masculinos. Todos têm titulação de pós-graduação e faixa etária entre 40 e 50 anos. Foram realizados os seguintes pré-processamentos usando o aplicativo Audacity[3]: eliminação de excesso de período sem falas, melhoria do volume e retirada de ruídos.

A fim de comparar qual possui a melhor eficiência, foi usada a métrica de Taxa de Erros das Palavras (WER, do inglês *Word Error Rate*). Para o WER, quanto menor o número, melhor será a qualidade do texto obtido. O WER é derivado da distância de Levenshtein, trabalhando no nível da palavra ao invés do nível do fonema. A partir da análise dos dados gerados pelo WER, constataram uma maior eficiência de conversão da API do Google (14,9%), seguido pela API da IBM (15,23%) e por último a API da Amazon (16,5%), nota-se que a diferença entre as taxas de erro é baixa. É importante ressaltar que a pesquisa foi exploratória e as conclusões foram feitas com uma pequena base de experimentos (18 gravações no total) e uma pequena amostra de pessoas (6 pessoas).

No trabalho de Lima *et al.* [de Lima et al. 2021] foram realizados testes para comparar a performance entre ferramentas de reconhecimento de fala, sendo elas: Microsoft Azure, Google Cloud, Wit, IBM Watson Speech to Text e Vosk. O Microsoft Azure foi testado de duas formas, uma com o uso da biblioteca nativa da empresa e outra com a biblioteca Speech Recognition. A base de testes utilizada é composta de trechos de 100 (cem) áudios com duração inferior a vinte segundos. Os arquivos são gravações telefônicas de conversas entre operadores do setor elétrico, estão em formato WAV, com um único canal e taxa de amostragem de 8.000 kHz.

Com relação a análise das transcrições, foram utilizadas as métricas, WER (do inglês, *Word Error Rate*), MER (do inglês, *Match Error Rate*) e WIL (do inglês, *Word information Lost*). Para avaliação geral dos serviços foi considerada a média das métricas. Através da observação dos resultados, constataram que o serviço da Wit apresentou o melhor desempenho em geral, seguido da Azure com biblioteca nativa. Pelos resultados gerais, o Azure com a biblioteca Speech Recognition fica equiparado com Google Cloud.

[3]https://www.audacityteam.org/

IBM e Vosk tiveram os piores resultados. Também foi analisado o desempenho para acertos de palavras-chaves, neste cenário, o melhor resultado foi da ferramenta Wit, seguido do Google Cloud.

Żelasko *et al.* [Żelasko et al. 2018] apresenta o problema de que muitos ASR não preveem pontuação ou capitalização. A falta de pontuação pode confundir tanto o leitor humano, quanto os algoritmos de processamento de linguagem natural que usarão estes resultados. O texto sem pontuação, além de poder ser confuso, pode ser difícil de ler, e algumas vezes perdem o real sentido do que foi dito.

A proposta do trabalho foi o de treinar duas variantes de modelos de Rede Neural Profunda (DNN, do inglês *Deep Neural Network*): a Memória de Curto-Prazo Longa Bidirecional (BLSTM, do inglês *Bidirectional Long Short-Term Memory*) e uma Rede Neural Convolucional (CNN, do inglês *Convolutional Neural Network*), para a predição de pontuação. Os modelos foram treinados no corpus inglês de Fisher [Cieri et al. 2004], que contém cerca de 11.000 conversas distintas com anotações de pontuação. Foram considerados o tempo e duração de cada palavra, solução que até então não havia sido utilizada em outro sistema para realizar a tarefa de predição de pontuação. Nos experimentos, as transcrições de corpus de Fisher são alinhadas no tempo e pontuadas usando um algoritmo de alinhamento de sequência. Foram usadas 4 (quatro) classes de pontuação, com a seguinte proporção na base de dados: espaço em branco (79,1%), ponto final (8,2%) e ponto de interrogação (1,2%), e vírgula (11,5%).

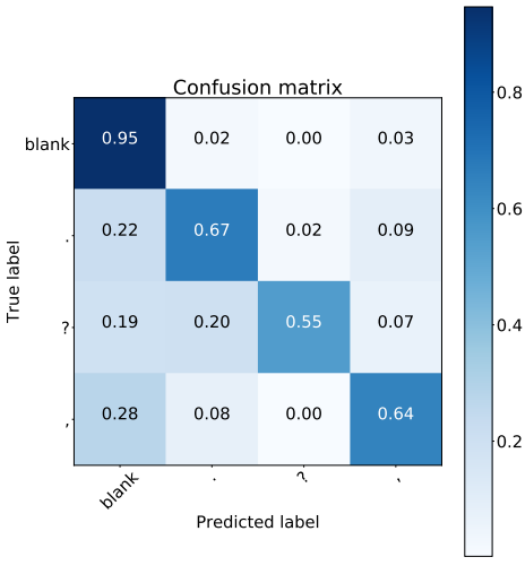

Figura 3. Matriz de confusão do melhor modelo: BLSTM alinhado no tempo.

A melhor acurácia foi com o modelo BLSTM considerando o alinhamento com o tempo das palavras. Os resultados são apresentados na Figura 3. Pela matriz de confusão, o acerto do espaço (*blank*) foi alta (0,95), mas não tanto para as demais pontuações. As CNNs apresentam melhor precisão e os BLSTMs tendem a ter uma melhor revocação. A CNN foi a que teve mais acertos nos casos de pontos de interrogação.

3. Materiais e Métodos

Neste trabalho foi feita a comparação entre duas APIs que realizam a transcrição de áudio para texto, sendo uma delas do Google, conhecida como Cloud Speech-to-Text e a Wit.ai que é utilizada pelo Facebook em bots de chat do messenger. Estas ferramentas processam arquivos de áudio a fim de transcrever a fala para texto. Foi utilizado a linguagem Python e a biblioteca Speech Recognition criada por [Zhang 2019] que serve para simplificar a interação do código desenvolvido com a API. A Figura 4 apresenta a arquitetura geral do ambiente de testes desenvolvido usando a linguagem de programação Python que usará as duas API para transcrição dos áudios da base de dados Braccent.

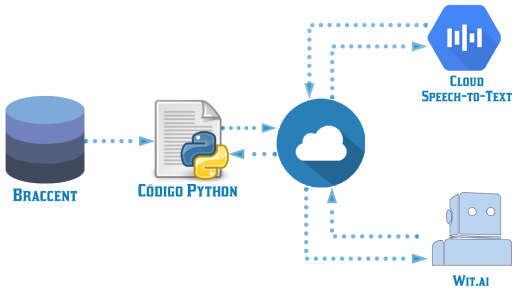

Figura 4. arquitetura geral do ambiente de testes.

Conforme ilustrado na Figura 4, o funcionamento do código ocorre de acordo com o seguinte fluxo:

1. O código desenvolvido em Python coleta o áudio da base de dados Braccent;
2. O código Python desenvolvido solicita o processamento do áudio através da nuvem para a respectiva API;
3. O código fica em modo *stand by* enquanto aguarda a resposta;
4. A API retorna a resposta;
5. O código armazena o texto referente a transcrição do áudio e;
6. Ao final, são computadas as métricas de avaliação.

3.1. Base de dados BRACCENT

A base de dados Braccent foi criada por [Batista 2019], e contém 1.743 amostras de fala. Os voluntários leem 16 sequências de frases, que foram criadas de forma a serem foneticamente balanceadas em seu conjunto e não necessariamente apresentam uma coerência semântica. O balanceamento fonético foi feito via análise da correlação de Spearman com a base Braccent e o CETENFolha[4], que foi de 89%, maior que os 80% do limiar descrito na literatura.

Há 7 sotaques brasileiros existentes na base, que são: nortista, baiano, fluminense, mineiro, carioca, nordestino e sulista. A classificação do sotaque regional de cada áudio foi realizada manualmente pelos alunos do curso de Letras do ano de 2018 do Instituto de

[4]CETENFolha (Corpus de Extractos de Textos Eletrônicos NILC/Folha de S. Paulo), que é um corpus de cerca de 24 milhões de palavras em português brasileiro, criado pelo projeto Processamento computacional do português com base nos textos do jornal Folha de São Paulo que fazem parte do corpus NILC/São Carlos, compilado pelo Núcleo Interinstitucional de Linguística Computacional (NILC).

Estudos da Linguagem na Universidade Estadual de Campinas. Foram 142 locutores em diferentes faixas etárias e níveis de escolaridade, e a quantidade de áudio de cada locutor variou entre 1 e 16.

É importante compreender que o trabalho da [Batista 2019, Batista et al. 2018] teve como objetivo a classificação do sotaque regional e que este trabalho versa sobre a transcrição da fala. Assim, no trabalho de origem, não houve a verificação se cada palavra da frase era pronunciado corretamente pelo voluntário. Para este trabalho, foi feita a validação, e foram selecionados apenas os áudios em que a frase era completamente pronunciada, analisado manualmente por um ser humano. Com isso, alguns áudios foram descartados, pois a pessoa não estava falando a frase identificada e haviam repetições de frases pela mesma pessoa, a Tabela 1 tem a quantidade de áudio por sotaque foi usada. É possível verificar pela tabela que há desbalanceamento quanto à quantidade de áudios por sotaque, com muito mais exemplares do sulista enquanto há poucos nortistas.

Tabela 1. Base de dados Braccent por sotaque, totalizando 1.648 áudios.

Sotaque	Baiano	Carioca	Fluminense	Mineiro	Nordestino	Nortista	Sulista
Total	173	81	110	137	333	24	790

O trabalho original usava 1.743 amostras de fala, enquanto este trabalho usa 1.648 áudios, sendo 95 áudios a menos. Em comparação com as informações do trabalho original, há 10 áudios a menos no Baiano; 1 a menos no Carioca; 4 a menos no Fluminense; 11 áudios a menos no sotaque Mineiro; 11 a menos no Nordestino; 3 a menos no Nortista e; 55 a menos no Sulista.

3.2. Métricas de Comparação

Neste trabalho foi utilizado a métrica de distância de Levenshtein para analisar o desempenho das API's. A distância de Levenshtein foi introduzida em 1995 por Kessler para medir a distância entre dialetos [Kessler 1995]. Dada pelo número mínimo de operações necessárias para transformar um trecho no outro. Existem três tipos de operações para realizar o processamento: inserção, exclusão e substituição de caracteres [Beijering et al. 2008].

Seu funcionamento ocorre pelo preenchimento de uma matriz nxm, na qual n e m são o número de caracteres das duas *strings* a serem comparadas. Como exemplo, vamos comparar as *strings* "TESTE" em "TECLA", que origina uma matriz $5x5$, conforme a matriz mais à esquerda da Figura 5. Dando início ao preenchimento das células restantes, existem duas possibilidades que devem ser observadas para realizar o preenchimento. Devemos verificar se as *substring* são iguais ou não. Analisando a célula $(1, 1)$, deve-se comparar os caracteres 'T' e 'T', nesse caso o campo é preenchido com o valor 0 (zero), pois são iguais. Como os caracteres adicionados em cada *substring* são os mesmos, a matriz resultante do algoritmo sempre é transposta.

		T	E	S	T	E
	0	1	2	3	4	5
T	1					
E	2					
C	3					
L	4					
A	5					

		T	E	S	T	E
	0	1	2	3	4	5
T	1	0	1			
E	2	1	0			
C	3					
L	4					
A	5					

		T	E	S	T	E
	0	1	2	3	4	5
T	1	0	1	2	3	4
E	2	1	0	1	2	3
C	3	2	1	1	2	3
L	4	3	2	2	2	3
A	5	4	3	3	3	3

Figura 5. Passo a passo da execução da métrica de Levenshtein.

Avançando um caractere, compara-se as *substrings* “T” com “TE” para a posição $(1, 2)$, a comparação de “TE” com “T” para a posição $(2, 1)$ e comparação de “TE” com “TE” para a posição $(2, 2)$. Cada célula é preenchida da seguinte forma: à esquerda (custo da inclusão), acima (custo da exclusão) e na diagonal (custo da substituição). O custo de inclusão é 1, o custo de exclusão é 1 e o custo de substituição é 0 (são iguais). O resultado é a matriz do meio da Figura 5.

Continua-se o preenchimento dos campos restantes seguindo o processo explicado anteriormente. Na próxima comparação, *substrings* “TEC” e “TES”, na diagonal $(3, 3)$ o valor é 1, pois deve-se substituir um caractere. E os custos de inclusão (células na linha até a diagonal ou células à esquerda) e exclusão (células na coluna até a diagonal ou células acima) são preenchidos. Cada célula da matriz representa o custo mínimo para transformar a respectiva parte da *string* na outra, portanto, a última célula da matriz representa o valor mínimo para transformar a *string* “TESTE” em “TECLA”. Ao final, chega-se à matriz mais a direita da Figura 5. A posição $(5, 5)$ é o resultado da distância de Levenshtein, que é o custo mínimo para transformar “TESTE” em “TECLA”, que são 3 operações, de acordo com o algoritmo.

Além do Levenshtein, foi utilizado Levenshtein Normalizado, que é obtido através da divisão do valor da distância Levenshtein pelo tamanho da maior *string*, resultando em um valor percentual da proximidade entre as duas *strings*. Utilizando o exemplo anterior, o Levenshtein Normalizado teria como resultado 0,6 (3 dividido por 5), logo, 60% de semelhança entre as duas *strings*.

4. Resultados e Discussão

Nas próximas seções apresentam-se os resultados e discussão de forma geral, posteriormente separados por sotaque regional. Em cada seção são apresentados os resultados considerando a pontuação das frases (vírgulas, pontos finais, aspas e parágrafos), e sem considerar a pontuação. Por exemplo:

- Frase de comparação considerando a pontuação: “Para ganhar, preciso decifrar o código até amanhã. Os papeis foram rasgados, mas ainda é possível ler informações importantes.”
- Frase de comparação sem considerar a pontuação: “para ganhar preciso decifrar o código até amanhã os papeis foram rasgados mas ainda é possível ler informações importantes”

Já é possível avaliar que os resultados sem considerar a pontuação serão melhores do que o resultado considerando a pontuação. Todos os resultados da ferramenta Google Cloud não possuem pontuações.

4.1. Comparação Geral

Nesta seção, consideramos os resultados de toda a base de dados testada. A Tabela 2 apresenta os resultados do Google Cloud e do Wit.ai considerando a pontuação e a Tabela 3 sem a pontuação, os melhores resultados estão em negrito para facilitar a comparação. São apresentadas a média, a mediana da distância de Levenshtein; a média, a mediana de Levenshtein normalizado, a quantidade de acertos (# Acertos) e o percentual de acertos (% de Acerto). Quanto menor a distância de Levenshtein, melhor o resultado e quanto maior o valor de Levenshtein normalizado, melhor o resultado. A quantidade de acertos é simplesmente uma comparação de igualdade de *strings*, exatamente iguais à frase original lida pelo voluntário, assim quanto maior o valor, melhor o resultado.

Tabela 2. Resultado das APIs considerando a pontuação

Métrica	Estatística	Google Cloud	Wit.ai
Levenshtein	Média	18,29	**5,85**
	Mediana	14	**4**
Normalized Levenshtein	Média	89%	**96%**
	Mediana	91%	**97%**
	# Acertos	0	**81**
	% de Acerto	0	**4,91%**

Tabela 3. Resultado das APIs sem considerar a pontuação

Métrica	Estatística	Google Cloud	Wit.ai
Levenshtein	Média	13,54	**3,21**
	Mediana	9	**2**
Normalized Levenshtein	Média	92%	**98%**
	Mediana	94%	**98%**
	# Acertos	126	**543**
	% de Acerto	7,64%	**32,94%**

Pelos resultados da Tabela 2 e a Tabela 3, a ferramenta Wit.ai apresentou melhores resultados em todas as métricas quando comparado ao Google Cloud. Acertar a pontuação é um desafio, logo os resultados considerando a pontuação são piores que sem considerá-la. Mesmo assim, o Wit.ai acertou 81 áudios completos com pontuação. O Google Cloud não retorna resultado com pontuação, logo, a indicação de 0 (zero) acertos. Ressalta-se que sem considerar a pontuação, a quantidade de acertos do Wit.ai aumenta em 6,7 vezes (543 áudios). Em ambos os resultados, é possível verificar que o Wit.ai teve resultados melhores, em todas as estatísticas, que o Google Cloud.

4.2. Resultado por sotaque regional

Nesta seção serão discutidos os resultados por sotaque. Inicialmente, apresentando os resultados do Google Cloud, depois do Wit.ai, e ao final comparando os resultados.

Tabela 4. Resultado do Google Cloud considerando a pontuação para os diferentes sotaques (Flu = Fluminense, Norde = Nordestino, Nort = Nortista, Sul = Sulista)

Google Cloud	Estatísticas	Baiano	Carioca	Flu	Mineiro	Norde	Nort	Sul
Levenshtein	Média	**13,84**	42,17	18,22	22,26	15,62	18,20	17,25
	Mediana	13	28	14	14	**12**	14,5	14
Normalized Levenshtein	Média	**0,92**	0,75	0,89	0,87	0,91	0,89	0,90
	Mediana	**0,92**	0,85	0,91	0,91	**0,92**	0,91	0,91
	# Acertos	0	0	0	0	0	0	0
	% Acertos	0	0	0	0	0	0	0

Tabela 5. Resultado do Google Cloud sem considerar a pontuação para os diferentes sotaques (Flu = Fluminense, Norde = Nordestino, Nort = Nortista, Sul = Sulista)

Google Cloud	Estatísticas	Baiano	Carioca	Flu	Mineiro	Norde	Nort	Sul
Levenshtein	Média	**9,05**	37,81	13,44	17,37	10,73	13,45	12,57
	Mediana	**7**	24	9	9	**7**	9,5	9
Normalized Levenshtein	Média	**0,94**	0,77	0,92	0,89	0,93	0,92	0,92
	Mediana	**0,95**	0,87	0,94	0,94	**0,95**	0,94	0,94
	# Acertos	15	3	8	9	30	0	**61**
	% Acerto	**8,67**	3,70	7,27	6,56	9	0	7,72

Tabela 6. Resultado da Wit.ai considerando a pontuação para os diferentes sotaques (Flu = Fluminense, Norde = Nordestino, Nort = Nortista, Sul = Sulista)

Wit.ai	Estatísticas	Baiano	Carioca	Flu	Mineiro	Norde	Nort	Sul
Levenshtein	Média	5,41	10,08	6,70	6,53	5,43	**5,37**	5,46
	Mediana	**4**	7	5	**4**	**4**	4,5	**4**
Normalized Levenshtein	Média	**0,96**	0,94	**0,96**	**0,96**	**0,96**	**0,96**	**0,96**
	Mediana	**0,97**	0,95	**0,97**	**0,97**	**0,97**	**0,97**	**0,97**
	# Acertos	10	1	8	9	9	2	**42**
	% Acertos	5,78	1,23	7,27	6,56	2,70	**8,33**	5,31

Tabela 7. Resultado da Wit.ai sem considerar a pontuação para os diferentes sotaques (Flu = Fluminense, Norde = Nordestino, Nort = Nortista, Sul = Sulista)

Wit.ai	Estatísticas	Baiano	Carioca	Flu	Mineiro	Norde	Nort	Sul
Levenshtein	Média	2,78	7,16	4,045	3,83	2,66	**2,58**	2,93
	Mediana	**2**	4	**2**	**2**	**2**	1,5	**2**
Normalized Levenshtein	Média	**0,98**	0,95	0,97	0,97	**0,98**	**0,98**	**0,98**
	Mediana	0,98	0,97	0,98	0,98	0,98	**0,99**	0,98
	# Acertos	58	14	40	44	113	9	**265**
	% Acerto	33,52	17,28	36,36	32,11	33,93	**37,5**	33,54

A Tabela 4 apresenta os resultados do Google Cloud, considerando a pontuação para os diferentes sotaques e a Tabela 5 são os resultados sem considerar a pontuação, os melhores resultados estão em negrito para facilitar a comparação. Para esta ferramenta, é interessante observar que obteve melhores resultados para o sotaque baiano, seguido do nordestino, com e sem a pontuação. O pior resultado foi para o sotaque carioca, pois apresenta valores bem mais altos na distância de Levenshtein e valores mais baixos no Levenshtein normalizado.

A Tabela 6 apresenta os resultados do Wit.ai considerando a pontuação para os diferentes sotaques e a Tabela 7 são os resultados sem considerar a pontuação. Em todos os casos, o Wit.ai apresentou resultados melhores que o Google Cloud. Considerando a pontuação, o Wit.ai foi bem em todos os sotaques, apresentando um resultado um pouco pior para o sotaque carioca (com valores maiores para a distância de Levenshtein). Sem considerar a pontuação, os valores das métricas melhoram, tanto a média quanto o a mediana do Levenshtein Normalizado estão acima de 0,95, e com percentual de acerto acima de 32%, exceto para o sotaque carioca que foi de 17,28%. O Wit.ai é melhor para o sotaque Nortista, seguido do Baiano e do Sulista.

Para facilitar a comparação é apresentada a Figura 6, um gráfico em rede comparando os resultados de Levenshtein, considerando a pontuação, em que cada ponta é um sotaque, a curva azul é o resultado do Wit.ai e a curva laranja é do Google Cloud. É possível notar que a curva de menor área é do Wit.ai, representando o melhor resultado em todos os sotaques, também se nota uma "ponta" nas curvas referente ao sotaque carioca em ambas as ferramentas (resultado pior, de maior valor), sendo mais proeminente no Google Cloud.

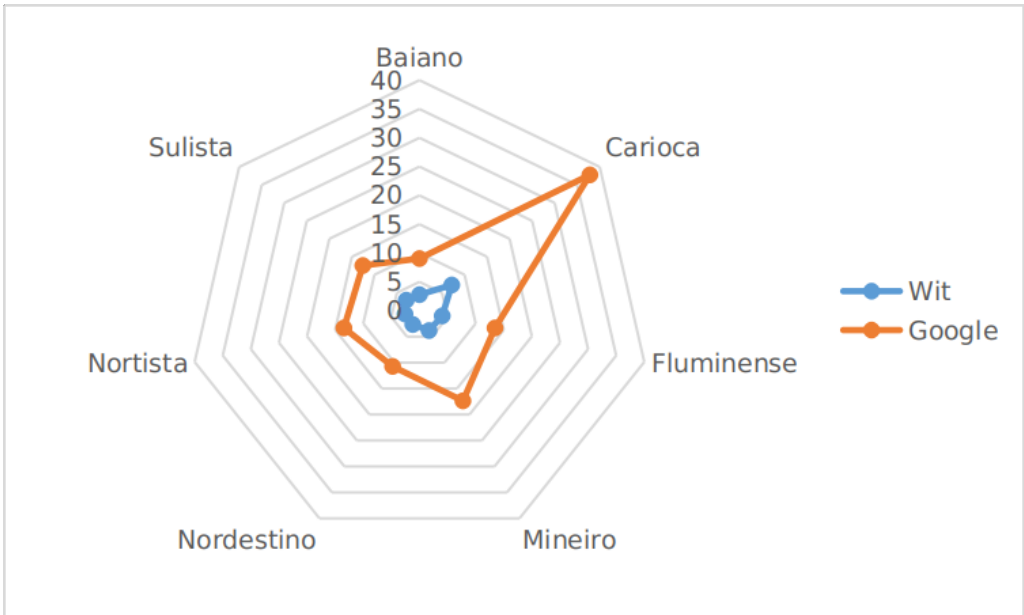

Figura 6. Gráfico em rede comparando os resultados de Levenshtein, sem considerar a pontuação, do Wit.ai e do Google Cloud, por sotaque.

Na análise dos resultados por sotaque, o Google Cloud obteve melhores resultados para o sotaque baiano seguido do nordestino, e os piores resultados foram para o sotaque carioca e não houve um único acerto para o sotaque nortista. O Wit.ai foi bem em todos os

sotaques, apresentando um resultado um pouco pior para o sotaque carioca. Analisando estes resultados, é possível se conjecturar que ambas as ferramentas não foram treinadas com o sotaque carioca e que este sotaque tem características fonéticas bem diferentes, a tal ponto de prejudicar os resultados da transcrição da fala.

5. Conclusões

Neste trabalho foi feito uma análise das ferramentas que realizam a transcrição de áudios. O objetivo da comparação é avaliar o nível de acurácia que cada API possui para o português do Brasil para os diferentes sotaques regionais brasileiros. Foram selecionadas as ferramentas: Google Cloud e Wit.ai, sendo as duas de empresas consolidadas (Google e Facebook respectivamente). Ambas disponibilizam APIs de uso gratuito para a quantidade de áudios deste trabalho. Como fonte de dados foi utilizado a base Braccent, possuindo 1.648 amostras de fala divididos entre diferentes, sendo eles: nortista, baiano, fluminense, mineiro, carioca, nordestino e sulista.

Para realizar a análise de acertos de cada ferramenta, foi utilizada a métrica de Levenshtein e a Levenshtein normalizada, que consistem em definir o menor número necessário de operações para transformar o texto transcrito para o original. O ideal para a métrica de Levenshtein seria o resultado ser igual a 0, significando que os dois trechos são idênticos. Portanto quanto maior o resultado dessa métrica, pior é a precisão da transcrição da API, refletindo que foram necessárias mais operações (inserção, exclusão ou substituição de caracteres) para transformar o trecho produzido pela ferramenta.

Por meio da biblioteca Speech Recognition, foi feita a chamada de ambas as ferramentas para realizar a transcrição dos áudios da base. Foi observado que, ao contrário da Wit.ai, o Google Cloud não apresenta os resultados com pontuação. Portanto foram realizadas duas análises, levando em consideração a pontuação e outra desconsiderando-a. Comparando os dados das duas APIs, é possível perceber que a Wit.ai apresenta melhores resultados que o Google Cloud. A média da métrica de Levenshtein Normalizado é de 0,96, enquanto o Google Cloud é de 0,89 considerando a pontuação. O Wit.ai acertou 81 áudios completos com pontuação. Ressalta-se que sem considerar a pontuação, essa quantidade aumenta em 6,7 vezes, acertando 543 áudios.

Na análise dos resultados por sotaque, o Google Cloud obteve melhores resultados para o sotaque baiano seguido do nordestino, e os piores resultados foram para o sotaque carioca e não houve um único acerto para o sotaque nortista. Em todos os sotaques, o Wit.ai apresentou resultados melhores que o Google Cloud. O Wit.ai foi bem em todos os sotaques, apresentando um resultado um pouco pior para o sotaque carioca. Analisando estes resultados, é possível se conjecturar que ambas as ferramentas não foram treinadas com o sotaque carioca e que este sotaque tem características fonéticas bem diferentes, a tal ponto a prejudicar a transcrição da fala. Ao final, considera-se que o Wit.ai apresentou resultados melhores em todos os cenários, além de transcrever a pontuação.

Embora os objetivos propostos no trabalho tenham sido alcançados, algumas melhorias são possíveis visando trabalhos futuros: utilizar outras bases de dados de áudios com sotaques; realizar experimentos com outros sistemas de transcrição de fala; e analisar os resultados utilizando outras métricas de comparação, tal como a taxa de acertos por palavras (WER, do inglês *Word Error Rate*).

6. Agradecimentos

Os autores agradecem à Nathalia Alves Rocha Batista, ao seu orientador Prof. Dr. Lee Luan Ling e coorientador Prof. Dr. Tiago Fernandes Tavares por terem dado acesso à base de dados para este trabalho.

Referências

Ahmed, A. et al. (2019). Vfnet: A convolutional architecture for accent classification. In *2019 IEEE 16th India Council International Conference (INDICON)*, pages 1–4. IEEE.

Batista, N. A. R. (2019). *Estudo sobre identificação automática de sotaques regionais brasileiros baseada em modelagens estatísticas e técnicas de aprendizado de máquina*. Dissertação, Curso de Mestrado em Engenharia Elétrica, na Área de Telecomunicações e Telemática, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas.

Batista, N. A. R. et al. (2018). Detecção automática de sotaques regionais brasileiros: A importância da validação cross-datasets. In *Anais do XXXVI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, pages 939–944, Campina Grande, PB. Sociedade Brasileira de Telecomunicações.

Beijering, K., Gooskens, C., and Heeringa, W. (2008). Predicting intelligibility and perceived linguistic distance by means of the levenshtein algorithm. *Linguistics in the Netherlands*, 25(1):13–24.

Biadsy, F., Hirschberg, J., and Habash, N. (2009). Spoken arabic dialect identification using phonotactic modeling. In *Proceedings of the eacl 2009 workshop on computational approaches to semitic languages*, pages 53–61.

Cieri, C., Miller, D., and Walker, K. (2004). The fisher corpus: a resource for the next generations of speech-to-text. In *LREC*, volume 4, pages 69–71.

de Lima, M., Coelho, B., and Takigawa, F. (2021). Ferramentas e recursos disponíveis para reconhecimento de fala em português brasileiro. *Anais do Computer on the Beach*, 12:475–479.

Iinuma, N. M. and de Oliveira Igarashi, M. (2019). Speech-to-text em ligações telefônicas. In *Proceedings of Brazilian Technology Symposium 2019 (BTSym'19)*, volume 1.

Juang, B.-H. and Rabiner, L. R. (2005). Automatic speech recognition–a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67.

Kessler, B. (1995). Computational dialectology in irish gaelic. *arXiv preprint cmp-lg/9503002*.

Lazaridis, A., el Khoury, E., Goldman, J.-P., Avanzi, M., Marcel, S., and Garner, P. N. (2014). Swiss french regional accent identification. In *Odyssey*.

Lima, J. R., da Costa Chagas, L. B., de Lira Tavares, O., and Cury, D. (2015). Reconhecimento de voz para inclusão de deficientes visuais em ambientes virtuais de aprendizagem. *Nuevas Ideas en Informática Educativa TISE*, pages 23–29.

López Herrera, G., Quesada Quirós, L., and Guerrero Blanco, L. A. (2017). Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*, pages 241–250. Springer.

Opidi, A. (2019). Top 10 best speech recognition apis: Google speech, ibm watson, speechapi, and others. https://blog.api.rakuten.net/top-10-best-speech-recognition-apis-google-speech-ibm-watson-speechapi-and-others/. Acesso em: 02 dez. 2020.

Raju, A., Hedayatnia, B., Liu, L., Gandhe, A., Khatri, C., Metallinou, A., Venkatesh, A., and Rastrow, A. (2018). Contextual language model adaptation for conversational agents. *arXiv preprint arXiv:1806.10215*.

Salau, A. O., Olowoyo, T. D., and Akinola, S. O. (2020). Accent classification of the three major nigerian indigenous languages using 1d cnn lstm network model. In *Advances in Computational Intelligence Techniques*, pages 1–16. Springer.

Sharan, R. V. and Moir, T. J. (2016). An overview of applications and advancements in automatic sound recognition. *Neurocomputing*, 200:22–34.

Shi, X. et al. (2021). The accented english speech recognition challenge 2020: open datasets, tracks, baselines, results and methods. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6918–6922. IEEE.

Viglino, T., Motlicek, P., and Cernak, M. (2019). End-to-end accented speech recognition. In *INTERSPEECH*, pages 2140–2144.

Wang, W., Zhang, C., and Wu, X. (2020). Sar-net: A end-to-end deep speech accent recognition network. *arXiv preprint arXiv:2011.12461*.

Weninger, F. et al. (2019). Deep learning based mandarin accent identification for accent robust ASR. In *Proceedings of INTERSPEECH*, pages 510–514.

Ynoguti, C. (1999). *Reconhecimento de Fala Contínua Utilizando Modelos Ocultos de Markov*. PhD thesis, Faculdade de Engenharia Elétrica, Unicamp.

Żelasko, P., Szymański, P., Mizgajski, J., Szymczak, A., Carmiel, Y., and Dehak, N. (2018). Punctuation prediction model for conversational speech. *arXiv preprint arXiv:1807.00543*.

Zhang, A. (2019). Speechrecognition 3.8.1. https://pypi.org/project/SpeechRecognition/. Acesso em: 01 nov. 2020.

Zhang, Z. et al. (2021). Accent recognition with hybrid phonetic features. *arXiv preprint arXiv:2105.01920*.